REGIONE CAMPANIA – LINEE DI INDIRIZZO PER L'IMPLEMENTAZIONE DEL SISTEMA INFORMATIVO SANITARIO REGIONALE

Allegato 1B
Sinfonia - Architettura Generale del sistema applicativo Flussi

Versione 1.00

21 Settembre 2018

Indice dei Contenuti

1. Introduzione

1.1. Generalità

Il presente documento fornisce una panoramica generale dell'architettura di Sinfonia in termini di sistemi e componenti applicative che lo costituiscono e relativa organizzazione strutturale e tecnologica. Il documento definisce:

- gli elementi cardine dell'architettura dei sistemi applicativi oggetto di fornitura, partendo da quanto previsto nel progetto esecutivo;
- gli obiettivi architetturali ai quali si conformano le soluzioni adottate ed oggetto di fornitura;
- i modelli architetturali e i pattern di riferimento per i sistemi e le relative componenti applicative;

Il documento definisce gli obiettivi architetturali del sistema, i sistemi cooperanti, l'architettura tecnologica generale e la distribuzione dei componenti software applicativi ed infrastrutturali sui diversi nodi dell'infrastruttura. Viene inoltre definita l'architettura applicativa di ciascun sistema, mediante specificazione sui diversi layer dei pattern di sviluppo, delle tecnologie e degli standard e meccanismi necessari per assolvere agli obiettivi architetturali.

Esso formalizza le diverse decisioni architetturali e progettuali definite per le componenti preliminarmente alla realizzazione, configurazione ed adeguamento delle stesse ai requisiti di processo e funzionali definiti nel workpackage di "*Analisi e Progettazione*".

1.2. Scopo e Ambito di Applicazione

Il documento è il risultato delle activity del workflow di Analysis & Design previste dalla metodologia adottata e costituisce l'input anche per quelle relative al workflow di Implementation & Test; pertanto è destinato a tutti i ruoli coinvolti nelle attività relative a quest'ultimo workflow.

Definisce principalmente gli aspetti riguardanti la logical view, l'implementaion view e la deployemnt view dell'intero sistema. Tali viste vengono quindi integrate e "complementate" dalle viste dei casi d'uso e dei processi di cui ai documenti di specifica dei requisiti software e di architettura di ciascuna componente applicativa per costituire la "Definizione architetturale" complessiva del sistema Sinfonia.

1.3. Ciclo di vita del documento

Il documento sarà revisionato ed aggiornato a seguito dello svolgimento delle attività di sviluppo delle componenti software qualora vengano rilevati, in fase di definizione di tali artifact ulteriori obiettivi e vincoli architetturali che richiedano la revisione dell'architettura definita da questo documento.

1.4. Riferimenti

1. Progetto Esecutivo per So.Re.Sa. S.p.A. Società Regionale per la Sanità Regione Campania Rif. Consip ID SIGEF 1607.
2. Decreto Dirigenziale n. 131 del 20.06.2018.

1.5. Glossario

Modello	Rappresentazione concettuale del sistema ottenuta attraverso l'utilizzo di costrutti linguistici e semantici propri di un linguaggio standardizzato di modellazione (come ad esempio UML).
Package	Elemento del modello che rappresenta un contenitore di altri elementi quali classi, componenti, interfacce, diagrammi, package, ecc.
Componente (o Area applicativa)	Modulo o sistema applicativo oggetto di fornitura o di terze parti.
WEB Tier	Livello architetturale del sistema software dedicato alla interazione con l'utente attraverso tecnologia e protocolli Internet.
Documento HTML	Documento SGML che soddisfa i requisiti delle specifiche W3C.
Stylesheet	Specifica del formato di presentazione di un documento XML con descrizione della trasformazione (opzionale) della struttura del documento di ingresso in un'altra struttura e della modalità di visualizzazione degli elementi della struttura. Il linguaggio per definire uno stylesheet è XSL (eXtensible Stylesheet Language).

Tabella 1 - Glossario

1.6. Acronimi

JEE (Java Enterprise Edition)	Versione enterprise della piattaforma Java.
DAO (Data Access Object)	Pattern che ha lo scopo di disaccoppiare la logica di business dalla logica di accesso ai dati.
HTTP (Hyper Text Transfer Protocol)	Protocollo standard di trasferimento di un ipertesto.
MVC (Model-View-Controller)	Pattern architetturale per lo sviluppo di interfacce utente dei sistemi software.

SOAP	Specifica per lo scambio di informazioni strutturate nell'implementazione di Web Services in reti di computer. Il formato dei messaggi è l'XML.
HTML(HyperText Markup Language)	Linguaggio usato per descrivere la struttura dei documenti ipertestuali disponibili nel World Wide Web.
JSON (JavaScript Object Notation)	È basato sul linguaggio JavaScript Standard ECMA-262 3ª edizione dicembre 1999, ma ne è indipendente. Viene usato in AJAX come alternativa a XML/XSLT.
XML (eXtensible Markup Language)	Metalinguaggio standardizzato dal World Wide Web Consortium (W3C).
XHTML (eXtensible HyperText Markup Language)	Versione aggiornata ed estesa dell'HTML.
JVM (Java Virtual Machine)	Macchina virtuale che esegue programmi in linguaggio Java bytecode.
DB	Database.
UDDI (Universal Description Discovery and Integration)	Registry (base dati ordinata ed indicizzata), basato su XML ed indipendente dalla piattaforma hardware, che permette alle aziende la pubblicazione dei propri dati e dei servizi (Web services) offerti su internet.
SQL (Structured Query Language)	SQL (Structured Query Language) è un linguaggio standardizzato per database basati sul modello relazionale (RDBMS)
SOA (Service Oriented Architecture)	Architettura software atta a definire l'uso dei servizi per supportare le richieste degli utenti.
SCA (Scalable Cooperative Architecture)	Architettura cooperativa e scalabile
BPM (Business Process Management)	Il Business process management è l'insieme di attività necessarie per definire, ottimizzare, monitorare e integrare i processi aziendali, al fine di creare un processo orientato a rendere efficiente ed efficace il business dell'azienda.
ESB (Enterprise Service Bus)	Un Enterprise Service Bus (ESB) è un'infrastruttura software che fornisce servizi di supporto ad architetture SOA complesse. Un ESB si basa su sistemi disparati, interconnessi con tecnologie eterogenee, e fornisce in maniera consistente servizi di orchestration, sicurezza, messaggistica, routing intelligente e trasformazioni, agendo come una dorsale attraverso la quale viaggiano servizi software

	e componenti applicativi.
JAX-WS (Java API for XML Web Service)	JAX-WS (Java API for XML Web Services) è un insieme di procedure (API) del linguaggio di programmazione Java dedicate allo sviluppo di servizi web. L'insieme fa parte della piattaforma Java EE. Come altre API della Java EE, JAX-WS usa annotazioni, introdotte nella Java SE 5, per semplificare lo sviluppo e implementazioni di client e terminali di servizi web. JAX-WS fa parte del kit di sviluppo Java per web services (Java Web Service Development Pack – JWSDP) e include Java Architecture for XML Binding (JAXB) e SOAP.
JAX-B (Java API for XML Binding)	JAXB è una delle API della Java Enterprise Edition, fa parte del Java Web Services Development Pack (JWSDP) ed è una delle tecnologie di base del progetto Web Services Interoperability Technology (WSIT) promosso dalla Sun Microsystems; inoltre, a partire dalla versione 1.6, JAXB è inclusa anche in Java SE. Java Architecture for XML Binding (JAXB) permette agli sviluppatori Java di effettuare il mapping tra classi e una loro corrispondente rappresentazione XML. JAXB fornisce la possibilità di serializzare oggetti Java in XML (marshalling) e di effettuare l'operazione inversa (unmarshalling), cioè permette di ottenere a partire da un file XML la corrispondente rappresentazione a oggetti Java. JAXB permette quindi di manipolare file XML senza la necessità di implementare alcuna routine specifica per il salvataggio e la lettura di dati. JAXB 1.0 fu sviluppato nell'ambito del Java Community Process come JSR 31. Dal 2006, JAXB 2.0 viene sviluppata come JSR 222. L'implementazione delle specifiche di JAXB è rilasciata sotto licenza CDDL.
WSDL (Web Service Description Language)	Il Web Services Description Language (WSDL) è un linguaggio formale in formato XML utilizzato per la creazione di "documenti" per la descrizione di Web Service.
PEC (Posta Elettronica Certificata)	PEC è l'acronimo di Posta Elettronica Certificata. È un sistema di "trasporto" di documenti informatici del tutto simile alla posta elettronica "tradizionale", cui però sono state aggiunte le caratteristiche per

	garantire agli utenti la certezza, a valore legale, dell'invio e della consegna dei messaggi e-mail al destinatario.
JSP (Java Server Pages)	JavaServer Pages è una tecnologia di programmazione Web in Java per lo sviluppo della logica di presentazione (tipicamente secondo il pattern MVC) di applicazioni Web, fornendo contenuti dinamici in formato HTML o XML. Si basa su un insieme di speciali tag, all'interno di una pagina HTML, con cui possono essere invocate funzioni predefinite sotto forma di codice Java (JSTL) e/o funzioni Javascript. In aggiunta, permette di creare librerie di nuovi tag che estendono l'insieme dei tag standard (JSP Custom Tag Library).
JSTL (Java server pages Standard Tag Library)	JavaServer Pages Standard Tag Library (JSTL) è una libreria inclusa come componente della piattaforma software di sviluppo per applicazioni Web Java EE. È un'estensione di JSP ed incorpora un insieme di tag HTML definiti tramite file XML e programmati in linguaggio Java. Questa libreria è stata rilasciata da società di sviluppo software quali la Sun Microsystems, utilizzabili per la creazione di JavaServer Pages. In alternativa alle librerie di tag standard si possono creare librerie di tag personalizzati, chiamate Custom Tag Library
JPA (Java Persistence API)	Le Java Persistence API, sono un framework per il linguaggio di programmazione Java che si occupa della gestione della persistenza dei dati di un DBMS relazionale nelle applicazioni che usano le piattaforme Java Standard Edition (J2SE) e Java Enterprise Edition (J2EE).
CSS (Cascading Style Sheets)	Il CSS (Cascading Style Sheets, in italiano fogli di stile), in informatica, è un linguaggio usato per definire la formattazione di documenti HTML, XHTML e XML ad esempio i siti web e relative pagine web. Le regole per comporre il CSS sono contenute in un insieme di direttive (Recommendations) emanate a partire dal 1996 dal W3C.
EJB (Enterprise Java Bean)	In informatica gli Enterprise JavaBean (EJB) sono i componenti software che implementano, lato server, la logica di business di un'applicazione web all'interno dell'architettura/piattaforma Java EE espletando servizi a favore della parte di front-end

