Sirius

Sequenziatore

Specifica Tecnica

Versione 1.0.0

Ingegneria Del Software AA 2013-2014

Informazioni documento

Titolo documento:	Specifica Tecnica
Data creazione:	2014-03-13
Versione attuale:	1.0.0
Utilizzo:	Esterno
Nome file:	*SpecificaTecnica_v1.0.0.pdf*
Redazione:	Quaglio Davide
	Botter Marco
	Marcomin Gabriele
	Giachin Vanni
Verifica:	Santangelo Davide
Approvazione:	Seresin Davide
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
1.0.0	*2014-03-29*	*Seresin Davide*	*Responsabile*	Approvato documento
0.1.0	*2014-03-29*	*Santangelo Davide*	*Verificatore*	Verificato documento
0.0.7	*2014-03-29*	*Giachin Davide*	*Progettista*	Aggiunto tracciamento package-componenti, requisiti-componenti, componenti-requisiti
0.0.6	*2014-03-28*	*Giachin Davide*	*Progettista*	Aggiunta descrizione package front-end
0.0.5	*2014-03-28*	*Marcomin Gabriele*	*Progettista*	Aggiunta definizione package front-end
0.0.4	*2014-03-27*	*Quaglio Davide*	*Progettista*	Aggiunta descrizione package back-end
0.0.3	*2014-03-27*	*Botter Marco*	*Progettista*	Aggiunti diagrammi di sequenza e definizione package back-end
0.0.2	*2014-03-27*	*Quaglio Davide*	*Progettista*	Aggiunta definizione di architettura
0.0.1	*2014-03-15*	*Giachin Vanni*	*Progettista*	Stesura introduzione

Indice

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 *smaptphone* 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*si 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.

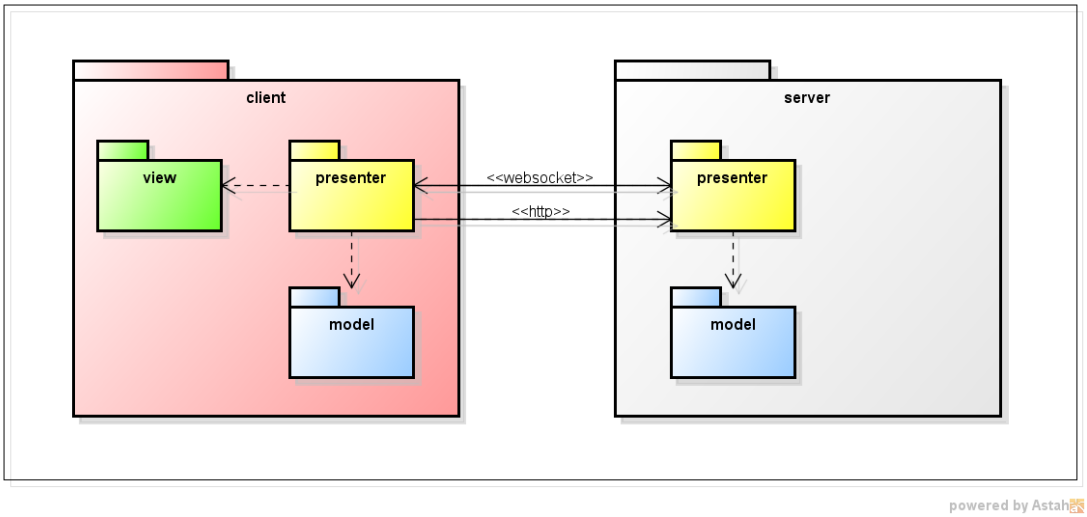

Figura 1: Diagramma UML architettura generale

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.process$_o$*wner* : *contienelecomponentitemplatenecessarieperlarealizzazionedel*
  contienelecomponentitemplatenecessarieperlarealizzazionedell'interfacciagraficadell'user.

2.3.2 Package sequenziatore.client.presenter

Il package$_G$ sequenziatore.client.presenter contiene tutte le componenti del Presenter della parte client$_G$ del sistema Sequenziatore; ed è composto da i seguenti package$_G$:

- sequenziatore.client.presenter.process$_o$*wner* : *contienelecomponentichecostituisconolacomponente*
  package$_G$:

- sequenziatore.client.presenter.process$_o$*wner.views* : *contieneleclassinecessarieperrealizzareegestirel'ag*

sequenziatore.client.presenter.user: contiene le componenti che realizzano la componente Presenter per l'utente autenticato; i sotto-package$_G$ di sequenziatore.client.presenter.user sono i seguenti:

- sequenziatore.client.presenter.user.views: contiene le classi necessarie per realizzare, mediante i template presenti nel package$_g$ sequenziatore.client.view.user, l'interfaccia utente e gestirne l'interazione con l'utente autenticato, gestendo inoltre la logica di business dell'applicazione;

2.3.3 Package sequenziatore.client.model

Il package$_G$ sequenziatore.client.model contiene tutte le classi della componente Model. Il package$_G$ sequenziatore.client.model contiene inoltre:

- sequenziatore.client.model.collections contiene le varie collezioni di dati contenuti nel package model; il nome del package ricalca inoltre il nome del supertipo di tutte le collezioni di strutture date usate in un sistema sviluppato usando il framework$_G$ Backbone.js$_G$

3 Descrizione singoli componenti

3.1 Package sequenziatore.client.view

3.1.1 Package sequenziatore.client.view.user

3.1.1.1 IMainUser

- **Nome:** `IMainUser`;
- **Descrizione:** Interfaccia che permette la gestione delle principali componenti dell'Interfaccia grafica dell'utente.

3.1.1.2 MainUser

- **Nome:** `MainUser`;
- **Descrizione:** Classe che permette la gestione delle principali componenti dell'interfaccia grafica dell'utente;

- **3.1.2 Package sequenziatore.client.view.process$_owner$**

3.1.3 Package sequenziatore::client::iview::iprocessowner

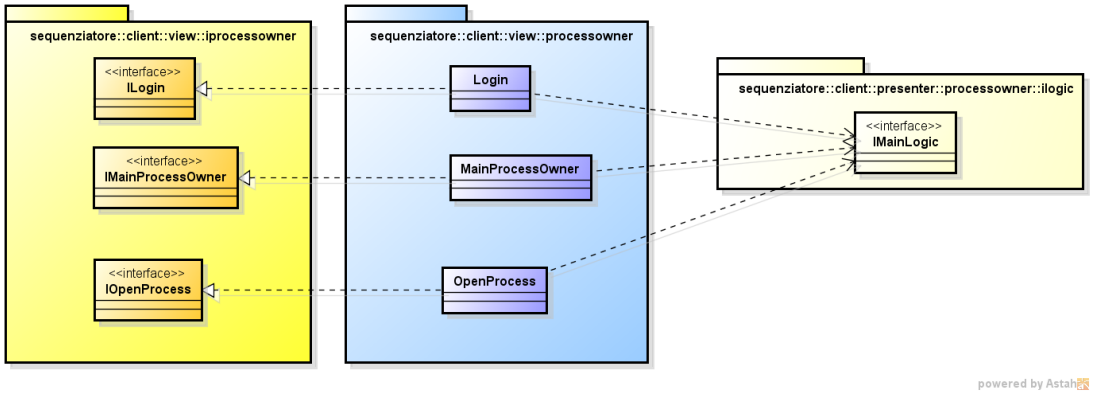

Figura 2: Diagramma view principale process owner

3.1.3.1 IMainProcessOwner

- **Nome:** `IMainProcessOwner`;
- **Package:** `sequenziatore::client::view::iprocessowner`;
- **Descrizione:** Interfaccia che permette la gestione delle principali componenti dell'interfaccia grafica dell'utente *process owner*$_G$.

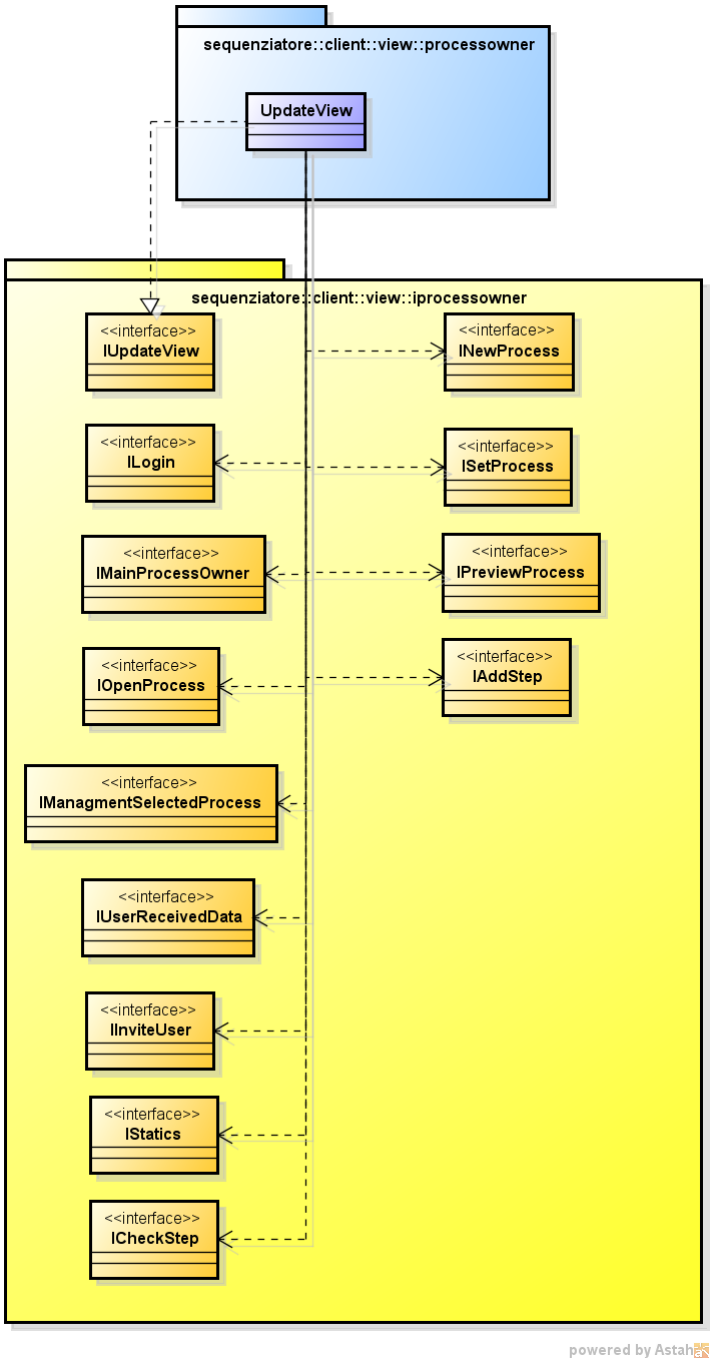

Figura 3: Diagramma aggiornamento view process owner

3.1.3.2 IUpdateView

- **Nome:** `IUpdateView`;
- **Package:** `sequenziatore::client::view::iprocessowner`;

- **Descrizione:** Interfaccia che permette di gestire l'aggiornamento dei *widget*$_G$ della componente *view*.

3.1.3.3 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*$_G$.

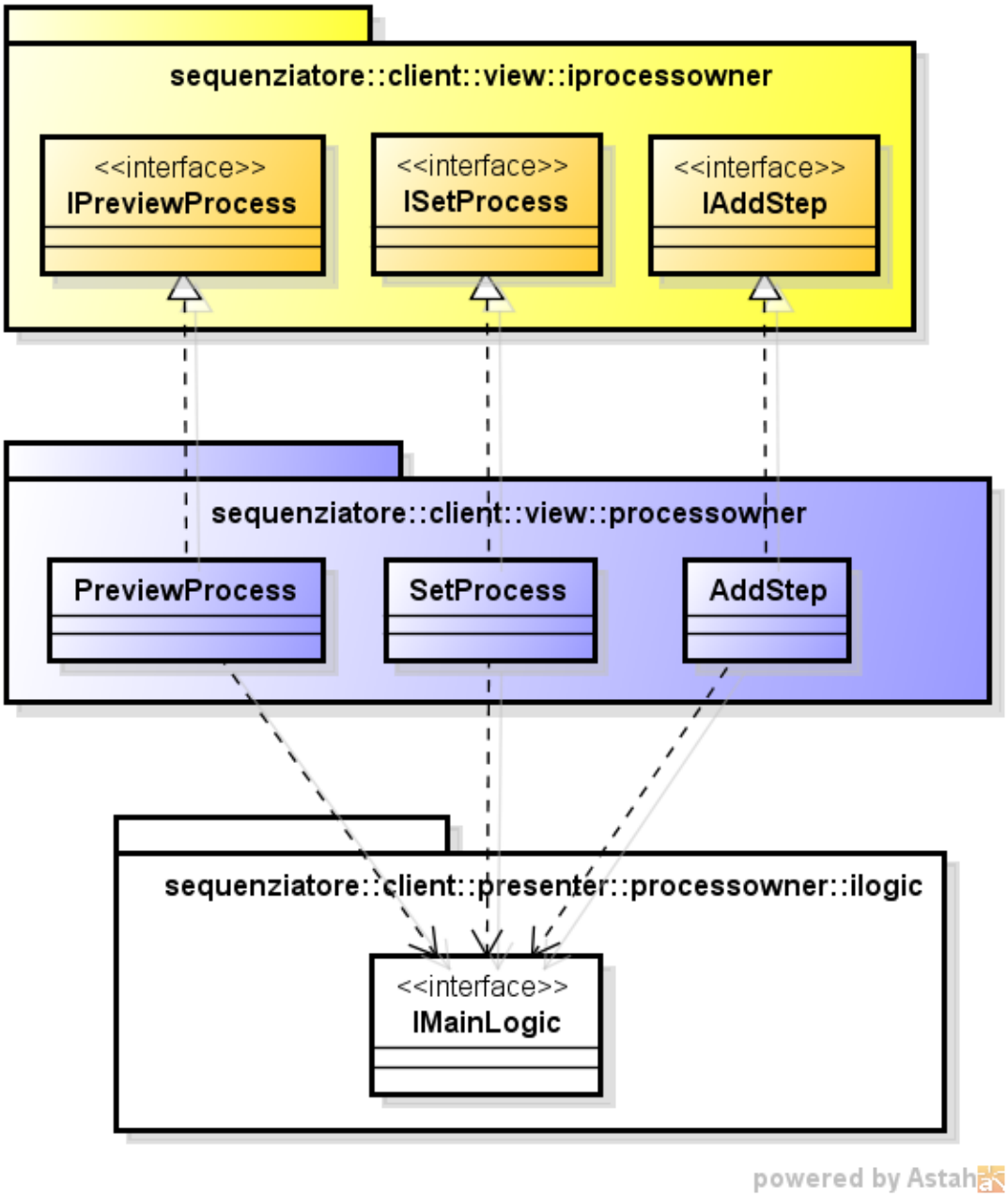

Figura 4: Diagramma view creazione nuovo processo

3.1.3.4 ISetProcess

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

3.1.3.5 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.

