



SIRIUS

---

SEQUENZIATORE

**Definizione di prodotto**

**Versione 1.0.0**

*Ingegneria Del Software AA 2013-2014*

## Informazioni documento

---

Titolo documento:	Definizione Di Prodotto
Data creazione:	2014-01-29
Versione attuale:	1.0.0
Utilizzo:	Interno
Nome file:	<i>DefinizioneDiProdotto_v1.0.0.pdf</i>
Redazione:	Quaglio Davide
Approvazione:	Santangelo Davide
Distribuito da:	Sirius
Destinato a:	Prof. Vardanega Tullio Prof. Cardin Riccardo Zucchetti S.p.A.

## Sommario

Tale documento andrà a trattare in modo approfondito le componenti e la struttura del prodotto il *Sequenziatore* trattate nel documento *Specifica Tecnica\_v1.0.0.pdf*

## Diario delle modifiche

Versione	Data	Autore	Ruolo	Descrizione
----------	------	--------	-------	-------------

## Indice

<b>1</b>	<b>Introduzione</b>	<b>1</b>
1.1	Scopo del documento . . . . .	1
1.2	Glossario . . . . .	1
<b>2</b>	<b>Standard di progetto</b>	<b>1</b>
<b>3</b>	<b>Specifica della componente view</b>	<b>1</b>
<b>4</b>	<b>Specifica della componente presenter</b>	<b>1</b>
4.1	Client . . . . .	1
4.2	Server . . . . .	1
4.2.1	Package com.sirius.sequenziatore.server.presenter.common . . . . .	2
4.2.2	Package com.sirius.sequenziatore.server.presenter.processowner . . . . .	3
4.2.3	Package com.sirius.sequenziatore.server.presenter.user . . . . .	4
<b>5</b>	<b>Specifica della componente model</b>	<b>5</b>

## 1 Introduzione

### 1.1 Scopo del documento

In questo documento si prefigge come obiettivo la definizione in modo approfondito della struttura, delle componenti e delle relazioni tra queste ultime del prodotto il *Sequenziatore* approfondendo quanto riportato nel documento di Specifica Tecnica

### 1.2 Glossario

Al fine di facilitare la comprensione del seguente documento, ed in generale di ogni documento che verrà fornito da parte del team *Sirius*, è stato creato appositamente un glossario (*Glossario\_v2.0.0.pdf*) contenente la definizione dei termini più complessi o di quelli che necessitano un approfondimento. Questi vocaboli sono contrassegnati in ogni documento dal pedice G (<sub>G</sub>).

## 2 Standard di progetto

## 3 Specifica della componente view

## 4 Specifica della componente presenter

Questa componente racchiude tutti i metodi che gestiscono le varie operazioni offerte dall' applicazione il *Sequenziatore* e viene suddivisa in due parti: Client e Server

### 4.1 Client

### 4.2 Server

Questa componente è incaricata di gestire la comunicazione con il client e di elaborarne le richieste restituendo i dati richiesti e quando necessario interroga la componente model per ottenere i dati dal database. Tale componente è composta dalle classi:

- `com.sirius.sequenziatore.server.presenter.common.SignUpController`
- `com.sirius.sequenziatore.server.presenter.common.LoginController`
- `com.sirius.sequenziatore.server.presenter.common.StepInfoController`
- `com.sirius.sequenziatore.server.presenter.common.ProcessInfoController`
- `com.sirius.sequenziatore.server.presenter.processowner.StepController`
- `com.sirius.sequenziatore.server.presenter.processowner.ProcessController`
- `com.sirius.sequenziatore.server.presenter.processowner.ApproveStepController`

- `com.sirius.sequenziatore.server.presenter.user.AccountController`
- `com.sirius.sequenziatore.server.presenter.user.UserStepController`
- `com.sirius.sequenziatore.server.presenter.user.UserProcessController`
- `com.sirius.sequenziatore.server.presenter.user.ReportController`

Nella prossime sessioni verranno trattate in dettaglio le seguenti classi dividendo l'esposizione per *package*, si evidenzia come la voce mappatura base sia l'estensione della mappatura su cui si programma il sistema che sarà `localhost:8080/sequenziatore/`, quindi tutte le mappature base saranno da considerarsi come aggiunte a seguito di `/sequenziatore/` e successivamente le varie varianti dei metodi. Tutte le classi *controller* del *presenter* dovranno essere marcate come `@Controller` per essere riconosciute in modo corretto da Spring.

#### 4.2.1 Package `com.sirius.sequenziatore.server.presenter.common`

**IMMAGINE DEL PACKAGE** All'interno di questa sezione verranno trattate tutte le classi contenute nel package *common*.

##### 4.2.1.1 Classe `SignUpController`

**Descrizione** Questa classe dovrà gestire tutte le richieste di registrazione al sistema, sarà incaricata di inserire i dati nel database e di avvertire il client della riuscita della registrazione.

