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;
- Developing Backbone.js Applications, Addy Osmani http://addyosmani.github.io/backbone-fundamentals;
- 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.

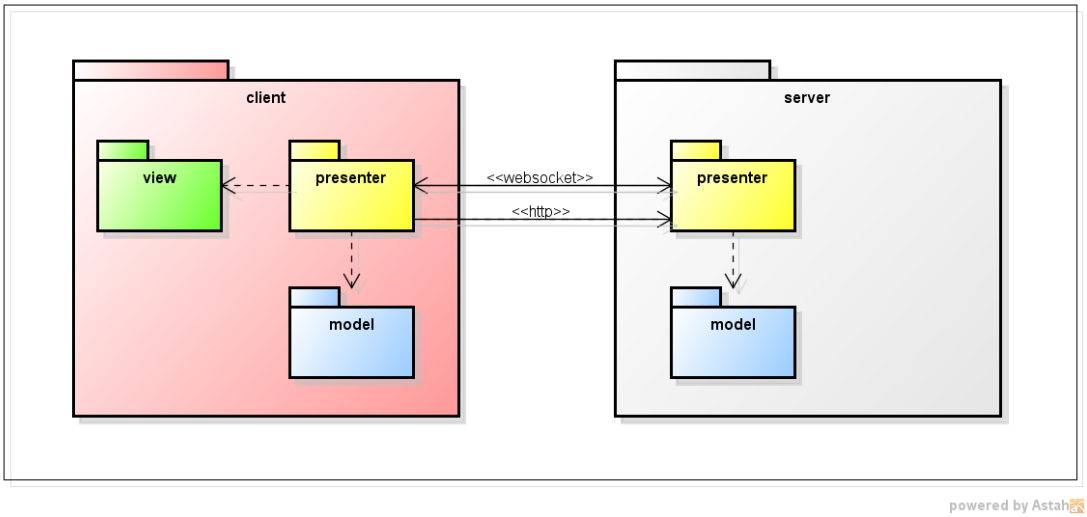


Figura 1: Diagramma UML architettura generale

## 2.3 Diagrammi dei package

Il seguente capitolo descrive le dipendenze intercorse fra i vari package$_G$ del sistema Sequenziatore.
Il sistema Sequenziatore è composto da due macro package:

1. sequenziatore.client: le componenti di questo package realizzano la parte front-end$_G$ del sistema Sequenziatore
2. sequenziatore.server: le componenti di questo package realizzano la parte back-end$_G$ del sistema Sequenziatore

### 2.3.1 Diagrammi del package com.sirius.sequenziatore.client

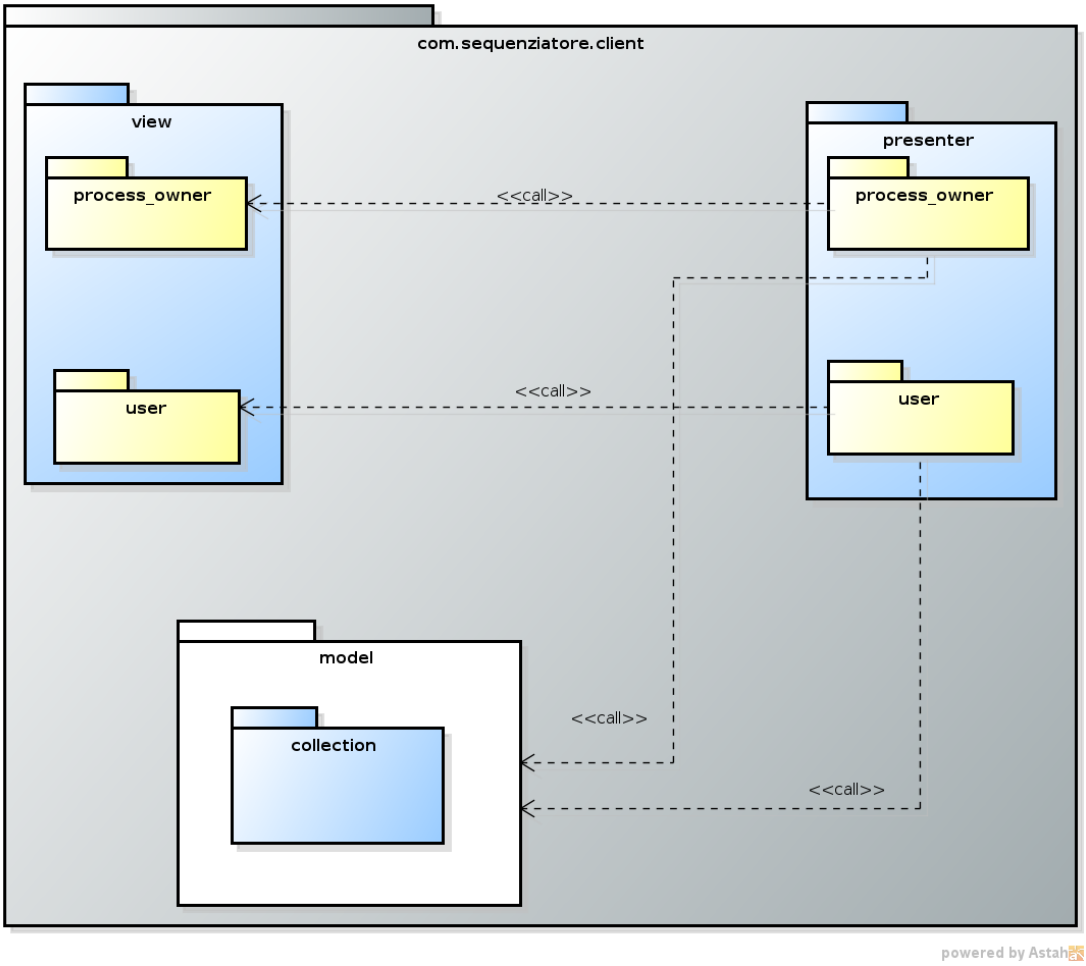


Figura 2: Diagramma package - *com.sirius.sequenziatore.client*

Il package sequenziatore.client è composto dai seguenti package:

- com.sirius.sequenziatore.client.view;

- com.sirius.sequenziatore.client.presenter;
- com.sirius.sequenziatore.client.model.

Come è facilmente intuibile, la struttura del package com.sirius.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.

#### 2.3.1.1 Package com.sirius.sequenziatore.client.view

Il package com.sirius.sequenziatore.client.view è composto da i seguenti package:

- com.sirius.sequenziatore.client.view.processowner: contiene le classi e interfacce necessarie a gestire l'interfaccia grafica e a generare gli eventi della parte grafica dell'utente process owner .
- com.sirius.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.

#### 2.3.1.2 Package com.sirius.sequenziatore.client.presenter

Il package com.sirius.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:

- com.sirius.sequenziatore.client.presenter.processowner: contiene le classi che costituiscono la componente Presenter per l'utente amministratore, il package com.sirius.sequenziatore.client.presenter.processowner permette gestisce gli eventi generati dalle componenti del package com.sirius.sequenziatore.client.view.processowner e aggiorna la parte grafica dell'utente process owner;
- com.sirius.sequenziatore.client.presenter.user: contiene le classi che permettono la gestiscione degli gli eventi generati dalle componenti del package$_g$ com.sirius.sequenziatore.client.view.user e l'aggiornamento della parte grafica per l'utente generico e autenticato.

#### 2.3.1.3 Package com.sirius.sequenziatore.client.model

Il package com.sirius.sequenziatore.client.model contiene tutte le classi della componente Model. Ogni classe del package com.sirius.sequenziatore.client.model possiede gli opportuni metodi per poter comunicare col server e poter accedere, modificare e salvare in modo persistente questi ultimi. Il package è contiene inoltre il package:

- com.sirius.sequenziatore.client.model.collection: è composto da classi che implementano collezioni di classi del package com.sirius.sequenziatore.client.model.

### 2.3.2 Diagrammi del package com.sirius.sequenziatore.server

I package che compongono il package com.sirius.sequenziatore.server sono:

# 3 Descrizione singoli componenti

## 3.1 Package com.sirius.sequenziatore.client

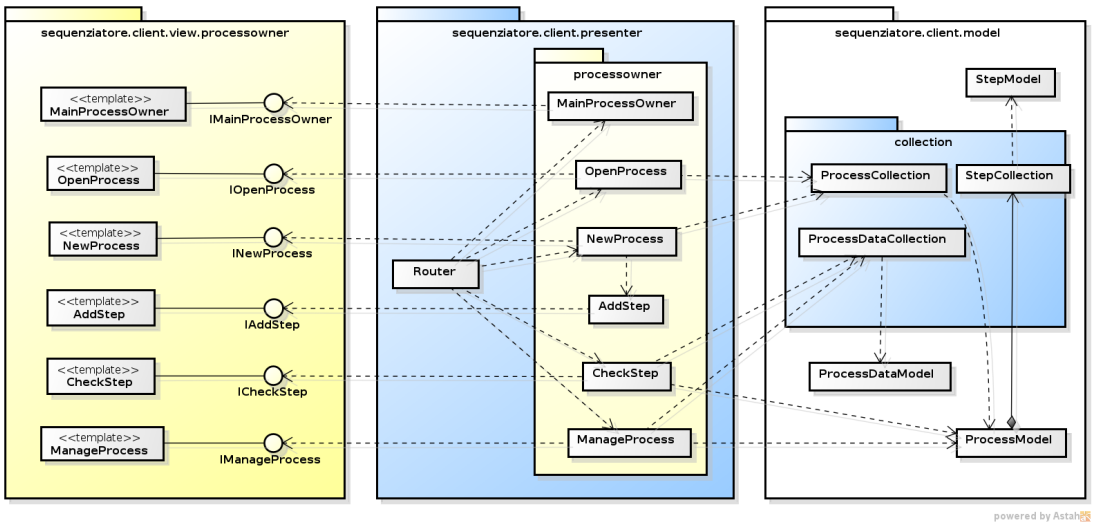


Figura 3: Diagramma componenti - *process owner*

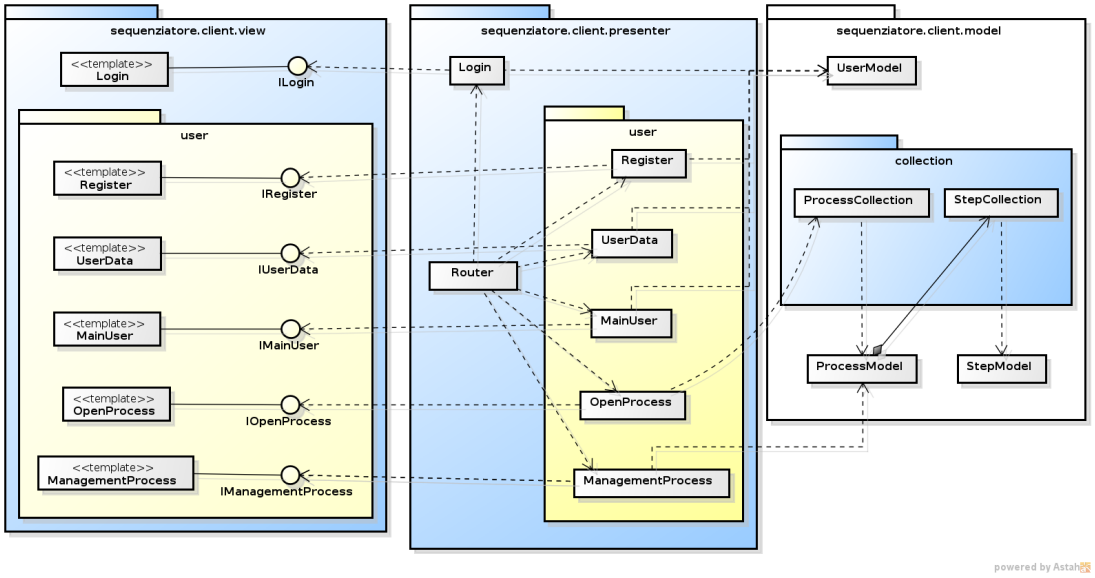


Figura 4: Diagramma componenti - *user*

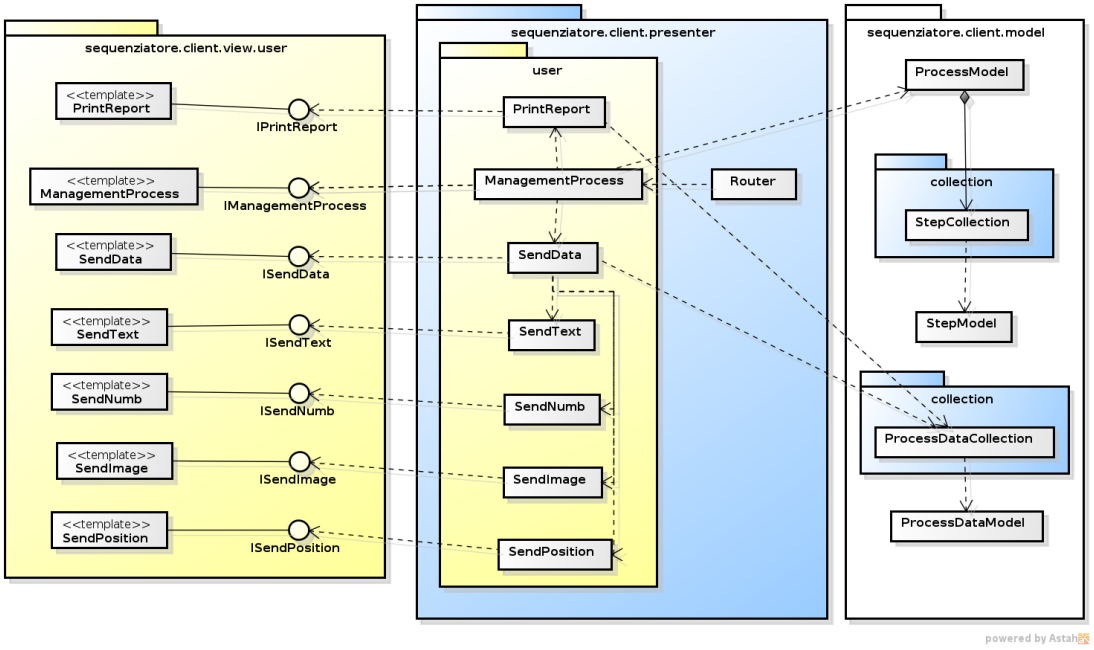


Figura 5: Diagramma componenti bis - *user*

## 3.2 Package com.sirius.sequenziatore.client.view

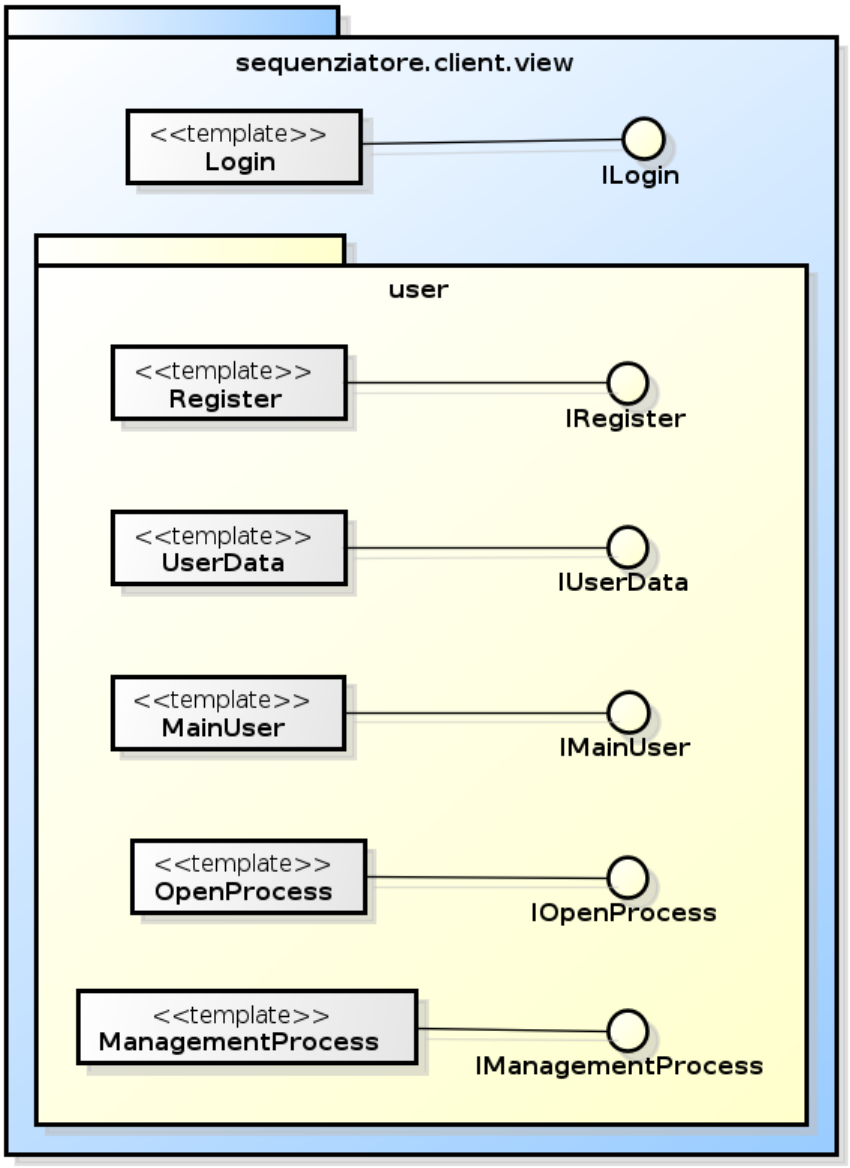


Figura 6: Diagramma principale *view* utente

#### 3.2.0.1 ILogin

- **Nome:** `ILogin`;
- **Package:** `com.sirius.sequenziatore.client.view`;
- **Descrizione:** Interfaccia che permette di gestire l'interfaccia grafica relativa alle richieste di autenticazione al sistema.

#### 3.2.0.2 Login

- **Nome:** `Login`;
- **Package:** `com.sirius.sequenziatore.client.view`;
- **Descrizione:** Componente che permette di gestire l'interfaccia grafica relativa alle richieste di autenticazione al sistema;
- **Relazioni con altri componenti:**

  Il componente implementa l'interfaccia `com.sirius.sequenziatore.client.view.ILogin`.

### 3.2.1 Package com.sirius.sequenziatore.client.view.user

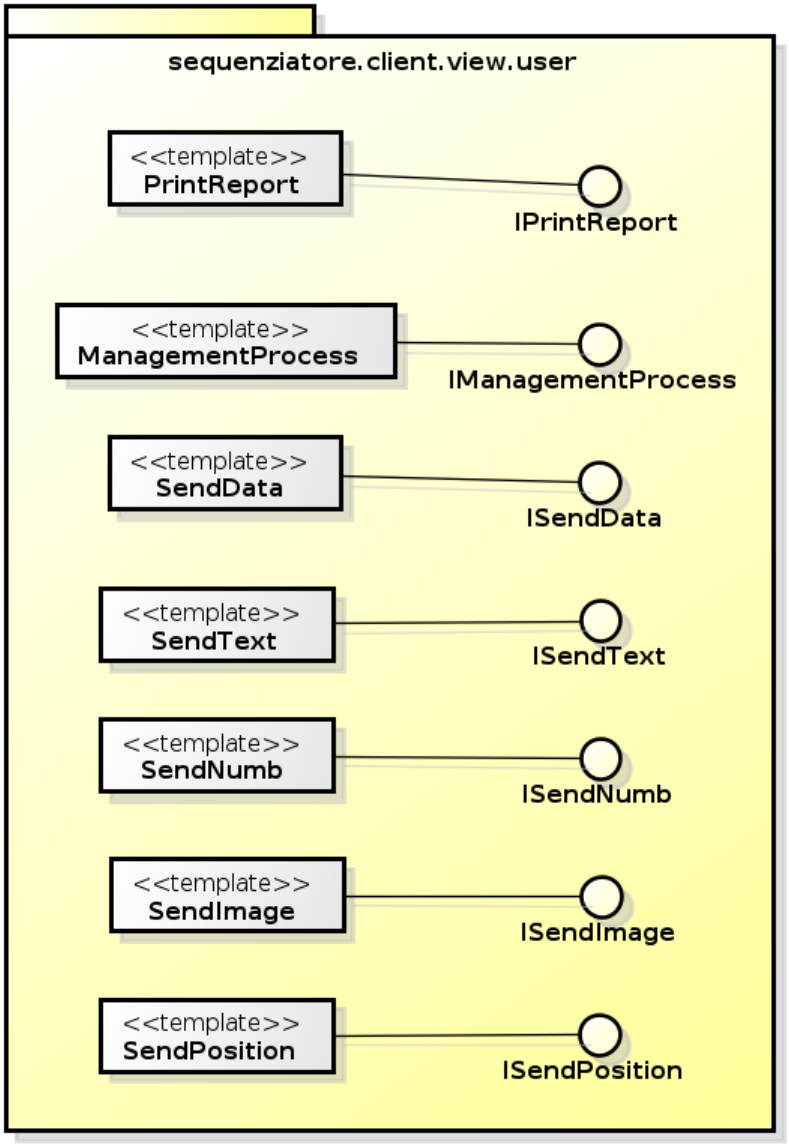


Figura 7: Diagramma *view* utente - gestione processi

#### 3.2.1.1 IMainUser

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

#### 3.2.1.2 MainUser

- **Nome:** `MainUser`;
- **Package:** `com.sirius.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 `com.sirius.sequenziatore.client.view.user.IMainUser`.

#### 3.2.1.3 IRegister

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

#### 3.2.1.4 Register

- **Nome:** `Register`;
- **Package:** `com.sirius.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 `com.sirius.sequenziatore.client.view.user.IRegister`.

#### 3.2.1.5 IUserData

- **Nome:** `IUserData`;
- **Package:** `com.sirius.sequenziatore.client.view.user`;
- **Descrizione:** Interfaccia che permette la realizzazione dei *widget* per la visualizzazione dei dati dell'utente e la relativa modifica dei dati, dove possibile.

#### 3.2.1.6 UserData

- **Nome:** `UserData`;
- **Package:** `com.sirius.sequenziatore.client.view.user`;
- **Descrizione:** Classe che permette la realizzazione dei *widget* che consentono visualizzazione e modifica dei dati dell'utente;
- **Relazioni con altri componenti:**

  La classe implementa l'interfaccia `com.sirius.sequenziatore.client.view.user.IUserData`.

#### 3.2.1.7 IOpenProcess

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

#### 3.2.1.8 OpenProcess

- **Nome:** `OpenProcess`;
- **Package:** `com.sirius.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 `com.sirius.sequenziatore.client.view.user.IOpenProcess`.

#### 3.2.1.9 IManagementProcess

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

#### 3.2.1.10 ManagementProcess

- **Nome:** `ManagementProcess`;
- **Package:** `com.sirius.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 `com.sirius.sequenziatore.client.view.user.IManagementProcess`.

#### 3.2.1.11 ISendData

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

#### 3.2.1.12 SendData

- **Nome:** `SendData`;
- **Package:** `com.sirius.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 `com.sirius.sequenziatore.client.view.user.ISendData`.

#### 3.2.1.13 ISendText

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

#### 3.2.1.14 SendText

- **Nome:** `SendData`;
- **Package:** `com.sirius.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 `com.sirius.sequenziatore.client.view.user.ISendText`.

#### 3.2.1.15 ISendNumb

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

#### 3.2.1.16 SendNumb

- **Nome:** `SendNumb`;
- **Package:** `com.sirius.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 `com.sirius.sequenziatore.client.view.user.ISendNumb`.

#### 3.2.1.17 ISendPosition

- **Nome:** `ISendPosition`;
- **Package:** `com.sirius.sequenziatore.client.view.user`;
- **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.

#### 3.2.1.18 SendPosition

- **Nome:** `SendPosition`;
- **Package:** `com.sirius.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 `com.sirius.sequenziatore.client.view.user.ISendPosition`.

#### 3.2.1.19 ISendImage

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

#### 3.2.1.20 SendImage

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

  La classe implementa l'interfaccia `com.sirius.sequenziatore.client.view.user.ISendImage`.

#### 3.2.1.21 IPrintProcess

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

#### 3.2.1.22 PrintProcess

- **Nome:** `PrintProcess`;
- **Package:** `com.sirius.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 `com.sirius.sequenziatore.client.view.user.IPrintProcess`.

### 3.2.2 Package com.sirius.sequenziatore.client.view.processowner

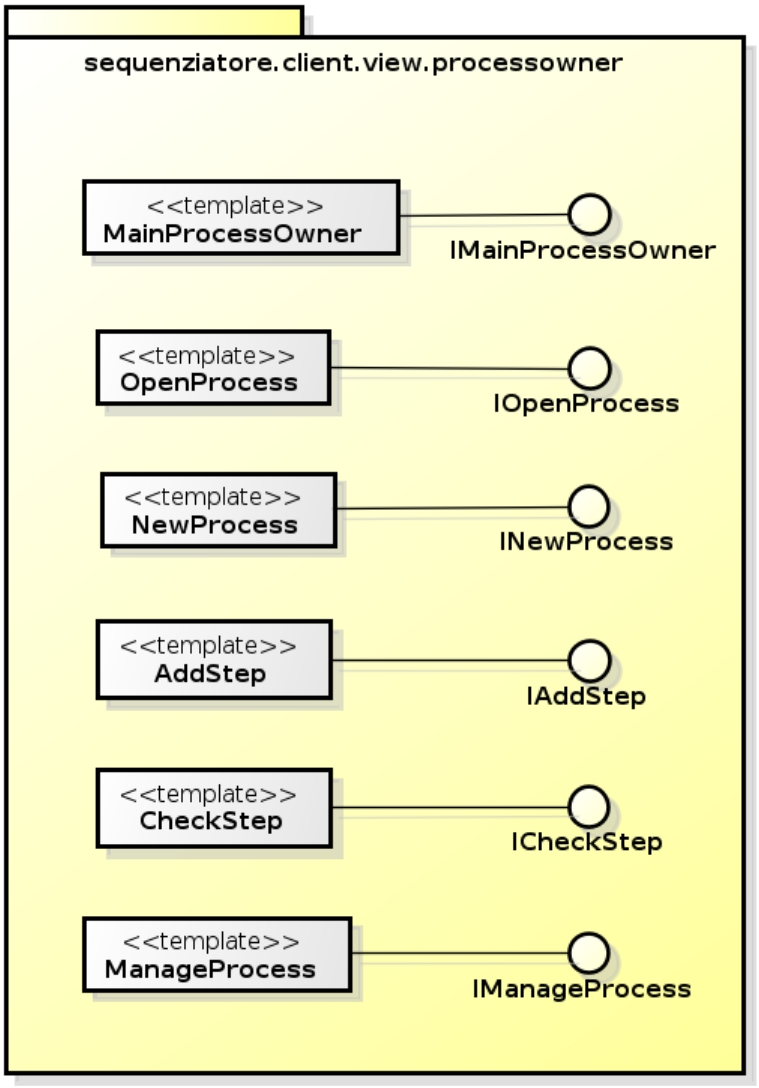


Figura 8: Diagramma *view Process Owner*

#### 3.2.2.1 IMainProcessOwner

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

#### 3.2.2.2 MainProcessOwner

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

  Il componente implementa l'interfaccia `com.sirius.sequenziatore.client.view.processowner.IMainProcessOwner`.

#### 3.2.2.3 INewProcess

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

#### 3.2.2.4 NewProcess

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

  Il componente implementa l'interfaccia `com.sirius.sequenziatore.client.view.processowner.INewProcess`.

#### 3.2.2.5 IAddStep

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

#### 3.2.2.6 AddStep

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

  Il componente implementa l'interfaccia `com.sirius.sequenziatore.client.view.processowner.IAddStep`.

#### 3.2.2.7 IOpenProcess

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

#### 3.2.2.8 OpenProcess

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

  Il componente implementa l'interfaccia `com.sirius.sequenziatore.client.view.processowner.IOpenProcess`.

#### 3.2.2.9 IManageProcess

- **Nome:** `IManageProcess`;
- **Package:** `com.sirius.sequenziatore.client.view.processowner`;
- **Descrizione:** Interfaccia che permette di realizzare i*widget* che consentono di gestire l'accesso ai dati inviati al$server_G$ dagli utenti;

#### 3.2.2.10 ManageProcess

- **Nome:** `ManageProcess`;
- **Package:** `com.sirius.sequenziatore.client.view.processowner`;
- **Descrizione:** Componente che permette di realizzare i*widget* che consentono di gestire l'accesso ai dati inviati al$server_G$ dagli utenti;
- **Relazioni con altri componenti:**

  Il componente implementa l'interfaccia `com.sirius.sequenziatore.client.view.processowner.IManageProcess`.

#### 3.2.2.11 ICheckStep

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

#### 3.2.2.12 CheckStep

- **Nome:** `CheckStep`;
- **Package:** `com.sirius.sequenziatore.client.view.processowner`;
- **Descrizione:** Componente che permette di realizzare i*widget* che consentono di gestire l'approvazione dei passi che richiedono intervento umano;
- **Relazioni con altri componenti:**

  Il componente implementa l'interfaccia `com.sirius.sequenziatore.client.view.processowner.ICheckStep`.