	ovvero per la logica di presentazione di un'applicazione web. Rappresentano dunque uno strato software residente su un application server all'interno di un'architettura software di tipo multi-tier. Le specifiche per gli EJB definiscono diverse proprietà che questi devono rispettare, tra cui la persistenza, il supporto alle transazioni, la gestione della concorrenza e della sicurezza e l'integrazione con altre tecnologie, come JMS, JNDI, e CORBA. Lo standard attuale, EJB 3, completato nella primavera del 2006, differisce notevolmente dalle versioni precedenti. Gli EJB necessitano di un EJB container tipicamente implementato all'interno degli application server assieme al servlet container per la parte di front-end.
Ajax (Asynchronous JavaScript and Xml)	In informatica AJAX, acronimo di Asynchronous JavaScript and XML, è una tecnica di sviluppo software per la realizzazione di applicazioni web interattive (Rich Internet Application). Lo sviluppo di applicazioni HTML con AJAX si basa su uno scambio di dati in background fra web browser e server, che consente l'aggiornamento dinamico di una pagina web senza esplicito ricaricamento da parte dell'utente. AJAX è asincrono nel senso che i dati extra sono richiesti al server e caricati in background senza interferire con il comportamento della pagina esistente. Normalmente le funzioni richiamate sono scritte con il linguaggio JavaScript. Tuttavia, e a dispetto del nome, l'uso di JavaScript e di XML non è obbligatorio, come non è necessario che le richieste di caricamento debbano essere necessariamente asincrone.
jQuery	jQuery è una libreria di funzioni Javascript per le applicazioni web, che si propone come obiettivo quello di semplificare la manipolazione, la gestione degli eventi e l'animazione delle pagine HTML. È un software liberamente distribuibile e gratuito, come previsto dalla licenza MIT.
JNDI (Java Naming and Directory Interface)	Java Naming and Directory Interface (JNDI) è una API Java per servizi di directory che ricopre un ruolo molto importante all'interno di un application server.

Tabella 2 - Acronimi

2. Vincoli ed obiettivi architetturali

Le soluzioni tecnologiche ed architetturali adottate per la definizione del sistema Sinfonia si basano sulla necessità di disporre di applicazioni che sono tra loro interoperabili e che permettono l'effettivo passaggio di informazioni tra gli attori coinvolti senza interruzioni di processo.

Il sistema informativo Sinfonia è implementato sulla base di uno stile architetturale orientato ai servizi (SOA) ove la cooperazione tra le varie componenti applicative di Sinfonia ed i sistemi esterni avviene attraverso l'esposizione e l'invocazione di servizi SOAP, orchestrati e coordinati attraverso uno strato ESB.

Una istanza del Sistema Applicativo Sinfonia è composta da uno strato applicativo di back end che espone servizi su cui è realizzato uno strato web che rende fruibili ad un utente finale le funzionalità offerte dal sistema. Le caratteristiche generali della soluzione si basano sugli strumenti tecnici ed architetturali che allo stato attuale appaiono più maturi, stabili e con elevato potenziale di crescita e diffusione come i modelli di comunicazione basata sugli standard nazionali per l'interoperabilità e cooperazione, i servizi per la sicurezza e la privacy basati su smart card e firma digitale, PEC.

Elementi fondamentali del modello architetturale sono:

- l'adozione della SOA (Services Oriented Architecture) e dei Web services che forniscono un approccio per la definizione, la pubblicazione e l'utilizzo dei servizi applicativi;
- gli accordi di servizio che specificano gli elementi funzionali e tecnici, necessari per l'invocazione dei servizi;
- lo strato ESB Enterprise Service Bus, che si occupa dell'orchestrazione e della coordinazione dei servizi offerti dalle varie componenti, oltre alle comuni facility previste da un ESB come messaggistica, routing, trasformazione dei servizi, security e single sign on.

Inoltre l'architettura del sistema si basa sui seguenti standard tecnologici di progetto e sviluppo:

- adozione di browser Internet standard per la visualizzazione dell'interfaccia utente dei servizi di back end;
- logica di presentazione web basata sull'impiego di pagine dinamiche (JSP) e delle tecnologie AJAX e JSON;
- logica di presentazione di tipo programmatico basata su Web Services SOAP;
- logica applicativa lato server implementata in architettura JEE, EJB3 e JPA.

L'accesso ai servizi offerti dal sistema Sinfonia è realizzato mediante:

- un sistema di front-end realizzato attraverso un portale che funge da entry point di Sinfonia e permette l'accesso a tutte le componenti applicative;
- un sistema di autenticazione utente di tipo SSO basato su un repository OpenLDAP ed un sistema CAS;
- un insieme di applicazioni web, realizzate mediante tecnologie enterprise come JSP con JSTL, jQuery, Ajax, CSS3 e (X)Html e fruibile dagli utenti mediante l'utilizzo di un browser web;

- un l'Enterprise Service Bus, che si occupa dell'orchestrazione e della coordinazione dei servizi offerti dalle varie componenti, oltre alle comuni facility previste da un ESB come messaggistica, routing, trasformazione dei servizi;
- un registro dei servizi UDDI (service inventory);
- i servizi di cooperazione applicativa, secondo lo standard SOAP, a favore sia di sistemi fruitori cooperanti appartenenti allo stesso dominio organizzativo che ospita il sistema informativo Sinfonia ma anche verso sistemi esterni.

Il modello adottato, fondato sui principi della cooperazione applicativa fra sistemi informativi eterogenei e sul concetto di dominio, prevede l'interconnessione di Sinfonia con lo strato ESB, allo scopo di abilitare l'accesso a tutti gli operatori sanitari della Regione e, in prospettiva, anche agli altri Enti.

La comunicazione tra i vari domini può essere mediata dalle Porte di Dominio, elemento di accesso standard alle risorse applicative dei domini interconnessi, capace di veicolare le richieste di servizio verso altri domini e, simmetricamente, di ricevere le richieste di servizio provenienti dagli altri domini. La sua capacità di interazione con il mondo esterno si fonda sulla comunicazione e sullo scambio di messaggi in formato XML su protocollo SOAP conforme alla Busta di e-government.

2.1. Obiettivi architetturali

Nel presente paragrafo vengono riportati gli obiettivi architetturali che saranno soddisfatti dal presente documento.

Denominazione	**Descrizione**
Standardizzazione dei service contract	Va assicurata la standardizzazione dei service contract, curando il modo in cui le funzionalità sono espresse, la definizione dei datatype, del modello dei dati e delle policy. Vi deve essere un'attenzione costante alla ottimizzazione ed alla appropriata granularità dei servizi, al fine di assicurare servizi consistenti affidabili e governabili.
Loose Coupling	Va garantita l'indipendenza del disegno e l'evoluzione della logica di business di ogni servizio e simultaneamente l'interoperabilità con i consumer dei servizi
Astrazione	Il disegno deve assicurare che i servizi occultino il più possibile i dettagli interni della loro implementazione
Riusabilità	I servizi devono essere progettati e realizzati in modo da favorire la loro riusabilità, considerandoli come risorse dell'organizzazione il più possibile agnostiche rispetto ad un contesto funzionale specifico
Autonomia	I servizi devo minimizzare la loro dipendenza sia per quanto concerne la logica che per l'ambiente di implementazione
Flessibilità	I componenti dovranno essere in grado di adeguarsi ai mutamenti tecnologici ed all'interazione con altri progetti presenti e futuri

Denominazione	Descrizione
Capacità di integrazione	I componenti di servizio (service component) sono il punto di partenza di un progetto ampio e complesso e dunque dovranno essere in grado di integrarsi, dal punto di vista tecnologico, con informazioni prodotte in sistemi diversi. A tal fine i componenti dovranno essere in grado di interfacciarsi con altri sistemi utilizzando standard riconosciuti.
Modularità	I componenti dovranno essere progettati, sia per quanto riguarda la parte hardware che la parte software, in maniera modulare per garantire una loro naturale evoluzione ed integrazione con altri sistemi
Affidabilità, robustezza e disponibilità	Il sistema deve essere disponibile in modo continuativo (H24 e 7x7), rispondendo agli SLA concordati; il sistema deve poter continuare a funzionare con adeguati livelli di alta affidabilità sfruttando la sua modularità anche in condizioni non specificate nei requisiti e in presenza di errori locali senza propagare i guasti a tutto il sistema
Manutenibilità	I componenti dovranno essere facilmente manutenibili; a tal fine il disegno progettuale dovrà essere chiaro, la documentazione completa e dovranno essere utilizzati software di base e strumenti di sviluppo ampiamente diffusi o standard de facto
Accesso utente tramite Front-End web-based	Tutti i componenti dovranno utilizzare schemi standard di applicativi
Semplicità d'uso	Il Front-End dovrà minimizzare l'intervento umano massimizzando, in ogni caso, l'ergonomia dell'interfaccia utente e favorire la facilità di utilizzo, presentando un ambiente intuitivo corredato di help on-line anche contestuale. Nella sua progettazione andranno tenute presenti le interfacce utente degli applicativi attuali in modo da minimizzare, per quanto possibile, il disorientamento iniziale degli utenti

Tabella 3 - Obiettivi architetturali

3. Architettura Generale di Sinfonia

Per la gestione dell'intero ciclo informatico e decisionale di una realtà costituita da organizzazioni ad elevata complessità gestionale, operativa e di processo come le organizzazioni che, a livello regionale e provinciale, governano tutti i processi della Sanità Pubblica (territoriale ed ospedaliera), sono necessarie soluzioni operative, di controllo e decisionali che possano garantire una visione in tempo reale della situazione di andamento della spesa, del bilancio, degli eventi sanitari ed epidemiologici, monitorando, per ciascun fenomeno, ogni singolo aspetto, con livelli di dettaglio e di aggregazione flessibili e variabili sulla base delle esigenze dell'utenza.