3.1.3.6 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.

3.1.3.7 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.

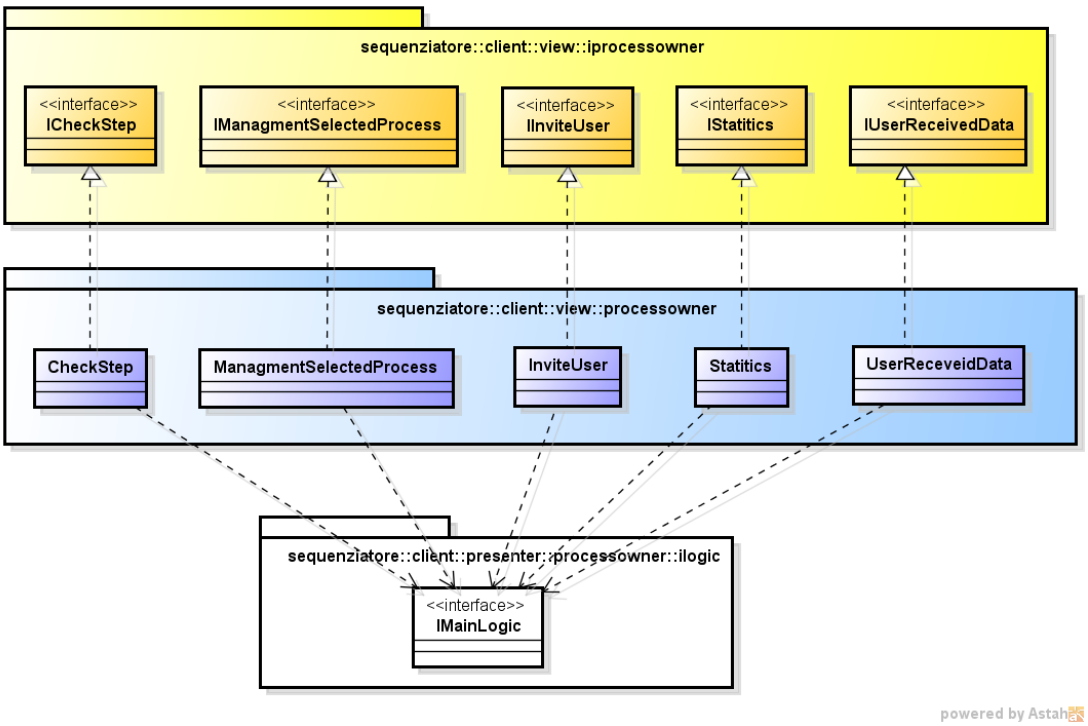

Figura 5: Diagramma view gestione processi creati

3.1.3.8 IManagmentSelectedProcess

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

3.1.3.9 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.

3.1.3.10 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.

3.1.3.11 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;

3.1.3.12 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.

3.1.3.13 MainProcessOwner

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

3.1.3.14 UpdateView

- **Nome:** `UpdateView`;
- **Package:** `sequenziatore.client::view::processowner`;
- **Descrizione:** Classe che permette di gestire l'aggiornamento dei *widget*$_G$ 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`.

3.1.3.15 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*$_G$;
- **Relazioni con altri componenti:**
  La classe implementa l'interfaccia `sequenziatore::client::view::iprocessowner::ILogin` e comunica con il*presenter* utilizzando metodi della classe `sequenziatore::client::presenter::processowner::logic::MainLogic`.

3.1.3.16 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'interfCheckStepaccia `sequenziatore::client::view::iprocessowner::ISetProcess` e comunica con il*presenter* utilizzando metodi della classe `sequenziatore::client::presenter::processowner::logic::MainLogic`.

3.1.3.17 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 il*presenter* utilizzando metodi della classe `sequenziatore::client::presenter::processowner::logic::MainLogic`.

3.1.3.18 PreviewProcess

- **Nome:** `PreviewProcess`;
- **Package:** `sequenziatore.client::view::processowner`;
- **Descrizione:** Classe che permette realizzare i*widget* 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 il*presenter* utilizzando metodi della classe `sequenziatore::client::presenter::processowner::logic::MainLogic`.

3.1.3.19 OpenProcess

- **Nome:** `OpenProcess`;
- **Package:** `sequenziatore.client::view::processowner`;
- **Descrizione:** Classe che permette di realizzare i*widget* 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 il*presenter* utilizzando metodi della classe `sequenziatore::client::presenter::processowner::logic::MainLogic`.

3.1.3.20 ManagmentSelectedProcess

- **Nome:** `ManagmentSelectedProcess`;
- **Package:** `sequenziatore.client::view::processowner`;
- **Descrizione:** Classe che permette di realizzare i*widget* che consentono di gestire i processi selezionati;
- **Relazioni con altri componenti:**
  La classe implementa l'interfaccia `sequenziatore::client::view::iprocessowner::IManagmentSelectedProcess` e comunica con il*presenter* utilizzando metodi della classe `sequenziatore::client::presenter::processowner::logic::MainLogic`.

3.1.3.21 CheckStep

- **Nome:** `CheckStep`;
- **Package:** `sequenziatore.client::view::processowner`;
- **Descrizione:** Classe che permette di realizzare i*widget* 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 il*presenter* utilizzando metodi della classe `sequenziatore::client::presenter::processowner::logic::MainLogic`.

3.1.3.22 UserReceivedData

- **Nome:** `UserReceivedData`;
- **Package:** `sequenziatore.client::view::processowner`;
- **Descrizione:** Classe che permette di realizzare i*widget* 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`.

3.1.3.23 Statistics

- **Nome:** `Statistics`;
- **Package:** `sequenziatore.client::view::processowner`;
- **Descrizione:** Classe che permette di realizzare i*widget* 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`.

3.1.3.24 InviteUser

- **Nome:** `InviteUser`;
- **Package:** `sequenziatore.client::view::processowner`;
- **Descrizione:** Classe che permette di realizzare i*widget* 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`.

3.2 Package sequenziatore.client.presenter

3.2.1 Package sequenziatore.client.presenter.user.views

3.2.1.1 UserRouter

- **Nome:** `UserRouter`;
- **Descrizione:** Classe che permette la modifica dello stato dell'interfaccia grafica in base all'evolversi dell'interazione fra utente e applicazione

  La classe cambia lo stato dell'interfaccia richiamando le seguente classi, le quali permetteranno la realizzazione del widget desiderato implementa l'interfaccia:

  * `Login`;
  * `Register`;
  * `MainUser`;
  * `UserData`;
  * `OpenProcessgic`;
  * `ManagmentProcess`;

3.2.1.2 Login

- **Nome:** `LoginLogic`;
- **Descrizione:** Classe che ha il compito di gestire le richieste di autenticazione e chiusura della sessione da parte dell'utente e realizzare l'opportuna interfaccia grafica;
- **Relazioni con altri componenti:**

  La classe, comunicando con l'interfaccia `ILogin`, realizza il widget per la l'autenticazione.

3.2.1.3 Register

- **Nome:** `Register`;
- **Descrizione:** Classe che ha il compito di gestire le richieste di registrazione da parte dell'utente;
- **Relazioni con altri componenti:**

  La classe, comunicando attraverso l'interfaccia `IRegister` realizza il widget per la registrazione.

3.2.1.4 UserData

- **Nome:** `UserData`;
- **Descrizione:** Classe che ha il compito di gestire la visualizzazione e la modifica dei dati dell'utente;
- **Relazioni con altri componenti:**
  La classe, per mezzo dell'interfaccia`IUserData`, realizza il widget preposto alla visualizzazione e modifica dei dati dell'utente.

3.2.1.5 OpenProcess

- **Nome:** `OpenProcess`;
- **Descrizione:** Classe che ha il compito di selezionare, ricercare e aprire un processo fra quelli eseguibili;
- **Relazioni con altri componenti:**
  La classe realizza e modifica l'opportuno widget mediante l'interfaccia `IOpenProcess`.

3.2.1.6 ManagmentProcess

- **Nome:** `ManagmentProcess`;
- **Descrizione:** Classe che ha il compito di gestire e accedere alle informazioni relative allo stato del processo selezionato.;
- **Relazioni con altri componenti:**
  La classe, mediante l'interfaccia `IManagmentProcess`, realizza e aggiorna il widget che permette la gestione del processo selezionato. Inoltre provvede a creare e chiamare le seguenti classi in base alle decisioni dell'utente:
  * `PrintReport`
  * `SendData`

3.2.1.7 PrintReport

- **Nome:** `PrintReport`;
- **Descrizione:** Classe che ha il compito di gestire la creazione di report sull'andamento dei processi in esecuzione;
- **Relazioni con altri componenti:**
  La classe chiama l'interfaccia `IPrintReport` per poter realizzare il widget per poter creare il report.

3.2.1.8 SendData

- **Nome:** `SendData`;
- **Descrizione:** Classe che ha il compito di gestire l'inserimento e l'invio di dati da parte degli utenti, per completare il passo corrente;
- **Relazioni con altri componenti:**
  La classe,mediante l'interfaccia `ISendData`, crea ed aggiorna il widget preposto per la selezione del vario tipo di dato da inviare. Crea ed invoca le seguenti classi preposte alla realizzazione e gestione dei widget per inviare i vari tipo di dati:
  * `SendText`
  * `SendNumb`
  * `SendImage`
  * `SendPosition`

3.2.1.9 SendText

- **Nome:** `SendText`;
- **Descrizione:** Classe che permette l'inserimento e l'invio di dati testuali da parte degli utenti;
- **Relazioni con altri componenti:**
  La classe,mediante l'interfaccia `ISendText`, realizza e aggiorna l'opportuno widget.

3.2.1.10 SendNumb

- **Nome:** `SendNumb`;
- **Descrizione:** Classe che ha il compito di permettere l'inserimento e l'invio di dati numerici da parte degli utenti, per completare il passo corrente;
- **Relazioni con altri componenti:**
  La classe,mediante l'interfaccia `ISendNumb`, realizza e aggiorna l'opportuno widget.

3.2.1.11 SendImage

- **Nome:** `SendImage`;
- **Descrizione:** Classe che gestisce l'inserimento e l'invio di immagini da parte degli utenti, richieste per completare il passo corrente;

- **Relazioni con altri componenti:**
  La classe,mediante l'interfaccia `ISendImage`, realizza e aggiorna l'opportuno widget.

3.2.1.12 SendPosition

- **Nome:** `SendPosition`;
- **Descrizione:** Classe che ha il compito di gestire il calcolo e l'invio della posizione geografica dell'utente;
- **Relazioni con altri componenti:**
  La classe,mediante l'interfaccia `ISendPosition`, realizza e aggiorna l'opportuno widget.

3.2.2 Package sequenziatore::client::presenter::processowner::logic

3.2.2.1 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::ManagmentProcessLogic`;
  * `sequenziatore::client::presenter::processowner::logic::CheckStepLogic`;
  * `sequenziatore::client::presenter::processowner::logic::StatisticsLogic`;

* `sequenziatore::client::presenter::processowner::logic::UserDataLogic`;
* `sequenziatore::client::presenter::processowner::logic::InviteUserLogic`.

3.2.2.2 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`.

3.2.2.3 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::iprocessowner::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`.

3.2.2.4 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::iprocessowner::ilogic::IAddStepLogic`, e utilizza metodi delle classi `sequenziatore::client::model::ProcessOwnerData` e `sequenziatore.client::view::processowner::UpdateView`.

3.2.2.5 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::iprocessowner::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`.

3.2.2.6 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::iprocessowner::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`.

3.2.2.7 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::iprocessowner::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`.

3.2.2.8 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::iprocessowner::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`.

3.2.2.9 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::iprocessowner::ilogic::IInviteUserLogic`, e utilizza metodi delle classi `sequenziatore::client::presenter::servercommunication::HttpCommunication`, `sequenziatore::client::presenter::servercommunication::WebsocketCommunication` e `sequenziatore.client::view::processowner::UpdateView`.

3.3 Package sequenziatore.client.model

3.3.0.10 ProcessModel

- **Nome:** `ProcessModel`;
- **Descrizione:** Classe che permette di gestire i dati di un processo, il salvataggio in locale e l'invio al back-end del sistema Sequenziatore.

3.3.0.11 ProcessDataModel

- **Nome:** `ProcessDataModel`;
- **Descrizione:**

3.3.0.12 StepModel

- **Nome:** `StepModel`;
- **Descrizione:** Classe che permette la gestione,il controllo rispettando i vincoli, invio al back-end e il salvataggio in locale dei dati di un passo di un processo.

3.3.0.13 UserModel

- **Nome:** `UserModel`;
- **Descrizione:** Classe che permette di gestire i dati di una sessione di un utente autenticato o di un process owner.

3.3.1 Package sequenziatore.client.model.collection

3.3.1.1 ProcessDataCollection

- **Nome:** `ProcessDataCollection`;
- **Descrizione:**
- **Relazioni con altri componenti:**

3.3.1.2 ProcessCollection

- **Nome:** `ProcessModel`;
- **Descrizione:** Classe che permette di gestire un insieme di processi.
- **Relazioni con altri componenti:**
  La classe definise una collezione di `sequenziatore.client.model.ProcessModel`.