## 3.3 Package com.sirius.sequenziatore.client.presenter

#### 3.3.0.13 Router

- **Nome:** `Router`;
- **Package:** `com.sirius.sequenziatore.client.presenter`;
- **Descrizione:** Classe che permette di coordinare l'inizializzazione e la renderizzazione delle pagine, gestendo gli eventi e le azioni di cambio pagina;
- **Relazioni con altri componenti:**

  La classe reperisce le informazioni di sessione dalla classe `com.sirius.sequenziatore.client.model::UserModel` e comunica con le seguenti classi se l'utente dispone dei diritti d'accesso necessari:

  - `com.sirius.sequenziatore.client.presenterLogin`;
  - `com.sirius.sequenziatore.client.presenter.userRegister`;
  - `com.sirius.sequenziatore.client.presenter.userMainUser`;
  - `com.sirius.sequenziatore.client.presenter.userUserData`;
  - `com.sirius.sequenziatore.client.presenter.userOpenProcessgic`;
  - `com.sirius.sequenziatore.client.presenter.userManagmentProcess`;
  - `com.sirius.sequenziatore.client.presenter.processownerMainProcessOwner`;
  - `com.sirius.sequenziatore.client.presenter.processownerOpenProcess`;
  - `com.sirius.sequenziatore.client.presenter.processownerNewProcess`;
  - `com.sirius.sequenziatore.client.presenter.processownerAddStep`;
  - `com.sirius.sequenziatore.client.presenter.processownerCheckStep`;
  - `com.sirius.sequenziatore.client.presenter.processownerManageProcess`;

#### 3.3.0.14 Login

- **Nome:** `LoginLogic`;
- **Descrizione:** Classe che ha il compito di gestire le richieste di autenticazione al sistema;

- **Relazioni con altri componenti:**

  La classe gestisce i dati di sessione comunicando con la classe `com.sirius.sequenziatore.client.modelUserModel` e realizza l'interfaccia grafica tramite metodi della classe `com.sirius.sequenziatore.client.viewLogin`.

### 3.3.1 Package com.sirius.sequenziatore.client.presenter.user

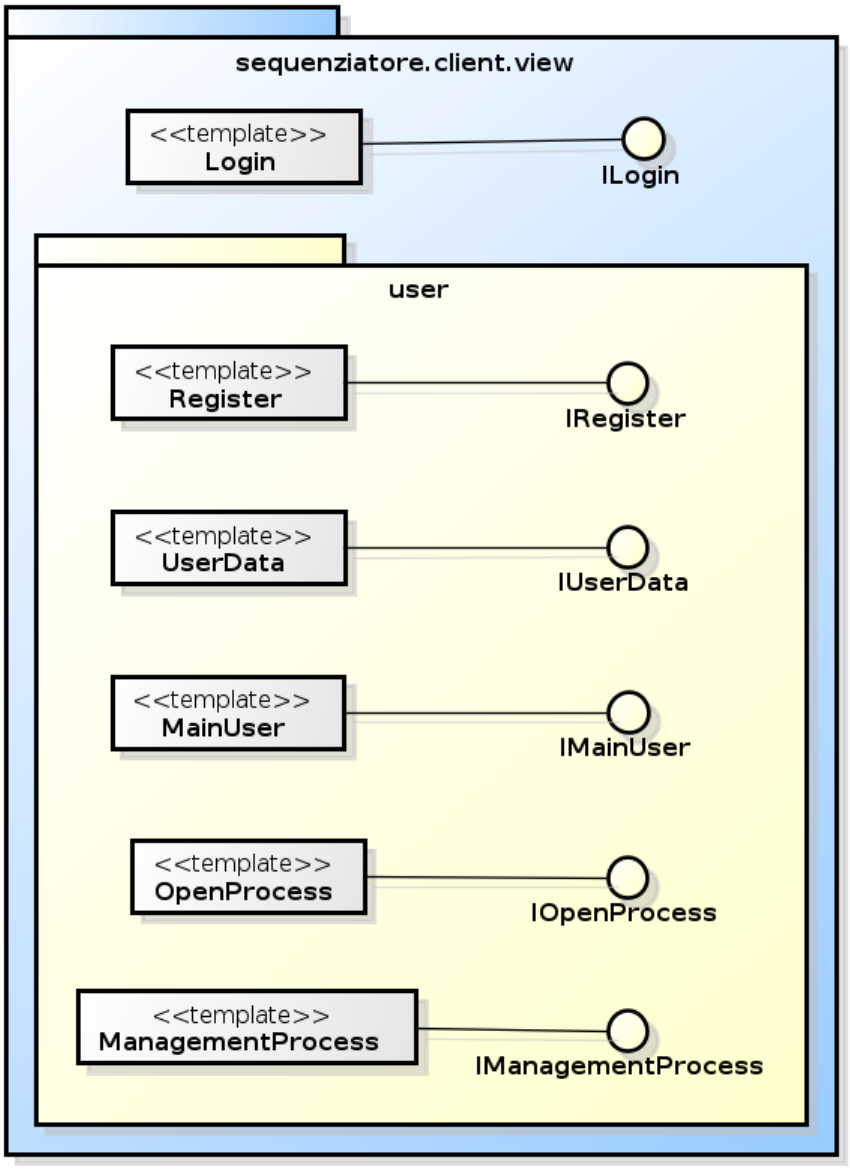


Figura 9: Diagramma principale *presenter* utente

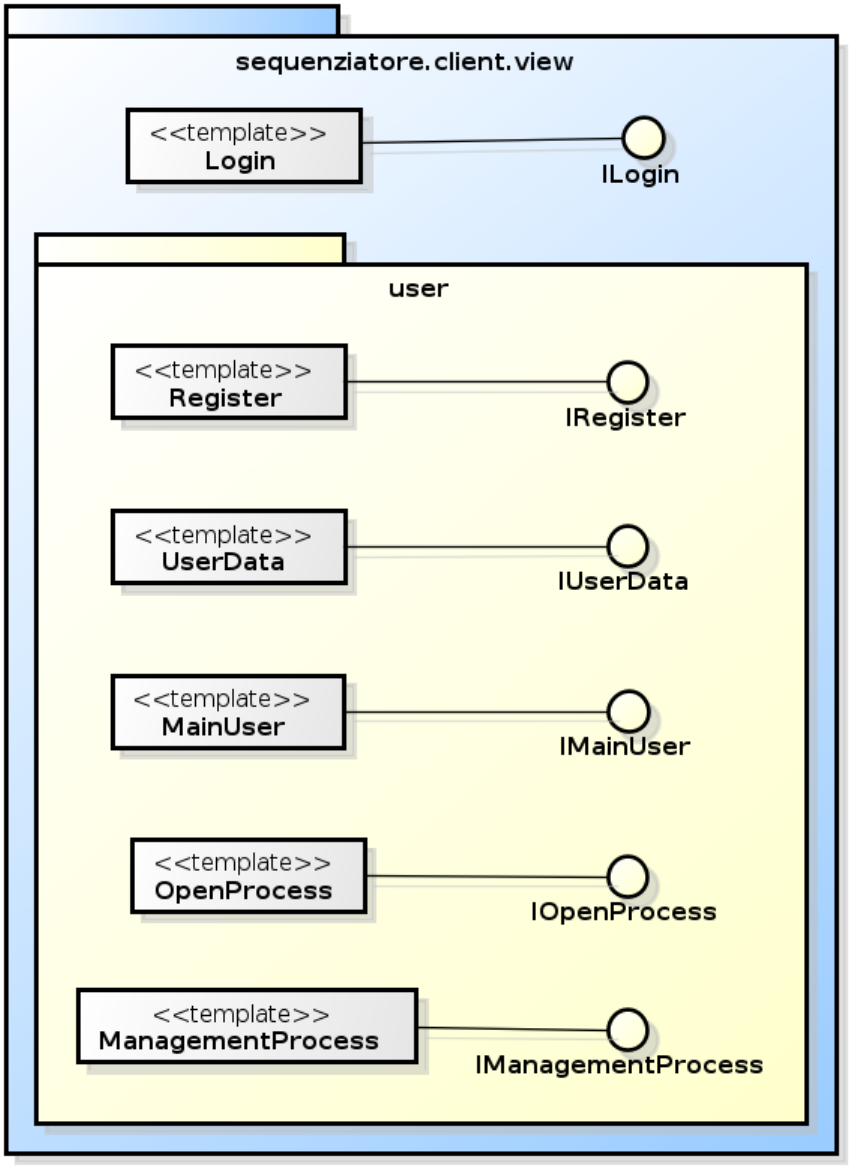


Figura 10: Diagramma *presenter* utente - gestione processi

#### 3.3.1.1 MainUser

- **Nome:** `MainUser`;
- **Package:** `com.sirius.sequenziatore.client.presenter.user`;
- **Descrizione:** Classe che ha il compito della gestione generale della logica delle funzionalità utente;
- **Relazioni con altri componenti:**

La classe comunica con l'interfaccia `com.sirius.sequenziatore.client.view.user.IMainUser` per la realizzazione dell'interfaccia grafica.

#### 3.3.1.2 Register

- **Nome:** `Register`;
- **Package:** `com.sirius.sequenziatore.client.presenter.user`;
- **Descrizione:** Classe che ha il compito di gestire le richieste di registrazione da parte dell'utente;
- **Relazioni con altri componenti:**

  La classe comunica con l'interfaccia `com.sirius.sequenziatore.client.view.user.IRegister` per la realizzazione dei *widget* per la registrazione, e con la classe `com.sirius.sequenziatore.client.model.UserModel` per comunicare col il $server_G$.

#### 3.3.1.3 UserData

- **Nome:** `UserData`;
- **Package:** `com.sirius.sequenziatore.client.presenter.user`;
- **Descrizione:** Classe che ha il compito di gestire la visualizzazione e la modifica dei dati dell'utente;
- **Relazioni con altri componenti:**

  La classe comunica con l'interfaccia `com.sirius.sequenziatore.client.view.user.IUserData` per realizzare il *widget* preposto alla visualizzazione e modifica dei dati dell'utente, e con la classe `com.sirius.sequenziatore.client.model.UserModel` per comunicare col il $server_G$.

#### 3.3.1.4 OpenProcess

- **Nome:** `OpenProcess`;
- **Package:** `com.sirius.sequenziatore.client.presenter.user`;
- **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 `com.sirius.sequenziatore.client.view.user.IOpenProcess` e utilizza la classe `com.sirius.sequenziatore.client.model.collection.ProcessCollection` per gestire e ottenere i dati dal $server_G$.

#### 3.3.1.5 ManagmentProcess

- **Nome:** `ManagmentProcess`;
- **Package:** `com.sirius.sequenziatore.client.presenter.user`;
- **Descrizione:** Classe che ha il compito di gestire e accedere alle informazioni relative allo stato del processo selezionato.;
- **Relazioni con altri componenti:**

  La classe comunica con l'interfaccia `com.sirius.sequenziatore.client.view.user.IManagmentProcess` per realizzare il *widget* che permette la gestione del processo selezionato, utilizza la classe `com.sirius.sequenziatore.client.model.ProcessModel` per gestire e ottenere i dati dal $server_G$, e provvede ad invocare le seguenti classi in base alle decisioni dell'utente:

  - `com.sirius.sequenziatore.client.presenter.user.PrintReport`;
  - `com.sirius.sequenziatore.client.presenter.user.SendData`.

#### 3.3.1.6 PrintReport

- **Nome:** `PrintReport`;
- **Package:** `com.sirius.sequenziatore.client.presenter.user`;
- **Descrizione:** Classe che ha il compito di gestire la creazione del report di fine processo;
- **Relazioni con altri componenti:**

  La classe comunica con l'interfaccia `com.sirius.sequenziatore.client.view.user.IPrintReport` per realizzare il *widget* per creare il report di fine processo, e utilizza la classe `com.sirius.sequenziatore.client.model.collection.ProcessDataCollection` per gestire e ottenere i dati dal $server_G$.

#### 3.3.1.7 SendData

- **Nome:** `SendData`;
- **Package:** `com.sirius.sequenziatore.client.presenter.user`;
- **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 comunica con l'interfaccia `com.sirius.sequenziatore.client.view.user.ISendData` per creare il *widget* che consente di inviare i dati, utilizza la classe `com.sirius.sequenziatore.client.model.collection.ProcessDataCollection` per gestire e ottenere i dati dal $server_G$, e infine invoca le seguenti classi che gestiscono l'invio di un tipo di dato specifico:

  - `com.sirius.sequenziatore.client.presenter.user.SendText`;
  - `com.sirius.sequenziatore.client.presenter.user.SendNumb`;
  - `com.sirius.sequenziatore.client.presenter.user.SendImage`;
  - `com.sirius.sequenziatore.client.presenter.user.SendPosition`.

#### 3.3.1.8 SendText

- **Descrizione:** Classe che permette l'inserimento e il controllo di dati testuali inseriti dagli utenti;
- **Relazioni con altri componenti:**

  La classe, mediante l'interfaccia `com.sirius.sequenziatore.client.view.user.ISendText`, realizza e aggiorna l'opportuno *widget*.

#### 3.3.1.9 SendNumb

- **Descrizione:** Classe che ha il compito di permettere l'inserimento e il controllo di dati numerici inseriti dagli utenti;
- **Relazioni con altri componenti:**

  La classe, mediante l'interfaccia `com.sirius.sequenziatore.client.view.user.ISendNumb`, realizza e aggiorna l'opportuno *widget*.

#### 3.3.1.10 SendImage

- **Descrizione:** Classe che gestisce l'inserimento e il controllo di immagini inserite dagli degli utenti;
- **Relazioni con altri componenti:**

  La classe, mediante l'interfaccia `com.sirius.sequenziatore.client.view.user.ISendImage`, realizza e aggiorna l'opportuno *widget*.

#### 3.3.1.11 SendPosition

- **Descrizione:** Classe che ha il compito di gestire il calcolo e il controllo della posizione geografica dell'utente;
- **Relazioni con altri componenti:**

  La classe, mediante l'interfaccia `com.sirius.sequenziatore.client.view.user.ISendPosition`, realizza e aggiorna l'opportuno *widget*.

