



SIRIUS

SEQUENZIATORE

Specifica Tecnica

Versione 1.0.0

Ingegneria Del Software AA 2013-2014

Informazioni documento

Titolo documento:	Specifica Tecnica
Data creazione:	2014-02-12
Versione attuale:	1.0.0
Utilizzo:	Esterno
Nome file:	<i>SpecificaTecnica_v1.0.0.pdf</i>
Redazione:	Quaglio Davide
Approvazione:	Giachin Vanni
Distribuito da:	Sirius
Destinato a:	Prof. Vardanega Tullio Prof. Cardin Riccardo Zucchetti S.p.A

Sommario

Descrizione dell'architettura e dei componenti relativi allo sviluppo del progetto *Sequenziatore*.

Diario delle modifiche

Versione	Data	Autore	Ruolo	Descrizione
0.0.1	2014-03-15	Giachin Vanni	?	Stesura introduzione

Indice

1	Introduzione	1
1.1	Scopo del Documento	1
1.2	Scopo del Prodotto	1
1.3	Glossario	1
1.4	Riferimenti	1
1.4.1	Normativi	1
1.4.2	Informativi	1
2	Definizione dell' architettura	3
2.1	Metodo e formalismo di specifica	3
2.2	Architettura generale	3
2.2.1	Componente View	3
2.2.2	Componente Presenter	3
2.2.3	Componente Model	4
2.3	Diagrammi dei package	5
2.3.1	Package sequenziatore.client.view	5
2.3.2	Package sequenziatore.client.presenter	5
2.3.3	Package sequenziatore.client.model	6
3	Descrizione singoli componenti	7
3.1	Package sequenziatore::client::iview	7
3.1.1	Package sequenziatore::client::iview::iuser	7
3.1.2	Package sequenziatore::client::iview::iprocessowner	10
3.2	Package sequenziatore::client::view	13
3.2.1	Package sequenziatore::client::view::user	13
3.2.2	Package sequenziatore::client::view::processowner	19
3.3	Package sequenziatore::client::ipresenter	24
3.3.1	Package sequenziatore::client::ipresenter::iuser::ilogic	24
3.3.2	Package sequenziatore::client::ipresenter::iprocessowner::ilogic	25
3.3.3	sequenziatore::client::ipresenter::iservercommunication	27
3.4	Package sequenziatore::client::presenter	27
3.4.1	Package sequenziatore::client::presenter::user::logic	27
3.4.2	Package sequenziatore::client::presenter::processowner::logic	30
3.4.3	sequenziatore::client::presenter::servercommunication	35
3.5	Package sequenziatore::client::imodel	37
3.5.1	Package sequenziatore::client::imodel::iprocessowner	37
3.5.2	Package sequenziatore::client::imodel::iuser	37
3.6	Package sequenziatore::client::model	37

3.6.1	Package sequenziatore::client::model	37
3.6.2	Package sequenziatore::client::model:processowner	38
3.6.3	Package sequenziatore::client::model:user	38
3.7	Package sequenziatore::server::presenter	40
3.7.1	Package sequenziatore::server::presenter::icommunication	40
3.7.2	Package sequenziatore::server::presenter::communication	40
3.7.3	Package sequenziatore::server::presenter::iuser	42
3.7.4	Package sequenziatore::server::presenter::user	43
3.7.5	Package sequenziatore::server::presenter::iprocessowner	45
3.7.6	Package sequenziatore::server::presenter::processowner	46
3.8	Package sequenziatore::server::model	48
3.8.1	Package sequenziatore::server::model::daouser	48
3.8.2	Package sequenziatore::server::model::daoprocessowner	49
3.8.3	Package sequenziatore::server::model::daoprocess	50
3.8.4	Package sequenziatore::server::model::daostep	51
4	Design pattern	53
4.1	Design pattern architetturali	53
4.1.1	Model View Presenter	53
4.2	Design pattern strutturali	53
4.2.1	Adapter	53
4.2.2	Decorator	53
4.2.3	Facade	53
4.2.4	Proxy	54
4.3	Design pattern creazionali	54
4.3.1	Singleton	54
4.3.2	Abstract Factory	54
4.4	Design pattern comportamentali	54
4.4.1	Command	54
4.4.2	Iterator	54
4.4.3	Observer	55
4.4.4	Strategy	55
4.4.5	Template method	55
5	Tracciamento	56
5.1	Tracciamento package - componenti	56
5.2	Tracciamento componenti - requisiti	58
5.3	Tracciamento requisiti - componenti	63

1 Introduzione

1.1 Scopo del Documento

Lo scopo di questo documento è la definizione delle specifiche progettuali del prodotto *software Sequenziatore*.

Viene quindi presentata l'architettura ad alto livello del sistema, e la descrizione delle singole componenti e dei *design pattern*_G utilizzati.

1.2 Scopo del Prodotto

Lo scopo del progetto *Sequenziatore*, è di fornire un servizio di gestione di processi definiti da una serie di passi da eseguirsi in sequenza o senza un ordine predefinito, utilizzabile da dispositivi mobili di tipo *smartphone* o *tablet*.

1.3 Glossario

Al fine di rendere più leggibili e comprensibili i documenti, i termini tecnici, di dominio, gli acronimi e le parole che necessitano di essere chiarite, sono riportate nel documento *Glossario_v2.0.0.pdf*.

Ciascuna occorrenza dei vocaboli presenti nel *Glossario* è seguita da una “G” maiuscola in pedice.

1.4 Riferimenti

1.4.1 Normativi

- Norme di Progetto: *NormeDiProgetto_v2.0.0.pdf*.
- Analisi dei Requisiti: *AnalisiDeiRequisiti_v2.0.0.pdf*.

1.4.2 Informativi

- Design Patterns: Elementi per il riuso di software ad oggetti - Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides (2002);
- Learning JavaScript Design Patterns, Addy Osmani, Volume 1.5.2:
<http://addyosmani.com/resources/essentialjsdesignpatterns/book>;
- Regolamento dei documenti, prof. Vardanega Tullio:
<http://www.math.unipd.it/~tullio/IS-1/2013/>;
- Dispense di ingegneria del software modulo A:
 - Progettazione software, prof. Vardanega Tullio:
<http://www.math.unipd.it/~tullio/IS-1/2013/Dispense/P09.pdf>;

- Diagrammi delle classi e degli oggetti, prof. Cardin Riccardo:
<http://www.math.unipd.it/~tullio/IS-1/2013/Dispense/E02a.pdf>;
- Diagrammi di sequenza, prof. Cardin Riccardo:
<http://www.math.unipd.it/~tullio/IS-1/2013/Dispense/E03a.pdf>;
- Diagrammi di attività, prof. Cardin Riccardo:
<http://www.math.unipd.it/~tullio/IS-1/2013/Dispense/E03b.pdf>;
- Introduzione ai design pattern, prof. Cardin Riccardo:
<http://www.math.unipd.it/~tullio/IS-1/2013/Dispense/E04.pdf>;
- Diagrammi dei package, prof. Cardin Riccardo:
<http://www.math.unipd.it/~tullio/IS-1/2013/Dispense/E05.pdf>;
- Dispense di ingegneria del software modulo B:
 - Design pattern: Model-View-Controller, prof. Cardin Riccardo:
http://www.math.unipd.it/~rcardin/pdf/Design%20Pattern%20-%20Model%20View%20Controller_4x4.pdf;
 - Design pattern strutturali, prof. Cardin Riccardo:
http://www.math.unipd.it/~rcardin/pdf/Design%20Pattern%20Strutturali_4x4.pdf;
 - Design pattern creazionali, prof. Cardin Riccardo:
http://www.math.unipd.it/~rcardin/pdf/Design%20Pattern%20Creazionali_4x4.pdf;
 - Design pattern comportamentali, prof. Cardin Riccardo:
http://www.math.unipd.it/~rcardin/pdf/Design%20Pattern%20Comportamentali_4x4.pdf;
 - Esercizi sugli errori rilevati in RP, prof. Cardin Riccardo:
http://www.math.unipd.it/~rcardin/pdf/Esercitazione%20-%20Errori%20comuni%20RP_4x4.pdf;

2 Definizione dell' architettura

2.1 Metodo e formalismo di specifica

L' architettura del sistema è la struttura del sistema, che comprende gli elementi *software*, la visibilità esterna di questi elementi e la relazione tra loro. Questo documento andrà ad esporre le componenti di alto livello del sistema che verranno poi approfondite nel periodo di Progettazione di dettaglio e codifica, per analizzare l' architettura del sistema il *Sequenziatore* seguirà l' approccio *top-down*, quindi innanzitutto si analizzerà il sistema fornendone una descrizione generale per poi scomporre le varie parti andando sempre più in dettaglio analizzando le singole componenti. Successivamente si analizzeranno i *design pattern* adottati e come verranno implementati. Per esporre al meglio l' architettura del sistema e il suo funzionamento di alto livello si utilizzeranno diagrammi dei *package*, delle classi, di attività e di sequenza seguendo quanto imposto dalle *NormeDiProgetto_v2.0.0.pdf*.

2.2 Architettura generale

Il sistema *Sequenziatore* è composto innanzitutto da due parti principali, un lato *Client* e un lato *Server*, per la loro progettazione si è tenuto conto dei principi della **riusabilità** e del **basso accoppiamento**, quindi si cercherà di progettare le due parti distintamente e senza dipendenze mantenendo all' oscuro il funzionamento del **server** al **client** e viceversa.

Dopo un' attenta analisi si è deciso di adottare il *design pattern* architetturale **MVP** seguendo la variante *Passive View*. Tale scelta è stata fatta per i seguenti motivi:

- ottenere una *view* priva di *application logic* che verrà delegata al *presenter*, questo semplificherà i test, infatti la vista sarà un semplice *mockup* e il *presenter* può essere testato separatamente dalla vista;
- offre un' architettura solida e mantenibile attraverso il disaccoppiamento massimo tra viste e modelli.

2.2.1 Componente View

Questa componente andrà a costituire la **GUI** del sistema e sarà divisa in due parti, lato amministratore e quello utente. Entrambe le parti non dovranno fare altro che offrire un' interfaccia agli utenti del sistema utilizzando HTML5, CSS e Javascript.

2.2.2 Componente Presenter

Il *presenter* andrà a rappresentare la *application logic* del sistema e sarà divisa tra il lato Server e il lato Client. Le funzionalità che andrà a ricoprire saranno:

- gestire parte della comunicazione tra le due parti;
- acquisire i dati inseriti dagli utenti ed elaborarli;
- mantenere aggiornata la vista in modo che rifletta i cambiamenti del model.

La maggior parte delle funzionalità saranno ricoperte dal *presenter* lato *client*, in quanto sarà responsabile di:

- aggiornare le viste dell' utente e dell' amministratore;
- controllare i dati inseriti dall' utente e quando possibile elaborarli;
- passare i dati che necessitano di elaborazione lato *server* al *presenter* dello stesso;
- ricevere le risposte dal lato *server* e fornire all' utente la vista aggiornata.

I ruoli del *presenter* lato *server* sono:

- ricevere le richieste dal *presenter* lato *client* ed elaborarle, restituendo poi il risultato sotto forma di **JSON**;
- informare il lato client delle modifiche effettuate sul model.

2.2.3 Componente Model

Questa componente andrà a rappresentare la *business logic* del sistema, e sarà suddivisa tra *client* in minima parte e *server*. I ruoli del componente lato *client* saranno di mantenere traccia dell' utente autenticato e di salvare, qualora si decida di implementare questa funzionalità, i dati come per esempio coordinate gps e immagini quando il dispositivo non disporrà di connessione internet.

2.3 Diagrammi dei package

Il seguente diagramma descrive le dipendenze intercorse fra i vari package_G del sistema Sequenziatore. I diagrammi dei package —g— descrivono le dipendenze che intercorrono tra i vari package_G che compongono il sistema. Figura 3: Diagramma dei package del prodotto MyTalk. Il sistema Sequenziatore è composto da due macro package_G:

1. sequenziatore.client: le componenti di questo package_G realizzano la parte front-end_G del sistema Sequenziatore
2. sequenziatore.server: le componenti di questo package_G realizzano la parte back-end_G del sistema Sequenziatore

Il package_G sequenziatore.client è composto dai seguenti package_G:

- sequenziatore.client.view;
- sequenziatore.client.presenter;
- sequenziatore.client.model.

Come è facilmente intuibile, la struttura del package_G sequenziatore.client si basa sulla struttura del design patter architetturale Model View Presenter, scelto dal team Sirius per poter separare la logica di presentazione dei dati dalla logica di business.

I package_G che compongono il package_G sequenziatore.server sono:

- sequenziatore.server.presenter;
- sequenziatore.server.model.

2.3.1 Package sequenziatore.client.view

Il package_G sequenziatore.client.view è composto da i seguenti package_G:

- sequenziatore.client.view.admin: contiene le classi e interfacce necessarie a gestire l'interfaccia grafica e a generare gli eventi della parte grafica dell'utente amministratore .
- sequenziatore.client.view.user: contiene le classi e interfacce necessarie a gestire l'interfaccia grafica e a generare gli eventi della parte grafica dell'utente.