3.3.1.3 StepCollection

- **Nome:** `StepCollection`;
- **Descrizione:** Classe che permette la gestione di un insieme di passi.
- **Relazioni con altri componenti:**

  La classe definise una collezione di `sequenziatore.client.model.StepModel`

3.4 Package com.sirius.sequenziatore.server.presenter

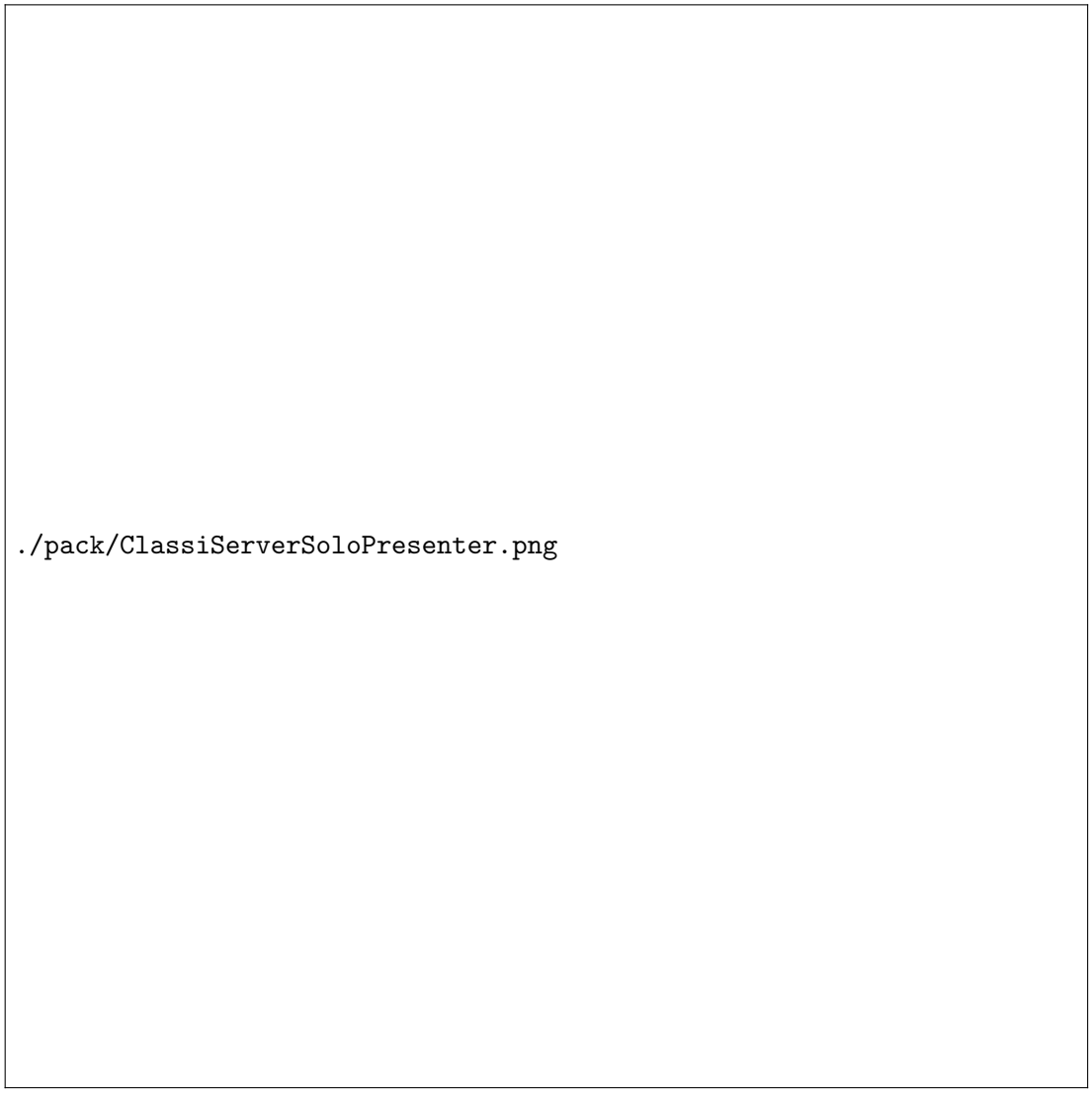

Figura 6: Diagramma presenter server

3.4.1 Package com.sirius.sequenziatore.server.presenter.common

Questo *package* contiene le classi che effettuano operazioni generali oppure comuni tra *Process Owner* e Utenti.

3.4.1.1 LoginController

- **Nome:** `LoginController`;
- **Package:** com.sirius.sequenziatore.server.presenter.common
- **Descrizione:** Classe che permette la gestione della login di un utilizzatore del sistema, controllando che i dati inseriti riferiscano a un utente correttamente

iscritto al sistema, ponendo attenzione se esso sia un *process owner* o un utente normale;

- **Relazione con altre componenti:** la classe invoca i metodi della classe:
  - com.sirius.sequenziatore.server.model.IDataAccessObject;

3.4.1.2 SignUpConroller

- **Nome:** `SignUpController`;
- **Package:** com.sirius.sequenziatore.server.presenter.common
- **Descrizione:** Classe che permette la gestione della registrazione di un nuovo utente nel sistema, nonostante la correttezza dei dati inseriti venga controllata dalla parte client, per sicurezza verrà effettuato un nuovo controllo anche sulla parte server prima di inserire un utente nel sistema;
- **Relazione con altre componenti:** la classe invoca i metodi della classe:
  - com.sirius.sequenziatore.server.model.IDataAccessObject;

3.4.1.3 StepInfoController

- **Nome:** `StepInfoController`;
- **Package:** com.sirius.sequenziatore.server.presenter.common
- **Descrizione:** Classe che fornisce a chi lo richiede lo scheletro di un passo, quindi andrà a fornire i dati da inserire per tale passo e altre informazioni;
- **Relazione con altre componenti:** la classe invoca i metodi della classe:
  - com.sirius.sequenziatore.server.model.IDataAccessObject;

3.4.1.4 ProcessInfoController

- **Nome:** `ProcessInfoController`;
- **Package:** com.sirius.sequenziatore.server.presenter.common
- **Descrizione:** Classe incaricata di fornire a chi lo richieda lo scheletro di un processo, come ad esempio numero di passi o condizioni per il suo completamento;
- **Relazione con altre componenti:** la classe invoca i metodi della classe:
  - com.sirius.sequenziatore.server.model.IDataAccessObject;

3.4.2 Package com.sirius.sequenziatore.server.presenter.user

3.4.2.1 AccountController

- **Nome:** `AccountController`;
- **Package:** com.sirius.sequenziatore.server.presenter.user
- **Descrizione:** classe che permette la modifica dei dati di un utente come password o altre informazioni inerenti ai dettagli personali di un utente;
- **Relazione con altre componenti:** la classe richiama i metodi della classe:
  - com.sirius.sequenziatore.server.model.IDataAccessObject;

3.4.2.2 UserProcessController

- **Nome:** `UserProcessController`;
- **Package:** com.sirius.sequenziatore.server.presenter.user
- **Descrizione:** classe che restituisce all' utente i dati di uno o più processi, può inoltre permettere l' inoltro della richiesta di un utente a iscriversi o disiscriversi a un processo;
- **Relazione con altre componenti:** la classe richiama i metodi della classe:
  - com.sirius.sequenziatore.server.model.IDataAccessObject;

3.4.2.3 UserStepController

- **Nome:** `UserStepController`;
- **Package:** com.sirius.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, inoltre è incaricato di restituire i dati inseriti di un passo quando richiesto da un utente;
- **Relazione con altre componenti:** la classe richiama i metodi della classe:
  - com.sirius.sequenziatore.server.model.IDataAccessObject;

3.4.2.4 ReportController

- **Nome:** `ReportController`;
- **Package:** com.sirius.sequenziatore.server.presenter.user
- **Descrizione:** Classe che fornisce i dati per generare 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 della classe:
  - com.sirius.sequenziatore.server.model.IDataAccessObject;

3.4.3 Package com.sirius.sequenziatore.server.presenter.processowner

3.4.3.1 ProcessController

- **Nome:** `ProcessController`;
- **Package:** com.sirius.sequenziatore.server.presenter.processowner
- **Descrizione:** Classe che riceve le richieste da parte del *process owner* per la gestione dei processi come ad esempio la creazione, la modifica e la eliminazione degli stessi;
- **Relazione con altre componenti:** la classe invoca i metodi della classe:
  - com.sirius.sequenziatore.server.model.IDataAccessObject;

3.4.3.2 StepController

- **Nome:** `StepController`;
- **Package:** com.sirius.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 singoli passi, per esempio permetterà di aggiungere o rimuovere dei campi richiesti;
- **Relazione con altre componenti:** la classe invoca i metodi della classe:
  - com.sirius.sequenziatore.server.model.IDataAccessObject;

3.4.3.3 ApproveStepController

- **Nome:** `ApproveStepController`;
- **Package:** com.sirius.sequenziatore.server.presenter.processowner
- **Descrizione:** Classe che permette al process owner la gestione dei passi da approvare, quindi con questa classe si forniranno la lista di passi da approvare e si gestirà la approvazione o il rifiuto dei suddetti in base all' esito del process owner;
- **Relazione con altre componenti:** la classe invoca i metodi della classe:
  - com.sirius.sequenziatore.server.model.IDataAccessObject;

3.5 Package sequenziatore::server::model

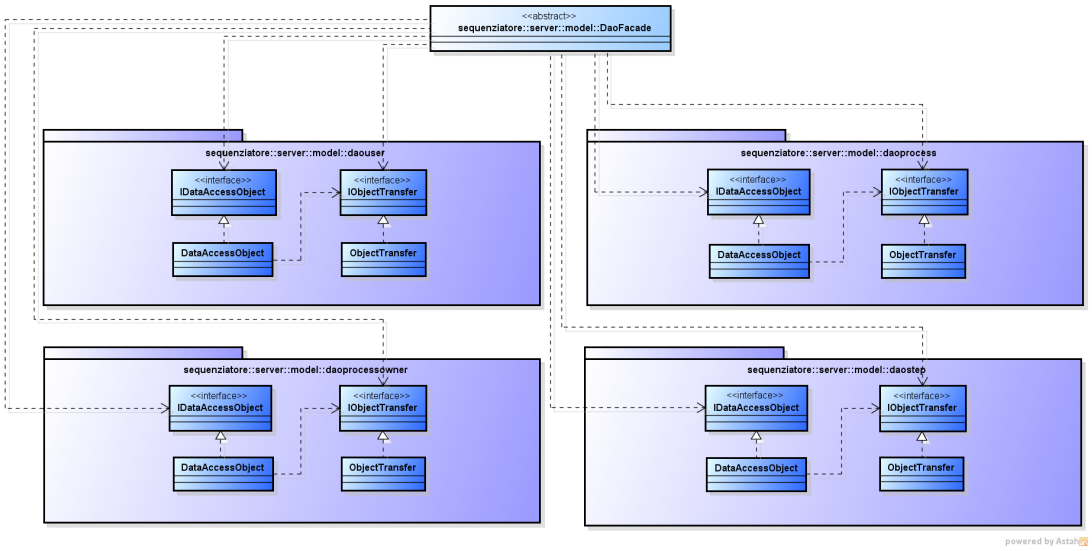

Figura 7: Diagramma model server

3.5.0.4 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

3.5.1 Package sequenziatore::server::model::daouser

3.5.1.1 IDataAccessObject

- **Nome:** `IDataAccessObject`;
- **Package:** sequenziatore::server::model::daouser
- **Descrizione:** Interfaccia con il compito di interagire con il database.

3.5.1.2 DataAccessObject

- **Nome:** `DataAccessObject`;
- **Package:** sequenziatore::server::model::daouser
- **Descrizione:** classe che si occupa di effettuare le richieste al database, acquisendo i dati richiesti o inserendone di nuovi.
- **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

3.5.1.3 IObjectTransfer

- **Nome:** `IObjectTransfer`;
- **Package:** sequenziatore::server::model::daouser
- **Descrizione:** Interfaccia che permette lo scambio di dati tra model e presenter.

3.5.1.4 ObjectTransfer

- **Nome:** `ObjectTransfer`;
- **Package:** sequenziatore::server::model::daouser
- **Descrizione:** Classe che permette lo scambio di dati tra la classe DataAccessObject e il presenter.
- **Relazione con altre componenti:** la classe implementa l'interfaccia sequenziatore::server::model::daouser::IObjectTransfer.

3.5.2 Package sequenziatore::server::model::daoprocessowner

3.5.2.1 IDataAccessObject

- **Nome:** `IDataAccessObject`;
- **Package:** sequenziatore::server::model::daoprocessowner
- **Descrizione:** Interfaccia con il compito di interagire con il database.

3.5.2.2 DataAccessObject

- **Nome:** `DataAccessObject`;
- **Package:** sequenziatore::server::model::daoprocessowner
- **Descrizione:** classe che si occupa di effettuare le richieste al database, acquisendo i dati richiesti o inserendone di nuovi.
- **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

3.5.2.3 IObjectTransfer

- **Nome:** `IObjectTransfer`;
- **Package:** sequenziatore::server::model::daoprocessowner
- **Descrizione:** Interfaccia che permette lo scambio di dati tra model e presenter.

3.5.2.4 ObjectTransfer

- **Nome:** `ObjectTransfer`;
- **Package:** sequenziatore::server::model::daoprocessowner
- **Descrizione:** Classe che permette lo scambio di dati tra la classe DataAccessObject e il presenter.
- **Relazione con altre componenti:** la classe implementa l'interfaccia sequenziatore::server::model::daoprocessowner::IObjectTransfer.

3.5.3 Package sequenziatore::server::model::daoprocess

3.5.3.1 IDataAccessObject

- **Nome:** `IDataAccessObject`;
- **Package:**sequenziatore::server::model::daoprocess
- **Descrizione:** Interfaccia con il compito di interagire con il database.

3.5.3.2 DataAccessObject

- **Nome:** `DataAccessObject`;
- **Package:** sequenziatore::server::model::daoprocess
- **Descrizione:** classe che si occupa di effettuare le richieste al database, acquisendo i dati richiesti o inserendone di nuovi.
- **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

3.5.3.3 IObjectTransfer

- **Nome:** `IObjectTransfer`;
- **Package:** sequenziatore::server::model::daoprocess
- **Descrizione:** Interfaccia che permette lo scambio di dati tra model e presenter.

3.5.3.4 ObjectTransfer

- **Nome:** `ObjectTransfer`;
- **Package:** sequenziatore::server::model::daoprocess
- **Descrizione:** Classe che permette lo scambio di dati tra la classe DataAccessObject e il presenter.
- **Relazione con altre componenti:** la classe implementa l'interfaccia sequenziatore::server::model::daoprocess::IObjectTransfer.

3.5.4 Package sequenziatore::server::model::daostep

3.5.4.1 IDataAccessObject

- **Nome:** `IDataAccessObject`;
- **Package:** sequenziatore::server::model::daostep
- **Descrizione:** Interfaccia con il compito di interagire con il database.