### 3.3.2 Package com.sirius.sequenziatore.client.presenter.processowner

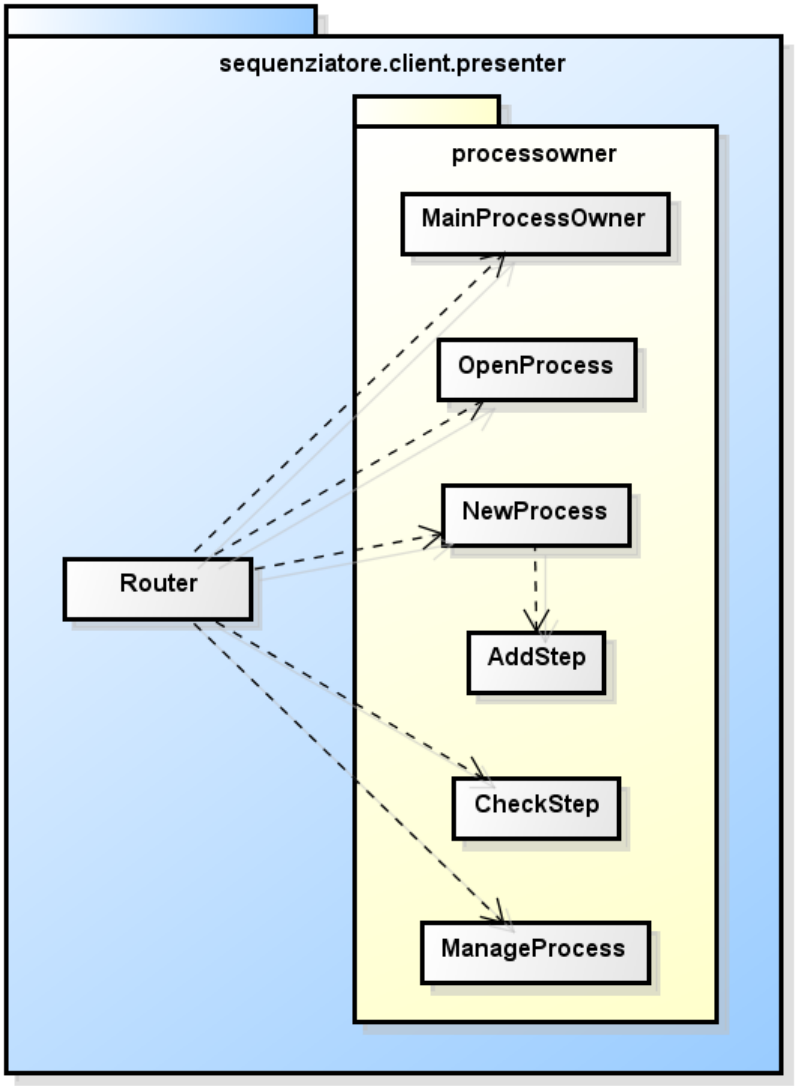


Figura 11: Diagramma *presenter Process Owner*

#### 3.3.2.1 MainProcessOwner

- **Nome:** `MainProcessOwner`;
- **Package:** `com.sirius.sequenziatore.client.presenter.processowner`;
- **Descrizione:** Classe che ha il compito della gestione generale della logica delle funzionalità *Process Owner*$_G$;
- **Relazioni con altri componenti:**

La classe comunica con l'interfaccia `com.sirius.sequenziatore.client.view.processowner.IMainProcessOwner` per la realizzazione dell'interfaccia grafica.

#### 3.3.2.2 OpenProcess

- **Nome:** `OpenProcess`;
- **Package:** `com.sirius.sequenziatore.client.presenter.processowner`;
- **Descrizione:** Classe che ha il compito di gestire la ricerca e la selezione di un processo;
- **Relazioni con altri componenti:**

  La classe comunica con l'interfaccia `com.sirius.sequenziatore.client.view.processowner.IOpenProcess` per la realizzazione dell'interfaccia grafica, e con la classe `com.sirius.sequenziatore.client.model.collection.ProcessCollection` per gestire e ottenere i dati dal $server_G$.

#### 3.3.2.3 NewProcess

- **Nome:** `NewProcess`;
- **Package:** `com.sirius.sequenziatore.client.presenter.processowner`;
- **Descrizione:** Classe che ha il compito di gestire la logica della definizione di un nuovo processo;
- **Relazioni con altri componenti:**

  La classe comunica con l'interfaccia `com.sirius.sequenziatore.client.view.processowner.INewprocess` per la realizzazione dell'interfaccia grafica, con la classe `com.sirius.sequenziatore.client.model.collection.ProcessCollection` comunicare con il $server_G$, e con la classe `com.sirius.sequenziatore.client.presenter.processownerAddStep`;.

#### 3.3.2.4 AddStep

- **Nome:** `AddStep`;
- **Package:** `com.sirius.sequenziatore.client.presenter.processowner`;
- **Descrizione:** Classe che ha il compito di gestire la logica di definizione dei passi di un processo;

- **Relazioni con altri componenti:**

  La classe comunica con l'interfaccia `com.sirius.sequenziatore.client.view.processowner.IAddStep` per la realizzazione dell'interfaccia grafica e utilizza la classe `com.sirius.sequenziatore.client.modelStep` per salvare i dati del passo in creazione.

#### 3.3.2.5 ManageProcess

- **Nome:** `ManageProcess`;
- **Package:** `com.sirius.sequenziatore.client.presenter.processowner`;
- **Descrizione:** Classe che ha il compito di gestire e accedere alle informazioni relative allo stato dei processi e ai dati inviati dagli utenti. Le operazioni di gestione dello stato comprendono la terminazione e l'eliminazione di un processo;
- **Relazioni con altri componenti:**

  La classe comunica con l'interfaccia `com.sirius.sequenziatore.client.view.processowner.IManageProcess` per la realizzazione dell'interfaccia grafica, e con le classi `com.sirius.sequenziatore.client.model.collection.ProcessDataCollection` e `com.sirius.sequenziatore.client.modelProcessModel` per gestire e ottenere i dati dal $server_G$.

#### 3.3.2.6 CheckStep

- **Nome:** `CheckStep`;
- **Package:** `com.sirius.sequenziatore.client.presenter.processowner`;
- **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 comunica con l'interfaccia `com.sirius.sequenziatore.client.view.processowner.ICheckStep` per la realizzazione dell'interfaccia grafica, e con le classi `com.sirius.sequenziatore.client.model.collection.ProcessDataCollection` e `com.sirius.sequenziatore.client.modelProcessModel` per gestire e ottenere i dati dal $server_G$.

## 3.4 Package com.sirius.sequenziatore.client.model

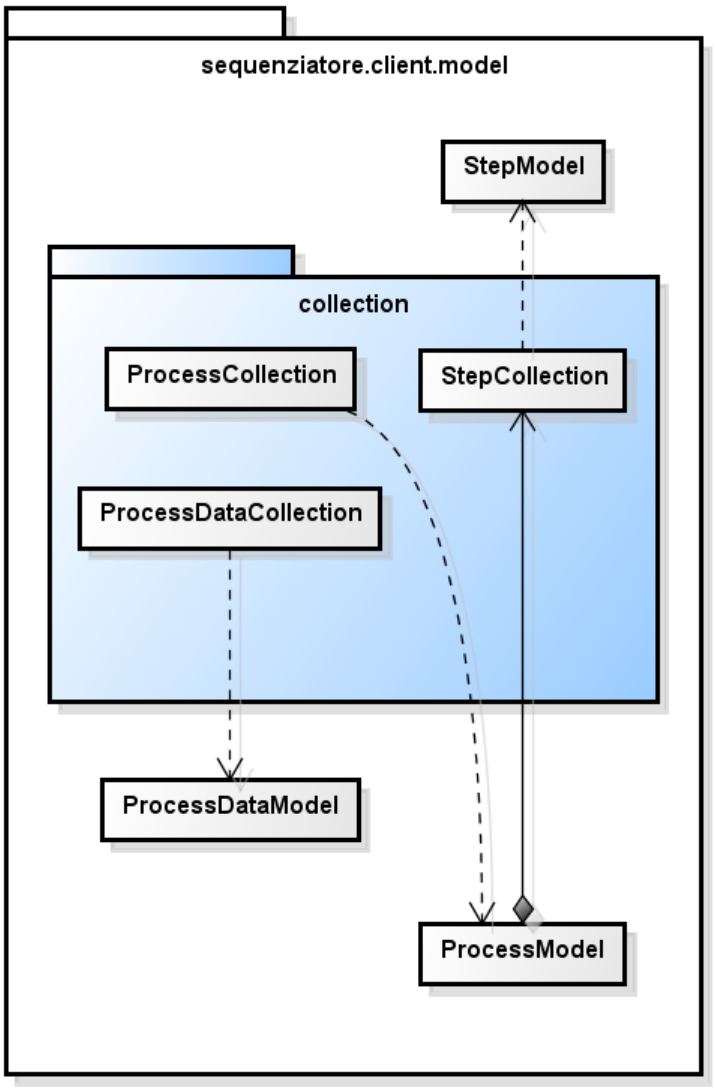


Figura 12: Diagramma modello lato *client*

#### 3.4.0.7 ProcessModel

- **Nome:** `ProcessModel`;
- **Package:** `com.sirius.sequenziatore.client.model`;
- **Descrizione:** Classe che permette di gestire i dati di un processo, e di salvarli o recuperarli dal $server_G$;
- **Relazioni con altri componenti:**

  La classe contiene un oggetto di tipo `com.sirius.sequenziatore.client.model.collection.StepCollection`.

#### 3.4.0.8 ProcessDataModel

- **Nome:** `ProcessDataModel`;
- **Package:** `com.sirius.sequenziatore.client.model`;
- **Descrizione:** Classe che permette di gestire i dati inviati da un utente relativi ad un processo, e di salvarli o recuperarli dal *server*$_G$.

#### 3.4.0.9 StepModel

- **Nome:** `StepModel`;
- **Package:** `com.sirius.sequenziatore.client.model`;
- **Descrizione:** Classe che permette di gestire i dati di un passo di un processo, e di salvarli o recuperarli dal *server*$_G$.

#### 3.4.0.10 UserModel

- **Nome:** `UserModel`;
- **Package:** `com.sirius.sequenziatore.client.model`;
- **Descrizione:** Classe che permette di gestire i dati di una sessione di un utente autenticato o di un *Process Owner*$_G$.

### 3.4.1 Package com.sirius.sequenziatore.client.model.collection

#### 3.4.1.1 ProcessCollection

- **Nome:** `ProcessCollection`;
- **Package:** `com.sirius.sequenziatore.client.model.collection`;
- **Descrizione:** Classe che permette di gestire un insieme di dati inviati da un utente relativi ad un processo;
- **Relazioni con altri componenti:**

  La classe definisce una collezione di `com.sirius.sequenziatore.client.model.ProcessModel`.

#### 3.4.1.2 ProcessDataCollection

- **Nome:** `ProcessDataCollection`;
- **Package:** `com.sirius.sequenziatore.client.model.collection`;
- **Descrizione:** Classe che permette di gestire un insieme di dati inviati dagli utenti;
- **Relazioni con altri componenti:**

  La classe definisce una collezione di `com.sirius.sequenziatore.client.model.ProcessDataModel`.

#### 3.4.1.3 StepCollection

- **Nome:** `StepCollection`;
- **Package:** `com.sirius.sequenziatore.client.model.collection`;
- **Descrizione:** Classe che permette di gestire un insieme di passi di un processo;
- **Relazioni con altri componenti:**

  La classe definisce una collezione di `com.sirius.sequenziatore.client.model.StepModel`.

## 3.5 Package com.sirius.sequenziatore.server.presenter

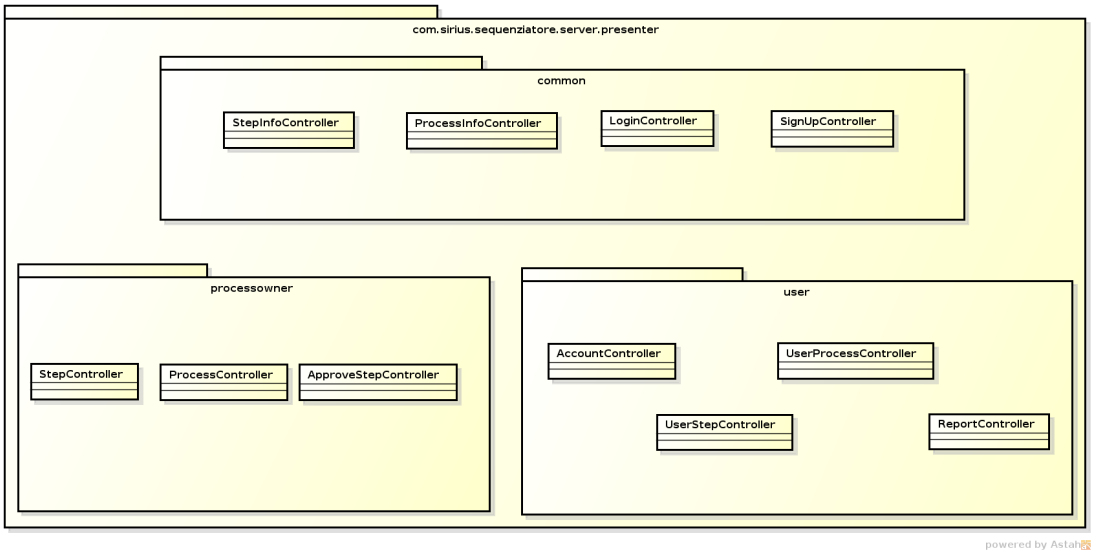


Figura 13: Diagramma presenter server

### 3.5.1 Package com.sirius.sequenziatore.server.presenter.common

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

#### 3.5.1.1 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:
- **Relazione con altre componenti:** la classe invoca i metodi delle classi:
  - com.sirius.sequenziatore.server.model.IDataAccessObject;
  - com.sirius.sequenziatore.server.model.ITransferObject

#### 3.5.1.2 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:
- **Relazione con altre componenti:** la classe invoca i metodi delle classi:
  - com.sirius.sequenziatore.server.model.IDataAccessObject;
  - com.sirius.sequenziatore.server.model.ITransferObject

#### 3.5.1.3 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 delle classi:
  - com.sirius.sequenziatore.server.model.IDataAccessObject;
  - com.sirius.sequenziatore.server.model.ITransferObject

#### 3.5.1.4 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:
- **Relazione con altre componenti:** la classe invoca i metodi delle classi:
  - com.sirius.sequenziatore.server.model.IDataAccessObject;
  - com.sirius.sequenziatore.server.model.ITransferObject