Il Sistema Sinfonia oggetto di fornitura si colloca, in Regione Campania, in un contesto di informatizzazione regionale in cui l'insieme delle soluzioni applicative ed infrastrutturali per il governo dei processi sanitari risulta essere "*variegato*" (e complesso) per tecnologie e pluralità di sistemi adottati nell'ambito dei diversi domini ed organizzazioni sanitarie e con una elevata tendenza alla distribuzione dei sistemi sul territorio.

In questo contesto si colloca l'intervento di Sinfonia con i suoi obiettivi di ***accrescimento dell'interoperabilità*** e ***del livello di federazione*** tra sistemi (e tra componenti applicative) e di ***allineamento***, per l'intera organizzazione sanitaria, indipendentemente dai vincoli territoriali e di dominio organizzativo, ***degli obiettivi e dei processi***, già nelle fasi di progettazione dei servizi applicativi.

La risposta a tali obiettivi è rappresentata dalla implementazione di soluzioni architetturali orientate ai servizi (SOA), che consentono di realizzare nuovi applicativi ed integrare applicativi esistenti all'interno di un processo di ridisegno complessivo del sistema IT.

4. Architettura del Sistema Sinfonia (Gestione Flussi)

Per ogni istanza Sinfonia il software applicativo è distribuito sui seguenti tier:

- Presentation Tier
- Business Tier

4.1. Presentation Tier

Il Presentation Tier ha la responsabilità di presentare i servizi sia in forma programmatica (Web Services) sia come Web Application per i servizi applicativi e sia come report.

I componenti di questo tier sono:

- Presentation Logic Web
- Presentation Logic Web Services
- Report

4.1.1 Presentation Logic della User Interface

La Presentation Logic della User Interface implementa l'interfaccia Web dei sistemi ed ha la responsabilità della presentazione dei servizi applicativi tramite interfaccia Web (XHTML) ad un operatore accreditato tramite una Workstation dotata di browser web.

La sua implementazione si basa sul Web-tier di un Application Server JEE, costituito da un Web Server e un Web Servlet Container che costituisce il contenitore standard per i componenti di front end della tecnologia JEE: Servlet e Java Server Pages (JSP).

Questo componente si occupa del dispatching delle richieste HTTP provenienti dal browser web e provvede alla generazione dinamica delle pagine (X)HTML di risposta. Ha la responsabilità della uniformità dell'interfaccia grafica, della gestione del page flow e del mantenimento dello stato conversazionale con il client.

La generazione dinamica delle pagine (X)HTML di risposta ed il controllo del flusso delle pagine web generate dinamicamente avviene seguendo il modello definito dal pattern MVC (Model View Controller).

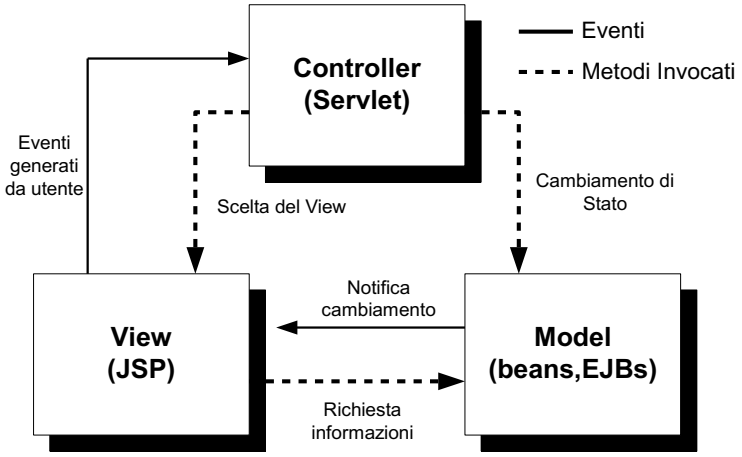

Figura 1 - Il Pattern MVC

In questo modello generale la Servlet (Controller) si comporta da motore web: in base all'input dell'utente, sul quale esegue una verifica di consistenza, decide quale processo di business invocare e seleziona la View successiva.

La View è realizzata attraverso pagine JSP che dinamicamente costruiscono il loro contenuto in base ai cambiamenti avvenuti nel Model, vale a dire recupera il risultato della transazione innescata dal Controller e lo visualizza all'utente.

Il Model è il sistema contenente la logica di business con cui l'utente interagisce innescando processi atti a conoscerne o alterarne lo stato.

L'uso del pattern MVC permette il completo disaccoppiamento fra la logica di presentation implementata nelle pagine JSP e la logica di controllo implementata nel codice della Servlet.

La specifica implementazione di MVC per i Servizi Applicativi prevede che:

- il Model renda disponibile il risultato di una transazione al Controller che a sua volta lo conserva nella sessione applicativa dell'utente;
- la JSP raccolga e renda visibile tale risultato all'utente titolare della sessione;

In particolare, il pattern è implementato mediante l'utilizzo del framework Spring MVC.

Nell'architettura dei servizi applicativi dei sistemi il ruolo del Model specificato nel pattern MVC è ricoperto, in linea con quanto dettato dal pattern Business Delegate, da un componente definito EJB Delegate a cui viene delegata la lookup, tramite JNDI, degli EJBs nell'EJB-tier che realizzano il Business Tier, disaccoppiando quindi in maniera completa la logica di controllo e di presentation dalla logica applicativa e di integrazione.

Rientra nella responsabilità del Controller la gestione del controllo di consistenza dei dati inseriti dall'utente e la gestione di situazioni anomale con presentazione all'utente di una pagina di errore. E' previsto l'uso di un Controller per ogni singolo caso d'uso individuato in fase di analisi delle componenti e, al fine di rendere più modulare e manutenibile il codice, le responsabilità del Controller sono distribuite su più classi di supporto.

I dati manipolati nello strato di *presentation* sono modellati da classi di business (Business Object) che rappresentano le entità del dominio.

Le pagine JSP fanno uso di codice JavaScript eseguito lato client, ad esempio per effettuare controlli sui dati inseriti dall'utente o per automatizzare interazioni tra l'utente e l'applicazione web. L'eventuale disattivazione del supporto JavaScript del browser non risulta comunque bloccante per l'applicazione.

Il controllo dell'accesso all'applicazione da parte dell'utente viene delegato a componenti applicative di Autenticazione e Autorizzazione Role Based secondo le modalità descritte nei relativi paragrafi.

Si riportano nel seguito le descrizioni dei pattern architetturali che saranno adottati per l'implementazione delle componenti della Web Application all'interno del Presentation-tier.

La tabella dell'Allegato 1, riporta i package della presentation logic con le relative responsabilità.

4.1.1.1 MVC e Business Delegate Web-tier

I pattern architetturali MVC e Business Delegate qui descritti vengono adottati per la realizzazione di tutti i casi d'uso che prevedono l'iterazione con l'utente mediante interfaccia web.

Tutti i casi d'uso sono realizzati coerentemente con lo schema riportato in figura.

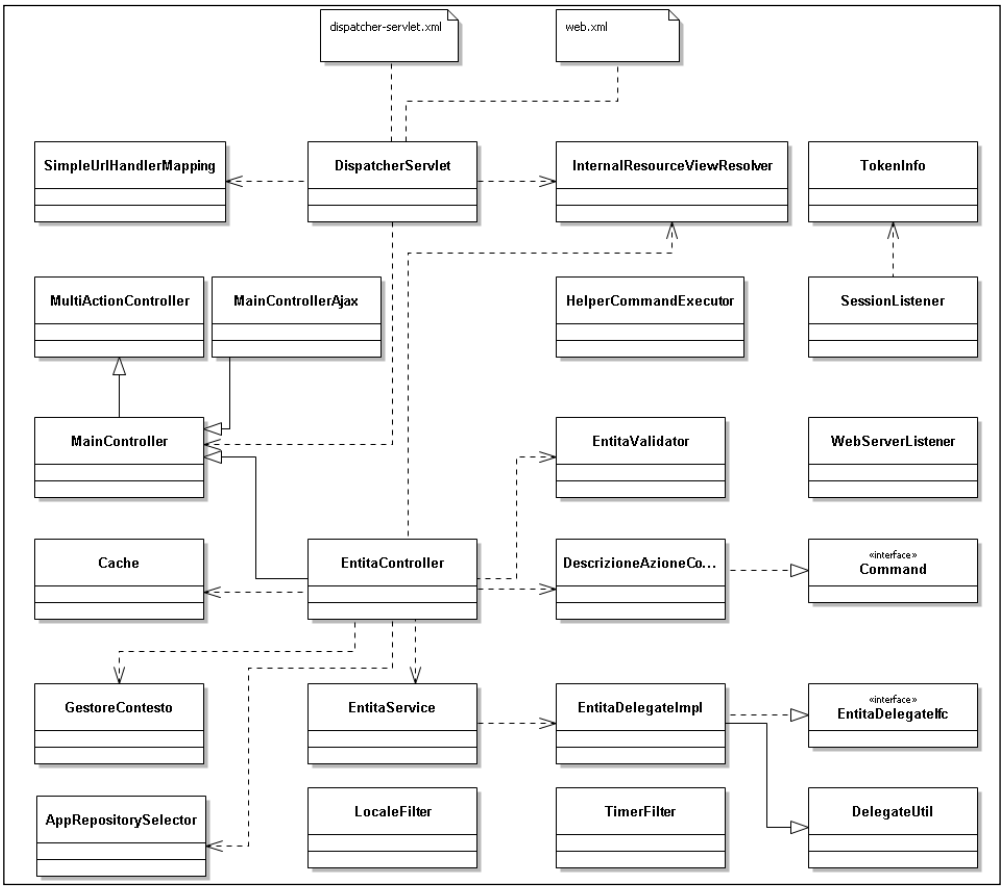

Figura 2 - Pattern MVC e Business Delegate

La tabella in Allegato 2 riporta la definizione delle responsabilità delle classi riportate in figura.