3.5.4.2 DataAccessObject

- **Nome:** `DataAccessObject`;
- **Package:** sequenziatore::server::model::daostep
- **Descrizione:** classe che si occupa di effettuare le richieste al database, acquisendo i dati richiesti o inserendone di nuovi.

- **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

3.5.4.3 IObjectTransfer

- **Nome:** `IObjectTransfer`;
- **Package:** sequenziatore::server::model::daostep
- **Descrizione:** Interfaccia che permette lo scambio di dati tra model e presenter.

3.5.4.4 ObjectTransfer

- **Nome:** `ObjectTransfer`;
- **Package:** sequenziatore::server::model::daostep
- **Descrizione:** Classe che permette lo scambio di dati tra la classe DataAccessObject e il presenter.
- **Relazione con altre componenti:** la classe implementa l'interfaccia sequenziatore::server::model::daostep::IObjectTransfer.

4 Design pattern

4.1 Model View Presenter

- **Scopo e descrizione:** 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 sopo 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.

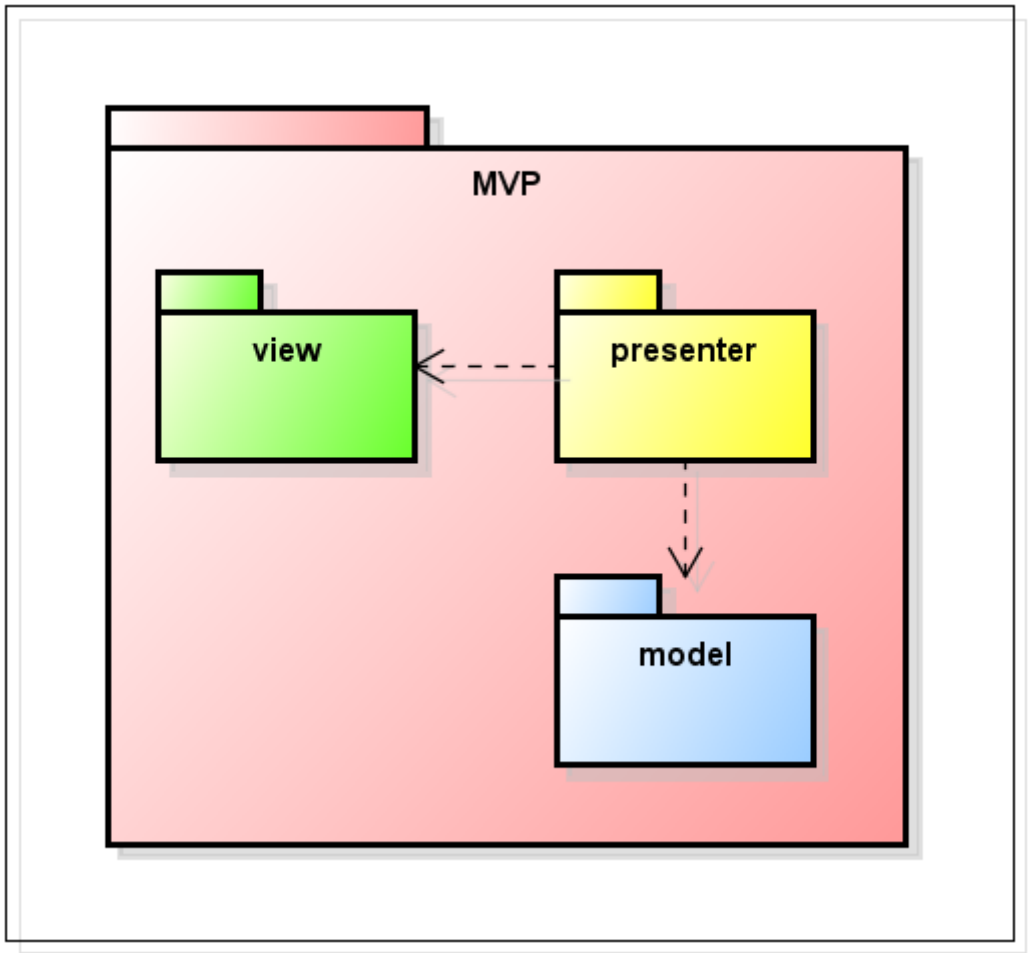

Figura 8: Diagramma UML pattern MVP

- **Contesto d'uso:** Il *pattern*$_G$ *Model View Presenter* (MVP) è la architettura di base del progetto.

4.2 Facade

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

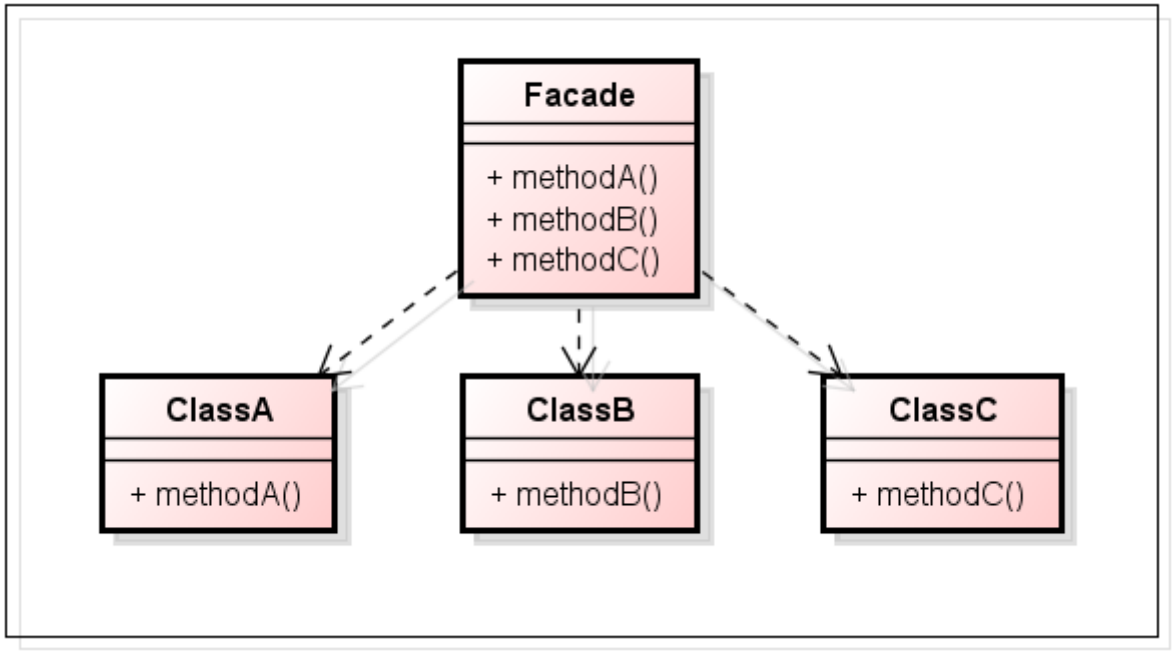

Figura 9: Diagramma UML pattern Facade

- **Contesto d'uso:** Il *pattern*$_G$ *Facade* è stato utilizzato nei *package*$_G$ *sequenziatore::server::presenter*, *sequenziatore::server::model*, *sequenziatore::client::presenter::user::logic* e *sequenziatore::client::view::user*.

4.3 Data Access Object

- **Scopo e descrizione:** Il *pattern*$_G$ *Data Access Object* (DAO) permette alla *business logic*$_G$ di essere indipendente dall'implementazione della persistenza dei dati. Il *pattern*$_G$ DAO è caratterizzato dai seguenti componenti:
  - **Data Access Object:** Realizza l'acesso fisico alla sorgente dei dati in modo trasparente al resto dell'applicazione;
  - **Object Transfer:** Rappresenta l'oggeto utilizzato per il trasferimento dei dati, sia in lettura, sia in scrittura.

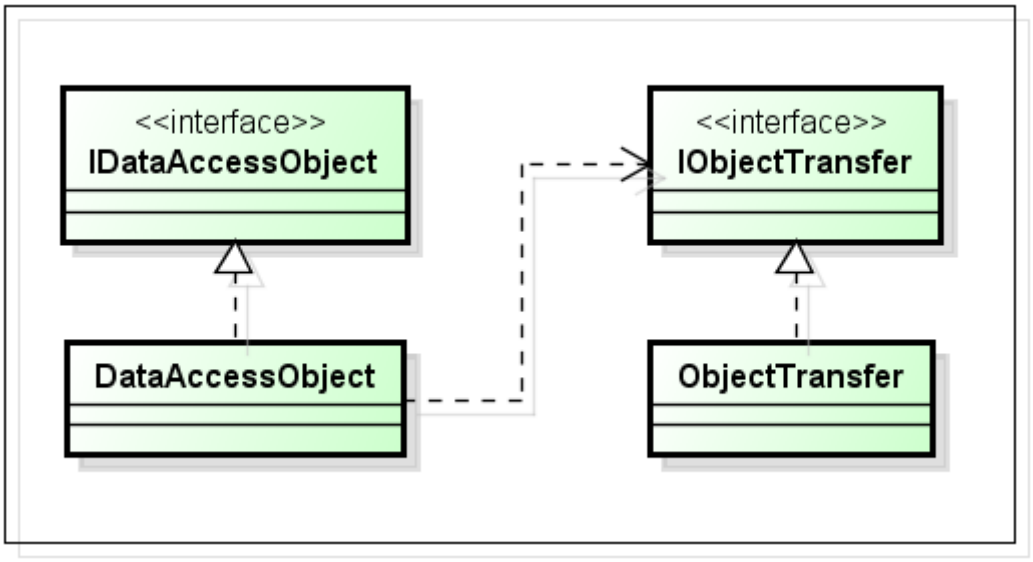

Figura 10: Diagramma UML pattern DAO

- **Contesto d'uso:** Il *pattern*$_G$ DAO è stato utilizzato nei *package*$_G$ *sequenziatore::server::model::daouser*, *sequenziatore::server::model::daoprocessowner*, *sequenziatore::server::model::daoprocess* e *sequenziatore::server::model::daostep*.

5 Diagrammi di attività

Di seguito vengono illustrati i diagrammi di attività che illustrano l'interazione degli utenti con il l'applicativo *Sequenziatore*. Si è cercato di creare diagrammi ad alto livello che descrivessero il principale flusso di azioni. Tali diagrammi sono in seguito stati suddivisi secondo sotto-diagrammi specifici, al fine di illustrare con maggior dettaglio il flusso di certe attività.

5.1 Diagrammi di attività: process owner

5.1.1 Creazione processo

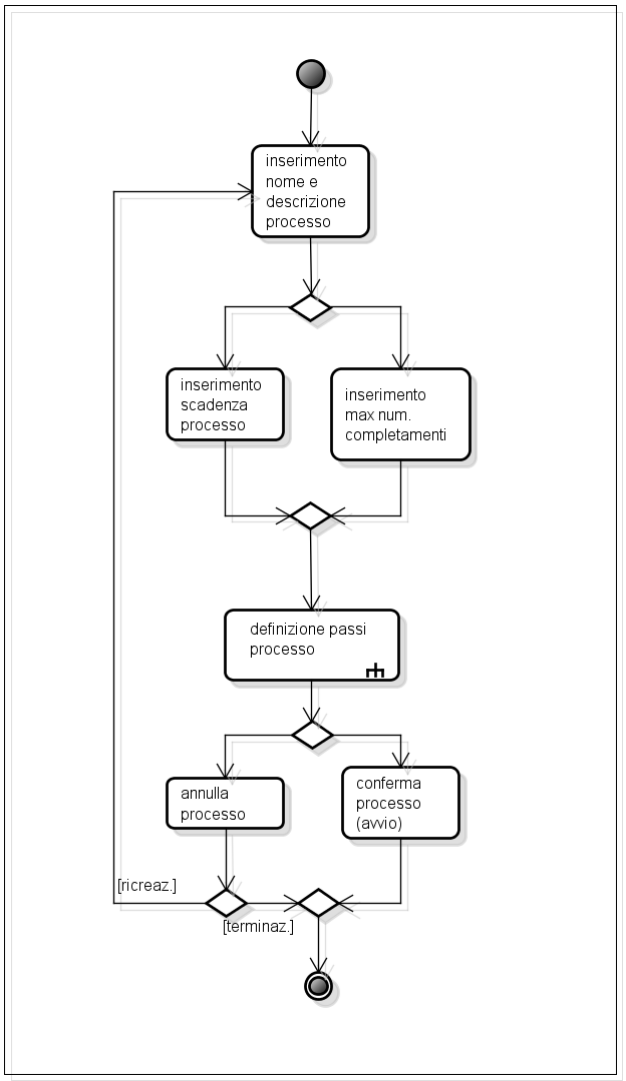

Figura 11: Attività process owner: creazione processo.