### 3.5.2 Package com.sirius.sequenziatore.server.presenter.user

#### 3.5.2.1 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;

#### 3.5.2.2 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;

#### 3.5.2.3 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;

#### 3.5.2.4 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;

### 3.5.3 Package com.sirius.sequenziatore.server.presenter.processowner

#### 3.5.3.1 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;

#### 3.5.3.2 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;

#### 3.5.3.3 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;

## 3.6 Package sequenziatore.server.model

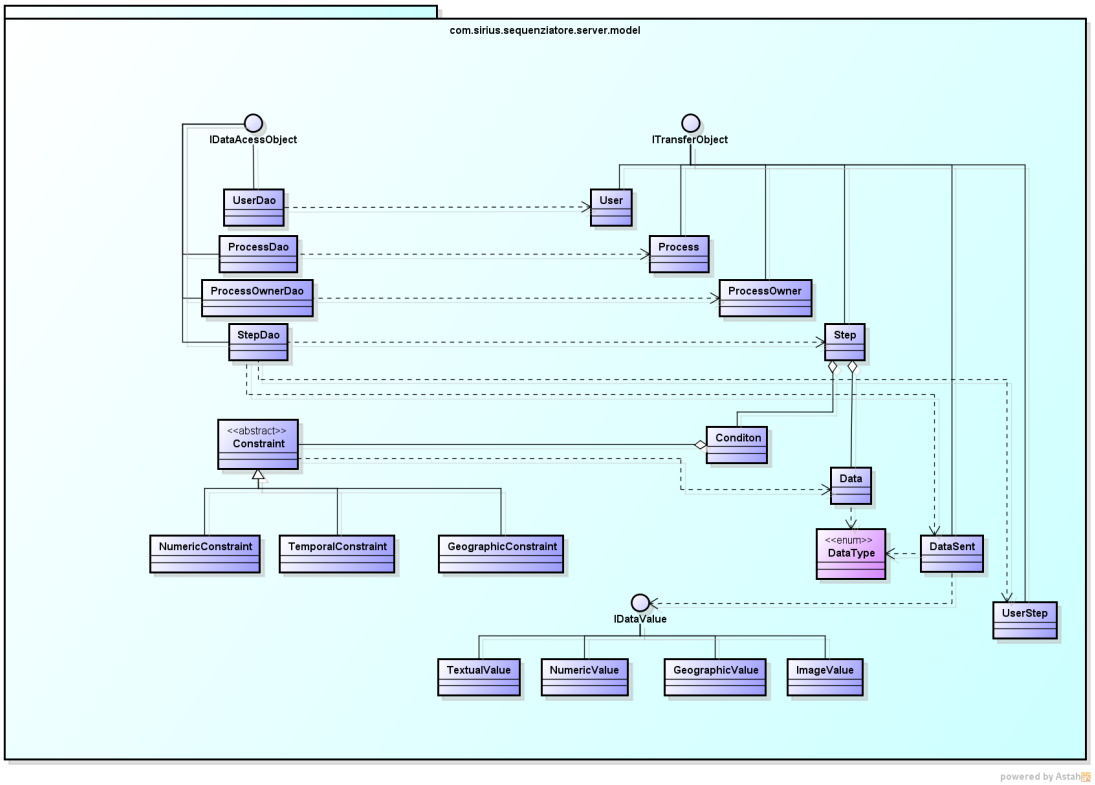


Figura 14: Diagramma model server

#### 3.6.0.4 IDataAcessObject

- **Nome:** `IDataAcessObject`;
- **Package:** `com.sirius.sequenziatore.server.model`;
- **Descrizione:** Interfaccia che permette di gestire la comunicazione e l'interrogazione con il *database*.

#### 3.6.0.5 ITransferObject

- **Nome:** `ITransferObject`;
- **Package:** `com.sirius.sequenziatore.server.model`;
- **Descrizione:** Interfaccia realizzata dai tipi che modellano i dati del *database*.

#### 3.6.0.6 UserDao

- **Nome:** `UserDao`;

- **Package:** `com.sirius.sequenziatore.server.model`;
- **Descrizione:** Classe che si occupa delle interrogazioni del *database* relative agli utenti del sistema.
- **Relazione con altre componenti:** la classe implementa la seguente interfaccia:
  - com.sirius.sequenziatore.server.model.IDataAcessObject.

  La classe invoca i metodi della classe:
  - com.sirius.sequenziatore.server.model.User.

#### 3.6.0.7 ProcessDao

- **Nome:** `ProcessDao`;
- **Package:** `com.sirius.sequenziatore.server.model`;
- **Descrizione:** Classe che si occupa delle interrogazioni del *database* relative ai processi.
- **Relazione con altre componenti:** la classe implementa la seguente interfaccia:
  - com.sirius.sequenziatore.server.model.IDataAcessObject.

  La classe invoca i metodi della classe:
  - com.sirius.sequenziatore.server.model.Process.

#### 3.6.0.8 ProcessOwnerDao

- **Nome:** `ProcessOwnerDao`;
- **Package:** `com.sirius.sequenziatore.server.model`;
- **Descrizione:** Classe che si occupa delle interrogazioni del *database* relative all'autenticazione del *ProcessOwner*.
- **Relazione con altre componenti:** la classe implementa la seguente interfaccia:
  - com.sirius.sequenziatore.server.model.IDataAcessObject.

  La classe invoca i metodi della classe:
  - com.sirius.sequenziatore.server.model.ProcessOwner.

#### 3.6.0.9 StepDao

- **Nome:** `StepDao`;
- **Package:** `com.sirius.sequenziatore.server.model`;
- **Descrizione:** Classe che si occupa delle interrogazioni del *database* relative a tutte le operazioni sui passi dei processi.
- **Relazione con altre componenti:** la classe implementa la seguente interfaccia:
  - com.sirius.sequenziatore.server.model.IDataAcessObject.

  La classe invoca i metodi della classe:
  - com.sirius.sequenziatore.server.model.Step;
  - com.sirius.sequenziatore.server.model.UserStep;
  - com.sirius.sequenziatore.server.model.DataSent.

#### 3.6.0.10 User

- **Nome:** `User`;
- **Package:** `com.sirius.sequenziatore.server.model`;
- **Descrizione:** Classe che modella gli utenti del sistema e che funge da interscambio dei dati di quest'ultimi con il *database*.
- **Relazione con altre componenti:** la classe implementa la seguente interfaccia:
  - com.sirius.sequenziatore.server.model.ITransferObject.

#### 3.6.0.11 Process

- **Nome:** `Process`;
- **Package:** `com.sirius.sequenziatore.server.model`;
- **Descrizione:** Classe che modella i processi del sistema e che funge da interscambio dei dati di quest'ultimi con il *database*.
- **Relazione con altre componenti:** la classe implementa la seguente interfaccia:
  - com.sirius.sequenziatore.server.model.ITransferObject.

#### 3.6.0.12 ProcessOwner

- **Nome:** `ProcessOwner`;
- **Package:** `com.sirius.sequenziatore.server.model`;
- **Descrizione:** Classe che modella il ProcessOwner e che funge da interscambio dei dati di quest'ultimo con il *database*.
- **Relazione con altre componenti:** la classe implementa la seguente interfaccia:
  - com.sirius.sequenziatore.server.model.ITransferObject.

#### 3.6.0.13 Step

- **Nome:** `Step`;
- **Package:** `com.sirius.sequenziatore.server.model`;
- **Descrizione:** Classe che modella i passi del sistema e che funge da interscambio dei dati di quest'ultimi con il *database*.
- **Relazione con altre componenti:** la classe implementa la seguente interfaccia:
  - com.sirius.sequenziatore.server.model.ITransferObject.

  La classe contiene istanze di:
  - com.sirius.sequenziatore.server.model.Condition;
  - com.sirius.sequenziatore.server.model.Data.

#### 3.6.0.14 DataSent

- **Nome:** `DataSent`;
- **Package:** `com.sirius.sequenziatore.server.model`;
- **Descrizione:** Classe che modella i dati ricevuti dagli utenti che funge da interscambio con il *database*.
- **Relazione con altre componenti:** la classe implementa la seguente interfaccia:
  - com.sirius.sequenziatore.server.model.ITransferObject.

  La classe invoca i metodi della classe:
  - com.sirius.sequenziatore.server.model.IDataValue.

#### 3.6.0.15 UserStep

- **Nome:** `UserStep`;
- **Package:** `com.sirius.sequenziatore.server.model`;
- **Descrizione:** Classe che modella i passi in corso e che funge da interscambio dei dati di quest'ultimi con il *database*.
- **Relazione con altre componenti:** la classe implementa la seguente interfaccia:
  - com.sirius.sequenziatore.server.model.ITransferObject.

#### 3.6.0.16 Condition

- **Nome:** `Condition`;
- **Package:** `com.sirius.sequenziatore.server.model`;
- **Descrizione:** Classe che modella le condizioni di avanzamento di un passo.
- **Relazione con altre componenti:**La classe contiene istanze di:
  - com.sirius.sequenziatore.server.model.Constraint;

#### 3.6.0.17 Data

- **Nome:** `Data`;
- **Package:** `com.sirius.sequenziatore.server.model`;
- **Descrizione:** Classe che modella i campi dato richiesti.

#### 3.6.0.18 Constraint

- **Nome:** `Constraint`;
- **Package:** `com.sirius.sequenziatore.server.model`;
- **Descrizione:** Classe astratta che modella i vincoli.

#### 3.6.0.19 NumericConstraint

- **Nome:** `NumericConstraint`;
- **Package:** `com.sirius.sequenziatore.server.model`;
- **Descrizione:** Classe che modella i vincoli numerici.
- **Relazione con altre componenti:** la classe estende la seguente classe:
  - com.sirius.sequenziatore.server.model.Constraint.

#### 3.6.0.20 TemporalConstraint

- **Nome:** `TemporalConstraint`;
- **Package:** `com.sirius.sequenziatore.server.model`;
- **Descrizione:** Classe che modella i vincoli temporali.
- **Relazione con altre componenti:** la classe estende la seguente classe:
  - com.sirius.sequenziatore.server.model.Constraint.

#### 3.6.0.21 GeographicConstraint

- **Nome:** `GeographicConstraint`;
- **Package:** `com.sirius.sequenziatore.server.model`;
- **Descrizione:** Classe che modella i vincoli geografici.
- **Relazione con altre componenti:** la classe estende la seguente classe:
  - com.sirius.sequenziatore.server.model.Constraint.

#### 3.6.0.22 IDataValue

- **Nome:** `IDataValue`;
- **Package:** `com.sirius.sequenziatore.server.model`;
- **Descrizione:** Interfaccia che modella i valori dei dati ricevuti.

#### 3.6.0.23 TextualValue

- **Nome:** `TextualValue`;
- **Package:** `com.sirius.sequenziatore.server.model`;
- **Descrizione:** Classe che modella i valori dei dati testuali.
- **Relazione con altre componenti:** la classe implementa la seguente interfaccia:
  - com.sirius.sequenziatore.server.model.IDataValue.

#### 3.6.0.24 NumericValue

- **Nome:** `NumericValue`;
- **Package:** `com.sirius.sequenziatore.server.model`;
- **Descrizione:** Classe che modella i valori dei dati numerici.
- **Relazione con altre componenti:** la classe implementa la seguente interfaccia:
  - com.sirius.sequenziatore.server.model.IDataValue.

#### 3.6.0.25 GeographicValue

- **Nome:** `TextualValue`;
- **Package:** `com.sirius.sequenziatore.server.model`;
- **Descrizione:** Classe che modella i valori dei dati geografici.
- **Relazione con altre componenti:** la classe implementa la seguente interfaccia:
  - com.sirius.sequenziatore.server.model.IDataValue.

#### 3.6.0.26 ImageValue

- **Nome:** `ImageValue`;
- **Package:** `com.sirius.sequenziatore.server.model`;
- **Descrizione:** Classe che modella i valori dei dati immagine.
- **Relazione con altre componenti:** la classe implementa la seguente interfaccia:
  - com.sirius.sequenziatore.server.model.IDataValue.

# 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.

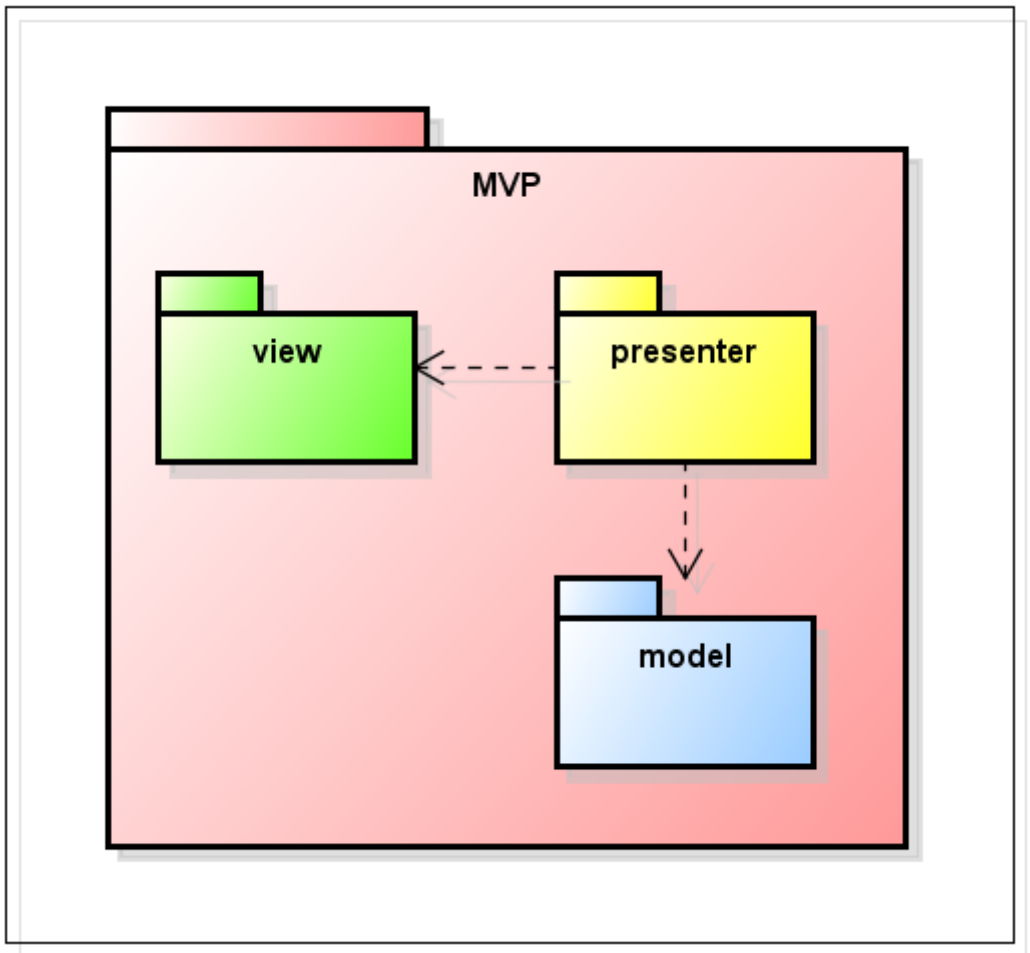


Figura 15: Diagramma UML pattern MVP

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

## 4.2 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.

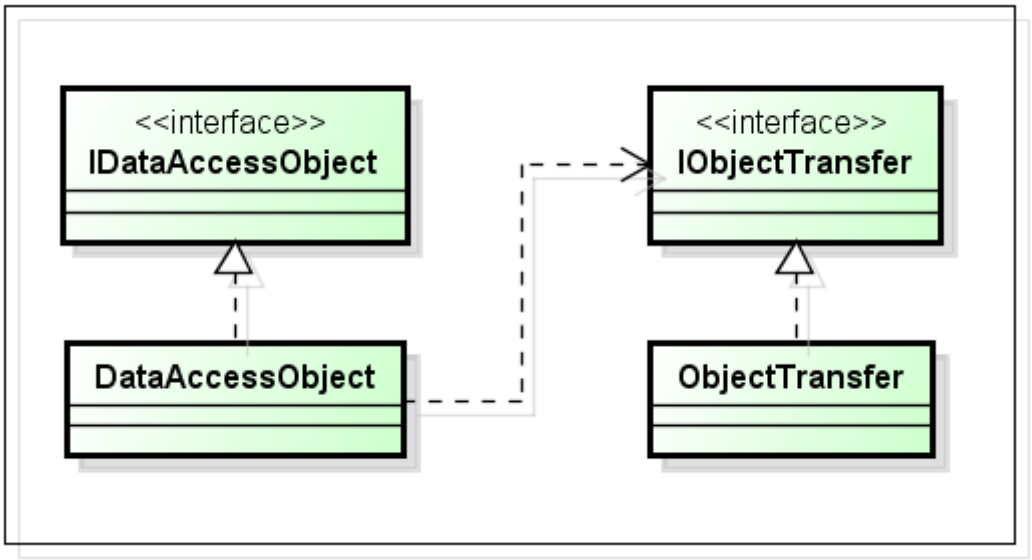


Figura 16: Diagramma UML pattern DAO

- **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*.