**Mappatura base:** `\signup`

**Attributi:**

**Metodi:**     • `+RegisterUser(Utente toBeRegistered)` , questo metodo gestirà un metodo **POST** e restituirà un errore qual'ora ci siano stati problemi nella registrazione;

##### 4.2.1.2 `LoginController`

**Descrizione** Questa classe gestirà le richieste di *log in*, dovrà controllare se l'utente esiste nel sistema e se le credenziali d'accesso siano corrette;

**Mappatura base:** `\login`

**Attributi:**

**Metodi:**     • `+CheckLogin(Utente toBeLogged)` , questo metodo gestirà un metodo di tipo **POST** , controllerà le credenziali di accesso e dovrà restituire un errore qual'ora ci siano stati problemi nella login;

#### 4.2.1.3 StepInfoController

**Descrizione** Questa classe restituirà lo scheletro, quindi la composizione del passo richiesto;

**Mappatura base:** `\step\{id}`

**Attributi:**

**Metodi:**   • `+Step GetStepInformation()` il metodo gestisce una richiesta di tipo **GET** restituendo la struttura del passo con id uguale all' id fornito dopo averla recuperata dal *database*;

#### 4.2.1.4 ProcessInfoController

**Descrizione**

**Mappatura base:** `\process\{id}`

**Attributi:**

**Metodi:**   •

### 4.2.2 Package `com.sirius.sequenziatore.server.presenter.processowner`

#### IMMAGINE PACKAGE

##### 4.2.2.1 StepController

**Descrizione** Questa classe dovrà fornire al *process owner* tutti i dati inseriti dagli utenti per un dato passo, quindi dovrà restituire una collezione di dati al process owner il quale potrà visionarli;

**Mappatura base:** `\stepdata\{idstep}\processowner`

**Attributi:**

**Metodi:**   •

##### 4.2.2.2 ProcessController

**Descrizione**

**Mappatura base:** `\process\controller`

**Attributi:**

**Metodi:**   •

#### 4.2.2.3 ApproveStepController

**Descrizione** Questa classe serve per fornire al *process owner* i dati da approvare e per gestire quali passi siano stati approvati quali no, qualora un passo non venga approvato, verrà rimosso dal *database*;

**Mappatura base:** `\approvedata`

**Attributi:**

**Metodi:**

- `+StepList GetStepToApprove()`, il metodo gestisce una richiesta di tipo *GET*, e restituirà un oggetto di tipo *StepList* contenente tutti i dati che richiedono approvazione;
- `+void ApproveResponse(Step)`, il metodo gestisce una richiesta di tipo *POST*, riceve un passo che ha subito la moderazione del *process owner*, tale passo verrà eliminato dal database se il processowner lo ha rifiutato altrimenti verrà approvato definitivamente;

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

##### IMMAGINE PACKAGE

#### 4.2.3.1 UserStepController

**Descrizione** Questa classe gestisce la ricezione di un passo da parte di un utente tramite una richiesta di tipo *POST*, tale passo dovrà essere inserito nel database, ponendo attenzione se è un passo che richiede approvazione o meno;

**Mappatura base:** `\stepdata\user`

**Attributi:**

**Metodi:** •

#### 4.2.3.2 UserProcessController

**Descrizione** Questa classe permette all'utente varie operazioni, innanzitutto l'iscrizione ad un processo, poi restituisce il passo a cui è arrivato e il suo stato per tale processo e infine fornisce una lista di processi con tutti i processi a cui si può iscrivere e i processi per i quali può chiedere di fare il *report*;

**Mappatura base:** `\user\{username}`

**Attributi:**



- Metodi:**
- +void ProcessSubscribe() questo metodo mappa su \subscribe\{processid} e gestisce una richiesta di tipo *POST* che permette ad un utente di iscriversi al processo voluto;
  - +Status GetProcessStatus() questo metodo mappa su \subscribe\{processid} e gestisce una richiesta *GET* che restituisce all'utente il proprio status per tale processo, restituendo il passo o i passi che può eseguire e quanti passi ha completato del processo;
  - +ProcessList GetListProcess() questo processo mappa su \processlist e gestisce una richiesta di tipo *GET* andando e restituire una lista di processi che contiene tutti i processi a cui è iscritto e quelli a cui si può iscrivere;

#### 4.2.3.3 ProcessInfoController

##### Descrizione

**Mappatura base:** \process\{id}

##### Attributi:

**Metodi:** •

#### 4.2.3.4 ProcessInfoController

##### Descrizione

**Mappatura base:** \process\{id}

##### Attributi:

**Metodi:** •

## 5 Specifica della componente model