4.1.2 Presentation Logic dei Web Services

La Presentation Logic dei Web Services ha la responsabilità di esporre i servizi applicativi tramite interfaccia programmatica basata su standard SOAP, realizzati con tecnologia JaxWs.

Lo scheletro di un controller Spring JaxWs, con le relative annotazioni necessarie a renderlo un web service viene tipicamente costruito automaticamente dall'ambiente di sviluppo. Inoltre annotazioni aggiuntive consentono di espandere le funzioni infrastrutturali del servizio come ad esempio la gestione trasparente di aspetti come la gestione degli errori (fault), il log e la sicurezza.

Ciascun web service sarà quindi costituito delle seguenti componenti principali:

- Controller Spring JaxWs
- VO (Value Object)
- Ws Client (Se i sistemi invocano un web service JaxWs)

4.1.2.1 Controller Spring JaxWs

Per poter esporre un servizio di cooperazione mediante un web service JaxWs è necessario definire classi Java denominate Controller Spring JaxWs, che ne implementano il comportamento. Tali classi di implementazione fanno uso delle java-annotation standard Spring.

Ogni metodo del Controller JaxWs implementa una operation del web service, o meglio uno specifico servizio JaxWs. Il nome del metodo corrisponderà al nome del servizio JaxWs esposto e quindi un client che lo invocherà dovrà effettuare una chiamata utilizzandone quel nome di servizio.

4.1.3 Report

Per la realizzazione dei report viene utilizzato JasperReports, uno strumento open source sviluppato in Java in grado di produrre reportistica secondo numerosi formati: PDF, HTML, Microsoft Excel, RTF, ODT, CSV, XML, ecc..

Per la progettazione dei report sono disponibili diversi strumenti di disegno, sia stand-alone che integrati negli ambienti di sviluppo come Eclipse. Il modulo di runtime che si occupa della generazione dei report è integrato nello strato di presentation delle componenti applicative.

JasperReports consente di accedere ad una o più fonti di dati (data source), tipicamente database remoti, tramite la specifica di riferimenti, nella forma di stringhe di connessione standard, che utilizzano i driver presenti sul sistema in cui il software andrà in esecuzione.

4.2. Business e Data Tier

Il Business Tier ha la responsabilità di realizzare la logica di business dell'applicazione. L'organizzazione delle classi di questo strato è per entità.

Per ogni entità è stato definito un enterprise java bean (EJB) di tipo session stateless. In particolare, si utilizzano gli EJB3 che, a mezzo annotations, permettono di definire un EJB prescindendo da onerosi file di configurazione (ejb-jar.xml, jboss.xml), nonché di utilizzare un sistema di Object Relational Mapping (ORM) standard denominato Java Persistence API (JPA).

A ciascuna classe EJB sono associate opportune classi di supporto che, contengono la logica applicativa per i servizi di lettura e transazionali.

L'accesso al database da parte degli EJB avviene tramite specifiche indicate dalle JPA ed Hibernate è il provider individuato per la loro implementazione. Tutte le transazioni sono 'transaction-scoped' ossia, il contesto di persistenza viene propagato per tutta la transazione e ha vita fino a quando l'intera transazione non si conclude. Il contesto di persistenza viene creato quando il metodo del primo Stateless Session Bean viene invocato e la transazione finisce, con essa il contesto di persistenza, quando questo metodo termina.

Per l'implementazione della logica di business vengono utilizzati Session EJB di tipo Stateless per due ragioni:

- le richieste provenienti dal Presentation Tier sono di tipo stateless;
- gli Stateless Session EJB sono più efficienti e performanti.

In Allegato 3 si riportano i package che realizzano la logica di business di tutte le componenti applicative.

Si riportano nel seguito la descrizione del pattern architetturale adottato per l'implementazione delle componenti della logica di business delle componenti dei sistemi applicativi di cui trattasi.

4.2.1 Session Facade EJB-tier

Lo schema che segue rappresenta il pattern utilizzato per l'implementazione della logica di business.

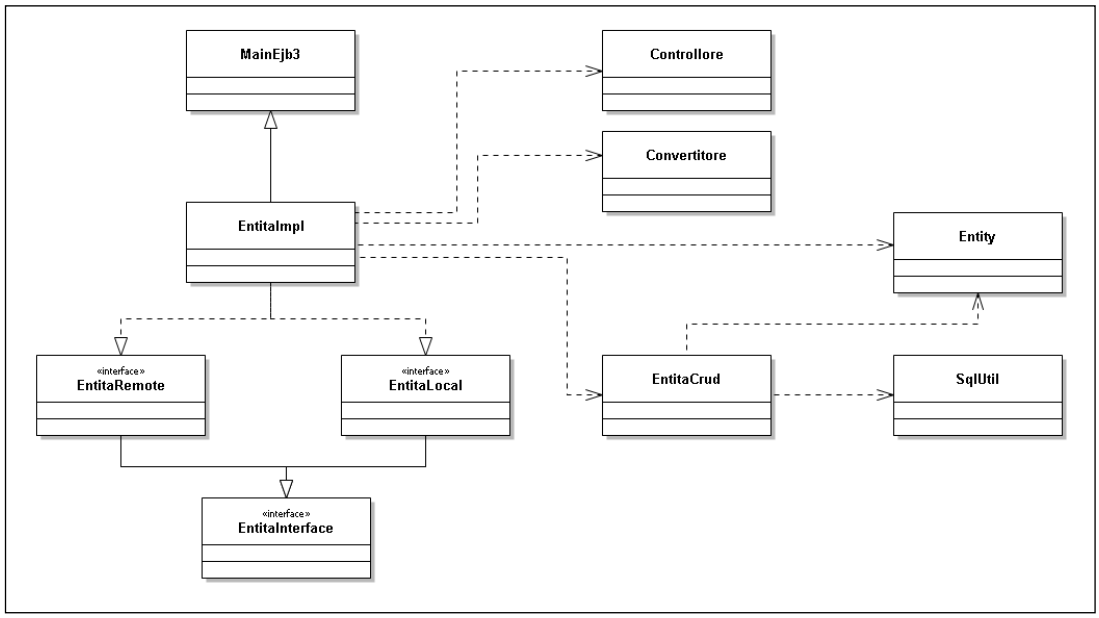

Figura 3 - Pattern Session Façade EJB-tier

La logica delle componenti che implementano l'EJB-tier è allocata nei package di cui all'allegato 4.

4.3. Elaborazioni Batch

Per elaborazione batch si intende un qualsiasi processo la cui esecuzione è asincrona rispetto alla richiesta attivata dall'utente mediante la web application e che durante la sua esecuzione non richieda interazione con l'utente. Pertanto non necessita di un ambiente operativo J2EE essendo sufficiente l'utilizzo di una JVM.

I casi d'uso che prevedono tali elaborazioni consentono all'utente, tramite interfaccia web, di richiedere una elaborazione batch consentendogli di esplicitare, se necessario, gli opportuni parametri di input. Successivamente un operatore di back office analizza le richieste pervenute e avvia l'esecuzione del processo batch. A processo concluso viene memorizzata su database l'avvenuta esecuzione del processo e l'esito (elaborato con successo o elaborato con errore). Lo stato della elaborazione viene reso disponibile all'utente mediante una funzionalità ad hoc.

L'autorizzazione dell'operatore di back office all'avvio di un processo batch è conseguente alla sua identificazione e autenticazione, mediante username e password, al sistema operativo ed al DBMS.

Spring Batch è un framework leggero che fornisce una solida base su cui costruire applicazioni batch robuste e scalabili.

Fornisce agli sviluppatori una serie di modelli collaudati che risolvono i problemi di batch comuni e consente agli sviluppatori di concentrarsi maggiormente sui requisiti di business e meno sulla complessità dell'infrastrutture batch.

Spring Batch contiene una grande varietà di componenti "out of box" configurabili che possono essere utilizzate per soddisfare la maggior parte dei più comuni casi di utilizzo batch. Una ampia configurazione XML e un modello di programmazione estensibile consentono una grande possibilità di personalizzazione delle componenti batch che possono quindi essere utilizzate come moduli per fornire rapidamente funzionalità comuni.

L'architettura del framework Spring Batch è quella degli ETL, ovvero Extract, Transform and Load.

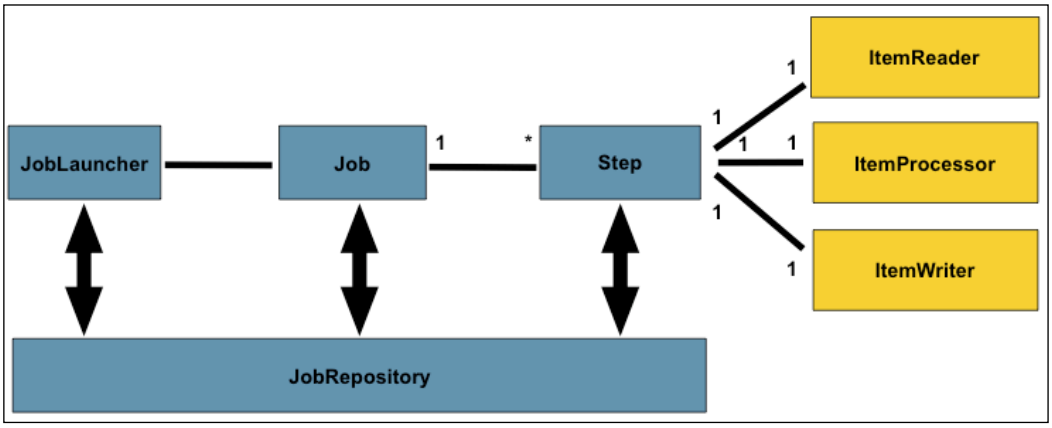

Figura 4 - Architettura Spring Batch

Ovvero un batch è definito come un Job (gestito da un repository), composto da n Step, ognuno dei quali esegue un processo di Lettura (Elaboration => ItemReader), Processazione (Transformation => ItemProcess) e Scrittura (Load => ItemWriter).