Descrizione: Il process owner$_G$ al fine di creare un nuovo processo dovrà dapprima inserire il nome e la descrizione del suddetto. Inseriti i primi campi potrà inserire o una data di scadenza o un numero massimo di completamenti del processo, alchè sarà tenuto a definire i passi del suddetto (per maggiori dettagli vedere: Figura 3, Attività process owner: creazione passo). Eseguiti i passi sopracitati potrà decidere se annullare il processo o darne la conferma

5.1.2 Gestione processo

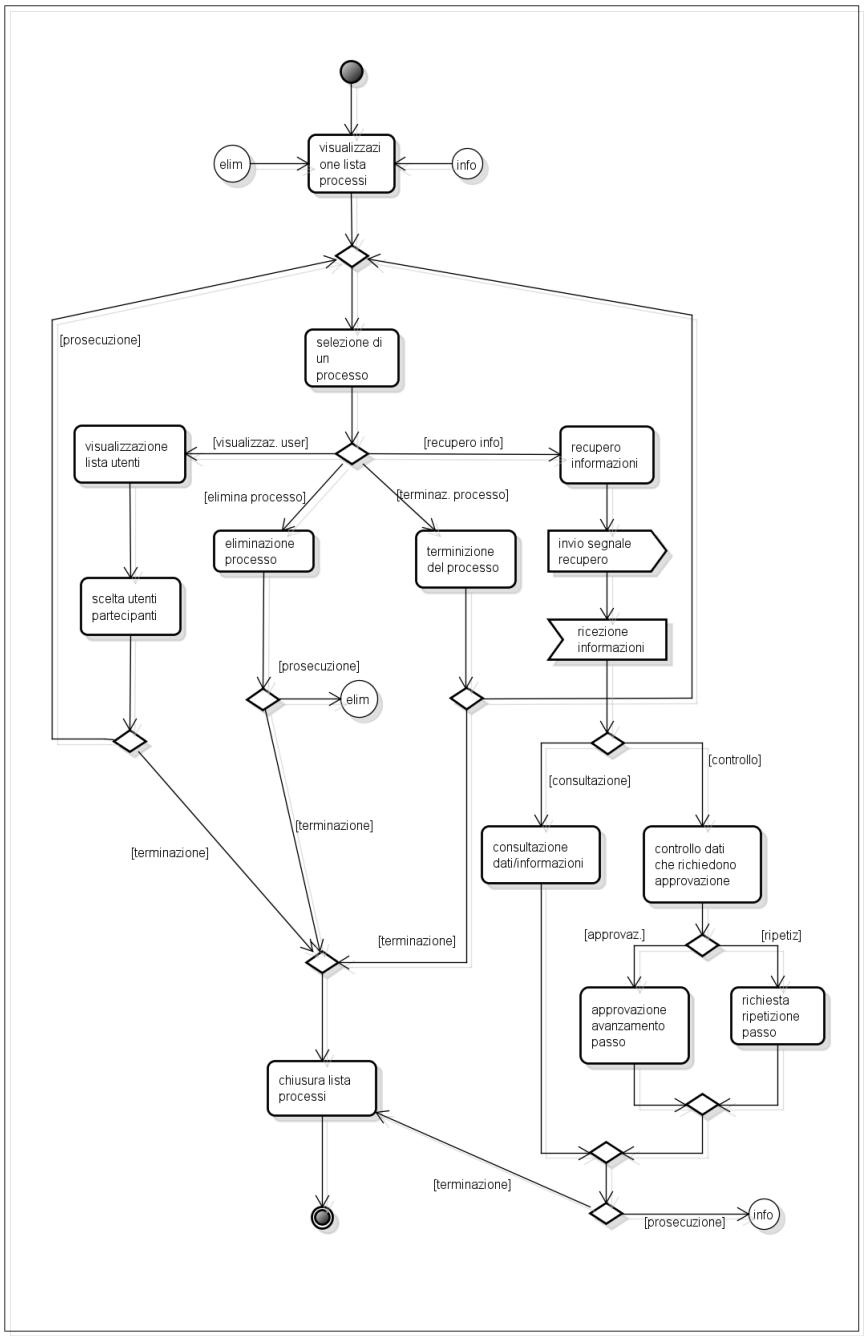

Figura 12: Attività process owner: gestione di un processo.

Descrizione: Brevemente il process owner dopo aver visualizzato la lista processi, può selezionare il processo di interesse per accedere alla sua gestione, ossia: visualizzare utenti (al fine di aggiungerli al processo), eliminare il processo, terminarlo oppure recuperare le informazioni relative al suddetto. Il recupero delle informazioni è

necessario per controllare i dati che richiedono la verifica umana. Nel momento in cui il process owner ha finito di gestire i processi, potrà chiudere l'applicazione.

5.1.3 Creazione passo

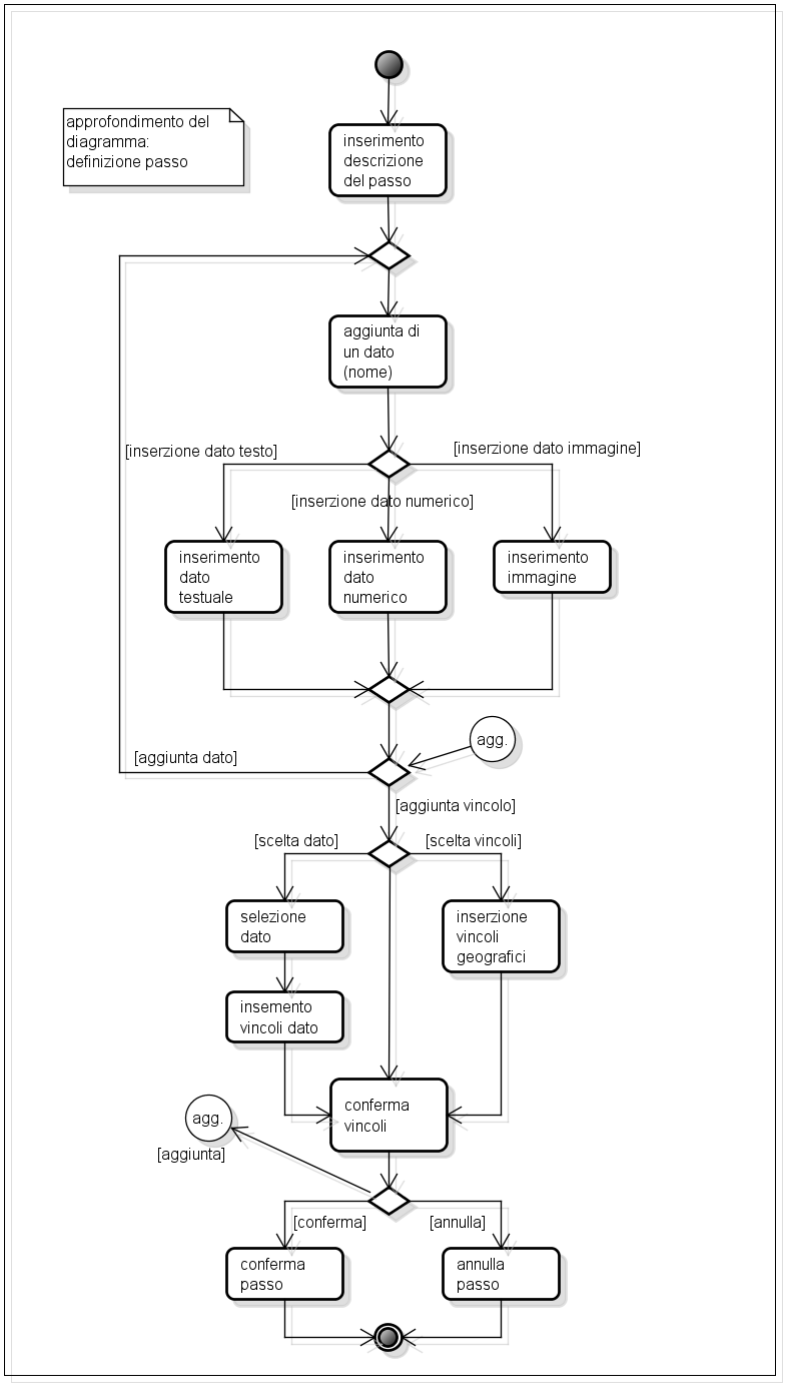

Figura 13: Attività process owner: creazione passo.

Descrizione: durante la creazione /modifica di un processo l'utente process owner potrà decidere di aggiungere dei passi, l'aggiunta di un passo comporta l'aggiunta dei dati che gli competono, che possono essere di tre tipologie. Compiuta l'aggiunta dei dati, sarà possibile imporre dei vincoli su questi dati, al fine di determinare se l'utente gli ha inseriti rispettandoli. In questa fase è inoltre possibile inserire un vincolo geografico (coordinate GPS). Attuato questo flusso di comandi il passo potrà essere avviato oppure annullato a discrezione del process owner.

5.1.4 Gestione passi

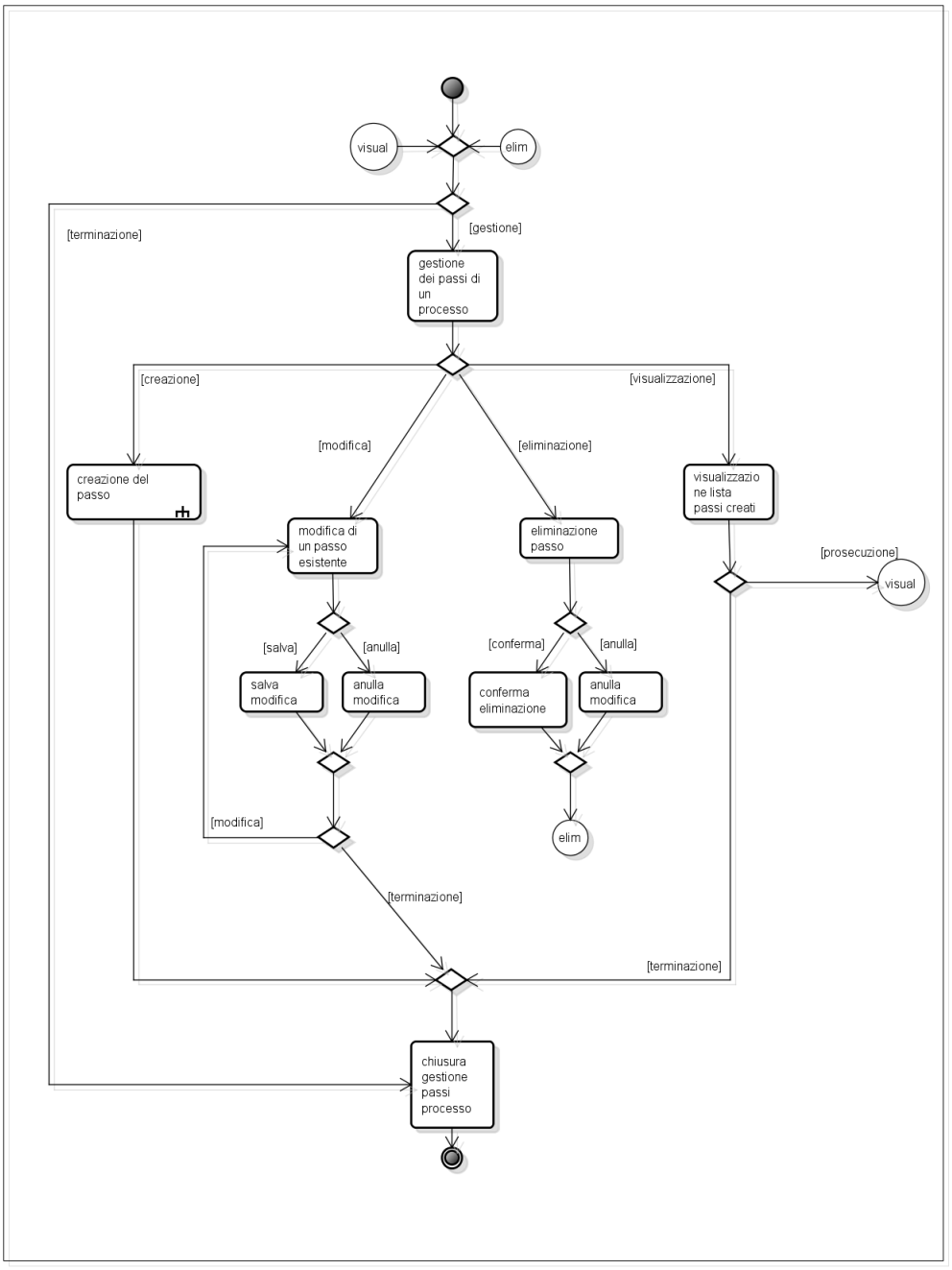

Figura 14: Attività process owner: gestione passi.

Descrizione: La gestione dei passi di un processo si dirama in 4 possibili scelte: la creazione di un nuovo passo, la modifica di un passo esistente, l'eliminazione di un passo e la visualizzazione dei passi creati. Per quanto concerne la modifica e l'eliminazione di un passo l'utente potrà scegliere se annullare o apportare

effettivamente le modifiche/eliminazione.

5.2 Diagrammi di attività: standard user

5.2.1 Registrazione

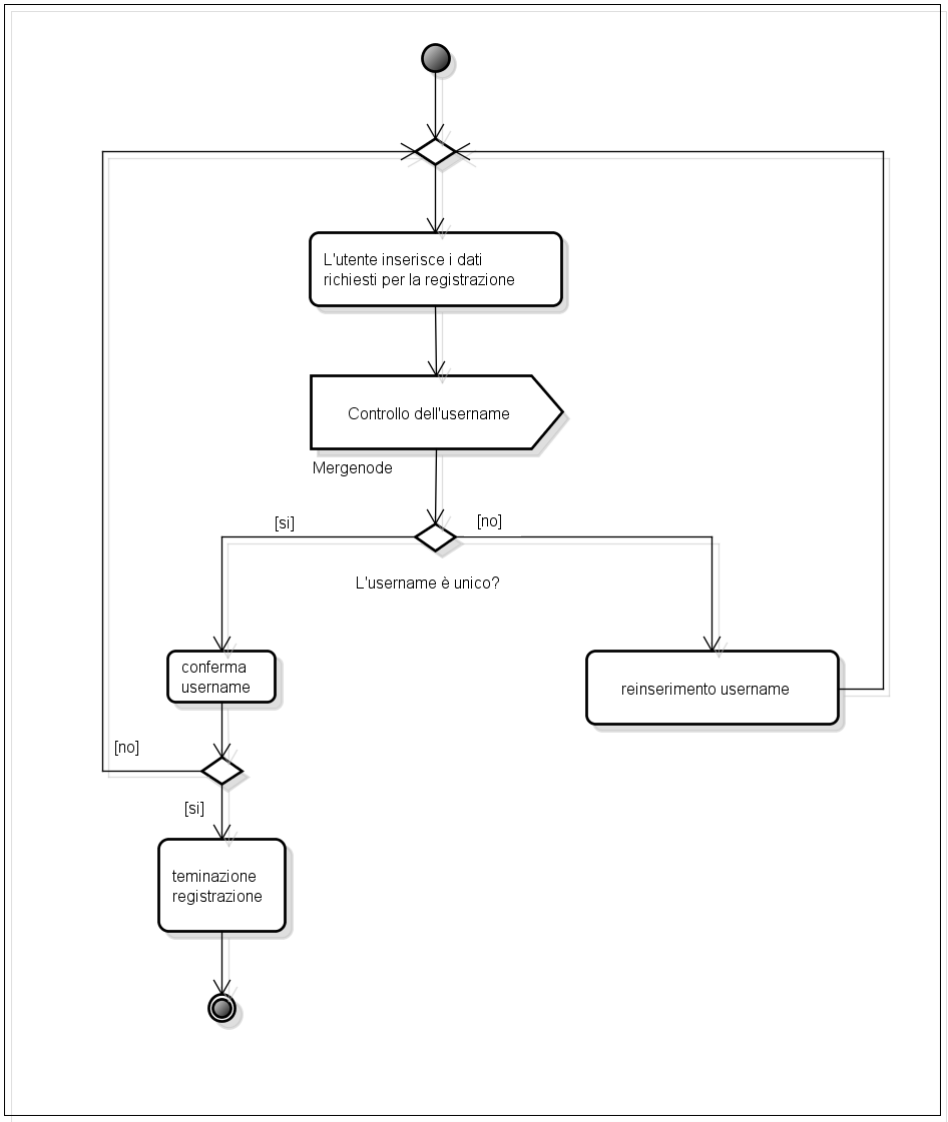

Figura 15: Attività user: Registrazione

Descrizione: L'utente inserisce i dati richiesti per la registrazione, se l'username scelto è unico, allora i dati vengono salvati, l'utente è registrato e può autenticarsi, in caso contrario viene richiesto di inserire un nuovo username.