2.3.2 Package sequenziatore.client.presenter

Il package_G sequenziatore.client.presenter contiene tutte le classi e interfacce del Presenter della parte client_G del sistema Sequenziatore; ed è composto da i seguenti package_G:

- `sequenziatore.client.presenter.admin`: contiene le classi che costituiscono la componente Presenter per l'utente amministratore, il package_G `sequenziatore.client.presenter.admin` è diviso ulteriormente nei sotto-package_G:
 - `sequenziatore.client.presenter.admin.logic`: gestisce gli eventi generati dalle componenti del package_G `sequenziatore.client.view.admin` e aggiorna la parte grafica dell'utente amministratore;
 - `sequenziatore.client.presenter.admin.serverCommunication`: contiene le componenti necessarie per interfacciarsi alla parte server_G del sistema Sequenziatore e gestire la comunicazione con quest'ultima.
- `sequenziatore.client.presenter.user`: contiene tutti i package_G e le classi che compongono la componente Presenter per l'utente generico e loggato, i sotto-package_G di `sequenziatore.client.presenter.user` sono i seguenti:
 - `sequenziatore.client.presenter.user.logic`: gestisce gli eventi generati dalle componenti del package_G `sequenziatore.client.view.user` e aggiorna la parte grafica per l'utente generico e loggato;
 - `sequenziatore.client.presenter.user.serverCommunication`: contiene le componenti necessarie, per la parte user, per interfacciarsi alla parte server_G del sistema Sequenziatore e gestire la comunicazione con quest'ultima.

2.3.3 Package `sequenziatore.client.model`

Il package_G `sequenziatore.client.model` contiene tutte le classi della componente Model. Il package_G è suddiviso in:

- `sequenziatore.client.model.localDataAdmin`: è composto dalle relative informazioni dell'utente amministratore autenticato al sistema Sequenziatore;
- `sequenziatore.client.model.localDataUser`: è composto dalle relative informazioni dell'utente autenticato al sistema Sequenziatore.

3 Descrizione singoli componenti

3.1 Package sequenziatore::client::iView

3.1.1 Package sequenziatore::client::view::iUser

3.1.1.1 IMainUser

- **Nome:** IMainUser;
- **Package:** sequenziatore::client::view::iUser;
- **Descrizione:** Interfaccia che permette la gestione delle principali componenti dell'Interfaccia grafica dell'utente.

3.1.1.2 IUpdateView

- **Nome:** IUpdateView;
- **Package:** sequenziatore::client::view::iUser;
- **Descrizione:** Interfaccia che permette di gestire l'aggiornameno dei *widget* della componente *view*;

3.1.1.3 ILogin

- **Nome:** ILogin;
- **Package:** sequenziatore::client::view::iUser;
- **Descrizione:** Interfaccia che permette di gestire l'interfaccia grafica relativa alle richieste di autenticazione e chiusura della sessione da parte dell'utente.

3.1.1.4 IRegister

- **Nome:** IRegister;
- **Package:** sequenziatore::client::view::iUser;
- **Descrizione:** Interfaccia che permette di gestire l'interfaccia grafica relativa alle richieste di registrazione da parte dell'utente.

3.1.1.5 IViewData

- **Nome:** IViewData;
- **Package:** sequenziatore::client::view::iUser;
- **Descrizione:** Interfaccia che permette la realizzazione dei *widget* per la visualizzazione dei dati dell'utente.

3.1.1.6 IEditData

- **Nome:** IEditData;
- **Package:** sequenziatore::client::view::iUser;
- **Descrizione:** Interfaccia che permette la realizzazione dei *widget* per la modifica dei dati personali dell'utente.

3.1.1.7 IChangePassword

- **Nome:** IChangePassword;
- **Package:** sequenziatore::client::view::iUser;
- **Descrizione:** Interfaccia che permette la realizzazione dei *widget* per la modifica della *password* dell'utente.

3.1.1.8 IOpenProcess

- **Nome:** IOpenProcess;
- **Package:** sequenziatore::client::view::iUser;
- **Descrizione:** Interfaccia che permette di realizzare i *widget* per consentire la ricerca e la selezione di processi.

3.1.1.9 IManagementSelectedProcess

- **Nome:** IManagementSelectedProcess;
- **Package:** sequenziatore::client::view::iUser;
- **Descrizione:** Interfaccia che permette di realizzare i *widget* per visualizzare lo stato corrente del processo selezionato e i vincoli per concludere il passo.

3.1.1.10 ISendData

- **Nome:** ISendData;
- **Package:** sequenziatore::client::view::iUser;
- **Descrizione:** Interfaccia che permette di realizzare i *widget* per inviare i dati richiesti per la conclusione del passo.

3.1.1.11 ISendText

- **Nome:** ISendData;
- **Package:** sequenziatore::client::view::iUser;
- **Descrizione:** Interfaccia che permette di realizzare i *widget* per inserire il testo da inviare per concludere il passo.

3.1.1.12 ISendNumb

- **Nome:** ISendNumb;
- **Package:** sequenziatore::client::view::iUser;
- **Descrizione:** Interfaccia che permette di realizzare i *widget* per inserire i dati numerici da inviare per concludere il passo.

3.1.1.13 ISendPosition

- **Nome:** ISendPosition;
- **Package:** sequenziatore::client::view::iUser;
- **Descrizione:** Interfaccia che permette di realizzare i *widget* per inviare la posizione geografica per la conclusione di un passo. Inoltre consente di visualizzare eventuali messaggi d'errore nella rilevazione delle coordinate.

3.1.1.14 ISendImage

- **Nome:** ISendImage;
- **Package:** sequenziatore::client::view::iUser;
- **Descrizione:** Interfaccia che permette di realizzare i *widget* per inserire le immagini richieste per la conclusione del passo.

3.1.1.15 ISendPhoto

- **Nome:** ISendPhoto;
- **Package:** sequenziatore::client::view::iUser;
- **Descrizione:** Interfaccia che permette di realizzare i *widget* per consentire all'utente di scattare le fotografie richieste dal passo in esecuzione, e di inviarle.

3.1.1.16 IEndSelectedProcess

- **Nome:** IEndSelectedProcess;
- **Package:** sequenziatore::client::view::iUser;
- **Descrizione:** Interfaccia che permette di realizzare i *widget* per visualizzare l'esito del processo e consentire le operazioni di conclusione del processo.

3.1.1.17 IPrintProcess

- **Nome:** IPrintProcess;
- **Package:** sequenziatore::client::view::iUser;
- **Descrizione:** Interfaccia che permette di realizzare i *widget* per consentire il salvataggio dei *report* di fine processo.

3.1.1.18 IPreviewProcess

- **Nome:** IPreviewProcess;
- **Package:** sequenziatore::client::view::iUser;
- **Descrizione:** Interfaccia che permette di realizzare i *widget* per consentire la visualizzazione dei *report* di fine processo.

3.1.2 Package sequenziatore::client::view::iprocessoowner

3.1.2.1 IMainProcessOwner

- **Nome:** IMainProcessOwner;
- **Package:** sequenziatore::client::view::iprocessoowner;
- **Descrizione:** Interfaccia che permette la gestione delle principali componenti dell'interfaccia grafica dell'utente *process owner_G*.

3.1.2.2 IUpdateView

- **Nome:** IUpdateView;
- **Package:** sequenziatore::client::view::iprocessoowner;
- **Descrizione:** Interfaccia che permette di gestire l'aggiornamento dei *widget_G* della componente *view*.

3.1.2.3 ILogin

- **Nome:** ILogin;
- **Package:** sequenziatore::client::view::iprocessowner;
- **Descrizione:** Interfaccia che permette di gestire l'interfaccia grafica relativa alle richieste di autenticazione e chiusura della sessione da parte dell'utente *process owner*.

3.1.2.4 ISetProcess

- **Nome:** ISetProcess;
- **Package:** sequenziatore::client::view::iprocessowner;
- **Descrizione:** Interfaccia che permette di gestire l'interfaccia grafica che consente di creare nuovi processi.

3.1.2.5 IAddStep

- **Nome:** IAddStep;
- **Package:** sequenziatore::client::view::iprocessowner;
- **Descrizione:** Interfaccia che permette di gestire l'interfaccia grafica che consente di definire un nuovo passo del processo in creazione.

3.1.2.6 IPreviewProcess

- **Nome:** IPreviewProcess;
- **Package:** sequenziatore::client::view::iprocessowner;
- **Descrizione:** Interfaccia che permette realizzare i *widget* che consentono di visualizzare l'anteprima del processo in creazione.

3.1.2.7 IOpenProcess

- **Nome:** IOpenProcess;
- **Package:** sequenziatore::client::view::iprocessowner;
- **Descrizione:** Interfaccia che permette di realizzare i *widget* che consentono di aprire un processo tramite ricerca o selezione da una lista.

3.1.2.8 IManagmentSelectedProcess

- **Nome:** IManagmentSelectedProcess;
- **Package:** sequenziatore::client::view::iprocessowner;
- **Descrizione:** Interfaccia che permette di realizzare i *widget* che consentono di gestire i processi selezionati.

3.1.2.9 ICheckStep

- **Nome:** ICheckStep;
- **Package:** sequenziatore::client::view::iprocessowner;
- **Descrizione:** Interfaccia che permette di realizzare i *widget* che consentono di gestire il controllo dei passi che richiedono intervento umano.

3.1.2.10 IStatistics

- **Nome:** IStatistics;
- **Package:** sequenziatore::client::view::iprocessowner;
- **Descrizione:** Interfaccia che permette di realizzare i *widget* che consentono di gestire l'accesso alle informazioni statistiche sui processi.

3.1.2.11 IUserReceivedData

- **Nome:** IUserReceivedData;
- **Package:** sequenziatore::client::view::iprocessowner;
- **Descrizione:** Classe che permette di realizzare i *widget* che consentono di gestire l'accesso ai dati inviati al *server_G* dagli utenti;

3.1.2.12 IInviteUser

- **Nome:** IInviteUser;
- **Package:** sequenziatore::client::view::iprocessowner;
- **Descrizione:** Interfaccia che permette di realizzare i *widget* che consentono di gestire i permessi di iscrizione ad un processo da parte degli utenti.

3.2 Package sequenziatore::client::view

3.2.1 Package sequenziatore::client::view::user

3.2.1.1 MainUser

- **Nome:** MainUser;
- **Package:** sequenziatore::client::view::user;
- **Descrizione:** Classe che permette la gestione delle principali componenti dell'interfaccia grafica dell'utente;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia `sequenziatore::client::view::iUser::IMainUser` e comunica con il *presenter* utilizzando metodi della classe `sequenziatore::client::presenter::user::logic::MainLogic`.

3.2.1.2 UpdateView

- **Nome:** UpdateView;
- **Package:** sequenziatore::client::view::user;
- **Descrizione:** Classe che permette di gestire l'aggiornameno dei *widget_G* della componente *view*;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia `sequenziatore::client::view::iUser::IUpdateView` e aggiorna le componenti della *view* comunicando con le seguenti classi:

- `sequenziatore::client::view::user::MainUser`;
- `sequenziatore::client::view::user::Login`;
- `sequenziatore::client::view::user::Register`;
- `sequenziatore::client::view::user::ViewData`;
- `sequenziatore::client::view::user::EditData`;
- `sequenziatore::client::view::user::ChangePassword`;
- `sequenziatore::client::view::user::OpenProcess`;
- `sequenziatore::client::view::user::ManagementSelectedProcess`;
- `sequenziatore::client::view::user::SendData`;

- sequenziatore::client::view::user::SendText;
- sequenziatore::client::view::user::SendNumb;
- sequenziatore::client::view::user::SendPosition;
- sequenziatore::client::view::user::SendImage;
- sequenziatore::client::view::user::SendPhoto;
- sequenziatore::client::view::user::EndSelectedProcess;
- sequenziatore::client::view::user::PrintProcess;
- sequenziatore::client::view::user::PreviewProcess.

3.2.1.3 Login

- **Nome:** Login;
- **Package:** sequenziatore::client::view::user;
- **Descrizione:** Classe che permette di gestire dell'interfaccia grafica relativa alle richieste di autenticazione e chiusura della sessione da parte dell'utente;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia

sequenziatore::client::view::iUser::ILogin e comunica con il *presenter* utilizzando metodi della classe

sequenziatore::client::presenter::user::logic::MainLogic.

3.2.1.4 Register

- **Nome:** Register;
- **Package:** sequenziatore::client::view::user;
- **Descrizione:** Classe che permette di gestire dell'interfaccia grafica relativa alle richieste di registrazione da parte dell'utente;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia

sequenziatore::client::view::iUser::IRegister e comunica con il *presenter* utilizzando metodi della classe

sequenziatore::client::presenter::user::logic::MainLogic.

3.2.1.5 ViewData

- **Nome:** ViewData;
- **Package:** sequenziatore::client::view::user;
- **Descrizione:** Classe che permette la realizzazione dei *widget* che consentono visualizzazione dei dati dell'utente;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia `sequenziatore::client::view::iUser::IViewData` e comunica con il *presenter* utilizzando metodi della classe `sequenziatore::client::presenter::user::logic::MainLogic`.

3.2.1.6 EditData

- **Nome:** EditData;
- **Package:** sequenziatore::client::view::user;
- **Descrizione:** Classe che permette la realizzazione dei *widget* che consentono la modifica dei dati personali dell'utente;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia `sequenziatore::client::view::iUser::IEditData` e comunica con il *presenter* utilizzando metodi della classe `sequenziatore::client::presenter::user::logic::MainLogic`.