## 4.3 Asynchronous Module Definition

- **Scopo e descrizione:** Il *pattern*$_G$ *Asynchronous Module Definition* (AMD), fornisce delle soluzioni per suddividere il codice *Javascript* $_G$ in moduli e di caricarli in modo asincrono, dove con modulo si intende un'unità che consente di incapsulare una porzione di codice. Di seguito viene riportato l'esempio di caricamento di due moduli `ModuloA` e `ModuloB` da una classe `MainClass`:

```
/* classe MainClass */
require(['ModuleA', 'ModuleB'], function(A, B) {
        var a = new A();
    var b = new B();
    a.print(); b.print();
});
```

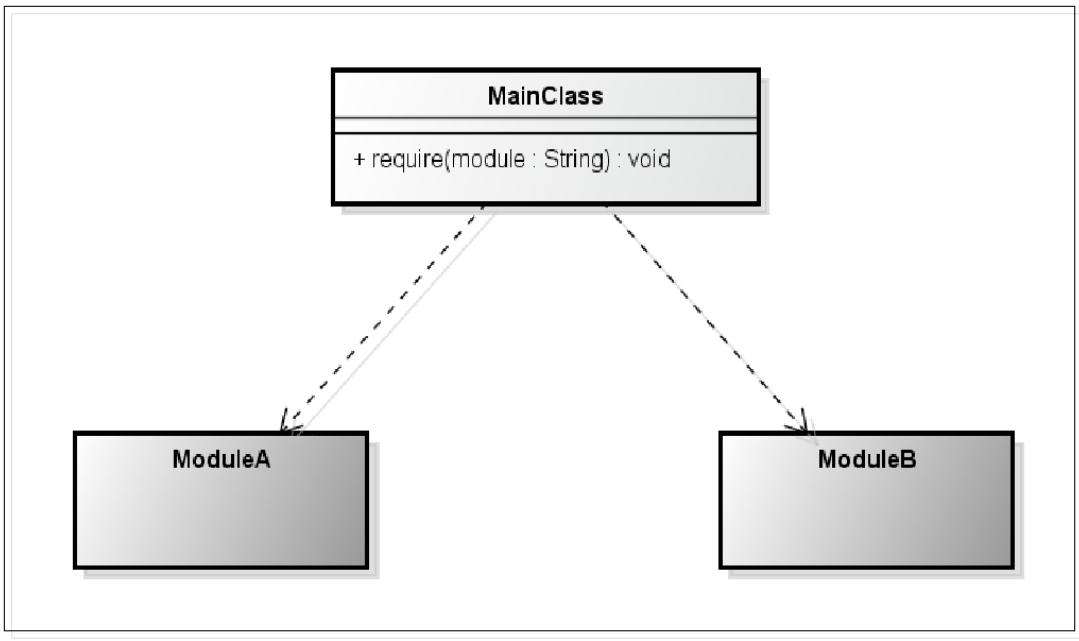


Figura 17: Diagramma pattern AMD

- **Contesto d'uso:** Il *pattern*$_G$ AMD è utilizzato dalla classe `com.sirius.sequenziatore.client.presenter.Router`, e consente di caricare le classi dei *package* `com.sirius.sequenziatore.client.presenter.processowner` e `com.sirius.sequenziatore.client.presenter.processowner` a seconda dei dai di sessione salvati dalla classe `com.sirius.sequenziatore.client.model.UserDataModel`.

# 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

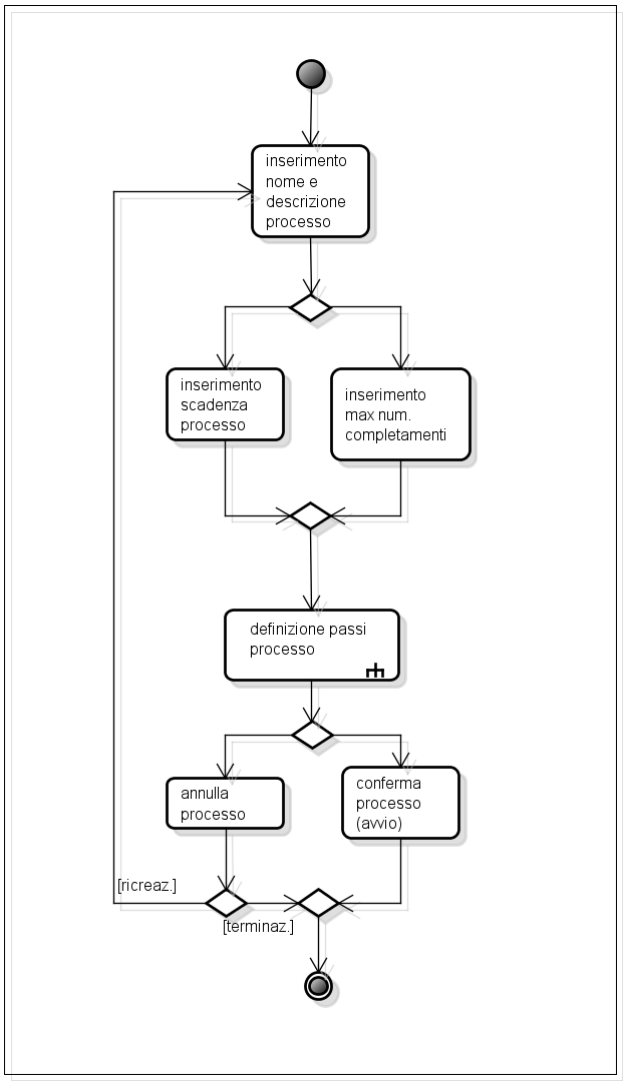


Figura 18: Attività process owner: creazione processo.

**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

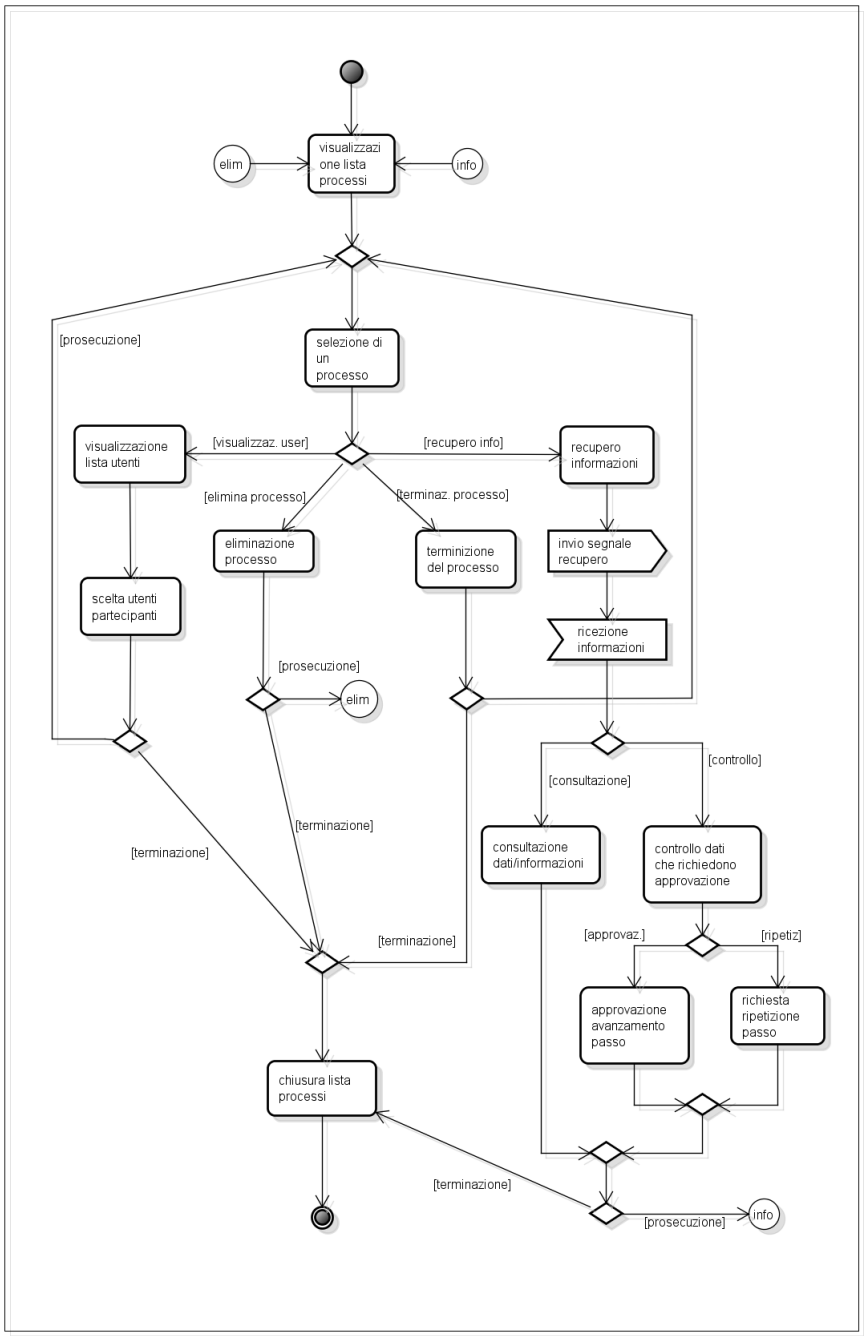


Figura 19: Attività process owner: gestione di un processo.

**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

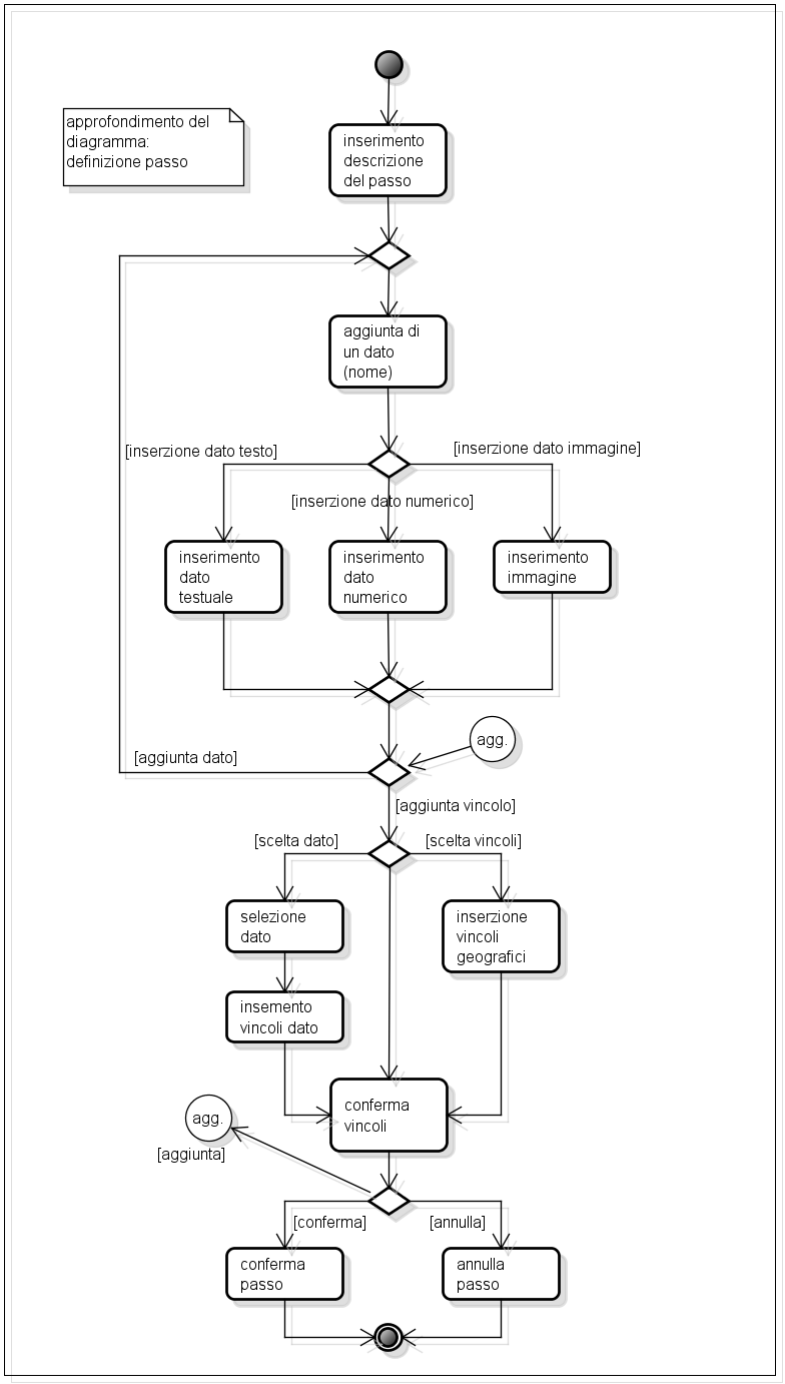


Figura 20: Attività process owner: creazione passo.