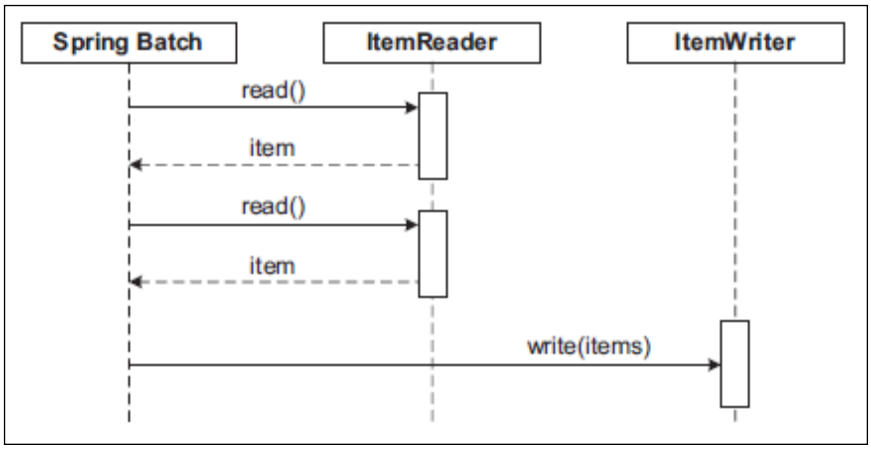

Figura 5 - Sequence Diagram Elaborazioni Batch

Dal punto di vista architetturale, le componenti che entrano in gioco nella richiesta e nell'esecuzione del batch sono distribuite su due layer:

- **Presentation Layer** che realizza i casi d'uso di richiesta di elaborazione batch;
- **Business Layer** che contiene la logica di start e di esecuzione del batch, che può essere un processo batch oppure un report batch, cioè un report non interattivo.

4.3.1 Presentation Layer

Il presentation layer è costituito dalle componenti di interfaccia web che consentono all'utente di richiedere l'esecuzione asincrona di un processo batch, fornendo gli opportuni parametri al processo. Tale richiesta, che è a tutti gli effetti un caso d'uso del sistema, viene memorizzata insieme ai parametri di input su database. In un momento successivo la richiesta viene presa in carico da un operatore di back office che avvia il processo batch, ne controlla l'esecuzione, ne analizza l'output e tramite interfaccia web può apporre un flag "visto" alle richieste evase nella tabella delle richieste dove è anche presente un flag di stato che prevede i seguenti valori:

1. "IN ELABORAZIONE" all'atto della richiesta di esecuzione del batch;
2. "ELABORATO CON SUCCESSO" dopo la corretta esecuzione del batch;
3. "ELABORATO CON ERRORE" se si sono verificati errori nella elaborazione del batch.

Ciascun processo batch provvede a registrare il proprio cambio di stato e produce un file di log consultabile dall'operatore di back office.

L'operatore che aveva effettuato la richiesta di esecuzione del processo batch può controllarne l'avvenuta esecuzione tramite interfaccia web, e, se previsto, eseguire il download del risultato del processo (report).

Dal punto di vista architetturale il presentation layer è del tutto simile ad un qualsiasi altro caso d'uso. Quindi per i casi d'uso di richiesta elaborazione batch valgono le scelte progettuali e architetturali effettuate per il presentation layer dell'intero sistema.

4.3.2 Business Layer

Il Business Layer delle elaborazioni batch si divide in due categorie diverse sia per il tipo di processo, sia per il risultato prodotto sia per la tecnologia adottata:

- **Processo Batch**: elaborazione che produce dati che vengono memorizzati sul db;

- **Report Batch:** elaborazione che produce file report in formato ASCII o pdf.

4.3.2.1 Processo Batch

Un processo batch:

- elabora dati presenti nel database in funzione dei parametri input forniti dall'utente che ha richiesto l'esecuzione del processo batch, anch'essi presenti nel database;
- produce risultati che vengono memorizzati nel database.

Un processo batch è costituito da un programma Java avviato da console, in un momento successivo alla richiesta, da un operatore di back office. E' un processo la cui esecuzione avviene all'interno di una macchina virtuale Java JVM standard J2SE. Per motivi di efficienza, dato il massiccio uso di accessi al database, la JVM di esecuzione può essere quella di una macchina in connessione intranet con Database Server se non una JVM presente sul database server.

4.3.2.2 Report Batch

Un report batch:

- elabora dati presenti nel database;
- l'elaborazione è in funzione dei parametri input forniti dall'utente che ha richiesto l'esecuzione del report batch, anch'essi presenti nel database;
- il layout e la logica di reporting del report sono definiti in un file con estensione .xml che risiede in un path accessibile dalla macchina in cui risiede la JVM di esecuzione del report batch;
- produce, come risultato della elaborazione un report, cioè un file di dati in formato conforme alle specifiche definite per il report; i dati sono disposti con un layout predefinito in fase di progettazione del report.

Il report batch è costituito da un programma Java avviato da console, in un momento successivo alla richiesta, da un operatore di backoffice, quindi è un processo la cui esecuzione avviene all'interno di una macchina virtuale Java JVM standard J2SE. Per motivi di efficienza, dato il massiccio uso di accessi al database, la JVM di esecuzione può essere quella di una macchina in connessione intranet con Database Server se non una JVM presente sul database server.

Tale programma utilizza le librerie di BIRT per integrare nell'applicazione la generazione dei report.

Il file con estensione .xml, che definisce il layout di un particolare report, risiede in un path accessibile dalla macchina in cui risiede la JVM di esecuzione del report batch ed è il prodotto della fase di design del report.

Il design di ogni report viene eseguito con l'ausilio del tool di sviluppo Report Designer all'interno dell'interfaccia IDE (Interactive Development Environment) di Eclipse.

In allegato 5 vengono riportati i package che contengono le classi di reporting delle componenti applicative.

4.4. Gestione Utenti ed Integrazione con sistema di SSO

Il sistema consente di accedere a tutti i servizi offerti, con meccanismi di autenticazione basati su username / password o smart card crittografiche (CIE, CNS, CRS, etc.). Il sistema gestisce,

con un'interfaccia, la gestione applicativa di ruoli e profili, che sarà possibile scegliere in fase di accesso. Gli utenti delle singole applicazioni presenti in Sinfonia accedono al sistema tramite un unico Front-End Web, la cui implementazione e autenticazione viene demandata al WSO2 Identity Server che interagisce in maniera federata con il Single-Sign On (SSO) regionale. Si rimanda a tale paragrafo per ulteriori approfondimenti in merito.

4.4.1 Identificazione, autenticazione ed autorizzazione degli utenti

L'identificazione, l'autenticazione e l'autorizzazione costituiscono i passi del processo attraverso il quale una entità accerta la corretta, o presunta, identità digitale di un utente.

Tutte le componenti applicative di Sinfonia per la fase di identificazione e autenticazione si integrano con il sistema di SSO.

La componente *Gestione Utenti*, invece, fornisce i servizi di amministrazione necessari a definire e profilare utenti nonché i servizi per l'autorizzazione dell'utente all'utilizzo delle singole funzionalità/servizi e alla visibilità di dati sensibili. La componente inoltre ha la responsabilità di produrre la reportistica analitica e riepilogativa relativa all'utilizzo dei servizi, all'assegnazione dei ruoli ed alla distribuzione degli utenti rispetto a ruoli e aziende sanitarie.

4.4.1.1 Definizione e Profilazione degli Utenti

La definizione e la profilazione di un utente è basata sulla sua identità e sui ruoli, detti Ruoli Istituzionali, che l'utente può assumere all'interno di una o più strutture.

La funzionalità di definizione e profilazione dell'utente fornisce la possibilità, ad un operatore autorizzato tramite interfaccia web, di definire o modificare più corrispondenze fra identità dell'utente, Ruolo Istituzionale e struttura in cui quel ruolo viene ricoperto.

La definizione e profilazione dell'utente in questi termini pone le basi per il processo di autorizzazione basata su ruolo e rende il meccanismo utilizzabile sia nel contesto Regionale che nel contesto aziendale.

4.4.1.2 Controllo delle autorizzazioni utente nello strato web

Il processo di definizione delle autorizzazioni è fondato sui Ruoli Istituzionali ed alla attribuzione a ciascun Ruolo Istituzionale di uno o più Ruoli Operativi. Un Ruolo Operativo viene definito come raggruppamento logicamente coerente di servizi.

Il sistema autorizzerà l'utente alla fruizione di un servizio solo se tale servizio è associato al Ruolo Operativo attribuito al Ruolo Istituzionale ricoperto dall'utente stesso (assegnato nella fase di definizione e profilazione).

Nell'ambito dello strato di presentation è stato utilizzato il meccanismo di verifica delle autorizzazioni offerto dal framework Spring Security basato su specifiche annotazioni da impiegare nel codice sorgente dei Controller.

4.5. Sicurezza accesso ai dati su DB

Poiché il database contiene i dati, la sua protezione è un aspetto di centrale importanza. In generale è sempre opportuno adottare ulteriori meccanismi di sicurezza esterni al DB, ma comunque aggiuntivi a quelli tipici del RDBMS. Oracle protegge i dati lì dove vengono conservati, all'interno del database, garantendone la protezione a un livello estremamente

elevato. L'RDBMS Oracle offre numerose funzionalità di sicurezza, dall'autenticazione utente alla gestione del privilegio ed al controllo dell'accesso.

4.5.1 Identificazione, autenticazione ed autorizzazione

Gli utenti accederanno al database per il tramite dell'application server che si connetterà al database mediante un pool di connessioni ed utilizzando una specifica utenza. Pertanto il DBMS si *limiterà* ad identificare ed autenticare l'application server verificando le autorizzazioni (Grant) di accesso ai dati in base al ruolo operativo dell'utente.