3.2.1.7 ChangePassword

- **Nome:** ChangePassword;
- **Package:** sequenziatore::client::view::user;
- **Descrizione:** Classe che permette la realizzazione dei *widget* che consentono la modifica della *password* dell'utente;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia `sequenziatore::client::view::iUser::IChangePassword` e comunica con il *presenter* utilizzando metodi della classe `sequenziatore::client::presenter::user::logic::MainLogic`.

3.2.1.8 OpenProcess

- **Nome:** OpenProcess;
- **Package:** sequenziatore::client::view::user;
- **Descrizione:** Classe che permette di realizzare i *widget* per consentire l'apertura di un processo tramite ricerca o selezionandolo da una lista;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia sequenziatore::client::view::iUser::IOpenProcess e comunica con il *presenter* utilizzando metodi della classe sequenziatore::client::presenter::user::logic::MainLogic.

3.2.1.9 ManagementSelectedProcess

- **Nome:** ManagementSelectedProcess;
- **Package:** sequenziatore::client::view::user;
- **Descrizione:** Classe che permette di realizzare i *widget* per consentire la visualizzazione dello stato del processo selezionato e i vincoli per concludere il passo in corso;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia sequenziatore::client::view::iUser::IManagementSelectedProcess e comunica con il *presenter* utilizzando metodi della classe sequenziatore::client::presenter::user::logic::MainLogic.

3.2.1.10 SendData

- **Nome:** SendData;
- **Package:** sequenziatore::client::view::user;
- **Descrizione:** Classe che permette di realizzare i *widget* per consentire l'invio dei dati richiesti per la conclusione del passo in esecuzione;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia sequenziatore::client::view::iUser::ISendData e comunica con il *presenter* utilizzando metodi della classe sequenziatore::client::presenter::user::logic::MainLogic.

3.2.1.11 SendText

- **Nome:** SendData;
- **Package:** sequenziatore::client::view::user;
- **Descrizione:** Classe che permette di realizzare i *widget* che consentono di inserire il testo da inviare per concludere il passo in esecuzione;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia

sequenziatore::client::view::iUser::ISendText e comunica con il *presenter* utilizzando metodi della classe

sequenziatore::client::presenter::user::logic::MainLogic.

3.2.1.12 SendNumb

- **Nome:** SendNumb;
- **Package:** sequenziatore::client::view::user;
- **Descrizione:** Classe che permette agli oggetti che la implementano di realizzare i *widget* che consentono di inserire i dati numerici da inviare per concludere il passo in esecuzione;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia

sequenziatore::client::view::iUser::ISendNumb e comunica con il *presenter* utilizzando metodi della classe

sequenziatore::client::presenter::user::logic::MainLogic.

3.2.1.13 SendPosition

- **Nome:** SendPosition;
- **Package:** sequenziatore::client::view::user;
- **Descrizione:** Classe che permette di realizzare i *widget* che consentono di inviare la posizione geografica richiesta per la conclusione del passo in esecuzione;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia

sequenziatore::client::view::iUser::ISendPosition e comunica con il *presenter* utilizzando metodi della classe

sequenziatore::client::presenter::user::logic::MainLogic.

3.2.1.14 SendImage

- **Nome:** SendImage;
- **Package:** sequenziatore::client::view::user;
- **Descrizione:** Classe che permette di realizzare i *widget* che consentono di inserire le immagini richieste per concludere il passo in esecuzione;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia
sequenziatore::client::view::iUser::ISendImage e comunica con il
presenter utilizzando metodi della classe
sequenziatore::client::presenter::user::logic::MainLogic.

3.2.1.15 SendPhoto

- **Nome:** SendPhoto;
- **Package:** sequenziatore::client::view::user;
- **Descrizione:** Classe che permette di realizzare i *widget* che consentono di scattare le fotografie richieste dal passo in esecuzione e di inviarle;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia
sequenziatore::client::view::iUser::ISendPhoto e comunica con il
presenter utilizzando metodi della classe
sequenziatore::client::presenter::user::logic::MainLogic.

3.2.1.16 EndSelectedProcess

- **Nome:** EndSelectedProcess;
- **Package:** sequenziatore::client::view::user;
- **Descrizione:** Classe che permette di realizzare i *widget* che consentono di visualizzare l'esito del processo e di effettuare le operazioni di conclusione del processo;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia
sequenziatore::client::view::iUser::IEndSelectedProcess e comunica
con il *presenter* utilizzando metodi della classe
sequenziatore::client::presenter::user::logic::MainLogic.

3.2.1.17 PrintProcess

- **Nome:** PrintProcess;
- **Package:** sequenziatore::client::view::user;
- **Descrizione:** Classe che permette di realizzare i *widget* che consentono il salvataggio dei *report* sull'esecuzione del processo;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia sequenziatore::client::view::iUser::IPrintProcess e comunica con il *presenter* utilizzando metodi della classe sequenziatore::client::presenter::user::logic::MainLogic.

3.2.1.18 PreviewProcess

- **Nome:** PreviewProcess;
- **Package:** sequenziatore::client::view::user;
- **Descrizione:** Classe che permette agli oggetti che la implementano di realizzare i *widget* per consentire la visualizzazione dei *report* sull'esecuzione del processo;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia sequenziatore::client::view::iUser::IPreviewProcess e comunica con il *presenter* utilizzando metodi della classe sequenziatore::client::presenter::user::logic::MainLogic.

3.2.2 Package sequenziatore::client::view::processowner

3.2.2.1 MainProcessOwner

- **Nome:** MainProcessOwner;
- **Package:** sequenziatore.client::view::processowner;
- **Descrizione:** Classe che permette la gestione delle principali componenti dell'interfaccia grafica dell'utente *process owner*;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia sequenziatore::client::view::iprocessowner::IMainProcessOwner e comunica con il *presenter* utilizzando metodi della classe sequenziatore::client::presenter::processowner::logic::MainLogic.

3.2.2.2 UpdateView

- **Nome:** UpdateView;
- **Package:** sequenziatore.client::view::processowner;
- **Descrizione:** Classe che permette di gestire l'aggiornamento dei *widget* della componente *view*;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia

`sequenziatore::client::view::iprocessowner::IUpdateView` e aggiorna le componenti della *view* comunicando con le seguenti classi:

- `sequenziatore::client::view::user::MainProcessOwner`;
- `sequenziatore::client::view::user::Login`;
- `sequenziatore::client::view::user::SetProcess`;
- `sequenziatore::client::view::user::AddStep`;
- `sequenziatore::client::view::user::PreviewProcess`;
- `sequenziatore::client::view::user::OpenProcess`;
- `sequenziatore::client::view::user::ManagmentSelectedProcess`;
- `sequenziatore::client::view::user::CheckStep`;
- `sequenziatore::client::view::user::UserReceivedData`;
- `sequenziatore::client::view::user::Statistics`;
- `sequenziatore::client::view::user::InviteUser`.

3.2.2.3 Login

- **Nome:** Login;
- **Package:** sequenziatore.client::view::processowner;
- **Descrizione:** Classe che permette di gestire l'interfaccia grafica relativa alle richieste di autenticazione e chiusura della sessione da parte dell'utente *process owner*;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia

`sequenziatore::client::view::iprocessowner::ILogin` e comunica con *ilpresenter* utilizzando metodi della classe

`sequenziatore::client::presenter::processowner::logic::MainLogic`.

3.2.2.4 SetProcess

- **Nome:** SetProcess;
- **Package:** sequenziatore.client::view::processowner;
- **Descrizione:** Classe che permette di gestire l'interfaccia grafica che consente di creare nuovi processi;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia `sequenziatore::client::view::iprocessowner::ISetProcess` e comunica con *ilpresenter* utilizzando metodi della classe `sequenziatore::client::presenter::processowner::logic::MainLogic`.

3.2.2.5 AddStep

- **Nome:** AddStep;
- **Package:** sequenziatore.client::view::processowner;
- **Descrizione:** Classe che permette di gestire l'interfaccia grafica che consente di definire un nuovo passo del processo in creazione;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia `sequenziatore::client::view::iprocessowner::IAddStep` e comunica con *ilpresenter* utilizzando metodi della classe `sequenziatore::client::presenter::processowner::logic::MainLogic`.

3.2.2.6 PreviewProcess

- **Nome:** PreviewProcess;
- **Package:** sequenziatore.client::view::processowner;
- **Descrizione:** Classe che permette realizzare *iwidget* che consentono di visualizzare l'anteprima del processo in creazione;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia `sequenziatore::client::view::iprocessowner::IPreviewProcess` e comunica con *ilpresenter* utilizzando metodi della classe `sequenziatore::client::presenter::processowner::logic::MainLogic`.

3.2.2.7 OpenProcess

- **Nome:** OpenProcess;
- **Package:** sequenziatore.client::view::processowner;
- **Descrizione:** Classe che permette di realizzare *iwidget* che consentono di aprire un processo tramite ricerca o selezionandolo da una lista;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia `sequenziatore::client::view::iprocessowner::IOpenProcess` e comunica con *ilpresenter* utilizzando metodi della classe `sequenziatore::client::presenter::processowner::logic::MainLogic`.

3.2.2.8 ManagmentSelectedProcess

- **Nome:** ManagmentSelectedProcess;
- **Package:** sequenziatore.client::view::processowner;
- **Descrizione:** Classe che permette di realizzare *iwidget* che consentono di gestire i processi selezionati;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia `sequenziatore::client::view::iprocessowner::IManagmentSelectedProcess` e comunica con *ilpresenter* utilizzando metodi della classe `sequenziatore::client::presenter::processowner::logic::MainLogic`.

3.2.2.9 CheckStep

- **Nome:** CheckStep;
- **Package:** sequenziatore.client::view::processowner;
- **Descrizione:** Classe che permette di realizzare *iwidget* che consentono di gestire l'approvazione dei passi che richiedono intervento umano;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia `sequenziatore::client::view::iprocessowner::ICheckStep` e comunica con *ilpresenter* utilizzando metodi della classe `sequenziatore::client::presenter::processowner::logic::MainLogic`.

3.2.2.10 UserReceivedData

- **Nome:** UserReceivedData;
- **Package:** sequenziatore.client::view::processowner;
- **Descrizione:** Classe che permette di realizzare *iwidget* che consentono di gestire l'accesso ai dati inviati al *server_G* dagli utenti;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia

`sequenziatore::client::view::iprocessowner::IUserReceivedData` e

comunica con il *presenter* utilizzando metodi della classe

`sequenziatore::client::presenter::processowner::logic::MainLogic`.

3.2.2.11 Statistics

- **Nome:** Statistics;
- **Package:** sequenziatore.client::view::processowner;
- **Descrizione:** Classe che permette di realizzare *iwidget* che consentono di gestire l'accesso alle informazioni statistiche sui processi;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia

`sequenziatore::client::view::iprocessowner::IStatistics` e comunica con il *presenter* utilizzando metodi della classe

`sequenziatore::client::presenter::processowner::logic::MainLogic`.

3.2.2.12 InviteUser

- **Nome:** InviteUser;
- **Package:** sequenziatore.client::view::processowner;
- **Descrizione:** Classe che permette di realizzare *iwidget* che consentono di gestire i permessi di iscrizione ad un processo da parte degli utenti;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia

`sequenziatore::client::view::iprocessowner::IInviteUser` e comunica con il *presenter* utilizzando metodi della classe

`sequenziatore::client::presenter::processowner::logic::MainLogic`.

3.3 Package sequenziatore::client::ipresenter

3.3.1 Package sequenziatore::client::ipresenter::iuser::ilogic

3.3.1.1 IMainLogic

- **Nome:** IMainLogic;
- **Package:** sequenziatore::client::presenter::iuser::ilogic;
- **Descrizione:** Interfaccia che permette di gestire gli eventi generati dalla componente *View*.

3.3.1.2 ILoginLogic

- **Nome:** ILoginLogic;
- **Package:** sequenziatore::client::presenter::iuser::ilogic;
- **Descrizione:** Interfaccia che ha il compito di gestire le richieste di autenticazione e chiusura della sessione da parte dell'utente.

3.3.1.3 IRegisterLogic

- **Nome:** IRegisterLogic;
- **Package:** sequenziatore::client::presenter::iuser::ilogic;
- **Descrizione:** Interfaccia che ha il compito di gestire le richieste di registrazione da parte dell'utente.

3.3.1.4 IManagmentDataLogic

- **Nome:** IManagmentDataLogic;
- **Package:** sequenziatore::client::presenter::iuser::ilogic;
- **Descrizione:** Interfaccia che ha il compito di gestire la visualizzazione e la modifica dei dati dell'utente.

3.3.1.5 IManagmentProcessLogic

- **Nome:** IManagmentProcessLogic;
- **Package:** sequenziatore::client::presenter::iuser::ilogic;
- **Descrizione:** Interfaccia che ha il compito di gestire e accedere alle informazioni relative allo stato dei processi.

3.3.1.6 ISendDataLogic

- **Nome:** ISendDataLogic;
- **Package:** sequenziatore::client::presenter::iuser::ilogic;
- **Descrizione:** Interfaccia che ha il compito di gestire l'inserimento e l'invio di dati da parte degli utenti.