**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

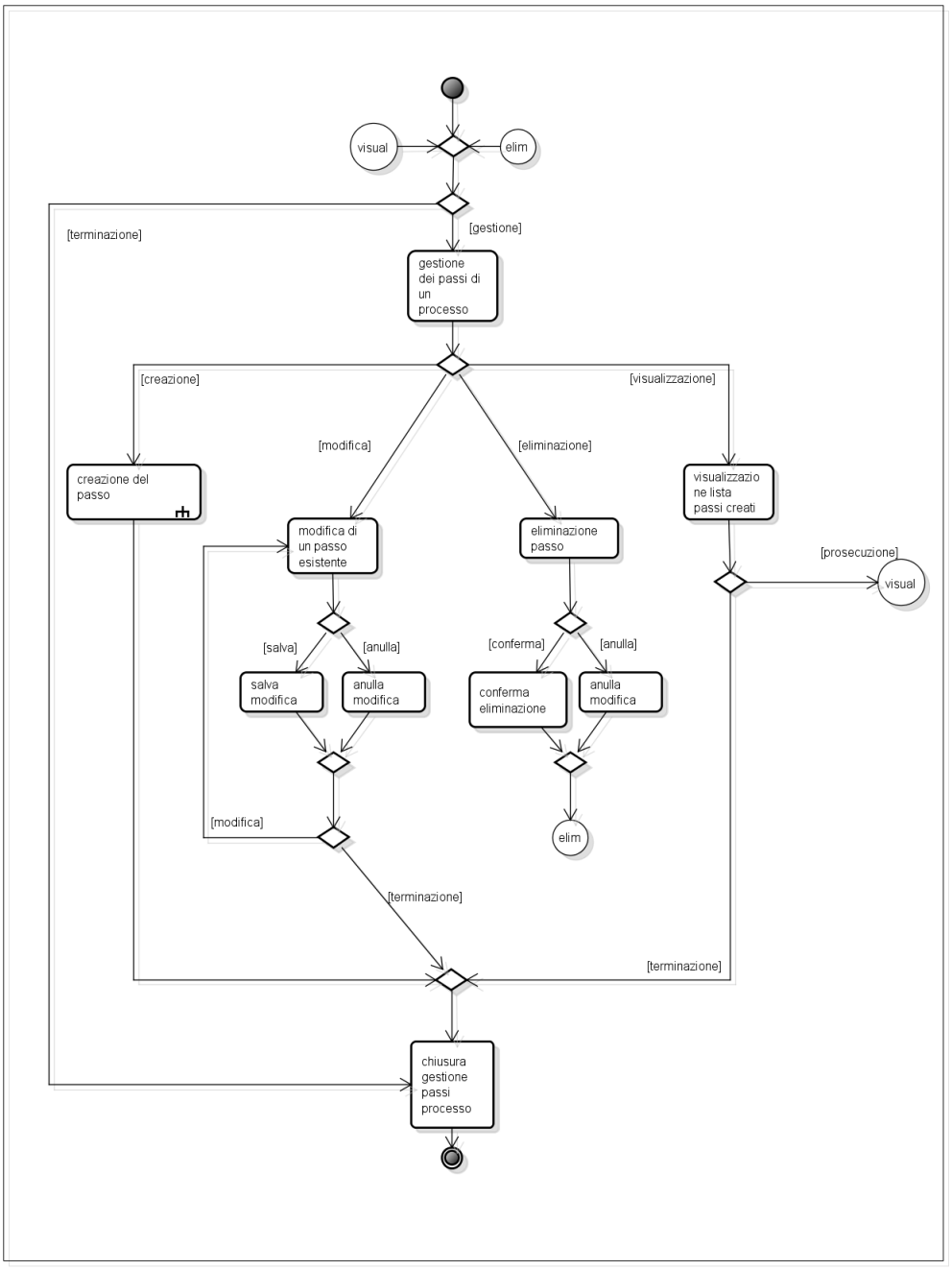


Figura 21: Attività process owner: gestione passi.