Il database dei sistemi gestisce il sistema delle autorizzazioni verificando i privilegi assegnati dal database administrator ai ruoli operativi. Quindi, dopo la fase di identificazione ed autenticazione, l'utente può eseguire operazioni su un oggetto del database solo se è stato espressamente autorizzato dall'amministratore. Per garantire la sicurezza e la privacy dei dati, saranno accordati all'utente solo i privilegi di cui necessita per svolgere le proprie funzioni, secondo il principio del "privilegio minimo". Riassumendo, i sistemi sono provvisto di un doppio sistema di profilazione dell'utente: uno al livello applicativo (solo l'utente autorizzato ad una specifica funzione potrà utilizzare la relativa funzionalità dell'applicazione) e il secondo all'interno del database (pur autorizzato al livello applicativo, vengono accordati all'utente i particolari privilegi sui dati necessari allo svolgimento del suo ruolo). Quindi, qualora un utente del sistema riuscisse anche ad aggirare il sistema di autorizzazione dell'applicativo, non riuscirebbe ad utilizzare le funzionalità in quanto non avrebbe le necessarie autorizzazioni al livello di database. Tale sistema garantisce quindi una profilazione dell'utente robusta a garanzia del principio di necessità nel trattamento dei dati (art.3 codice della privacy) che costituisce la precondizione di qualsiasi Sistema Informativo per la garanzia dei dati personali.

4.6. Cifratura

Alcuni requisiti di legge richiedono particolari misure quando dati personali o identificativi siano abbinati a informazioni di tipo sensibile. Ad esempio, l'accesso al nome dell'assistito in quanto tale può non richiedere particolari precauzioni, ma la combinazione del nome o del dato identificativo con informazioni di tipo sensibile può richiedere ulteriori misure di sicurezza come la crittografia.

I meccanismi di cripting dei dati sensibili al livello fisico nel database difendono da eventuali attacchi alla sicurezza che dovessero sopraggiungere dall'esterno del database stesso (ad es. qualcuno che riuscisse ad accedere direttamente al livello di Sistema Operativo ai datafile del database bypassando tutti i meccanismi di sicurezza messi a disposizione da Oracle).

Per questa eventualità il sistema RDBMS si avvale della feature **"Oracle Transparent Data Encription**" mediante il quale si può implementare in maniera trasparente il processo di cifratura dei dati sensibili direttamente nel motore RDBMS. Tale meccanismo consente di applicare la cifratura in maniera selettiva su specifiche colonne oppure a livello di intero tablespace per proteggere tabelle, indici e altri dati con algoritmi di cifratura robusti (3DES o AES fino a 256 bits) e senza la complessità della gestione di chiavi di cifratura.

Inoltre, anche la cifratura all'interno della Base Dati, se pur un valido meccanismo di sicurezza, non risolve il problema del furto dei supporti contenenti i backup. Per risolvere questo problema, il sistema, mediante la option Oracle Advanced Security, consentirà la cifratura dei backup direttamente sul supporto di salvataggio rendendo indecifrabili le informazioni in essi contenute in caso di accessi fraudolenti ai supporti sui quali vengono salvati i backup.

Il sistema implementerà la cifratura dei canali di comunicazione tra il database server e gli application server mediante gli algoritmi di cifratura (SSL/TSL) messi a disposizione da Oracle.

Il sistema implementerà, inoltre, il mascheramento dinamico dei dati sensibili mediante l'utilizzo della feature "Oracle Data Redaction" inclusa nella option Oracle Advanced Security. Sarà possibile creare policy che specificano le condizioni che devono essere soddisfatte prima che i dati vengano mascherati e restituiti all'utente. Durante la definizione di tali policy, si potrà specificare quali colonne mascherare, il tipo di protezione che deve essere applicato (totale, parziale, random ecc.) ed i ruoli istituzionali ai quali il dato viene mascherato.

In alternativa, laddove l'utilizzo della feature "Oracle Data Redaction" non consenta di soddisfare pienamente il requisito funzionale del mascheramento dati, si procederà al mascheramento dinamico dei dati sensibili in maniera applicativa.

4.7. Disaccoppiamento tra dati sensibili e anagrafici

Oltre al tradizionale meccanismo di ruoli e privilegi ed ai meccanismi di cripting dei dati sensibili si è ritenuto opportuno adottare, nella quasi totalità dei casi, anche il meccanismo di disaccoppiamento logico dei dati.

Tale meccanismo è ottenuto disaccoppiando le tabelle contenenti dati sensibili da quelle contenenti dati identificativi e correlandole tra loro mediante l'utilizzo di "codici non parlanti" (con tale terminologia ci si riferisce a codici non esplicativi della semantica del dato o a codici possano ricondurre immediatamente alla semantica del dato) oppure, utilizzando sempre codici non parlanti nei casi in cui i dati identificativi dell'utente e le informazioni sensibili siano contenute nella stessa tabella (es. codice fiscale dell'assistito e i codici esenzione sono contenuti nella stessa tabella, ma questi ultimi sono codificati con una codifica del tutto interna al sistema e non conosciuta dall'operatore). Solo in rare eccezioni, a causa della grossa mole dei dati e dell'elevata attività transazionale correlata, si è deciso di non applicare il disaccoppiamento per non impattare pesantemente sulle performance del sistema e si utilizzano quindi soltanto i meccanismi di ruoli e privilegi e di cripting dei dati.

4.8. Tracciamento e Monitoraggio

La componente Tracciabilità e Monitoraggio fornisce servizi trasversali a tutte le componenti applicative, con l'obiettivo di raccogliere (per consentirne poi la successiva analisi) le informazioni riguardanti gli utenti che accedono al sistema, i servizi da essi richiesti, data ed ora della richiesta, modalità con cui accedono al sistema e fruiscono dei servizi, l'esito del servizio richiesto.

La componente ha la responsabilità di conservare traccia degli eventi di fruizione dei servizi offerti dalle componenti applicative. Le informazioni raccolte, opportunamente aggregate ed elaborate, permettono di:

- misurare i carichi di lavoro del sistema;
- monitorare il livello delle prestazioni (per ogni servizio il tempo medio di esecuzione);
- elencare le situazioni anomale;
- produrre rapporti sui servizi erogati relativamente ad aree applicative
- monitorare le modifiche ai dati del database.

Questa componente non ha responsabilità di monitoraggio sistemistico che viene delegato alle componenti di ambiente e di sistema.

4.8.1 Ambito del tracciamento

Sono oggetto di tracciamento i seguenti eventi:

- utilizzo del singolo caso d'uso, query, report;
- singola funzione elementare del caso d'uso laddove applicabile;
- utilizzo del singolo web service sia per web services di consultazione sia di tipo transazionale;
- operazioni di CUD sulle persistenze del sistema.

Relativamente agli use case di consultazione, alle query e ai report si provvederà a tracciare i dati riguardanti i filtri di ricerca/consultazione. Non verrà effettuato alcun tracciamento del risultato.

Per quanto riguarda il tracciamento delle operazioni CUD sulle persistenze, si provvederà a tracciare, per ogni tabella sottoposta ad attività di tracing quanto segue:

- nome della tabella soggetta a tracciamento;
- tipo di operazione (insert, update, delete);
- istanze della tabella prima della modifica oppure in caso di inserimento di un nuovo record l'intera istanza inserita; nel caso di cancellazione l'istanza cancellata;
- istanza successiva alla modifica, nel caso di modifica;
- id dell'utente che ha effettuato la modifica;
- data ed ora della modifica.

4.8.2 Punti di tracciamento

I punti di tracciamento per use case, query, report e web services sono allocati sul layer web.

Il seguente diagramma evidenzia, nelle componenti già descritte, i metodi dedicati alle funzionalità di tracciabilità e monitoraggio ottenute tramite l'utilizzo della libreria log4j.

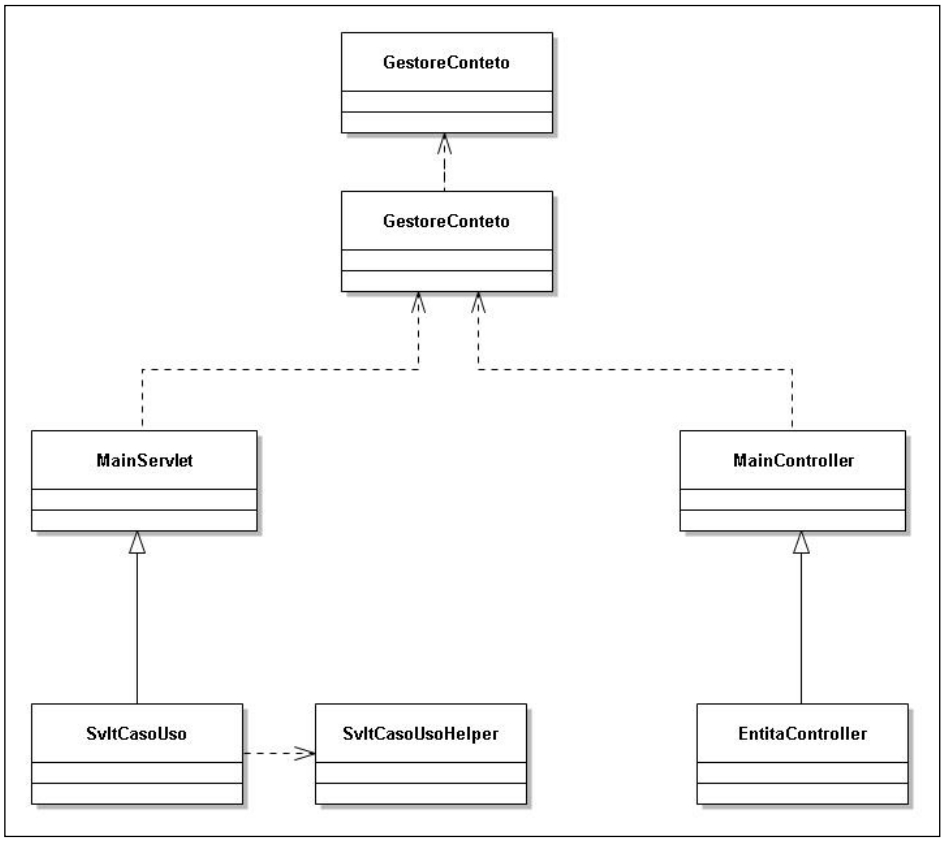

Figura 6 - Tracciabilità e Monitoraggio

In Allegato 6 vengono riportati i package che contengono le classi che hanno la responsabilità della gestione dei log applicativi.

4.8.3 Procedura di tracciamento

La strategia di tracciamento prevede due passi:

1. log su file (sul file system dell'application server);
2. procedura batch notturna che legge i file, inserisce i record di tracciamento in una apposita tabella del DB ed effettua il backup dei file.