3.3.1.7 IReportLogic

- **Nome:** IReportLogic;
- **Package:** sequenziatore::client::presenter::iuser::ilogic;
- **Descrizione:** Interfaccia che ha il compito di gestire la creazione di report sull'andamento dei processi in esecuzione.

3.3.2 Package sequenziatore::client::ipresenter::iprocessowner::ilogic

3.3.2.1 IMainLogic

- **Nome:** IMainLogic;
- **Package:** sequenziatore::client::presenter::iprocessowner::ilogic;
- **Descrizione:** Interfaccia che permette di gestire gli eventi generati dalla componente *View*;

3.3.2.2 ILoginLogic

- **Nome:** ILoginLogic;
- **Package:** sequenziatore::client::presenter::iprocessowner::ilogic;
- **Descrizione:** Interfaccia che ha il compito di gestire le richieste di autenticazione e chiusura della sessione da parte dell'utente *process owner*.

3.3.2.3 INewProcessLogic

- **Nome:** INewProcessLogic;
- **Package:** sequenziatore::client::presenter::iprocessowner::ilogic;
- **Descrizione:** Interfaccia che ha il compito di gestire la logica della definizione di un nuovo processo.

3.3.2.4 IAddStepLogic

- **Nome:** IAddStepLogic;
- **Package:** sequenziatore::client::presenter::iprocessowner::ilogic;
- **Descrizione:** Interfaccia che ha il compito di definire la logica di gestione dei passi di un processo.

3.3.2.5 IManagmentProcessLogic

- **Nome:** IManagmentProcessLogic;
- **Package:** sequenziatore::client::presenter::iprocessowner::ilogic;
- **Descrizione:** Interfaccia che ha il compito di gestire e accedere alle informazioni relative allo stato dei processi.

3.3.2.6 ICheckStepLogic

- **Nome:** ICheckStepLogic;
- **Package:** sequenziatore::client::presenter::iprocessowner::ilogic;
- **Descrizione:** Interfaccia che ha il compito di definire la logica del controllo di un passo che richiede intervento umano per essere approvato.

3.3.2.7 IStatisticLogic

- **Nome:** IStatisticLogic;
- **Package:** sequenziatore::client::presenter::iprocessowner::ilogic;
- **Descrizione:** Interfaccia che ha il compito di gestire l'accesso alle informazioni statistiche sui processi.

3.3.2.8 IUserDataLogic

- **Nome:** IUserDataLogic;
- **Package:** sequenziatore::client::presenter::iprocessowner::ilogic;
- **Descrizione:** Interfaccia che ha il compito gestire l'accesso alle informazioni sui passi superati dagli utenti.

3.3.2.9 IInviteUserLogic

- **Nome:** IInviteUserLogic;
- **Package:** sequenziatore::client::presenter::iprocessowner::ilogic;
- **Descrizione:** Interfaccia che ha il compito di gestire i permessi di iscrizione ad un processo degli utenti.

3.3.3 sequenziatore::client::ipresenter::iservercommunication

3.3.3.1 IHttpCommunication

- **Nome:** IHttpCommunication;
- **Package:** sequenziatore::client::presenter::servercommunication;
- **Descrizione:** Interfaccia che permette di gestire l'invio e ricezione di dati al *server_G* tramite protocollo HTTP_G.

3.3.3.2 IWebSocketCommunication

- **Nome:** IWebSocketCommunication;
- **Package:** sequenziatore::client::presenter::servercommunication;
- **Descrizione:** Interfaccia che permette di gestire l'invio e ricezione di dati dal *server_G* tramite *websocket_G*.

3.3.3.3 IDataFormatter

- **Nome:** IDataFormatter;
- **Package:** sequenziatore::client::presenter::servercommunication;
- **Descrizione:** Interfaccia che permette di gestire la formattazione dei dati da inviare e ricevuti dal *server_G*.

3.4 Package sequenziatore::client::presenter

3.4.1 Package sequenziatore::client::presenter::user::logic

3.4.1.1 MainLogic

- **Nome:** MainLogic;
- **Package:** sequenziatore::client::presenter::user::logic;

- **Descrizione:** Classe che permette di gestire gli eventi generati dalla componente *View*;

- **Relazioni con altri componenti:**

La classe implementa l'interfaccia

`sequenziatore::client::presenter::iuser::ilogic::IMainLogic` e delega la gestione della logica di dettaglio alle seguenti classi:

- `sequenziatore::client::presenter::user::logic::LoginLogic`;
- `sequenziatore::client::presenter::user::logic::RegisterLogic`;
- `sequenziatore::client::presenter::user::logic::ManagmentDataLogic`;
- `sequenziatore::client::presenter::user::logic::ManagmentProcessLogic`;
- `sequenziatore::client::presenter::user::logic::SendDataLogic`;
- `sequenziatore::client::presenter::user::logic::ReportLogic`;

3.4.1.2 LoginLogic

- **Nome:** `LoginLogic`;
- **Package:** `sequenziatore::client::presenter::user::logic`;
- **Descrizione:** Classe che ha il compito di gestire le richieste di autenticazione e chiusura della sessione da parte dell'utente;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia

`sequenziatore::client::presenter::iuser::ilogic::ILoginLogic`, e utilizza metodi delle classi `sequenziatore::client::presenter::servercommunication::HttpCommunication` e `sequenziatore::client::view::user::UpdateView`.

3.4.1.3 RegisterLogic

- **Nome:** `RegisterLogic`;
- **Package:** `sequenziatore::client::presenter::user::logic`;
- **Descrizione:** Classe che ha il compito di gestire le richieste di registrazione da parte dell'utente;

- **Relazioni con altri componenti:**

La classe implementa l'interfaccia

`sequenziatore::client::presenter::iuser::ilogic::IRegisterLogic`, e utilizza metodi delle classi `sequenziatore::client::presenter::servercommunication::HttpCommunication` e `sequenziatore::client::view::user::UpdateView`.

3.4.1.4 ManagmentDataLogic

- **Nome:** `ManagmentDataLogic`;

- **Package:** `sequenziatore::client::presenter::user::logic`;

- **Descrizione:** Classe che ha il compito di gestire la visualizzazione e la modifica dei dati dell'utente;

- **Relazioni con altri componenti:**

La classe implementa l'interfaccia `sequenziatore::client::presenter::iuser::ilogic::IManagmentDataLogic`, e utilizza metodi delle classi

`sequenziatore::client::presenter::servercommunication::HttpCommunication` e `sequenziatore::client::view::user::UpdateView`.

3.4.1.5 ManagmentProcessLogic

- **Nome:** `ManagmentProcessLogic`;

- **Package:** `sequenziatore::client::presenter::user::logic`;

- **Descrizione:** Classe che ha il compito di gestire e accedere alle informazioni relative allo stato dei processi. Le operazioni di gestione dello stato comprendono l'eliminazione dei processi terminati;

- **Relazioni con altri componenti:**

La classe implementa l'interfaccia `sequenziatore::client::presenter::iuser::ilogic::IManagmentProcessLogic`, e utilizza metodi delle classi

`sequenziatore::client::presenter::servercommunication::HttpCommunication`,
`sequenziatore::client::presenter::servercommunication::WebsocketCommunication`,
`sequenziatore::client::model::localdata_user::UserData` e `sequenziatore::client::view::user::UpdateView`.

3.4.1.6 SendDataLogic

- **Nome:** SendDataLogic;
- **Package:** sequenziatore::client::presenter::user::logic;
- **Descrizione:** Classe che ha il compito di gestire l'inserimento e l'invio di dati da parte degli utenti;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia

sequenziatore::client::presenter::iuser::ilogic::ISendDataLogic, e utilizza metodi delle classi sequenziatore::client::presenter::servercommunication::HttpCommunication, sequenziatore::client::presenter::servercommunication::WebsocketCommunication, sequenziatore::client::model::UserData e sequenziatore::client::view::user::UpdateView.

3.4.1.7 ReportLogic

- **Nome:** ReportLogic;
- **Package:** sequenziatore::client::presenter::user::logic;
- **Descrizione:** Classe che ha il compito di gestire la creazione di report sull'andamento dei processi in esecuzione;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia

sequenziatore::client::presenter::iuser::ilogic::IManagmentLogic, e utilizza metodi delle classi sequenziatore::client::presenter::servercommunication::HttpCommunication, sequenziatore::client::presenter::servercommunication::WebsocketCommunication, sequenziatore::client::model::localdata_user::UserData e sequenziatore::client::view::user::UpdateView.

3.4.2 Package sequenziatore::client::presenter::processowner::logic

3.4.2.1 MainLogic

- **Nome:** MainLogic;
- **Package:** sequenziatore::client::presenter::processowner::logic;

- **Descrizione:** Classe che permette di gestire gli eventi generati dalla componente *View*;

- **Relazioni con altri componenti:**

La classe implementa l'interfaccia

`sequenziatore::client::presenter::iprocessowner::ilogic.IMainLogic`
e delega la gestione della logica di dettaglio alle seguenti classi:

- `sequenziatore::client::presenter::processowner::logic::LoginLogic`;
- `sequenziatore::client::presenter::processowner::logic::NewProcessLogic`;
- `sequenziatore::client::presenter::processowner::logic::AddStepLogic`;
- `sequenziatore::client::presenter::processowner::logic::ManagementProcessLogic`;
- `sequenziatore::client::presenter::processowner::logic::CheckStepLogic`;
- `sequenziatore::client::presenter::processowner::logic::StatisticsLogic`;
- `sequenziatore::client::presenter::processowner::logic::UserDataLogic`;
- `sequenziatore::client::presenter::processowner::logic::InviteUserLogic`.

3.4.2.2 LoginLogic

- **Nome:** `LoginLogic`;
- **Package:** `sequenziatore::client::presenter::processowner::logic`;
- **Descrizione:** Classe che ha il compito di gestire le richieste di autenticazione e chiusura della sessione da parte dell'utente *process owner*;

- **Relazioni con altri componenti:**

La classe implementa l'interfaccia `sequenziatore::client::presenter::iprocessowner::ilogic::ILoginLogic`, e utilizza metodi delle classi `sequenziatore::client::presenter::servercommunication::HttpCommunication` e `sequenziatore.client::view::processowner::UpdateView`.

3.4.2.3 NewProcessLogic

- **Nome:** NewProcessLogic;
- **Package:** sequenziatore::client::presenter::processowner::logic;
- **Descrizione:** Classe che ha il compito di gestire la logica della definizione di un nuovo processo, comunicando con il *server_G* quando richiesto;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia `sequenziatore::client::presenter::i-processowner::ilogic::INewProcessLogic`, e utilizza metodi delle classi `sequenziatore::client::presenter::servercommunication::HttpCommunication`, `sequenziatore::client::presenter::servercommunication::WebsocketCommunication`, `sequenziatore::client::model::localdata_process_owner::ProcessOwnerData` e `sequenziatore.client::view::processowner::UpdateView`.

3.4.2.4 AddStepLogic

- **Nome:** AddStepLogic;
- **Package:** sequenziatore::client::presenter::processowner::logic;
- **Descrizione:** Classe che ha il compito di definire la logica di gestione dei passi di un processo;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia `sequenziatore::client::presenter::i-processowner::ilogic::IAddStepLogic`, e utilizza metodi delle classi `sequenziatore::client::model::ProcessOwnerData` e `sequenziatore.client::view::processowner::UpdateView`.

3.4.2.5 ManagmentProcessLogic

- **Nome:** ManagmentProcessLogic;
- **Package:** sequenziatore::client::presenter::processowner::logic;
- **Descrizione:** Classe che ha il compito di gestire e accedere alle informazioni relative allo stato dei processi. Le operazioni di gestione dello stato comprendono la terminazione e l'eliminazione di un processo;

- **Relazioni con altri componenti:**

La classe implementa l'interfaccia `sequenziatore::client::presenter::i-processowner::ilogic::IManagmentLogic`, e utilizza metodi delle classi `sequenziatore::client::presenter::servercommunication::HttpCommunication`, `sequenziatore::client::presenter::servercommunication::WebsocketCommunication`, `sequenziatore::client::model::localdata_process_owner::ProcessOwnerData` e `sequenziatore.client::view::processowner::UpdateView`.

3.4.2.6 CheckStepLogic

- **Nome:** `CheckStepLogic`;
- **Package:** `sequenziatore::client::presenter::processowner::logic`;
- **Descrizione:** Classe che ha il compito di definire la logica del controllo di un passo che richiede intervento umano per essere approvato;

- **Relazioni con altri componenti:**

La classe implementa l'interfaccia `sequenziatore::client::presenter::i-processowner::ilogic::ICheckStepLogic`, e utilizza metodi delle classi `sequenziatore::client::presenter::servercommunication::HttpCommunication`, `sequenziatore::client::presenter::servercommunication::WebsocketCommunication`, `sequenziatore::client::model::localdata_process_owner::ProcessOwnerData` e `sequenziatore.client::view::processowner::UpdateView`.