5.2.2 Login

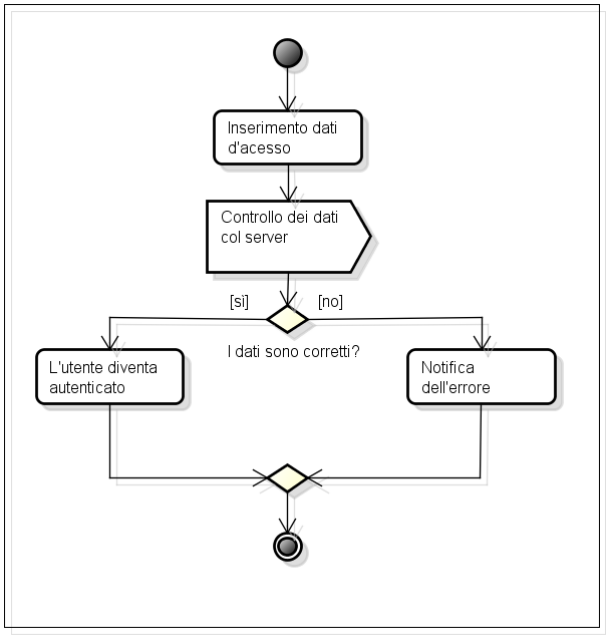

Figura 16: Attività user: Login

Descrizione: L'utente non autenticato inserisce i suoi dati d'accesso, se sono corretti, l'utente viene autenticato, altrimenti gli viene notificato l'errore.

5.2.3 Modifica dati utente

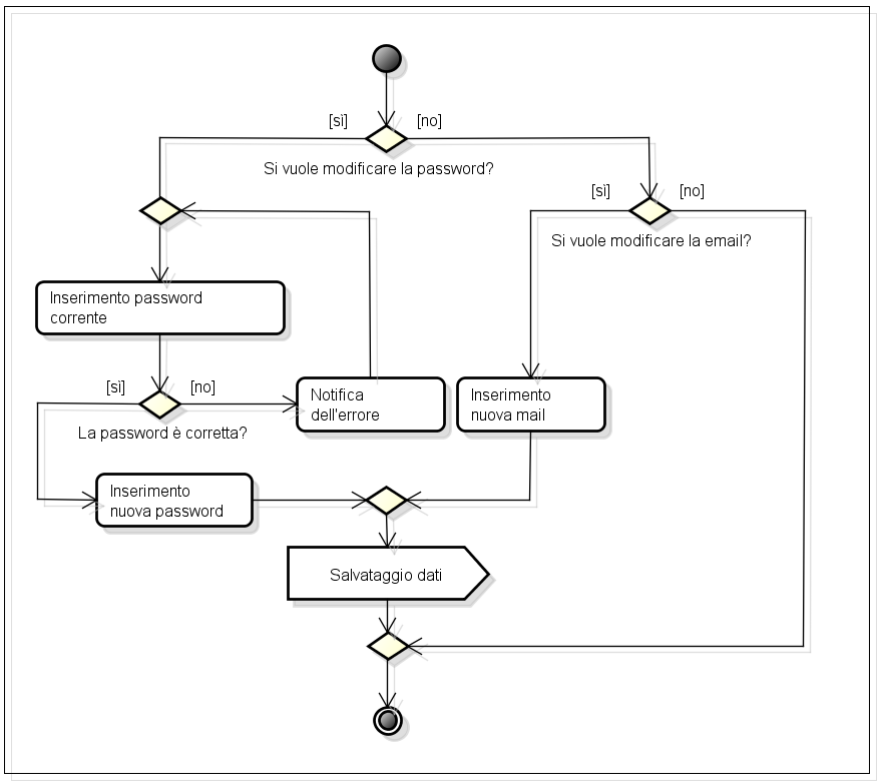

Figura 17: Attività user: Modifica dati utente

Descrizione: I dati che l'utente può modificare una volta reistrato sono la sua password e la sua email. Se l'utente vuole modificare la password gli viene priam richiesta la password corrente, se non è corretta gli viene notificato un errore e la richiesta viene ripetuta, in caso contrario l'utente inserisce una nuova password. Se invece l'utente vuole modificare la sua email, gli viene semplicemnte richiesta una nuova mail. In caso di modifica di password o email i dai vengono risalvati sul server.

5.2.4 Gestione dei processi

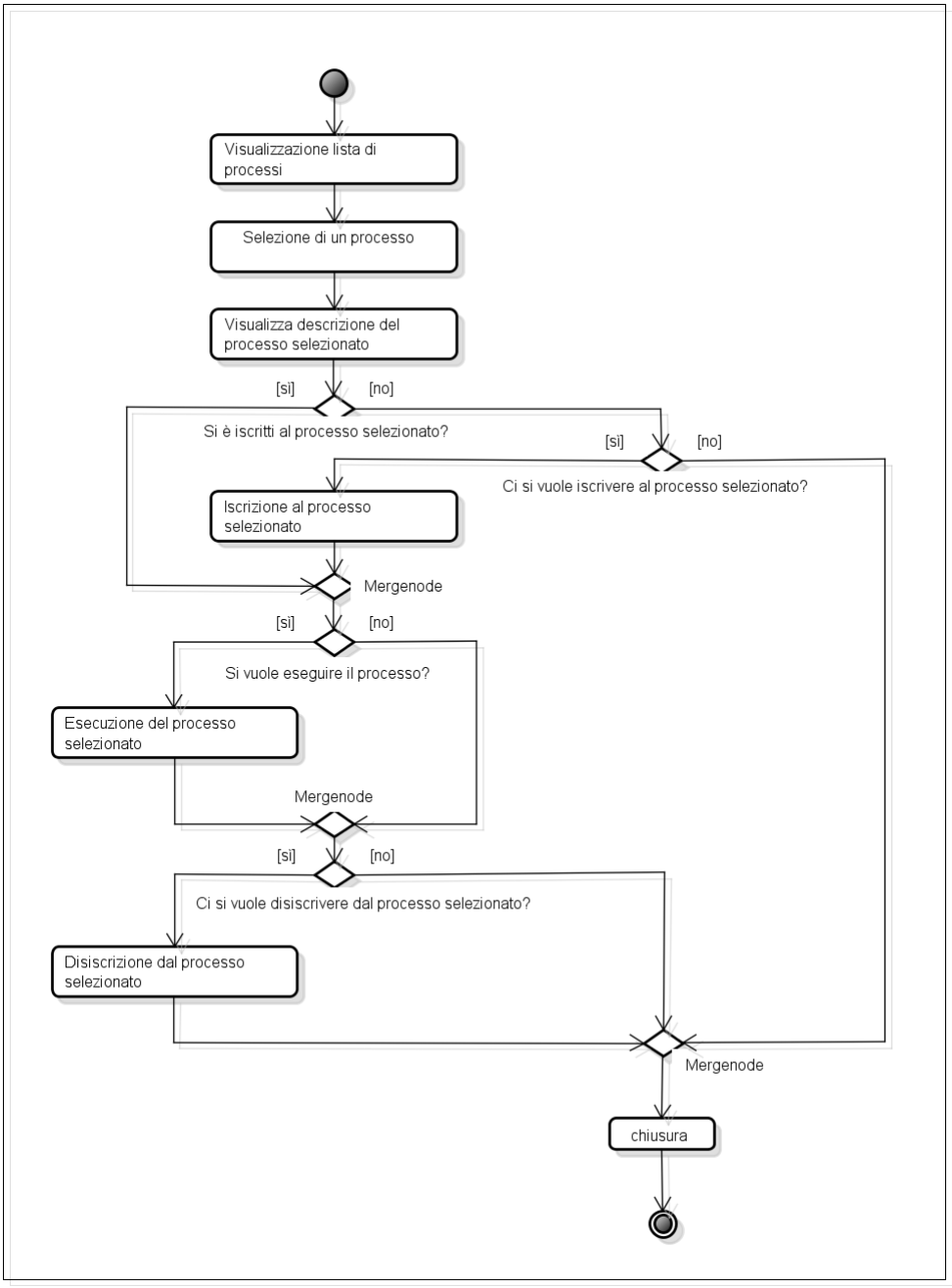

Figura 18: Attività user: Gestione dei processi

Descrizione: Il sistema dopo aver ricevuto dal server i dati sui processi che l'utente può gestire, ne visualizza una lista, l'utente seleziona un processo dalla lista di cui riceve successivamente la descrizione. Se l'utente è iscritto al processo selezionato può eseguire il processo e/o pùò disiscriversi da questo processo. Se non è iscritto invece

può decidere di iscriversi, e una volta iscritto gli vengono offerte le stesse attività descritte nel caso precedente.

5.2.5 Esecuzione di un processo

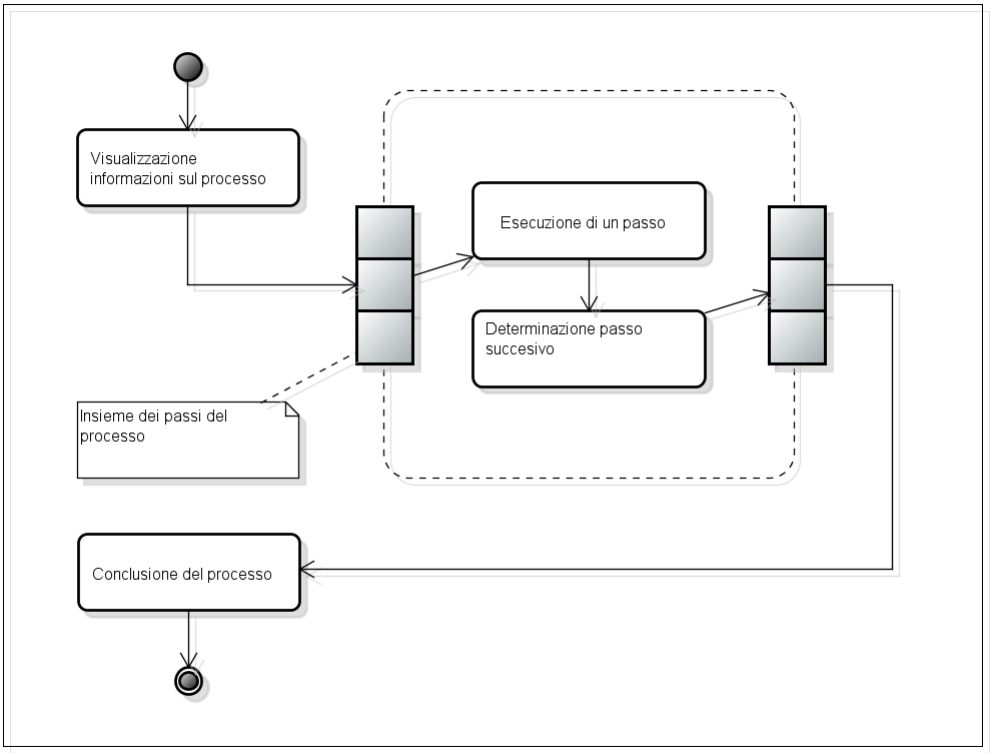

Figura 19: Attività user: Esecuzione di un processo

Descrizione: All'utente vengono visualizzate le informazioni sul processo in esecuzione, dopodichè, per ogni passo del processo, l'utente segue il passo (si veda il diagramma delle attività Esecuzione di un passo per i dettagli), e il sistema determina il passo succesivo. Infine, al terminie dei passi che il sistema ha determinato da eseguire, il processo viene concluso (si veda il digramma delle attività Conclusione di un processo per i dettagli).

5.2.6 Conclusione di un processo

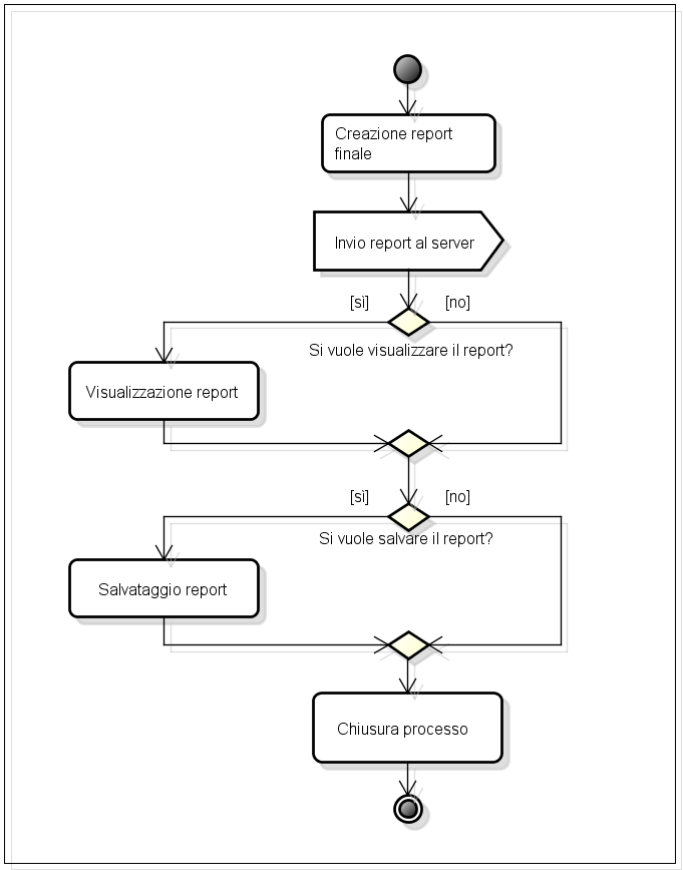

Figura 20: Attività user: conclusione di un processo

Descrizione: Il sistema genera un report sui passi eseguiti e sui dati raccolti, questo report viene inviato al server. Successivamente l'utente può scegliere se visualizzare il report e se salvarne una copia sul proprio dispositivo. Infine il processo viene chiuso.