**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

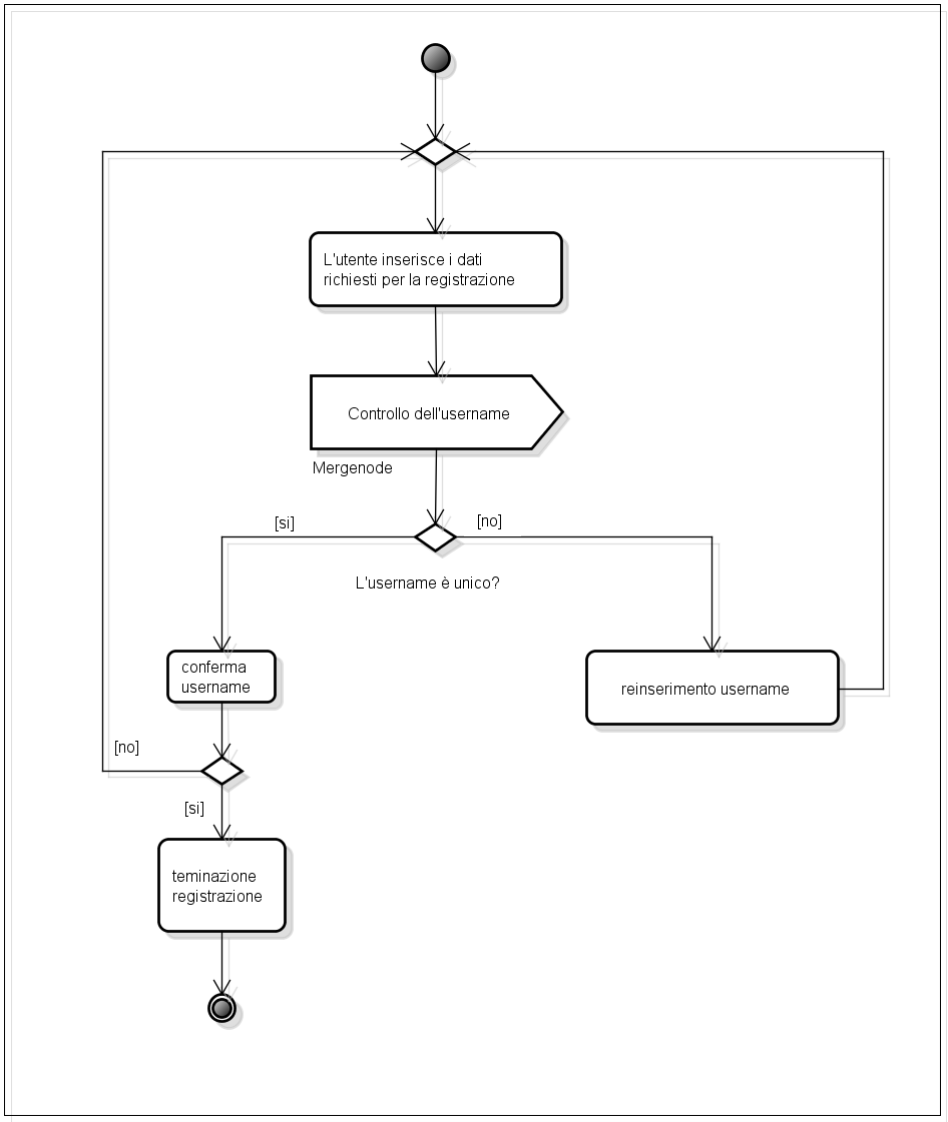


Figura 22: Attività user: Registrazione

**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

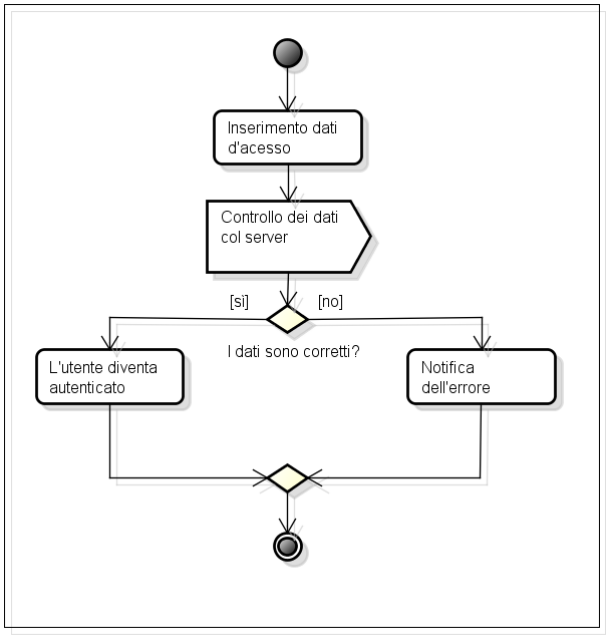


Figura 23: Attività user: Login

**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

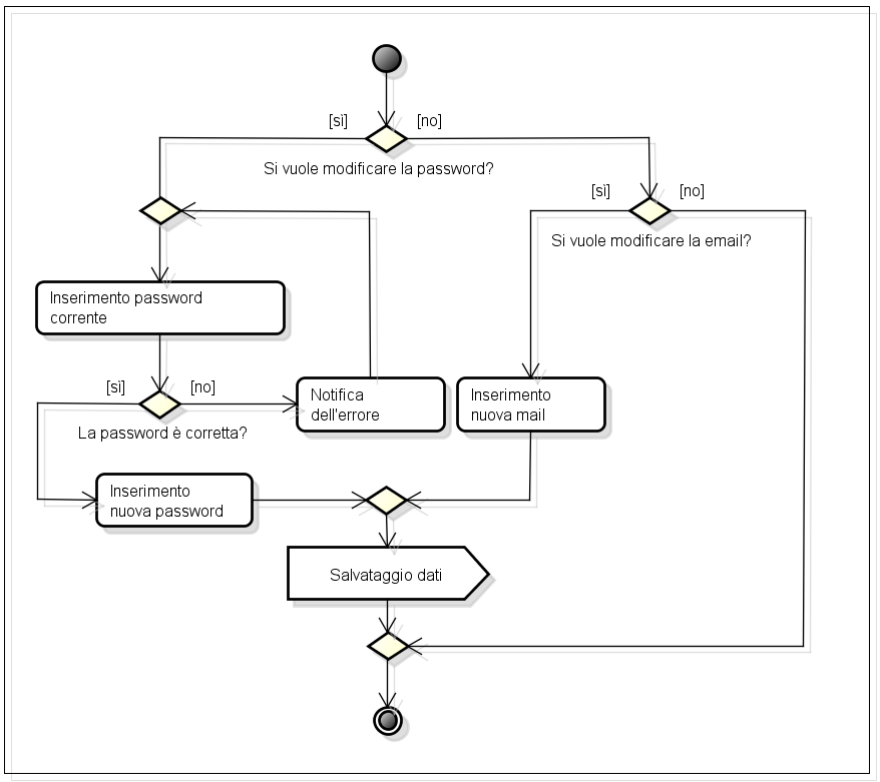


Figura 24: Attività user: Modifica dati utente

**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

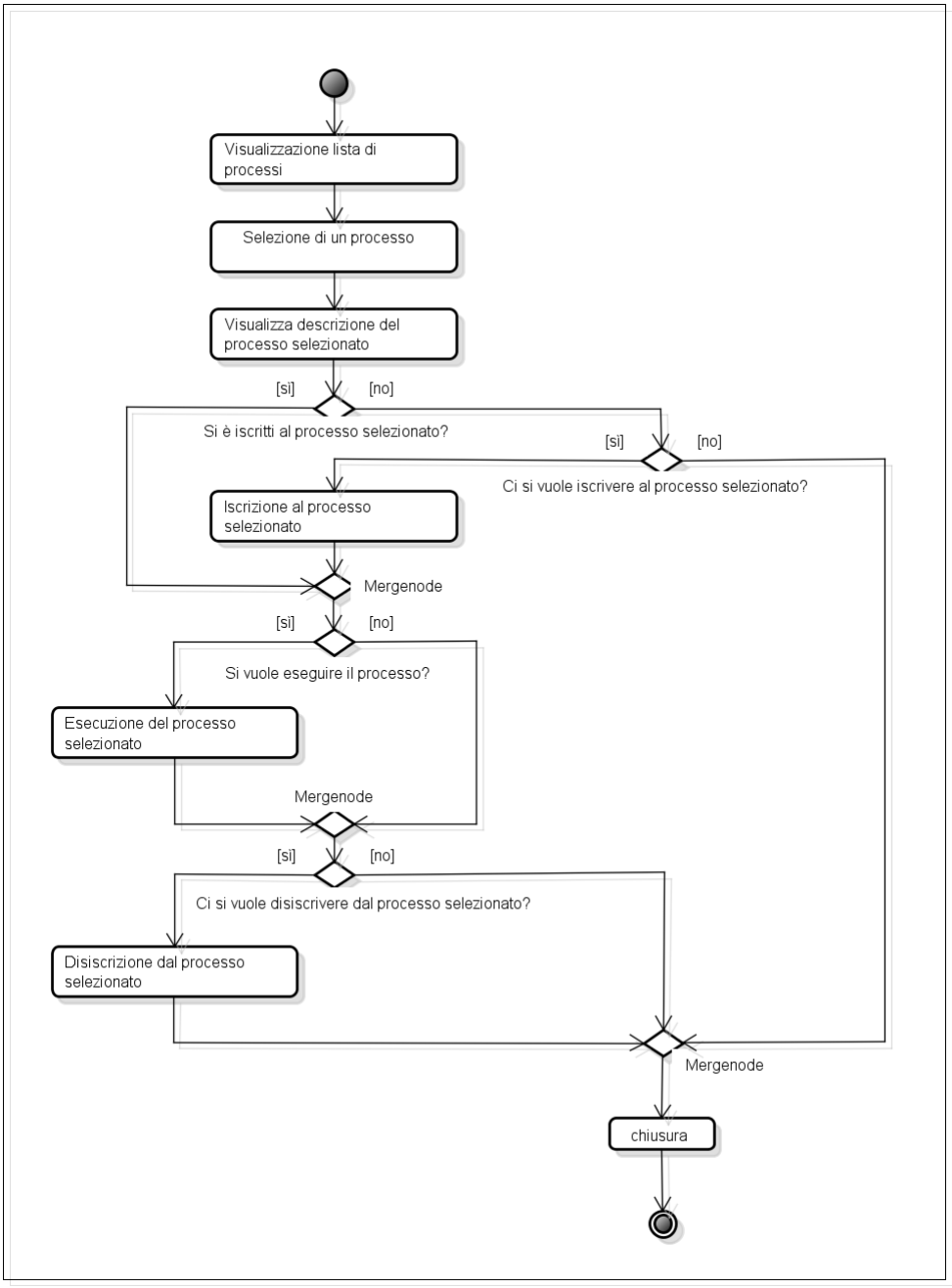


Figura 25: Attività user: Gestione dei processi

**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

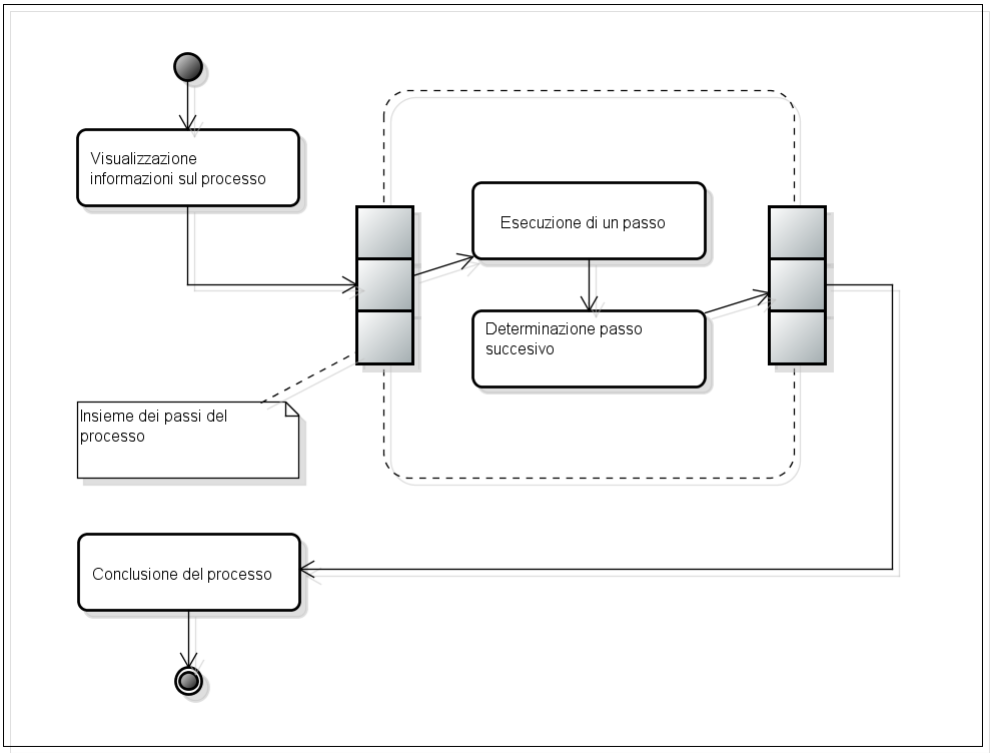


Figura 26: Attività user: Esecuzione di un processo

**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 successivo. 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

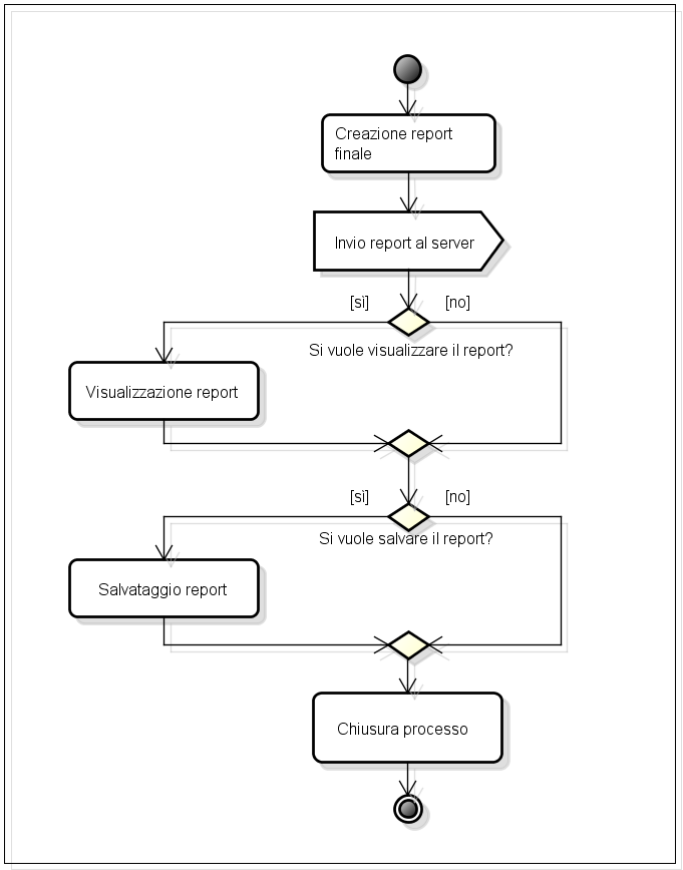


Figura 27: Attività user: conclusione di un processo

**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

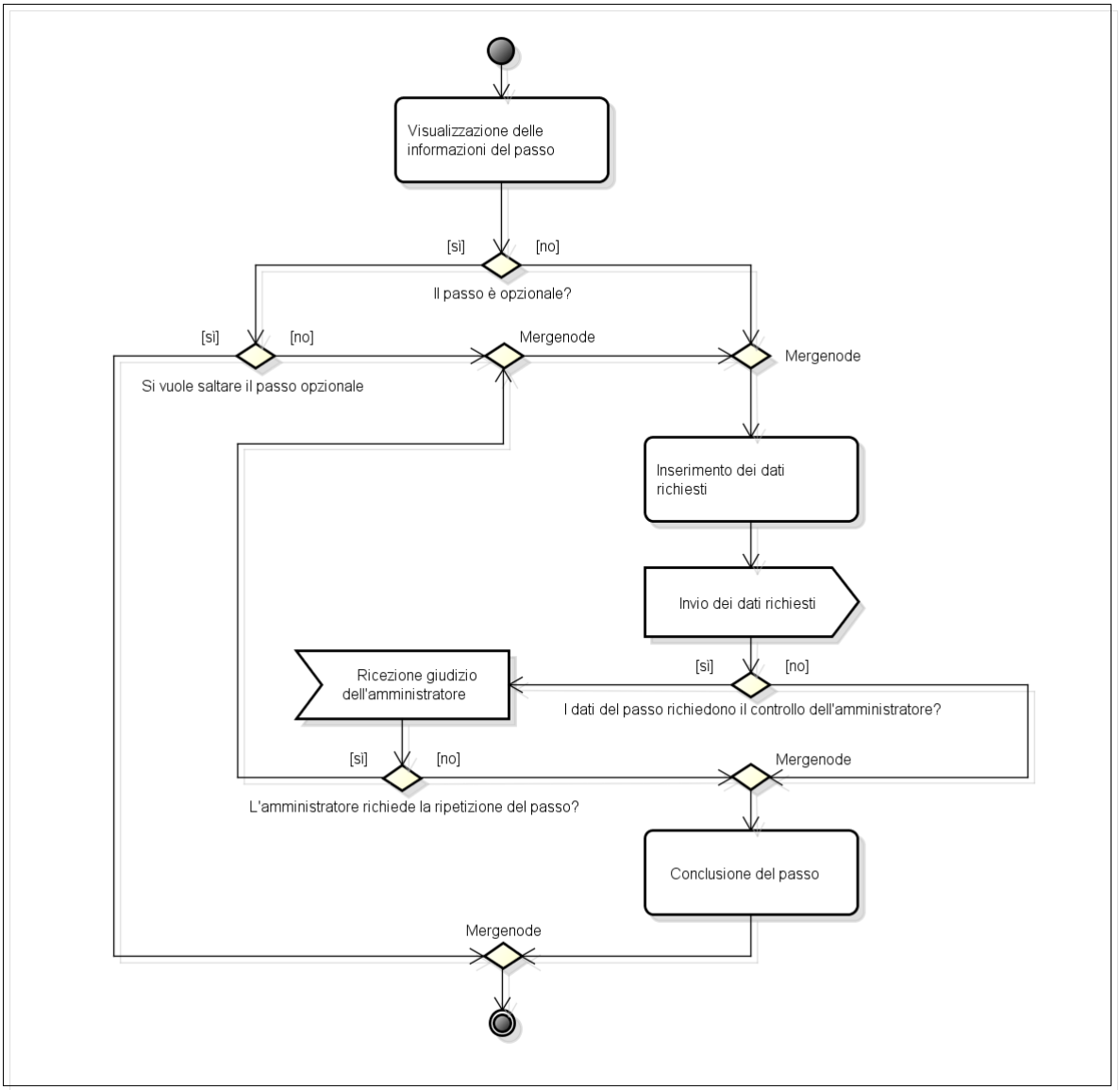


Figura 28: Attività user: Esecuzione di un passo

**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 |
|---|---|
| com.sirius.sequenziatore.client.view | V1 - `ILogin`<br>V2 - `Login` |
| com.sirius.sequenziatore.client.view.user | VU1 - `IMainUser`<br>VU2 - `MainUser`<br>VU3 - `IRegister`<br>VU4 - `Register`<br>VU5 - `IUserData`<br>VU6 - `UserData`<br>VU7 - `IOpenProcess`<br>VU8 - `OpenProcess`<br>VU9 - `IManagementProcess`<br>VU9 - `ManagementProcess`<br>VU10 - `ISendData`<br>VU11 - `SendData`<br>VU12 - `ISendText`<br>VU13 - `SendText`<br>VU14 - `ISendNumb`<br>VU15 - `SendNumb`<br>VU16 - `ISendPosition`<br>VU17 - `SendPosition`<br>VU18 - `ISendImage`<br>VU19 - `SendImage`<br>VU20 - `IPrintProcess`<br>VU21 - `PrintProcess` |
| com.sirius.sequenziatore.client.view.processowner | VA1 - `IMainProcessOwner`<br>VA2 - `MainProcessOwner`<br>VA3 - `INewProcess`<br>VA4 - `NewProcess`<br>VA5 - `IAddStep`<br>VA6 - `AddStep`<br>VA7 - `IOpenProcess`<br>VA8 - `OpenProcess`<br>VA9 - `IManageProcess` |

| | |
|---|---|
| | VA10 - `ManageProcess`<br>VA11 - `ICheckStep`<br>VA12 - `CheckStep` |
| com.sirius.sequenziatore.client.presenter() | P1 - `Router`<br>P2 - `Login` |
| com.sirius.sequenziatore.client.presenter.user | PU1 - `MainUser`<br>PU2 - `Register`<br>PU3 - `UserData`<br>PU4 - `OpenProcess`<br>PU5 - `ManagementProcess`<br>PU6 - `PrintReport`<br>PU7 - `SendData`<br>PU8 - `SendText`<br>PU9 - `SendImage`<br>PU10 - `SendPosition`<br>PU11 - `SendNumber` |
| com.sirius.sequenziatore.client.presenter.processowner | PA1 - `MainProcessOwner`<br>PA2 - `NewProcess`<br>PA3 - `AddStep`<br>PA4 - `OpenProcess`<br>PA5 - `ManageProcess`<br>PA6 - `CheckStep` |
| com.sirius.sequenziatore.client.model | M1 - `ProcessModel`<br>M2 - `ProcessDataModel`<br>M3 - `StepModel`<br>M4 - `UserModel` |
| com.sirius.sequenziatore.client.model.collection | MC1 - `ProcessCollection`<br>MC2 - `ProcessDataCollection`<br>MC3 - `StepCollection` |
| com.sirius.sequenziatore.server.presenter.communication | SP2 - `HttpCommunication`<br>SP3 - `WebsocketCommunication` |

| | |
|---|---|
| com.sirius.sequenziatore.server.presenter.user | SPU1 - `AccountManager`<br>SPU2 - `UserProcessManager`<br>SPU3 - `UserStepManager`<br>SPU4 - `Report` |
| com.sirius.sequenziatore.server.presenter.processowner | SPA1 - `Login`<br>SPA2 - `ProcessManager`<br>SPA3 - `StepManager`<br>SPA4 - `UserManager`<br>SPA5 - `report` |
| com.sirius.sequenziatore.server.model.daouser | SMU1 - `DataAccessObject`<br>SMU2 - `ObjectTransfer` |
| com.sirius.sequenziatore.server.model.daoprocessowner | SMA1 - `DataAccessObject`<br>SMA2 - `ObjectTransfer` |
| com.sirius.sequenziatore.server.model.daoprocess | SM1 - `DataAccessObject`<br>SM2 - `ObjectTransfer` |
| com.sirius.sequenziatore.server.model.daostep | SM3 - `DataAccessObject`<br>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<br><br>VU15, CPU6 | Il sistema dovrà permettere all'utente autenticato l'inserimento di una immagine richiesta | VU14 |
| 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<br>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 geolocalizzazione, fondamentale nello sviluppo del nostro sistema, oltre alle altre novità che introduce.

## A.2 CSS3

*CSS3* è un linguaggio *style sheet* verrà utilizzato per la rappresentazione grafica delle pagine, in modo da separarla dai contenuti. In questo modo verranno migliorate comprensione, manutenibilità e portabilità.

## A.3 Javascript

L' utilizzo di *Javascript*$_G$ è stato richiesto espressamente dal proponente, è un linguaggio di *scripting* non compilato ma interpretato direttamente dal *browser*$_G$. Nello sviluppo del nostro progetto ci permetterà di sviluppare l'applicazione lato lato *client*$_G$ e di comunicare con il *server*$_G$.

## A.4 Backbone.js

*Backbone.js* è un *framework* basato sul paradigma *model-view-presenter*, utilizzata per lo sviluppo dell'applicazione *Sequenziatore*. Il *framework* è particolarmente leggero e necessita come unica dipendenza della libreria *Underscore.js*. *Backbone.js* è creato per sviluppare applicazioni *web* di tipo *single page*, e consente di strutturare il codice, grazie alle classi *Model, View* e *Router* estendibili dal programmatore. Il gruppo *Sirius* ha scelto questo *framework*, in quanto si presta alle esigenze architetturali del progetto, e inoltre è molto ben documentato.

## A.5 Underscore.js

*Underscore.js* è una libreria necessaria al *framework Backbone.js*. Viene utilizzata in particolare per gestire la comunicazione tra *Backbone* e i *template* utilizzati nel *package view*.

## A.6 Require.js

*Require.js* è una libreria utilizzata per gestire le dipendenze tra le componenti e le librerie, e per implementare il *pattern Asynchronous Module Definition* . La libreria è stata scelta per l'ottima compatibilità con il *framework Backbone.js*.

### A.7 JQuery

*Jquery* è una libreria *Javascript* per applicazioni web. La libreria consente di interagire con gli elementi *DOM*, di gestire eventi e implementare funzionalità *AJAX*.

### A.8 JQueryMobile

Questa libreria verrà usata per lo sviluppo di un *front-end* per dispositivi di tipo *responsive*, accessibili da *smartphne*, *tablet* e computer. La scelta del team di questa libreria è data dal fatto che è affermata nel mondo del *web development*.

### A.9 JAVA 7

*Java*$_G$ 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.10 JSON

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

### A.11 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.12 MySQL

*MySQL*$_G$ 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.13 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*, rispettivamente il contenitore *servlet* e il connettore *HTTP*.