3.4.2.7 StatisticLogic

- **Nome:** `StatisticLogic`;
- **Package:** `sequenziatore::client::presenter::processowner::logic`;
- **Descrizione:** Classe che ha il compito di gestire l'accesso alle informazioni statistiche sui processi, come il numero di utenti partecipanti e il numero di completamenti;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia `sequenziatore::client::presenter::i-processowner::ilogic::IStatisticLogic`, e utilizza metodi delle classi `sequenziatore::client::presenter::servercommunication::HttpCommunication`, `sequenziatore::client::presenter::servercommunication::WebsocketCommunication`, `sequenziatore::client::model::localdata_process_owner::ProcessOwnerData` e `sequenziatore.client::view::processowner::UpdateView`.

3.4.2.8 UserDataLogic

- **Nome:** `UserDataLogic`;
- **Package:** `sequenziatore::client::presenter::processowner::logic`;
- **Descrizione:** Classe che ha il compito gestire l'accesso alle informazioni sui passi superati dagli utenti;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia `sequenziatore::client::presenter::i-processowner::ilogic::IUserDataLogic`, e utilizza metodi delle classi `sequenziatore::client::presenter::servercommunication::HttpCommunication`, `sequenziatore::client::presenter::servercommunication::WebsocketCommunication`, `sequenziatore::client::model::localdata_process_owner::ProcessOwnerData` e `sequenziatore.client::view::processowner::UpdateView`.

3.4.2.9 InviteUserLogic

- **Nome:** `InviteUserLogic`;
- **Package:** `sequenziatore::client::presenter::processowner::logic`;
- **Descrizione:** Classe che ha il compito di gestire i permessi di iscrizione ad un processo degli utenti;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia `sequenziatore::client::presenter::i-processowner::ilogic::IInviteUserLogic`, e utilizza metodi delle classi

```
sequenziatore::client::presenter::servercommunication::HttpCommuni-
cation,
sequenziatore::client::presenter::servercommunication::Websocket-
Communication e
sequenziatore.client::view::processowner::UpdateView.
```

3.4.3 sequenziatore::client::presenter::servercommunication

3.4.3.1 HttpCommunication

- **Nome:** HttpCommunication;
- **Package:** sequenziatore::client::presenter::servercommunication;
- **Descrizione:** Classe che permette di gestire l'invio e ricezione di dati al *server_G* tramite protocollo HTTP_G;

- **Relazioni con altri componenti:**

La classe implementa l'interfaccia `sequenziatore::client::presenter::i-servercommunication::IHttpCommunication` e formatta i dati da inviare e ricevuti tramite la classe `sequenziatore::client::presenter::servercommunication::JSONFormatter`.

3.4.3.2 WebsocketCommunication

- **Nome:** WebsocketCommunication;
- **Package:** sequenziatore::client::presenter::servercommunication;
- **Descrizione:** Classe che permette di gestire l'invio e ricezione di dati dal *server_G* tramite *websocket_G*;

- **Relazioni con altri componenti:**

La classe implementa l'interfaccia `sequenziatore::client::presenter::i-servercommunication::IWebsocketCommunication` e formatta i dati da inviare e ricevuti tramite la classe `sequenziatore::client::presenter::servercommunication::JSONFormatter`.

3.4.3.3 JSONFormatter

- **Nome:** JSONFormatter;
- **Package:** sequenziatore::client::presenter::servercommunication;

- **Descrizione:** Classe che permette di gestire la formattazione dei dati da inviare e ricevuti dal *server_G*;

- **Relazioni con altri componenti:**

La classe implementa l'interfaccia `sequenziatore::client::presenter::i-servercommunication::IDataFormatter`.

3.5 Package sequenziatore::client::imodel

3.5.1 Package sequenziatore::client::imodel::iprocessowner

3.5.1.1 IProcessOwnerData

- **Nome:** IProcessOwnerData;
- **Package:** sequenziatore::client::model::ilocaldata_process_owner;
- **Descrizione:** Interfaccia che permette di gestire i dati di un processo e il salvataggio in locale di tali dati.

3.5.2 Package sequenziatore::client::imodel::iuser

3.5.2.1 IUserData

- **Nome:** IUserData;
- **Package:** sequenziatore::client::model::ilocaldata_user;
- **Descrizione:** Interfaccia che permette di gestire i dati di un processo e il salvataggio in locale di tali dati.

3.6 Package sequenziatore::client::model

3.6.1 Package sequenziatore::client::model

3.6.1.1 IProcess

- **Nome:** IProcess;
- **Package:** sequenziatore::client::model;
- **Descrizione:** Interfaccia che permette di gestire i dati di un processo.

3.6.1.2 IStep

- **Nome:** IStep;
- **Package:** sequenziatore::client::model;
- **Descrizione:** Interfaccia che permette di gestire i dati di un passo di un processo.

3.6.1.3 Process

- **Nome:** Process;
- **Package:** sequenziatore::client::model;
- **Descrizione:** Classe che permette di gestire i dati di un processo;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia `sequenziatore::client::model::IProcess` e utilizza oggetti di tipo `sequenziatore::client::model::Step`.

3.6.1.4 Step

- **Nome:** Step;
- **Package:** sequenziatore::client::model;
- **Descrizione:** Classe che permette di gestire i dati di un passo;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia `sequenziatore::client::model::IStep`.

3.6.2 Package sequenziatore::client::model:processowner

3.6.2.1 ProcessOwnerData

- **Nome:** ProcessOwnerData;
- **Package:** sequenziatore::client::model::localdata_process_owner;
- **Descrizione:** Classe che permette di gestire i dati di un processo e il salvataggio in locale di tali dati;
- **Relazioni con altri componenti:**

La classe implementa l'interfaccia `sequenziatore::client::model::ilocaldata_process_owner::IProcessOwnerData` e utilizza oggetti di tipo `sequenziatore::client::model::Process`.

3.6.3 Package sequenziatore::client::model:user

3.6.3.1 UserData

- **Nome:** UserData;
- **Package:** sequenziatore::client::model::localdata_user;

- **Descrizione:** Classe che permette di gestire un processo e il salvataggio in locale di tali dati;

- **Relazioni con altri componenti:**

La classe implementa l'interfaccia

`sequenziatore::client::model::ilocaldata_user::IUserData` e utilizza oggetti di tipo `sequenziatore::client::model::Process`.

3.7 Package sequenziatore::server::presenter

3.7.0.2 PresenterFacade

- **Nome:** PresenterFacade
- **Tipo:** Abstract;
- **Package:** sequenziatore::server::presenter
- **Descrizione:** Classe astratta che ha il ruolo di decidere a che pacchetto inoltrare le richieste;
- **Relazione con altre componenti:** la classe richiama metodi delle classi:
 - sequenziatore::server::presenter::iprocessowner::ProcessOwnerFacade;
 - sequenziatore::server::presenter::iprocessowner::UserFacade.

3.7.1 Package sequenziatore::server::presenter::icommunication

3.7.1.1 ICommunication

- **Nome:** ICommunication;
- **Tipo:** Interface;
- **Package:** sequenziatore::server::presenter::icommunication
- **Descrizione:** interfaccia che gestisce le comunicazioni con il *presenter* lato *client*;

3.7.1.2 IDataFormatter

- **Nome:** IDataFormatter;
- **Tipo:** Interface;
- **Package:** sequenziatore::server::presenter::icommunication
- **Descrizione:** interfaccia che gestisce le comunicazioni con il presenter lato *client* tramite richieste http;

3.7.2 Package sequenziatore::server::presenter::communication

3.7.2.1 HttpCommunication

- **Nome:** HttpCommunication;
- **Tipo:** Class;

- **Package:** sequenziatore::server::presenter::communication
- **Descrizione:** classe responsabile della gestione delle comunicazioni con il presenter lato *client* tramite richieste http;
- **Relazione con altre componenti:** la classe implementa l' interfaccia `:sequenziatore::server::presenter::communication::ICommunication` e richiama metodi delle classi:
 - sequenziatore::server::presenter::PresenterFacade.

3.7.2.2 WebSocketCommunication

- **Nome:** WebSocketCommunication;
- **Tipo:** Class;
- **Package:** sequenziatore::server::presenter::communication
- **Descrizione:** classe responsabile della gestione delle comunicazioni con il presenter lato *client* tramite WebSocket;
- **Relazione con altre componenti:** la classe implementa l' interfaccia `:sequenziatore::server::presenter::communication::ICommunication` e richiama metodi delle classi:
 - sequenziatore::server::presenter::PresenterFacade.

3.7.3 Package sequenziatore::server::presenter::iuser

3.7.3.1 ProcessOwnerFacade

- **Nome:** ProcessOwnerFacade;
- **Tipo:** abstract;
- **Package:** sequenziatore::server::presenter::iuser
- **Descrizione:** classe che decide in base alla richiesta ricevuta a che classe assegnarne l'elaborazione di tale richiesta all'interno del package processowner
- **Relazione con altre componenti:** la classe richiama metodi delle classi:
 - sequenziatore::server::presenter::user::AccountManager tramite l'interfaccia sequenziatore::server::presenter::user::IAccountManager;
 - sequenziatore::server::presenter::user::UserProcessManager tramite l'interfaccia sequenziatore::server::presenter::user::IUserProcessManager;
 - sequenziatore::server::presenter::user::UserStepManager tramite l'interfaccia sequenziatore::server::presenter::user::IUserStepManager;
 - sequenziatore::server::presenter::user::Report tramite l'interfaccia sequenziatore::server::presenter::user::IReport;

3.7.3.2 IAccountManager

- **Nome:** IAccountManager;
- **Package:** sequenziatore::server::presenter::iuser
- **Descrizione:** Interfaccia che permette la gestione del proprio account all'utente.

3.7.3.3 IUserProcessManager

- **Nome:** IUserProcessManager;
- **Package:** sequenziatore::server::presenter::iuser
- **Descrizione:** Interfaccia che permette la gestione dei processi di un utente;

3.7.3.4 IUserStepManager

- **Nome:** IUserStepManager;
- **Package:** sequenziatore::server::presenter::iuser
- **Descrizione:** Interfaccia che gestisce l'esecuzione di un passo da parte di un utente.

3.7.3.5 IReport

- **Nome:** IReport;
- **Package:** sequenziatore::server::presenter::iuser
- **Descrizione:** Interfaccia che permette la creazione del report per l'utente.

3.7.4 Package sequenziatore::server::presenter::user

3.7.4.1 AccountManager

- **Nome:** AccountManager;
- **Package:** sequenziatore::server::presenter::user
- **Descrizione:** classe che permette la modifica dei dati e il controllo del *log in* di un utente;
- **Relazione con altre componenti:** la classe implementa l'interfaccia sequenziatore::server::presenter::iuser::IAccountManager e richiama i metodi delle classi:
 - sequenziatore::server::model::DaoFacade;

3.7.4.2 UserProcessManager

- **Nome:** UserProcessManager;
- **Package:** sequenziatore::server::presenter::user
- **Descrizione:** classe che permette l'inoltro della richiesta di un utente a iscriversi o disiscriversi a un processo;
- **Relazione con altre componenti:** la classe implementa l'interfaccia sequenziatore::server::presenter::iuser::IUserProcessManager e richiama i metodi delle classi:
 - sequenziatore::server::model::DaoFacade;

3.7.4.3 UserStepManager

- **Nome:** UserStepManager;
- **Package:** sequenziatore::server::presenter::user
- **Descrizione:** Gestisce l'esecuzione di un passo da parte di un utente inoltrando la richiesta di inserire i dati nel *database* e in caso sia richiesto notifica l'amministratore che deve controllare se il passo è stato completato;

- **Relazione con altre componenti:** la classe implementa l' interfaccia sequenziatore::server::presenter::iuser::IUserStepManager e richiama i metodi delle classi:

- sequenziatore::server::model::DaoFacade;

3.7.4.4 Report

- **Nome:** Report;
- **Package:** sequenziatore::server::presenter::user
- **Descrizione:** Classe che genera il report dell' utente riferito al processo richiesto;
- **Relazione con altre componenti:** la classe implementa l' interfaccia sequenziatore::server::presenter::iuser::IReport e richiama i metodi delle classi:

- sequenziatore::server::model::DaoFacade;

3.7.5 Package sequenziatore::server::presenter::iprocessowner

3.7.5.1 ProcessOwnerFacade

- **Nome:** ProcessOwnerFacade;
- **Tipo:** abstract;
- **Package:** sequenziatore::server::presenter::iprocessowner
- **Descrizione:** classe che decide in base alla richiesta ricevuta a che classe assegnarne l'elaborazione di tale richiesta all'interno del package processowner
- **Relazione con altre componenti:** la classe richiama metodi delle classi:
 - sequenziatore::server::presenter::processowner::Login tramite l'interfaccia sequenziatore::server::presenter::processowner::ILogin;
 - sequenziatore::server::presenter::processowner::ProcessManager tramite l'interfaccia sequenziatore::server::presenter::processowner::IProcessManager;
 - sequenziatore::server::presenter::processowner::StepManager tramite l'interfaccia sequenziatore::server::presenter::processowner::IStepManager;
 - sequenziatore::server::presenter::processowner::UserManager tramite l'interfaccia sequenziatore::server::presenter::processowner::IUserManager;
 - sequenziatore::server::presenter::processowner::Report tramite l'interfaccia sequenziatore::server::presenter::processowner::IReport;