4.8.4 Persistenza

Tutte le informazioni di tracciamento sono registrate in una apposita tabella del DB. Questa tabella è il punto di raccolta di tutte le informazioni dei canali di tracciamento. Le informazioni di auditing sono memorizzate su un'altra tabella del DB.

4.8.5 Canali di tracciamento

Per canale di tracciamento si intende il flusso di informazioni riguardanti una componente del sistema.

È definito un canale di tracciamento per ogni componente applicativa. Ad un canale di tracciamento corrisponderà uno specifico file che conterrà tutte le informazioni tracciate dal

sistema secondo quanto definito dalle regole **Ambito del Tracciamento** e **Record di Tracciamento.** Dunque ogni componente software che contiene punti di tracciamento utilizzerà il canale di tracciamento della componente applicativa di appartenenza.

4.8.6 Record di tracciamento

Il singolo record di tracciamento contiene, per ogni evento, i seguenti dati:

1. il momento esatto in cui si è verificato l'evento (data, ora, minuti e secondi);
2. l'identificativo dell'operatore che lo ha generato;
3. l'identificativo della sessione http gestito dal web server (jboss) in corso;
4. l'identificativo della transazione in corso, determinato dal software applicativo;
5. il nome del caso d'uso, query, report o web service;
6. il nome della pagina attraversata;
7. il nome dell'operazione (RICERCA, INSERIMENTO, CANCELLAZIONE, MODIFICA, DETTAGLIO);
8. il flag inizio esecuzione operazione;
9. il flag fine esecuzione operazione;
10. il flag esito operazione;
11. i valori della richiesta, nel caso di consultazione;
12. l'eventuale stack-trace dell'eccezione.

4.9. Funzionalità e meccanismi per la configurazione applicativa

Nella conduzione di sistemi informativi può essere necessario intervenire attraverso una *console amministrativa* per modificarne il comportamento senza necessariamente modificare il software applicativo che realizza funzioni e servizi.

La componente *Amministrazione Applicativa* fornisce gli strumenti per configurare e personalizzare il sistema, in tutti quei casi in cui l'implementazione permetta di scegliere il comportamento desiderato tra più opzioni disponibili. Le funzioni incluse in questa componente, dipendentemente dal loro ambito di applicazione (componente applicativa, intero sistema), sono disponibili tramite interfaccia web, rispettivamente ad un soggetto avente il ruolo di amministratore di sistema o di componente applicativa.

L'amministrazione del sistema, nei termini appena descritti, consente la gestione di opportuni parametri mediante i quali discriminare tra diverse opzioni disponibili per modificare i comportamenti, oltre che dell'intero sistema (nei casi in cui è applicabile), dei servizi e delle funzionalità di ciascuna componente applicativa.

Tutte le operazioni necessarie alla gestione sono eseguite senza interruzioni dell'erogazione del servizio: i parametri di configurazione sono pertanto modificabili a run-time.

Per consentire lo svolgimento di specifiche attività – quali ad esempio l'aggiornamento della base dati - l'area Amministrazione Applicativa consente di sospendere in maniera selettiva l'erogazione dei servizi di una specifica area applicativa, senza interrompere i servizi delle altre aree.

L'area implementa componenti e metodi richiamabili dalle diverse altre aree applicative per la lettura dei valori dei parametri di configurazione. La lettura dei valori dei parametri può avvenire a diversi livelli del software:

- nello strato WEB le componenti di tipo *it.exprivia.secsisr.<sistema>.<sigla area>.web.componente.servlet.casouso.SvltCasoUsoHelper* (helper dei casi d'uso delle altre aree applicative) accedono alla classe it.exprivia.secsisr.<sistema>.<sigla area>.web.proxy.parametroconfigurazione.DelegatoParametroConfigurazione.java
- nello strato EJB le componenti di tipo *it.exprivia.secsisr.<sistema>.<sigla area>.ejb.entita.bean.EntitaEJB* (EJB locali di entità) accedono alla classe it.exprivia.secsisr.<sistema>.<sigla area>.ejb.entita.dao.ParametroConfigurazioneDao
- nei batch che girano nella JVM di Oracle le componenti di tipo *it.exprivia.secsisr.<sistema>.<sigla area>.batch.areaapplicativa.batchcasouso.BatchCasoUso* (classi main delle elaborazioni batch) accedono alla classe it.exprivia.secsisr.<sistema>.<sigla area>.ejb.entita.dao.ParametroConfigurazioneDao

Il seguente diagramma mostra le relazioni fra le diverse componenti, nel rispetto delle naming convention e dei pattern adottati e già descritti.

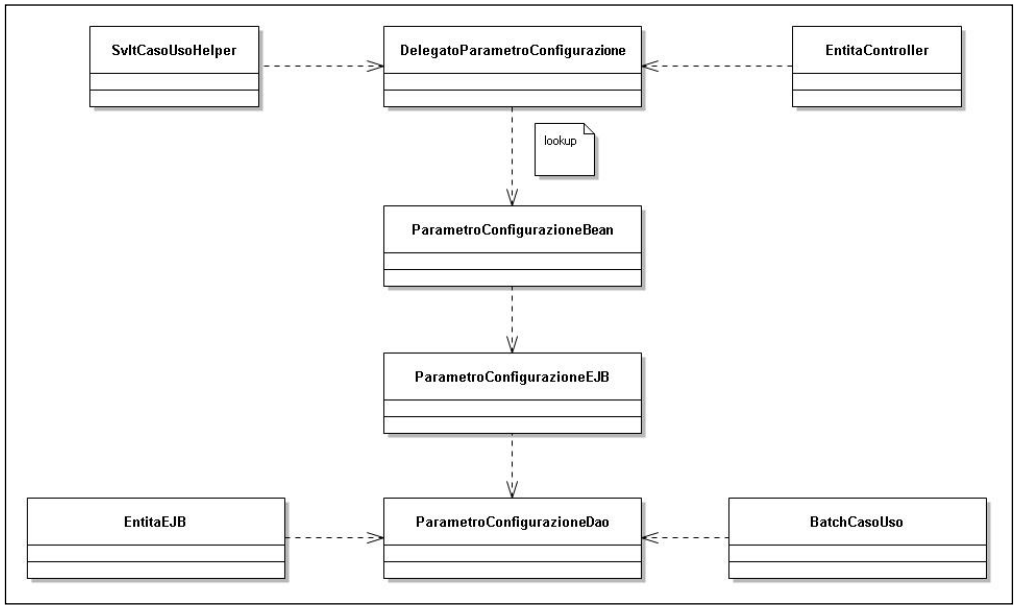

Figura 7 – Amministrazione Applicativa - relazioni fra le diverse componenti

4.10. Gestione delle Notifiche

Il trattamento di notifiche di documenti risponde all'esigenza di segnalare la creazione o modifica di una informazione, la modifica di un documento o la produzione di un nuovo documento da parte di un ente o amministrazione ad altri enti o amministrazioni.

La notifica documenti avviene con le seguenti modalità:

- registrazione e consultazione via web application con o senza allegati;
- via e-mail con o senza allegati;
- via e-mail con avviso di consultazione e download dell'allegato via web application.

Il presentation layer è costituito dalle componenti di interfaccia web che consentono all'utente di

- notificare un documento o evento;
- consultare le notifiche ricevute ed inviate e relativi download.

Dal punto di vista architetturale presentation layer e business layer sono del tutto simili a quelli di un qualsiasi altro caso d'uso. Quindi per i casi d'uso di notifica e consultazione eventi valgono le scelte progettuali e architetturali effettuate per presentation layer e business layer delle restanti componenti applicative.

Per quanto riguarda invece le notifiche a mezzo e-mail, tutte le componenti applicative si integrano grazie al modulo J-Communicator con il sistema di Posta Elettronica Certificata, consentendo agli utenti di poter fruire direttamente all'interno dei moduli applicativi web delle funzionalità di gestione della messaggistica mediante caselle PEC.

Per maggiori informazioni sul modulo JCommunicator si faccia riferimento al par. 8.3 **PEC**, posta elettronica certificata.

5. Anonimizzazione e Pseudonimizzazione di flussi informativi

La componente di Anonimizzazione e Pseudonimizzazione è una componente al servizio di tutto il sistema Sinfonia, che provvede a gestire i processi tecnici ed amministrativi connessi con le problematiche di anonimizzazione e di pseudonimizzazione dei dati sensibili e giudiziari.

Si tratta di un sistema di supporto a tutto il sistema Sinfonia in tutti quei i casi in cui risulterà necessaria, in accordo con il modello organizzativo e di processo e delle specifiche funzionali, la pseudonimizzazione di dati o gruppi di dati elementari, o la pseudonimizzazione di interi *frammenti verticali* di flussi informativi.

La componente consente di:

- generare un codice detto PILUR (Pseudonimo Identificativo Logico Univoco Regionale) e conservarne l'associazione con i dati identificativi diretti della persona, al fine di sostituire con tale codice i dati identificativi diretti del cittadino (pseudonimizzazione) all'interno flussi informativi contenenti dati sensibili;
- gestire le richieste di decodifica del PILUR;
- gestire le richieste di rilascio del PILUR a fronte di un codice fiscale;
- gestire le richieste cartacee di decodifica e di rilascio del PILUR;
- pseudonimizzare un flusso, ossia sostituire, in un flusso informativo, i dati identificativi personali del cittadino con il PILUR generato e la successiva registrazione in archivio dell'associazione tra PILUR e i dati identificativi personali.
- consentire la valutazione (accettazione/rifiuto) delle richieste di decodifica del PILUR e di rilascio del PILUR.