5.2.7 Esecuzione di un passo

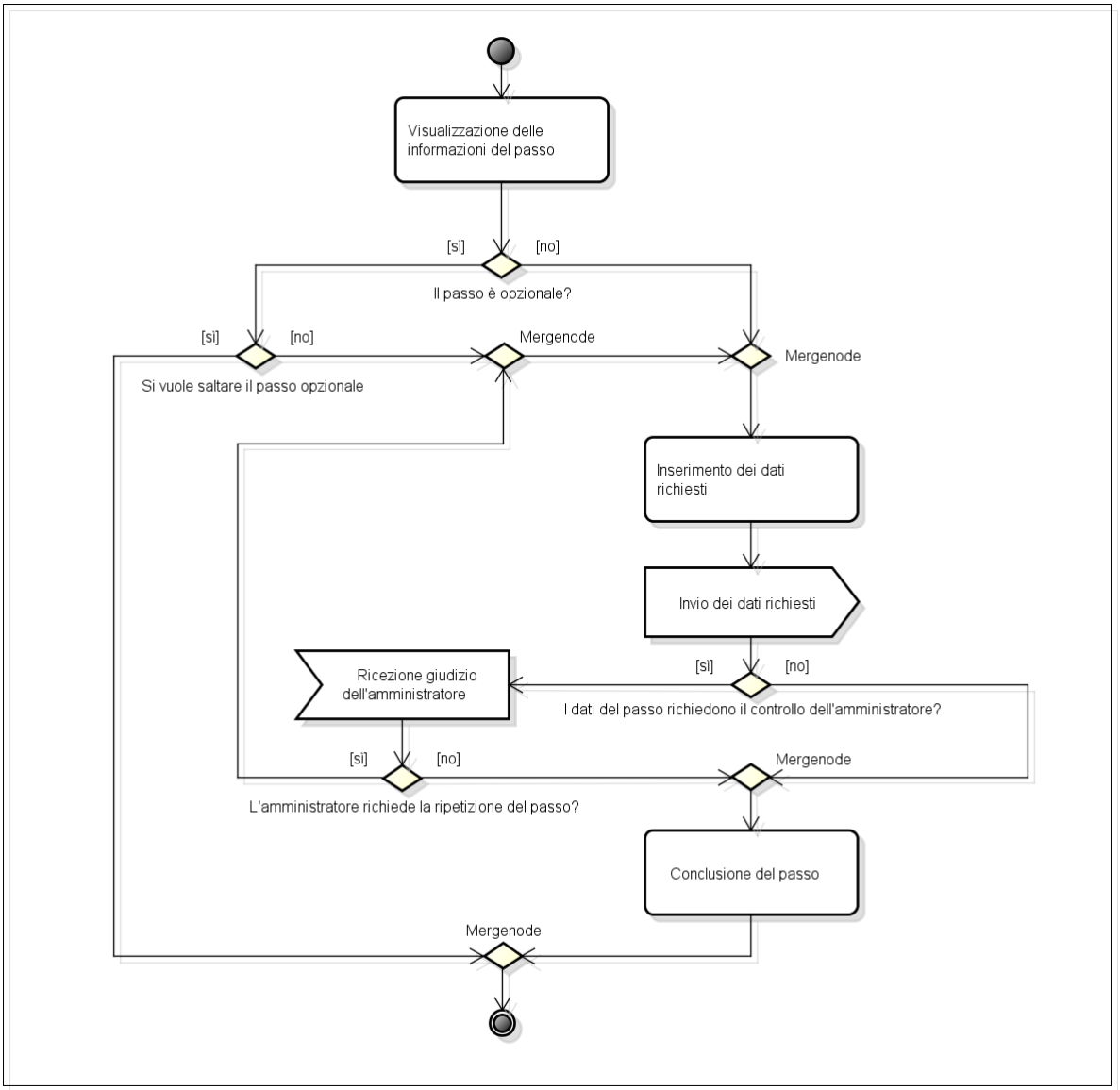

Figura 21: Attività user: Esecuzione di un passo

Descrizione: All'utente vengono visualizzate le informazioni sul passo, poi se il passo è opzionale l'utente può decidere di saltarlo. Nel caso il passo non sia opzionale o che l'utente non voglia saltarlo, l'utente inserisce i dati richisti per il completamento del passo, i quali vengono successivamente inviati al server. Se i dati inviati richiedono il controllo dell'amministratore, il passo non può essere completato fino alla ricezione del suo giudizio che può richiedere di ripetere l'esecuzione del passo. Nel caso che i dati soddisfino il l'amministratore o non fosse richiesto il controllo, il passo viene concluso.

6 Tracciamento

6.1 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::processowner	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::process-owner::logic	CPA1 - `MainLogic` CPA2 - `LoginLogic` CPA3 - `NewProcessLogic` CPA4 - `AddStepLogic` CPA5 - `ManagementProcessLogic` CPA6 - `CheckStepLogic` CPA7 - `StatisticLogic` CPA8 - `UserDataLogic` CPA9 - `InviteUserLogic`
sequenziatore::client::presenter::server-communication	CP1 - `HttpCommunication` CP2 - `WebsocketCommunication` CP3 - `JSONFormatter`
sequenziatore::client::model	CM1 - `Process` CM2 - `Step`
sequenziatore::client::model::localdata_user	CMU3 - `UserData`
sequenziatore::client::model::localdata_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::process-owner	SPA1 - `Login` SPA2 - `ProcessManager` SPA3 - `StepManager` SPA4 - `UserManager` SPA5 - `report`

sequenziatore::server::model::daouser	SMU1 - `DataAccessObject` SMU2 - `ObjectTransfer`
sequenziatore::server::model::daoprocess-owner	SMA1 - `DataAccessObject` SMA2 - `ObjectTransfer`
sequenziatore::server::model::daoprocess	SM1 - `DataAccessObject` SM2 - `ObjectTransfer`
sequenziatore::server::model::daostep	SM3 - `DataAccessObject` SM4 - `ObjectTransfer`

Tabella 1: Tabella package/componenti

6.2 Tracciamento requisiti - componenti

Requisito	**Descrizione**	**Componente**
FOBU 1	Il sistema dovrà permettere all'utente di registrarsi	VU4, CPU3, CP1, CP3, SP1, SP3, SPU1, SMU1, SMU2
FOBU 1.1	L'utente dovrà inserire un *username* che lo identifichi univocamente all'interno del sistema	VU4, CPU3
FOBU 1.1.1	L'utente dovrà inserire un *username* composto da almeno 6 caratteri	VU4, CPU3
FOBU 1.2	L'utente dovrà inserire una *password* d'accesso	VU4, CPU3
FOBU 1.2.1	L'utente dovrà inserire una *password* 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 *email*	VU4, CPU3
FDEU 1.6.1	La *email* inserita dovrà corrispondere ad un indirizzo di posta elettronica esistente	VU4, CPU3
FOBU 2	Il sistema dovrà permettere all'utente di autenticarsi	VU3, CPU2, CP1, CP3, SP1, SP3, 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 $server_G$	VU3, CPU2
FOBU 2.2	L'utente dovrà inserire il proprio *username* per autenticarsi	VU3, CPU2
FOBU 2.3	L'utente dovrà inserire la propria *password* per autenticarsi	VU3, CPU2
FOPL 3	Il sistema dovrà permettere all'utente autenticato di gestire le proprie credenziali	VU5, VU6, CPU4, CP1, CP3, SP1, SP3, SPU1, SMU1, SMU2
FOPL 3.1	L'utente autenticato potrà visualizzare le proprie credenziali	VU5, CPU4, SPU1, CP1, CP3, SP1, SP3, SMU1, SMU2
FOPL 3.1.1	L'utente autenticato visualizzerà il proprio *username*	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 *email*	VU5, CPU4

FOPL 3.2	Il sistema dovrà permettere all'utente autenticato di modificare i propri dati	VU6, CPU4, SPU1, CP1, CP3, SP1, SP3, SMU1, SMU2
FOPL 3.2.1	Il sistema dovrà permettere all'utente autenticato di modificare la propria *password*	VU6, VU7, CPU4, SPU1, SMU1, SMU2
FOPL 3.2.1.1	L'utente autenticato potrà inserire la nuova *password*	VU6, VU7, CPU4
FOPL 3.2.1.2	L'utente autenticato potrà inserire la *password* corrente	VU6, VU7, CPU4
FOPL 3.2.1.3	Il sistema dovrà comunicare all'utente autenticato se la *password* 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, CP1, CP2, CP3, SP1, SP2, SP3, 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, CP1, CP2, CP3, SP1, SP2, SP3, SM1, SM2
FOBL 4.1.1	Il sistema dovrà permettere all'utente autenticato di selezionare ed aprire una lista di processi	VU8, CPU5, SPU2, CP1, CP2, CP3, SP1, SP2, SP3, 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, CP1, CP2, CP3, SP1, SP2, SP3, 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, CP1, CP2, CP3, SP1, SP2, SP3, 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, CP1, CP2, CP3, SP1, SP2, SP3, 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, CP1, CP2, CP3, SP1, SP2, SP3, 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, SPU3, CMU3
FOBL 4.4.4.2.1	L'utente autenticato potrà visualizzare se il passo in esecuzione richiede l'approvazione del *process owner*$_G$ 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	VU10, CPU6, SPU2, SM3, SM4
FOBL 4.4.4.3.1	Il sistema dovrà permettere all'utente autenticato l'inserimento di una immagine richiesta	VU14

VU15, CPU6		
FDEL 4.4.4.3.1.1	Il sistema dovrà permettere all'utente autenticato di scattare una foto per inserire l'immagine richiesta	VU15, CPU6
FOBL 4.4.4.3.1.2	Il sistema dovrà permettere all'utente autenticato di inserire una immagine caricandola dai suoi file	VU14, CPU6
FOBL 4.4.4.3.2	L'utente può inserire dati testuali richiesti dal passo in esecuzione	VU11, CPU6
FOBL 4.4.4.3.3	Il sistema dovrà permettere all'utente autenticato di inserire dati numerici richiesti dal passo in esecuzione	VU12, CPU6
FOBL 4.4.4.4	L'utente autenticato potrà inviare al sistema i dati richiesti per l'esecuzione del passo in corso	VU10, CPU6, CP1, CP2, CP3, SP1, SP2, SP3, SPU2, SM3, SM4
FOBL 4.4.4.4.1	L'utente autenticato potrà inviare al sistema i dati dati testuali inseriti	VU11, CPU6, CP1, CP2, CP3, SP1, SP2, SP3, SM3, SM4
FOBL 4.4.4.4.2	L'utente autenticato potrà inviare al sistema le immagini inserite	VU15, VU14, CPU6, CP1, CP2, CP3, SP1, SP2, SP3, SM3, SM4
FOBL 4.4.4.4.3	L'utente autenticato potrà inviare al sistema i dati numerici inseriti	VU12, CPU6, CP1, CP2, CP3, SP1, SP2, SP3, SM3, SM4
FOBL 4.4.4.4.4	L'utente autenticato potrà inviare al sistema le coordinate della sua posizione	VU13, CPU6, CP1, CP2, CP3, SP1, SP2, SP3, SM3, SM4
FOBL 4.4.4.4.5	L'utente autenticato potrà inviare al sistema la data e ora al momento della richiesta di invio dati	VU10, CPU6, CP1, CP2, CP3, SP1, SP2, SP3, SM3, SM4

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 SPU3, CMU3	VU10, CPU6
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	VU10, CPU6, CMU3
FOPL 4.4.4.4.6.2	Il sistema, in presenza di connessione, dovrà permettere all'utente autenticato di inviare i dati precedentemente salvati	VU10, CPU6, CMU3
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	VU9, CPU5, CP2, CP1, CP2, CP3, SP1, SP2, SP3, SPU3, SP3
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 *process owner*$_G$	VU9, CPU5
FOBL 4.4.5	Il sistema dovrà permettere all'utente autenticato di concludere un processo terminato o del quale ha eseguito tutti i passi	VU16, CPU5, SPU2, SM1, SM2
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$_G$	VU16, VU17, VU18, CPU5, SPU4, CP1, CP3, SP1, SP3, SM1, SM2
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	VU16, CPU5, SM1, SM2

FOPL 4.4.6	Il sistema dovrà permettere all'utente autenticato di saltare il passo in esecuzione se facoltativo	VU9, CPU6, SPU3, SM1, SM2, SM3, SM4
FOBL 4.5	Il sistema dovrà permettere all'utente autenticato di disiscriversi da un processo a cui è iscritto	VU9, CPU6, CP1, CP3, SP1, SP3, SPU1, SM1, SM2
FOBL 5	L'utente potrà terminare la propria sessione, diventando utente generico	VU9, CPU6, SM1, SM2
FOBA 1	Il sistema dovrà permettere al *process owner*$_G$ la creazione di processi	VA4, CPA4, CM1, CM2, SPA2, SM1
FOBA 1.1	Il *process owner*$_G$ dovrà inserire un nome che identifichi univocamente il processo che vuole creare	VA4, CPA4, CM1, SM1, SM2
FOBA 1.2	Il *process owner*$_G$ dovrà inserire la descrizione del processo che vuole creare	VA4, CPA4, CM1, SM1, SM2
FOBA 1.3	Il sistema dovrà permettere al *process owner*$_G$ di definire i criteri di terminazione di un processo durante la sua creazione	VA4, CPA4, CM1
FOBA 1.3.1	Il *process owner*$_G$ dovrà inserire il numero massimo di completamenti del processo in creazione	VA4, CPA4, CM1, SM1, SM2
FOBA 1.3.2	Il *process owner*$_G$ potrà inserire la data di terminazione del processo in creazione	VA4, CPA4, CM1, SM1, SM2
FOBA 1.4	Il sistema dovrà permettere al *process owner*$_G$ di gestire i passi del processo in creazione	VA4, VA6, CPA4, CM1, SM1, SM2, CM2, SM3, SM4
FOBA 1.4.1	Il sistema dovrà permettere al *process owner*$_G$ di creare un passo del processo in creazione	VA5, CPA4, CM2
FOBA 1.4.1.1	Il *process owner*$_G$ dovrà inserire la descrizione del passo in creazione	VA5, CPA4, CM2
FOBA 1.4.1.2	Il sistema dovrà permettere al *process owner*$_G$ di inserire uno o più dati al passo in creazione	VA5, CPA4, CM2