3.7.5.2 IProcessManager

- **Nome:** IProcessManager;
- **Package:** sequenziatore::server::presenter::iprocessowner
- **Descrizione:** Interfaccia che permette la gestione dei processi al *process owner*;

3.7.5.3 ILogin

- **Nome:** ILogin;
- **Package:** sequenziatore::server::presenter::iprocessowner
- **Descrizione:** Interfaccia che permette la gestione della login del *process owner*;

3.7.5.4 IStepManager

- **Nome:** IStepManager;
- **Package:** sequenziatore::server::presenter::iprocessowner
- **Descrizione:** Interfaccia che permette la gestione dei passi al *process owner*;

3.7.5.5 IUserManager

- **Nome:** IUserManager;
- **Package:** sequenziatore::server::presenter::iprocessowner
- **Descrizione:** Interfaccia per la gestione degli utenti iscritti ai processi;

3.7.5.6 IReport

- **Nome:** IReport;
- **Package:** sequenziatore::server::presenter::iprocessowner
- **Descrizione:** Interfaccia che permette la gestione dei report al *process owner*;

3.7.6 Package sequenziatore::server::presenter::processowner

3.7.6.1 Login

- **Nome:** Login;
- **Package:** sequenziatore::server::presenter::iprocessowner
- **Descrizione:** Classe che permette la gestione della login del *process owner*;
- **Relazione con altre componenti:** la classe implementa l' interfaccia sequenziatore::server::presenter::iprocessowner::ILogin ed invoca i metodi delle classi:
 - sequenziatore::server::model::DaoFacade;

3.7.6.2 ProcessManager

- **Nome:** ProcessManager;
- **Package:** sequenziatore::server::presenter::processowner
- **Descrizione:** Classe che riceve le richieste del *process owner* per la gestione dei processi come creazione, modifica e eliminazione degli stessi;
- **Relazione con altre componenti:** la classe implementa l' interfaccia sequenziatore::server::presenter::iprocessowner::IProcessManager ed invoca i metodi delle classi:
 - sequenziatore::server::model::DaoFacade;

3.7.6.3 StepManager

- **Nome:** StepManager;
- **Package:** sequenziatore::server::presenter::processowner
- **Descrizione:** Classe che permette l'elaborazione delle richieste del *process owner* per quanto concerne la creazione, la rimozione e la modifica di passi;
- **Relazione con altre componenti:** la classe implementa l'interfaccia sequenziatore::server::presenter::iprocessowner::IStepManager ed invoca i metodi delle classi:
 - sequenziatore::server::model::DaoFacade;

3.7.6.4 UserManager

- **Nome:** UserManager;
- **Package:** sequenziatore::server::presenter::iprocessowner
- **Descrizione:** Classe che permette la gestione degli utenti iscritti alla piattaforma, permettendogli di rimuovere utenti da processi,;
- **Relazione con altre componenti:** la classe implementa l'interfaccia sequenziatore::server::presenter::iprocessowner::IUserManager ed invoca i metodi delle classi:
 - sequenziatore::server::model::DaoFacade;

3.7.6.5 Report

- **Nome:** Report;
- **Package:** sequenziatore::server::presenter::iprocessowner
- **Descrizione:** Classe che permette la gestione delle richieste dei report al *process owner*, permettendogli di visualizzare i risultati raggiunti in un processo;
- **Relazione con altre componenti:** la classe implementa l'interfaccia sequenziatore::server::presenter::iprocessowner::IReport ed invoca i metodi delle classi:
 - sequenziatore::server::model::DaoFacade;

3.8 Package sequenziatore::server::model

3.8.0.6 DaoFacade

- **Nome:** DaoFacade;
- **Tipo:** abstract;
- **Package:** sequenziatore::server::model
- **Descrizione:** Classe astratta che decide a che pacchetto assegnare la richiesta di esecuzione *query*;
- **Relazione con altre componenti:** la classe invoca i metodi delle seguenti classi:
 - sequenziatore::server::model::daoprocessowner::ObjectTransfer tramite l' interfaccia sequenziatore::server::model::daoprocessowner::IObjectTransfer
 - sequenziatore::server::model::daoprocessowner::DataAccessObject tramite l' interfaccia sequenziatore::server::model::daoprocessowner::IDataAccessObject
 - sequenziatore::server::model::daostep::ObjectTransfer tramite l' interfaccia sequenziatore::server::model::daostep::IObjectTransfer
 - sequenziatore::server::model::daostep::DataAccessObject tramite l' interfaccia sequenziatore::server::model::daostep::IDataAccessObject
 - sequenziatore::server::model::daouser::ObjectTransfer tramite l' interfaccia sequenziatore::server::model::daouser::IObjectTransfer
 - sequenziatore::server::model::daouser::DataAccessObject tramite l' interfaccia sequenziatore::server::model::daouser::IDataAccessObject
 - sequenziatore::server::model::daoprocess::ObjectTransfer tramite l' interfaccia sequenziatore::server::model::daoprocess::IObjectTransfer
 - sequenziatore::server::model::daoprocess::DataAccessObject tramite l' interfaccia sequenziatore::server::model::daoprocess::IDataAccessObject

3.8.1 Package sequenziatore::server::model::daouser

3.8.1.1 IDataAccessObject

- **Nome:** IDataAccessObject;
- **Package:** sequenziatore::server::model::daouser
- **Descrizione:** Interfaccia che permette la gestione del proprio account all' utente.

3.8.1.2 DataAccessObject

- **Nome:** DataAccessObject;
- **Package:** sequenziatore::server::model::daouser
- **Descrizione:** Interfaccia che permette la gestione del proprio account all' utente.
- **Relazione con altre componenti:** la classe implementa l' interfaccia sequenziatore::server::model::daouser::IDataAccessObject ed invoca metodi delle classi:
 - sequenziatore::server::model::daouser::ObjectTransfer tramite l' interfaccia sequenziatore::server::model::daouser::IObjectTransfer

3.8.1.3 IObjectTransfer

- **Nome:** IObjectTransfer;
- **Package:** sequenziatore::server::model::daouser
- **Descrizione:** Interfaccia che permette la gestione del proprio account all' utente.

3.8.1.4 ObjectTransfer

- **Nome:** ObjectTransfer;
- **Package:** sequenziatore::server::model::daouser
- **Descrizione:** Interfaccia che permette la gestione del proprio account all' utente.
- **Relazione con altre componenti:** la classe implementa l'interfaccia sequenziatore::server::model::daouser::IObjectTransfer.

3.8.2 Package sequenziatore::server::model::daoprocessowner

3.8.2.1 IDataAccessObject

- **Nome:** IDataAccessObject;
- **Package:** sequenziatore::server::model::daoprocessowner
- **Descrizione:** Interfaccia che permette la gestione del proprio account all' utente.

3.8.2.2 DataAccessObject

- **Nome:** DataAccessObject;
- **Package:** sequenziatore::server::model::daoprocessowner
- **Descrizione:** Interfaccia che permette la gestione del proprio account all' utente.
- **Relazione con altre componenti:** la classe implementa l' interfaccia sequenziatore::server::model::daoprocessowner::IDataAccessObject ed invoca metodi delle classi:
 - sequenziatore::server::model::daoprocessowner::ObjectTransfer tramite l' interfaccia sequenziatore::server::model::daoprocessowner::IObjectTransfer

3.8.2.3 IObjectTransfer

- **Nome:** IObjectTransfer;
- **Package:** sequenziatore::server::model::daoprocessowner
- **Descrizione:** Interfaccia che permette la gestione del proprio account all' utente.

3.8.2.4 ObjectTransfer

- **Nome:** ObjectTransfer;
- **Package:** sequenziatore::server::model::daoprocessowner
- **Descrizione:** Interfaccia che permette la gestione del proprio account all' utente.
- **Relazione con altre componenti:** la classe implementa l'interfaccia sequenziatore::server::model::daoprocessowner::IObjectTransfer.

3.8.3 Package sequenziatore::server::model::daoprocess

3.8.3.1 IDataAccessObject

- **Nome:** IDataAccessObject;
- **Package:**sequenziatore::server::model::daoprocess
- **Descrizione:** Interfaccia che permette la gestione del proprio account all' utente.

3.8.3.2 DataAccessObject

- **Nome:** DataAccessObject;
- **Package:** sequenziatore::server::model::daoprocess
- **Descrizione:** Interfaccia che permette la gestione del proprio account all' utente.
- **Relazione con altre componenti:** la classe implementa l' interfaccia sequenziatore::server::model::daoprocess::IDataAccessObject ed invoca metodi delle classi:
 - sequenziatore::server::model::daoprocess::ObjectTransfer tramite l' interfaccia sequenziatore::server::model::daoprocess::IObjectTransfer

3.8.3.3 IObjectTransfer

- **Nome:** IObjectTransfer;
- **Package:** sequenziatore::server::model::daoprocess
- **Descrizione:** Interfaccia che permette la gestione del proprio account all' utente.

3.8.3.4 ObjectTransfer

- **Nome:** ObjectTransfer;
- **Package:** sequenziatore::server::model::daoprocess
- **Descrizione:** Interfaccia che permette la gestione del proprio account all' utente.
- **Relazione con altre componenti:** la classe implementa l'interfaccia sequenziatore::server::model::daoprocess::IObjectTransfer.

3.8.4 Package sequenziatore::server::model::daostep

3.8.4.1 IDataAccessObject

- **Nome:** IDataAccessObject;
- **Package:** sequenziatore::server::model::daostep
- **Descrizione:** Interfaccia che permette la gestione del proprio account all' utente.

3.8.4.2 DataAccessObject

- **Nome:** DataAccessObject;
- **Package:** sequenziatore::server::model::daostep
- **Descrizione:** Interfaccia che permette la gestione del proprio account all' utente.
- **Relazione con altre componenti:** la classe implementa l' interfaccia sequenziatore::server::model::daostep::IDataAccessObject ed invoca metodi delle classi:
 - sequenziatore::server::model::daostep::ObjectTransfer tramite l' interfaccia sequenziatore::server::model::daostep::IObjectTransfer

3.8.4.3 IObjectTransfer

- **Nome:** IObjectTransfer;
- **Package:** sequenziatore::server::model::daostep
- **Descrizione:** Interfaccia che permette la gestione del proprio account all' utente.

3.8.4.4 ObjectTransfer

- **Nome:** ObjectTransfer;
- **Package:** sequenziatore::server::model::daostep
- **Descrizione:** Interfaccia che permette la gestione del proprio account all' utente.
- **Relazione con altre componenti:** la classe implementa l'interfaccia sequenziatore::server::model::daostep::IObjectTransfer.

4 Design pattern

4.1 Design pattern architetturali

4.1.1 Model View Presenter

- **Scopo:** Il *pattern_G* architetturale *Model View Presenter* (MVP) è un derivato del *Model View Controller* (MVC), focalizzato sulla valorizzazione della logica della presentazione. Entrambi i pattern hanno lo scopo di disaccoppiare la logica dell'applicazione dalla rappresentazione grafica.

Il *pattern_G* MVP prevede la suddivisione dell'applicazione in tre componenti:

- **Model:** Definisce il modello dati e le regole di accesso e di modifica;
- **View:** Si occupa della rappresentazione dell'interfaccia utente;
- **Presenter:** Contiene la logica dell'applicazione, si occupa delle comunicazioni tra vista e modello e dell'aggiornamento della vista.

- **Contesto d'uso:**

4.2 Design pattern strutturali

4.2.1 Adapter

- **Scopo:** Il *pattern_G* strutturale *Adapter* permette di utilizzare un componente software la cui interfaccia deve essere adattata per potersi integrare ad un'altra presente nell'applicazione esistente.

Tale *pattern_G* può essere basato sia su classi che su oggetti, perciò, l'istanza della classe da adattare, può derivare tramite ereditarietà o composizione.

- **Contesto d'uso:**

4.2.2 Decorator

- **Scopo:** Il *pattern_G* strutturale *Decorator* permette di aggiungere dinamicamente funzionalità ad un oggetto base, con la possibilità di comporre arbitrariamente.

Tale *pattern_G* si pone come alternativa all'uso dell'ereditarietà singola o multipla;

- **Contesto d'uso:**

4.2.3 Facade

- **Scopo:** Il *pattern_G* strutturale *Facade* prevede l'utilizzo di un'interfaccia unica e semplice per un sottosistema complesso, diminuendo la complessità del sistema;

- **Contesto d'uso:**

4.2.4 Proxy

- **Scopo:** Il *pattern_G* strutturale *Proxy* viene utilizzato per accedere ad un oggetto complesso di cui si vogliono controllare gli accessi, tramite un oggetto semplice, che espone gli stessi metodi dell'oggetto che maschera;
- **Contesto d'uso:**

4.3 Design pattern creazionali

4.3.1 Singleton

- **Scopo:** Il *pattern_G* creazionale *Singleton* viene utilizzato quando si ha la necessità di avere una sola istanza di una classe e di avere un punto di accesso globale ad essa;
- **Contesto d'uso:**

4.3.2 Abstract Factory

- **Scopo:** Il *pattern_G* creazionale *Abstract Factory* fornisce un'interfaccia per creare famiglie di prodotti senza specificare classi concrete. Le classi che concretizzano tale interfaccia, vengono costruite una sola volta, e consentono di utilizzare una varietà di elementi che presentano le stesse funzionalità con diverse implementazioni;
- **Contesto d'uso:**

4.4 Design pattern comportamentali

4.4.1 Command

- **Scopo:** Il *pattern_G* comportamentale *Command* permette di separare l'invocazione di un comando dai suoi dettagli implementativi;
- **Contesto d'uso:**

4.4.2 Iterator

- **Scopo:** Il *pattern_G* comportamentale *Iterator* fornisce l'accesso sequenziale agli elementi di un aggregato senza esporne l'implementazione;
- **Contesto d'uso:**

4.4.3 Observer

- **Scopo:** Il *pattern_G* comportamentale *Observer* viene utilizzato quando si vuole realizzare una dipendenza tra un soggetto e più oggetti, in cui il cambiamento di stato del un soggetto, viene notificato a tutti gli oggetti dipendenti;
- **Contesto d'uso:**

4.4.4 Strategy

- **Scopo:** Il *pattern_G* comportamentale *Strategy* viene utilizzato per definire una famiglia di algoritmi, incapsularli e renderli intercambiabili;
- **Contesto d'uso:**

4.4.5 Template method

- **Scopo:** Il *pattern_G* comportamentale *Template method* viene utilizzato per definire lo scheletro di un algoritmo, lasciando l'implementazione di alcuni passi alle sottoclassi. In particolare consente di specificare l'ordine delle operazioni da effettuare ma di delegare la loro implementazione o parte di essa alle sottoclassi;
- **Contesto d'uso:**

5 Tracciamento

5.1 Tracciamento package - componenti

Package	Componente
sequenziatore::client::view::user	VU1 - MainUser
	VU2 - UpdateView
	VU3 - Login
	VU4 - Register
	VU5 - ViewData
	VU6 - EditData
	VU7 - ChangePassword
	VU8 - OpenProcess
	VU9 - ManagementSelectedProcess
	VU10 - SendData
	VU11 - SendText
	VU12 - SendNumb
	VU13 - SendPosition
	VU14 - SendImage
	VU15 - SendPhoto
	VU16 - EndSelectedProcess
	VU17 - PrintProcess
	VU18 - PreviewProcess
sequenziatore.client::view::process-owner	VA1 - MainProcessOwner
	VA2 - UpdateView
	VA3 - Login
	VA4 - SetProcess
	VA5 - AddStep
	VA6 - PreviewProcess
	VA7 - OpenProcess
	VA8 - ManagementSelectedProcess
	VA9 - CheckStep
	VA10 - UserReceivedData
	VA11 - Statistics
	VA12 - InviteUser
sequenziatore::client::presenter::user::logic	CPU1 - MainLogic

	CPU2 - LoginLogic
	CPU3 - RegisterLogic
	CPU4 - ManagementDataLogic
	CPU5 - ManagementProcessLogic
	CPU6 - SendDataLogic
	CPU7 - ReportLogic
sequenziatore::client::presenter::processowner::logic	CPA1 - MainLogic
	CPA2 - LoginLogic
	CPA3 - NewProcessLogic
	CPA4 - AddStepLogic
	CPA5 - ManagementProcessLogic
	CPA6 - CheckStepLogic
	CPA7 - StatisticLogic
	CPA8 - UserDataLogic
	CPA9 - InviteUserLogic
sequenziatore::client::presenter::servercommunication	CP1 - HttpCommunication
	CP2 - WebSocketCommunication
	CP3 - JSONFormatter
sequenziatore::client::model	CM1 - Process
	CM2 - Step
sequenziatore::client::model::local-data_user	CMU3 - UserData
sequenziatore::client::model::local-data_process_owner	CMA4 - ProcessOwnerData
sequenziatore::server::presenter::communication	SP2 - HttpCommunication
	SP3 - WebSocketCommunication
sequenziatore::server::presenter::user	SPU1 - AccountManager
	SPU2 - UserProcessManager
	SPU3 - UserStepManager
	SPU4 - Report
sequenziatore::server::presenter::admin	SPA1 - Login
	SPA2 - ProcessManager

	SPA3 - StepManager
	SPA4 - UserManager
	SPA5 - report
sequenziatore::server::model::daouser	SMU1 - DataAccessObject
	SMU2 - ObjectTransfer
sequenziatore::server::model::dao-processowner	SMA1 - DataAccessObject
	SMA2 - ObjectTransfer
sequenziatore::server::model::dao-process	SM1 - DataAccessObject
	SM2 - ObjectTransfer
sequenziatore::server::model::dao-step	SM3 - DataAccessObject
	SM4 - ObjectTransfer

Tabella 1: Tabella package/componenti

5.2 Tracciamento componenti - requisiti

Requisito	Componente
FOBA 1	UCA 1
FOBA 1.1	UCA 1.1
	UCA 1.1.1
FOBA 1.2	UCA 1.1
	UCA 1.1.2
FOBA 1.3	UCA 1.2
FOBA 1.3.1	UCA 1.2.1
FOBA 1.3.2	UCA 1.2.2
FOBA 1.4	UCA 1.3
FOBA 1.4.1	UCA 1.3.1
FOBA 1.4.1.1	UCA 1.3.1.1
FOBA 1.4.1.2	UCA 1.3.1.2
FOBA 1.4.1.2.1	UCA 1.3.1.2.1

FOBA 1.4.1.2.2	UCA 1.3.1.2.2
FOBA 1.4.1.2.2.1	UCA 1.3.1.2.2
FOBA 1.4.1.2.2.2	UCA 1.3.1.2.2
FOBA 1.4.1.2.2.3	UCA 1.3.1.2.2
FOBA 1.4.1.3	UCA 1.3.1.3
FOBA 1.4.1.3.1	UCA 1.3.1.3.1
FDEA 1.4.1.3.1.1	UCA 1.3.1.3.1.1
FOBA 1.4.1.3.1.2	UCA 1.3.1.3.1.2
FOBA 1.4.1.3.1.2.1	UCA 1.3.1.3.1.2
FOPA 1.4.1.3.1.2.2	UCA 1.3.1.3.1.2
FOBA 1.4.1.3.1.3	UCA 1.3.1.3.1.3
FDEA 1.4.1.3.1.4	UCA 1.3.1.3.1.4
FOPA 1.4.1.3.1.4.1	UCA 1.3.1.3.1.4
FDEA 1.4.1.3.1.4.2	UCA 1.3.1.3.1.4
FOPA 1.4.1.3.1.4.3	UCA 1.3.1.3.1.4
FOPA 1.4.1.3.1.5	UCA 1.3.1.3.1.5
FOBA 1.4.1.3.2	UCA 1.3.1.3.2
FOBA 1.4.2	UCA 1.3.2
FDEA 1.4.3	UCA 1.3.3
FDEA 1.4.3.1	UCA 1.3.3.1
FDEA 1.4.3.2	UCA 1.3.3.2
FDEA 1.4.3.3	UCA 1.3.3.3
FDEA 1.4.3.3.1	UCA 1.3.3.3.1
FDEA 1.4.3.3.1.1	UCA 1.3.3.3.1.1
FDEA 1.4.3.3.1.2	UCA 1.3.3.3.1.2
FDEA 1.4.3.3.1.3	UCA 1.3.3.3.1.3
FDEA 1.4.3.3.1.4	UCA 1.3.3.3.1.4
FOPA 1.4.3.3.1.5	UCA 1.3.3.3.1.5
FDEA 1.4.3.3.2	UCA 1.3.3.3.2
FDEA 1.4.4	UCA 1.3.4

FOBA 1.5	UCA 1.4
FDEA 2	UCA 2
FDEA 2.1	UCA 2.1
FOPA 2.1.2	UCA 2.1.3
FDEA 2.1.3	UCA 2.1.2
FOPA 2.2	UCA 2.2
FOPA 2.2.1	UCA 2.2.1
FOPA 2.2.2	UCA 2.2.2
FOPA 2.3	UCA 2.3
FOPA 2.3.1	UCA 2.3.1
FOPA 2.3.1.1	UCA 2.3.1
FOPA 2.3.1.2	UCA 2.3.1
FOPA 2.3.1.3	UCA 2.3.1
FOPA 2.3.1.4	UCA 2.3.1
FDEA 2.3.2	UCA 2.3.2
FDEA 2.3.2.1	UCA 2.3.2.1
FDEA 2.3.2.2	UCA 2.3.2.2
FDEA 2.3.3	UCA 2.3.3
FDEA 2.4	UCA 2.4
FOBA 2.4.1	UCA 2.4.1
FDEA 2.4.2	UCA 2.4.2.1
FDEA 2.4.3	UCA 2.4.2.2
FDEA 2.4.4	UCA 2.4.2
FDEA 2.5	UCA 2.5
FDEA 2.6	UCA 2.5
FOBA 3	UCA 3
FOBU 1	UCU 1
FOBU 1.1	UCU 1.1
FOBU 1.1.1	UCU 1.1

FOBU 1.2	UCU 1.2
FOBU 1.2.1	UCU 1.2
FOBU 1.3	UCU 1.3
FOBU 1.4	UCU 1.4
FOBU 1.5	UCU 1.5
FOBU 1.5.1	UCU 1.5
FOBU 1.6	UCU 1.6
FDEU 1.6.1	UCU 1.6
FOBU 2	UCU 2
FOBU 2.1	UCU 2
FOBU 2.2	UCU 2.1
FOBU 2.3	UCU 2.2
FOPL 3	UCL 1
FOPL 3.1	UCL 1.1
FOPL 3.1.1	UCL 1.1.1
FOPL 3.1.2	UCL 1.1.2
FOPL 3.1.3	UCL 1.1.3
FOPL 3.1.4	UCL 1.1.4
FOPL 3.2	UCL 1.2
FOPL 3.2.1	UCL 1.2.1
FOPL 3.2.1.1	UCL 1.2.1.1
FOPL 3.2.1.2	UCL 1.2.1.2
FOPL 3.2.1.3	UCL 1.2.1.2
FOBL 4	UCL 2
FOBL 4.1.1	UCL 2.1.1
FOBL 4.1.1.1	UCL 2.1.1.1
FOBL 4.1.1.2	UCL 2.1.1.2
FDEL 4.1.1.3	UCL 2.1.1.3
FDEL 4.1.1.4	UCL 2.1.1.4
FOBL 4.1.2	UCL 2.1.2

FDEL 4.1.3	UCL 2.1.3
FOBL 4.2	UCL 2.2
FOBL 4.3	UCL 2.3
FOBL 4.4	UCL 2.4
FOBL 4.4.1.1	UCL 2.4.1.1
FOBL 4.4.1.2	UCL 2.4.1.2
FOBL 4.4.2	UCL 2.4.2
FOBL 4.4.2.1	UCL 2.4.2.1
FOBL 4.4.2.2	UCL 2.4.2.2
FOBL 4.4.2.3	UCL 2.4.2.3
FOBL 4.4.2.4	UCL 2.4.2.4
FOBL 4.4.3	UCL 2.4.3
FOBL 4.4.4	UCL 2.4.4
FOBL 4.4.4.1	UCL 2.4.4.1
FOBL 4.4.4.1.1	UCL 2.4.4.1
FOBL 4.4.4.1.2	UCL 2.4.4.1.1
FOBL 4.4.4.2	UCL 2.4.4.2
FOBL 4.4.4.2.1	UCL 2.4.4.2.1
FOBL 4.4.4.2.2	UCL 2.4.4.2.2
FOBL 4.4.4.2.2.1	UCL 2.4.4.2.2
FOPL 4.4.4.2.2.2	UCL 2.4.4.2.2
FOBL 4.4.4.2.3	UCL 2.4.4.2.3
FDEL 4.4.4.2.4	UCL 2.4.4.2.4
FOPL 4.4.4.2.4.1	UCL 2.4.4.2.4
FDEL 4.4.4.2.4.2	UCL 2.4.4.2.4
FOPL 4.4.4.2.4.3	UCL 2.4.4.2.4
FOBL 4.4.4.3	UCL 2.4.4.3
FOBL 4.4.4.3.1	UCL 2.4.4.3.1
FDEL 4.4.4.3.1.1	UCL 2.4.4.3.1.1

FOBL 4.4.4.3.1.2	UCL 2.4.4.3.1.2
FOBL 4.4.4.3.2	UCL 2.4.4.3.2
FOBL 4.4.4.3.3	UCL 2.4.4.3.3
FOBL 4.4.4.4	UCL 2.4.4.4
FOBL 4.4.4.4.1	UCL 2.4.4.4
FOBL 4.4.4.4.2	UCL 2.4.4.4
FOBL 4.4.4.4.4	UCL 2.4.4.4
FOBL 4.4.4.4.5	UCL 2.4.4.4
FOBL 4.4.4.4.6	UCL 2.4.4.4
FOBL 4.4.4.4.6.1	UCL 2.4.4.4
FOBL 4.4.4.4.6.2	UCL 2.4.4.4
FOBL 4.4.4.5	UCL 2.4.4.5
FOBL 4.4.4.6	UCL 2.4.4.6
FOPL 4.4.5.1	UCL 2.4.5.1
FOBL 4.4.5.2	UCL 2.4.5.2
FOPL 4.4.6	UCL 2.4.6
FOBL 4.5	UCL 2.5
FOBL 5	UCL 3