FOBA 1.4.1.2.1	Il *process owner*$_G$ potrà inserire un nome al dato che vuole aggiungere al passo in creazione	VA5, CPA4, CM2
FOBA 1.4.1.2.2	Il *process owner*$_G$ dovrà scegliere il tipo del dato che vuole aggiungere al passo in creazione	VA5, CPA4, CM2
FOBA 1.4.1.2.2.1	Il *process owner*$_G$ potrà scegliere un dato testuale come tipo del dato aggiunto al passo in creazione	VA5, CPA4, CM2
FOBA 1.4.1.2.2.2	Il *process owner*$_G$ potrà scegliere un dato numerico come tipo del dato aggiunto al passo in creazione	VA5, CPA4, CM2
FOBA 1.4.1.2.2.3	Il *process owner*$_G$ potrà scegliere un'immagine come tipo del dato aggiunto al passo in creazione	VA5, CPA4, CM2
FOBA 1.4.1.3	Il sistema dovrà permettere al *process owner*$_G$ di definire uno o più criteri di superamento del passo in creazione	VA5, CPA4, CM2
FOBA 1.4.1.3.1	Per ogni criterio di superamento, il *process owner*$_G$ dovrà definire una o più condizioni di avanzamento	VA5, CPA4, CM2
FDEA 1.4.1.3.1.1	Per ogni criterio di superamento, il *process owner*$_G$ potrà scegliere se i dati ricevuti dall'utente richiederanno il suo controllo per concludere il passo in creazione	VA5, CPA4, CM2
FOBA 1.4.1.3.1.2	Per ogni criterio di superamento, il *process owner*$_G$ potrà inserire un vincolo sulla posizione geografica dell'utente al momento dell'invio dei dati	VA5, CPA4, CM2
FOBA 1.4.1.3.1.2.1	Il sistema dovrà permettere al *process owner*$_G$ di stabilire una precisa posizione geografica	VA5, CPA4, CM2

FOPA 1.4.1.3.1.2.2	Il *process owner*$_G$ potrà inserire un raggio di tolleranza rispetto alla posizione geografica inserita durante la definizione delle condizioni di avanzamento di un passo	VA5, CPA4, CM2
FOBA 1.4.1.3.1.3	Per ogni criterio di superamento, il *process owner*$_G$ potrà stabilire uno o più intervalli temporali in cui l'utente può inviare i dati richiesti	VA5, CPA4, CM2
FDEA 1.4.1.3.1.4	Per ogni criterio di superamento, il *process owner*$_G$ potrà inserire dei vincoli sui dati numerici presenti nel passo in creazione	VA5, CPA4, CM2
FOPA 1.4.1.3.1.4.1	Il *process owner*$_G$ potrà stabilire un numero minimo e massimo di cifre durante la definizione dei vincoli su un dato numerico	VA5, CPA4, CM2
FDEA 1.4.1.3.1.4.2	Il *process owner*$_G$, durante la definizione dei vincoli su un dato numerico, potrà stabilire se tale numero potrà contenere cifre decimali	VA5, CPA4, CM2
FOPA 1.4.1.3.1.4.3	Il *process owner*$_G$, durante la definizione dei vincoli su un dato numerico, potrà stabilire un limite superiore e inferiore per tale numero	VA5, CPA4, CM2
FOPA 1.4.1.3.1.5	Il *process owner*$_G$ potrà stabilire la facoltatività dell'esecuzione di un passo	VA5, CPA4, CM2
FOBA 1.4.1.3.2	Il sistema dovrà permettere al *process owner*$_G$ di scegliere il passo eseguibile dall'utente una volta soddisfatto il criterio di superamento in definizione	VA5, CPA4, CM2
FOBA 1.4.2	Il *process owner*$_G$ potrà visualizzare la lista dei passi creati durante la creazione di un nuovo processo	VA5, CPA4, CM2

FDEA 1.4.3	Il *process owner*$_G$, durante la creazione di un nuovo processo, potrà modificare un passo esistente	VA5, CPA4, CM2
FDEA 1.4.3.1	Il sistema dovrà permettere al *process owner*$_G$ di modificare la descrizione di un passo di un processo in creazione	VA5, CPA4, CM2
FDEA 1.4.3.2	Il sistema dovrà permettere al *process owner*$_G$ di modificare la descrizione dei dati di un passo di un processo in creazione	VA5, CPA4, CM2
FDEA 1.4.3.3	Il sistema dovrà permettere al *process owner*$_G$ di modificare i criteri di superamento dei passi del processo in creazione	VA5, CPA4, CM2
FDEA 1.4.3.3.1	Il sistema dovrà permettere al *process owner*$_G$ di modificare le condizioni di avanzamento dei passi del processo in creazione	VA5, CPA4, CM2
FDEA 1.4.3.3.1.1	Il sistema dovrà permettere al *process owner*$_G$ di modificare i vincoli sull'approvazione dei passi del processo in creazione	VA5, CPA4, CM2
FDEA 1.4.3.3.1.2	Il sistema dovrà permettere al *process owner*$_G$ di modificare i vincoli dei passi del processo in creazione, relativi alla posizione dell'utente al momento dell'invio dei dati	VA5, CPA4, CM2
FDEA 1.4.3.3.1.3	Il sistema dovrà permettere al *process owner*$_G$ di modificare gli intervalli temporali in cui l'utente potrà inviare i dati, stabiliti nei passi del processo in creazione	VA5, CPA4, CM2
FDEA 1.4.3.3.1.4	Il sistema dovrà permettere al *process owner*$_G$ di modificare i vincoli sui dati numerici dei passi del processo in creazione	VA5, CPA4, CM2

FOPA 1.4.3.3.1.5	Il sistema dovrà permettere al *process owner*$_G$ di modificare le impostazioni sulla facoltatività dei passi del processo in creazione	VA5, CPA4, CM2
FDEA 1.4.3.3.2	Il sistema dovrà permettere al *process owner*$_G$ di sostituire il passo eseguibile al soddisfacimento dei criteri di superamento dei passi del processo in creazione	VA5, CPA4, CM2
FDEA 1.4.4	Il sistema dovrà permettere al *process owner*$_G$ di eliminare un passo del processo in creazione	VA5, CPA4, CM2
FOBA 1.5	Il sistema dovrà permettere al *process owner*$_G$ di avviare un processo in creazione che contiene almeno un passo	VA4, SPA2, SM1, SM2, CP1, CP2, CP3, SP1, SP2, SP3
FDEA 2	Il sistema dovrà permettere al *process owner*$_G$ la gestione dei processi creati	VA8, VA11, CPA5, CPA7, SM1, SM2
FDEA 2.1	Il sistema dovrà permettere al *process owner*$_G$ di scegliere un processo avviato	VA7, VA8, VA11, CPA5, CPA7, SM1, SM2, CP1, CP3, SP1, SP3
FOPA 2.1.2	Il sistema dovrà permettere al *process owner*$_G$ di ricercare un processo inserendone il nome	VA7, VA8, VA11, CPA5, CPA7, SM1, SM2, CP1, CP3, SP1, SP3
FDEA 2.1.3	Il sistema dovrà permettere al *process owner*$_G$ di selezionare un processo da gestire	VA7, VA8, VA11, CPA5, CPA7, SM1, SM2
FOPA 2.2	Il sistema dovrà permettere al *process owner*$_G$ di selezionare gli utenti a cui permettere l'iscrizione al processo gestito	VA12, CPA9, SM1, SMU1, SMU2

FOPA 2.2.1	Il *process owner*$_G$ potrà visualizzare la lista degli utenti registrati al sistema	VA12, CPA9, SM1, SMU1, SMU2, CP1, CP2, SP1, SP3
FOPA 2.2.2	Il sistema dovrà permettere al *process owner*$_G$ di selezionare dalla lista gli utenti a cui consentire l'iscrizione al processo gestito	VA12, CPA9, SM1, SMU1, SMU2
FDEA 2.3	Il sistema dovrà permettere al *process owner*$_G$ di consultare informazioni sul processo gestito	VA8, VA11, CPA5, CPA7, SM1, SM2
FOPA 2.3.1	Il sistema dovrà permettere al *process owner*$_G$ di recuperare informazioni sul processo gestito	VA8, VA11, CPA5, CPA7, SM1, SM2, CP1, CP3, SP1, SP3
FOPA 2.3.1.1	Il sistema dovrà permettere al *process owner*$_G$ di visualizzare la descrizione del processo gestito	VA11, CPA7, SM1, SM2
FOPA 2.3.1.2	Il sistema dovrà permettere al *process owner*$_G$ di visualizzare i criteri di terminazione del processo gestito	VA11, CPA7, SM1, SM2
FOPA 2.3.1.3	Il sistema dovrà permettere al *process owner*$_G$ di visualizzare i dati dei passi del processo gestito	VA11, CPA7, SM1, SM2, CP1, CP3, SP1, SP3
FOPA 2.3.1.4	Il sistema dovrà permettere al *process owner*$_G$ di visualizzare le condizioni di superamento dei passi del processo gestito	VA11, CPA7, SM1, SM2
FDEA 2.3.2	Il sistema dovrà permettere al *process owner*$_G$ di visualizzare lo stato dell'esecuzione del processo	VA11, CPA7, SM1, SM2, CP1, CP3, SP1, SP3
FDEA 2.3.2.1	Il sistema dovrà permettere al *process owner*$_G$ di visualizzare il numero di utenti iscritti al processo gestito	VA11, CPA7, SM1, SM2

FDEA 2.3.2.2	Il sistema dovrà permettere al *process owner*$_G$ di visualizzare il numero di completamenti del processo gestito	VA11, CPA7, SM1, SM2
FDEA 2.3.3	Il sistema dovrà permettere al *process owner*$_G$ di visualizzare i dati inviati dagli utenti che hanno comportato il superamento di un passo del processo gestito	VA11, CPA7, CM1, CM2, SM1, SM2, SM3, SM4, CP1, CP3, SP1, SP3
FDEA 2.4	Il sistema dovrà permettere al *process owner*$_G$ di controllare i dati inviati dagli utenti che richiedono la sua approvazione	VA9, CPA5, SM1, SM2
FOBA 2.4.1	Il sistema dovrà permettere al *process owner*$_G$ di visualizzare i dati inviati dagli utenti che richiedono la sua approvazione	VA9, CPA5, SM1, SM2, CP1, CP3, SP1, SP3
FDEA 2.4.2	Il sistema dovrà permettere al *process owner*$_G$ di approvare i dati controllati	VA9, CPA5, SM1, SM2
FDEA 2.4.3	Il sistema dovrà permettere al *process owner*$_G$ di respingere i dati controllati	VA9, CPA5, SM1, SM2
FDEA 2.4.4	Il sistema dovrà inviare l'esito del controllo agli utenti che hanno inviato dei dati che richiedono approvazione	VA9, CPA5, SM1, SM2, CP1, CP2, CP3, SP1, SP2, SP3
FDEA 2.5	Il sistema dovrà permettere al *process owner*$_G$ di terminare un processo avviato	VA8, CPA5, SM1, SM2, CP1, CP2, CP3, SP1, SP2, SP3
FDEA 2.6	Il sistema dovrà permettere al *process owner*$_G$ di eliminare un processo terminato dall'insieme dei processi creati	VA8, CPA5, SM1, SM2
FOBL 3	Il *process owner*$_G$ potrà terminare la propria sessione, diventando utente generico	VA3, CPA2, SMA1, SMA2

Tabella 2: Tabella requisiti-Componenti

A Tecnologie utilizzate

A.1 HTML5

HTML5$_G$, richiesto espressamente dal proponente all' interno del capitolato d'appalto, verrà utilizzato per la struttura base della pagine, inoltre ci permetterà di utilizzare il controllo della geo localizzazione, fondamentale nello sviluppo del nostro sistema, oltre alle altre novità che introduce.

A.2 CSS3

CSS3 è un linguaggio *style sheet* verrà utilizzato per rappresentare la struttura della presentazione in modo da mantenerla separata dai contenuti delle varie pagine, in questo modo verranno migliorate comprensione, manutenibilità e portabilità.

A.3 Javascript

L' utilizzo di Javascript è stato richiesto espressamente dal proponente, è un linguaggio di *scripting* lato client non compilato ma interpretato direttamente dal browser. Nello sviluppo del nostro progetto ci permetterà di non ricaricare la pagina ad ogni modifica degli utenti, e gestirà anche la comunicazione, quando richiesto, con il lato server per ricevere dati di cui può necessitare.

A.4 JAVA 7

Java è un linguaggio orientato agli oggetti che permette di essere quanto più indipendenti possibili dalla piattaforma di esecuzione. Nello sviluppo del nostro sistema verrà utilizzato nella la creazione del *back end*, in particolare per la creazione delle *Servlet*.

A.5 JSON

JavaScript Object Notation è il formato scelto pe lo scambio dati tra client e server, è molto facile da utilizzare e si integra bene con la programmazione in AJAX e il suo uso con Javascript è semplice infatti il *parsing* di tale tipo di dato viene effettuato con la semplice chiamata ad un metodo.

A.6 JDBC

Java DataBase Connectivity è un connettore per database in grado di consentire l'accesso alle basi di dati da un programma scritto in Java. Fornisce i metodi per interrogare e modificare i dati nella base di dati.

A.7 JQueryMobile

Questo *framework* verrà usato per lo sviluppo di un front end per dispositivi di tipo *responsive* accessibili da *smartphne*, *tablet* e computer. La scelta del team di questo *framework* è data dal fatto che sembra una tecnologia affermata nel mondo del *web development* e dal semplice utilizzo, inoltre permette la scrittura di meno righe di codice rispetto a Javascript puro.

A.8 MySQL

MySQL è un Relational database management system(RDBMS), il team ha scelto questo tipo di base di dati in quanto di semplice utilizzo e già utilizzata da tutti i membri del gruppo.

A.9 Apache Tomcat

Apache Tomcat è un contenitore servlet *open source* che offre una piattaforma per l'esecuzione di applicazioni web sviluppate in java. La versione 4.x comprende Catalina e Coyote, ripsettivamente il contenitore servlet e il connettore HTTP.