Tabella 2: Tabella componenti/requisiti

5.3 Tracciamento requisiti - componenti

Requisito	Descrizione	Componente
FOBU 1	Il sistema dovrà permettere all'utente di registrarsi	VU4 CPU3 SPU1 SMU1 SMU2
FOBU 1.1	L'utente dovrà inserire un <i>username</i> che lo identifichi univocamente all'interno del sistema	VU4 CPU3

FOBU 1.1.1	L'utente dovrà inserire un <i>username</i> composto da almeno 6 caratteri	VU4 CPU3
FOBU 1.2	L'utente dovrà inserire una <i>password</i> d'accesso	VU4 CPU3
FOBU 1.2.1	L'utente dovrà inserire una <i>password</i> composta almeno da 8 caratteri alfanumerici	VU4 CPU3
FOBU 1.3	L'utente dovrà inserire il proprio nome	VU4 CPU3
FOBU 1.4	L'utente dovrà inserire il proprio cognome	VU4 CPU3
FOBU 1.5	L'utente dovrà inserire la propria data di nascita	VU4 CPU3
FOBU 1.5.1	La data di nascita inserita dall'utente dovrà essere antecedente alla data di iscrizione	VU4 CPU3
FOBU 1.6	L'utente dovrà inserire una sua <i>email</i>	VU4 CPU3
FDEU 1.6.1	La <i>email</i> inserita dovrà corrispondere ad un indirizzo di posta elettronica esistente	VU4 CPU3
FOBU 2	Il sistema dovrà permettere all'utente di autenticarsi	VU3 CPU2 SPU1 SMU1 SMU2
FOBU 2.1	Il sistema dovrà negare l'autenticazione se i dati inseriti dall'utente sono errati o non esistenti all'interno del <i>server_G</i>	VU3 CPU2
FOBU 2.2	L'utente dovrà inserire il proprio <i>username</i> per autenticarsi	VU3 CPU2

FOBU 2.3	L'utente dovrà inserire la propria <i>password</i> per autenticarsi	VU3 CPU2
FOPL 3	Il sistema dovrà permettere all'utente autenticato di gestire le proprie credenziali	VU5 VU6 CPU4 SPU1 SMU1 SMU2
FOPL 3.1	L'utente autenticato potrà visualizzare le proprie credenziali	VU5 CPU4 SPU1 SMU1 SMU2
FOPL 3.1.1	L'utente autenticato visualizzerà il proprio <i>username</i>	VU5 CPU4
FOPL 3.1.2	L'utente autenticato visualizzerà il proprio nome	VU5 CPU4
FOPL 3.1.3	L'utente autenticato visualizzerà il proprio cognome	VU5 CPU4
FOPL 3.1.4	L'utente autenticato visualizzerà la propria data di nascita	VU5 CPU4
FOPL 3.1.5	L'utente autenticato visualizzerà la propria <i>email</i>	VU5 CPU4
FOPL 3.2	Il sistema dovrà permettere all'utente autenticato di modificare i propri dati	VU6 CPU4 SPU1

		SMU1 SMU2
FOPL 3.2.1	Il sistema dovrà permettere all'utente autenticato di modificare la propria <i>password</i>	VU6 VU7 CPU4 SPU1 SMU1 SMU2
FOPL 3.2.1.1	L'utente autenticato potrà inserire la nuova <i>password</i>	VU6 VU7 CPU4
FOPL 3.2.1.2	L'utente autenticato potrà inserire la <i>password</i> corrente	VU6 VU7 CPU4
FOPL 3.2.1.3	Il sistema dovrà comunicare all'utente autenticato se la <i>password</i> inserita non è corretta	VU6 VU7 CPU4
FOBL 4	Il sistema dovrà permettere all'utente autenticato di gestire i processi disponibili	VU8 VU9 VU10 VU11 VU12 VU13 VU14 VU15 VU16 VU17 VU18 CPU5 CPU6 CPU7 SPU2

		SPU3
		SPU4
		SM1
		SM2
		SM3
		SM4
FOBL 4.1	Il sistema dovrà permettere all'utente autenticato di scegliere un processo da una lista selezionata o da i risultati di una ricerca	VU8
		CPU5
		SPU2
		SM1
		SM2
FOBL 4.1.1	Il sistema dovrà permettere all'utente autenticato di selezionare ed aprire una lista di processi	VU8
		CPU5
		SPU2
		SM1
		SM2
FOBL 4.1.1.1	Il sistema dovrà permettere all'utente autenticato di selezionare ed aprire la lista dei processi in esecuzione	VU8
		CPU5
		SPU2
		SM1
		SM2
FOBL 4.1.1.2	Il sistema dovrà permettere all'utente autenticato di selezionare ed aprire la lista dei processi disponibili	VU8
		CPU5
		SPU2
		SM1
		SM2
FDEL 4.1.1.3	L'utente autenticato riceverà da parte del sistema la segnalazione di processi terminabili	VU8
		CPU5
		SPU2

		SM1
		SM2
FDEL 4.1.1.4	L'utente autenticato riceverà da parte del sistema la segnalazione dei nuovi processi disponibili	VU8
		CPU5
		SPU2
		SM1
		SM2
FOBL 4.1.2	L'utente autenticato potrà selezionare un processo dalla lista di processi aperta	VU8
		CPU5
		SPU2
		SM1
		SM2
FDEL 4.1.3	Il sistema dovrà permettere all'utente autenticato di ricercare dei processi fra tutti quelli a cui può partecipare	VU8
		CPU5
		SPU2
		SM1
		SM2
FOBL 4.2	L'utente autenticato potrà visualizzare la descrizione di un processo selezionato	VU9
		CPU5
		SPU2
		SM1
		SM2
FOBL 4.3	Il sistema dovrà permettere all'utente autenticato di iscriversi a un processo precedentemente selezionato	VU9
		CPU5
		SPU2
		SM1
		SM2

FOBL 4.4	Il sistema dovrà permettere all'utente autenticato di eseguire il processo scelto a cui è iscritto	VU9 CPU5 SPU2 SM1 SM2
FOBL 4.4.1	Il sistema dovrà permettere all'utente autenticato di visualizzare i criteri di terminazione di un processo	VU9 CPU5 SPU2 SM1 SM2
FOBL 4.4.1.1	L'utente autenticato potrà visualizzare il numero di completamenti del processo necessari e sufficienti a causarne la terminazione	VU9 CPU5 SPU2 SM1 SM2
FOBL 4.4.1.2	L'utente autenticato potrà visualizzare l'eventuale data di scadenza del processo selezionato	VU9 CPU5 SPU2 SM1 SM2
FOBL 4.4.2	L'utente autenticato potrà visualizzare le informazioni sullo stato corrente di avanzamento del processo selezionato	VU9 CPU5 SPU2 SM1 SM2
FOBL 4.4.2.1	L'utente autenticato potrà visualizzare il numero di passi già completati del processo selezionato	VU9 CPU5

FOBL 4.4.2.2	L'utente autenticato potrà visualizzare il numero di totale dei passi del processo selezionato	VU9 CPU5
FOBL 4.4.2.3	L'utente autenticato potrà visualizzare il numero di utenti che hanno già terminato il processo selezionato	VU9 CPU5
FOBL 4.4.2.4	L'utente autenticato potrà visualizzare il numero di utenti iscritti al processo selezionato	VU9 CPU5
FOBL 4.4.3	L'utente autenticato potrà visualizzare la lista dei passi in corso, cioè quelli iniziali o quelli immediatamente successivi agli ultimi passi superati	VU9 CPU5 SPU2 SM1 SM2 SM3 SM4
FOBL 4.4.4	Il sistema dovrà permettere all'utente autenticato di eseguire un passo del processo scelto	VU9 CPU5 SPU2 CMU3 SM1 SM2 SM3 SM4
FOBL 4.4.4.1	L'utente autenticato potrà visualizzare le informazioni del passo in esecuzione	VU9 CPU5 CMU3 SPU2 SM1 SM2

		SM3
		SM4
FOBL 4.4.4.1.1	L'utente autenticato potrà visualizzare la descrizione del passo in esecuzione	VU9
		CPU5
		CMU3
FOBL 4.4.4.1.2	L'utente autenticato potrà visualizzare l'eventuale nome dei dati del passo esecuzione	VU9
		CPU5
		CMU3
FOBL 4.4.4.2	L'utente autenticato potrà visualizzare i vincoli da rispettare per superare il passo in esecuzione	VU9
		CPU5
		CMU3
FOBL 4.4.4.2.1	L'utente autenticato potrà visualizzare se il passo in esecuzione richiede l'approvazione del <i>process owner_G</i> per essere concluso	VU9
		CPU5
		CMU3
FOBL 4.4.4.2.2	L'utente autenticato potrà visualizzare i vincoli sui dati geografici richiesti	VU9
		CPU5
		CMU3
FOBL 4.4.4.2.2.1	L'utente autenticato potrà visualizzare la posizione in cui dovrà trovarsi durante l'invio dei dati del passo in esecuzione	VU9
		CPU5
		CMU3
FOPL 4.4.4.2.2.2	L'utente autenticato potrà visualizzare l'eventuale raggio di tolleranza rispetto alla posizione geografica richiesta per l'esecuzione del passo	VU9
		CPU5
		CMU3
FOBL 4.4.4.2.3	L'utente autenticato potrà visualizzare l'eventuale intervallo temporale in cui può inviare i dati	VU9
		CPU5

		CMU3
FDEL 4.4.4.2.4	L'utente autenticato potrà visualizzare i vincoli sui dati numerici	VU9
		CPU5 CMU3
FOPL 4.4.4.2.4.1	L'utente autenticato potrà visualizzare il numero minimo e massimo di cifre dei valori numerici richiesti	VU9
		CPU5 CMU3
FDEL 4.4.4.2.4.2	L'utente autenticato potrà visualizzare se i valori numerici richiesti possono contenere cifre decimali	VU9
		CPU5 CMU3
FOPL 4.4.4.2.4.3	L'utente autenticato potrà visualizzare l'eventuale limite superiore e inferiore dei valori numerici richiesti	VU9
		CPU5 CMU3
FOBL 4.4.4.3	Il sistema dovrà permettere all'utente autenticato di inserire i dati richiesti per l'esecuzione del passo in corso	
FOBL 4.4.4.3.1	Il sistema dovrà permettere all'utente autenticato l'inserimento di una immagine richiesta	
FDEL 4.4.4.3.1.1	Il sistema dovrà permettere all'utente autenticato di scattare una foto per inserire l'immagine richiesta	
FOBL 4.4.4.3.1.2	Il sistema dovrà permettere all'utente autenticato di inserire una immagine caricandola dai suoi file	
FOBL 4.4.4.3.2	L'utente può inserire dati testuali richiesti dal passo in esecuzione	
FOBL 4.4.4.3.3	Il sistema dovrà permettere all'utente autenticato di inserire dati numerici richiesti dal passo in esecuzione	

FOBL 4.4.4.4	L'utente autenticato potrà inviare al sistema i dati richiesti per l'esecuzione del passo in corso
FOBL 4.4.4.4.1	L'utente autenticato potrà inviare al sistema i dati testuali inseriti
FOBL 4.4.4.4.2	L'utente autenticato potrà inviare al sistema le immagini inserite
FOBL 4.4.4.4.3	L'utente autenticato potrà inviare al sistema i dati numerici inseriti
FOBL 4.4.4.4.4	L'utente autenticato potrà inviare al sistema le coordinate della sua posizione
FOBL 4.4.4.4.5	L'utente autenticato potrà inviare al sistema la data e ora al momento della richiesta di invio dati
FOPL 4.4.4.4.6	Il sistema dovrà permettere all'utente autenticato di raccogliere i dati in assenza di connessione e di inviarli a collegamento ripristinato
FOPL 4.4.4.4.6.1	Il sistema, in assenza di connessione, dovrà permettere all'utente autenticato di salvare i dati richiesti dal passo esecuzione
FOPL 4.4.4.4.6.2	Il sistema, in presenza di connessione, dovrà permettere all'utente autenticato di inviare i dati precedentemente salvati
FOBL 4.4.4.5	Il sistema dovrà notificare all'utente autenticato se i dati che ha inviato sono corretti, se non soddisfano i vincoli di superamento del passo o se sono in attesa di approvazione
FOBL 4.4.4.6	Il sistema dovrà permettere all'utente autenticato di concludere un passo del quale ha ricevuto l'approvazione sui dati da parte del sistema o dal <i>process owner_G</i>
FOBL 4.4.5	Il sistema dovrà permettere all'utente autenticato di concludere un processo terminato o del quale ha eseguito tutti i passi

FOPL 4.4.5.1	Il sistema dovrà permettere all'utente autenticato la creazione di un report finale su un processo terminato o del quale ha eseguito tutti i passi in formato PDF _G
FOBL 4.4.5.2	Il sistema dovrà permettere all'utente autenticato di eliminare un processo un processo terminato o del quale ha eseguito tutti i passi, dalla lista dei processi gestiti
FOPL 4.4.6	Il sistema dovrà permettere all'utente autenticato di saltare il passo in esecuzione se facoltativo
FOBL 4.5	Il sistema dovrà permettere all'utente autenticato di disiscriversi da un processo a cui è iscritto
FOBL 5	L'utente potrà terminare la propria sessione, diventando utente generico

Tabella 3: Tabella requisiti-Componenti