6. Dettagli architettura Gestione Flussi

6.1. Gestione Flussi

- La componente di Gestione Flussi è strutturata in layer architetturali concentrici, definiti da: **Layer di FrontEnd – per l'acquisizione delle informazioni tramite operazioni utente, la gestione e visualizzazione delle stesse, la richiesta dei flussi informativi;**
- **Layer di ESB/EMS – Enterprise Service Bus**, Enterprice Message Service per la configurazione di *gateway* di comunicazione in ricezione ed in trasmissione, nonché di *code* di lavoro JMS per i processi asincroni; **Layer di BPM – Business Process Management**, per la definizione di *work-flow* operativi più complessi;
- **Layer di ETL – Extract-Transform-Load**, funzionale alle attività di acquisizione, modifica e validazione dei dati in ricezione, di *data staging*, ed infine di predisposizione dei dati in trasmissione;

Alcuni moduli — laterali a tali *layer* — renderanno disponibili le funzionalità specifiche di:

- ***Protocollo*** — in termini di *logging* operativo — delle operazioni di ricezione e trasmissione.
- ***Archiviazione*** dei flussi informativi in ricezione ed in trasmissione.

La figura seguente rappresenta in termini schematici tali *layer* architetturali:

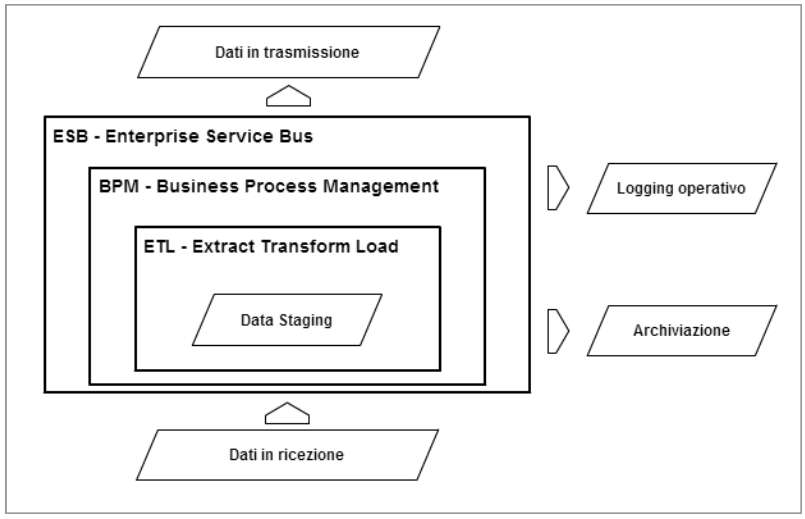

Figura 8 – Layer architetturali di Gestione Flussi

6.1.1 *Layer* di *FrontEnd*

Livello attraverso il quale si permetterà agli utenti di inserire a Sistema i flussi dati, di poterli modificare e gestire.

Sarà realizzato con AngularJS integrato in Liferay .

I benefici dell'adozione di questo strato sono:

- la facilità di visualizzazione e gestione per l'utente delle informazioni
- la possibilità di guidare l'utente nell'acquisizione e gestione dei dati inseriti
- la possibilità per l'utente di modificare le informazioni inserite a Sistema tramite un'interfaccia user friendly
- la possibilità per l'utente di recuperare tutti i file caricati direttamente tramite l'interfaccia Web, senza la necessità di dover accedere in altro modo a dischi di archiviazione.

6.1.2 *Layer* di *ESB – Enterprise Service Bus / EMS – Enterprice Message Service*

I processi di acquisizione dei dati più complessi dalle molteplici fonti informative - in presa diretta dal contesto operazionale regionale, attraverso cooperazione applicativa, o attraverso importazione di data file - saranno gestiti attraverso una architettura di enterprise service bus, che abiliterà al trasporto dei dati verso il layer di staging, semplificandone l'integrazione, tale layer verrà implementato mediante tecnologia TIBCO.

I benefici dell'adozione di una architettura di *enterprise service* bus sono:

- la separazione della *business logic* dai protocolli di comunicazione e dai formati di messaggio
- il *bridging* dei protocolli di comunicazione
- il *routing* dei messaggi basato su contenuti e regole di *delivering*
- la gestione del *payload* dei messaggi (*encryption*, *compression*, *encoding*, …).

Verranno create delle *queue* sull'EMS Tibco per permettere la comunicazione tra il FrontEnd e le diverse componenti, realizzando così il disaccoppiamento tra le stesse a fronte delle richieste effettuate dagli utenti al Sistema. Questo permetterà all'intera soluzione di essere più robusta a fronte di eventuali guasti.

6.1.3 *Layer* di *BPM – Business Process Management*

La definizione di processi di flusso — in termini di workflow dinamico piuttosto che di elaborazione statica — comporta l'implementazione di:

- *Policy* di schedulazione temporale dei dati in ricezione e di quelli in trasmissione
- registro dei mittenti e dei destinatari del processo trasmissivo
- conservazione in copia del flusso ricevuto e di quello corrispondente poi validato e trasmesso

- *Policy* specifiche e differenziate di gestione degli scarti di dato non valido: i flussi di dati che presentino scarti saranno notificati al mittente senza essere trasmessi al destinatario e potranno essere:
  - re-inviati (entro le date concordate) interamente dal mittente con le correzioni effettuate. Tale file subirà nuovamente il processo di validazione.
  - Modificati mediante form predefinite per la modifica parziale dei dati. Questi ultimi verranno risottomessi nel sistema per la validazione degli stessi e, in caso di esito positivo, trasmessi secondo le tempistiche e le modalità concordate.

6.1.4 *Layer* di *ETL – Extract-Transform-Load*

La ricezione, gestione, validazione e trasmissione dei flussi informativi saranno gestiti come processi *ETL* che prevedano:

- una fase di estrazione-ricezione dei dati da una o più sorgenti
- una fase di manipolazione dei dati — che comprende procedure di *data quality*, *assesment* e *cleansing*
- una fase di caricamento-trasmissione dei dati verso i destinatari finali.

6.1.5 Contesto infrastrutturale

La componente *Gestione Flussi* si colloca nel contesto infrastrutturale generale, secondo lo schema seguente:

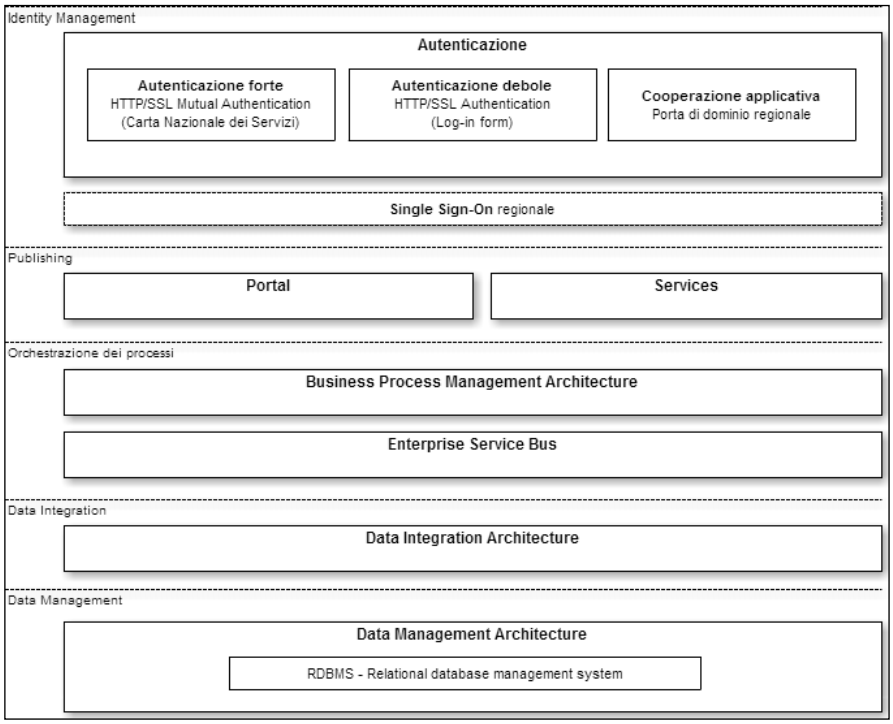

Figura 9 - Contesto infrastrutturale di Gestione Flussi

6.1.6 Entità funzionali di processo

Le entità funzionali di processo - in termini di flusso informativo - sono:

- **Flusso informativo in ricezione.**
- **Flusso informativo in trasmissione.**

Le entità funzionali di processo - in termini invece di processo - sono:

- **Processo di ricezione di flusso informativo.**
- **Processo di trasmissione del flusso informativo.**

6.1.6.1 Flusso informativo in ricezione.

Un flusso informativo in ricezione è definito da un set di *file* di *record* informativi riferiti ad uno specifico periodo di competenza, ed è caratterizzato da:

- un set di metadati quali ad esempio:
  - Denominazione del flusso informativo.
  - Periodo di competenza.
  - Azienda di competenza
  - Identificativo del mittente.

  - *Timestamp* di ricezione.

- File dei record di flusso informativo.

Per i sistemi esterni ogni file inviato, sarà considerato comunque come afferente a tutto il periodo di competenza. L'invio di un successivo file relativo allo stesso periodo e flusso informativo sarà considerato come sostitutivo del precedente.

6.1.6.2 Flusso informativo in trasmissione

Un flusso informativo in trasmissione è definito da un *file* di *record* informativi, caratterizzato da:

- un set di metadati quali ad esempio:
  - *Denominazione del flusso informativo.*
  - *Periodo di competenza.*
  - *Azienda di competenza*
  - *Identificativo del richiedente.*
  - *Timestamp di trasmissione.*
- File dei record di flusso informativo.

6.1.6.3 Processo di ricezione di flusso informativo

È il processo di ricezione — periodico — di un *file* di *record* informativi.

Il processo di ricezione è caratterizzato da un periodo di competenza e da un'azienda di competenza.

In questo periodo potranno esservi più eventi di ricezione del flusso: un evento di ricezione per il flusso iniziale, ed uno o più eventi successivi per flussi *sostitutivi*.

In tale intervallo temporale, inoltre, si potrà procedere con l'*editing* da operatore dei *record* di flusso informativo — in termini di operazioni di *update* e *delete* di *record* informativi specifici.

Dopo che l'utente preposto richiederà la generazione del flusso informativo, non potranno essere più ricevuti flussi sostitutivi né si potrà procedere con operazioni di *editing* da parte dell'operatore; qualora questo si renda necessario si dovrà procedere attraverso una procedura dedicata che è attivabile solo per tali finalità e da operatori preposti.

Il processo di ricezione dei flussi informativi è schematizzato nella figura seguente:

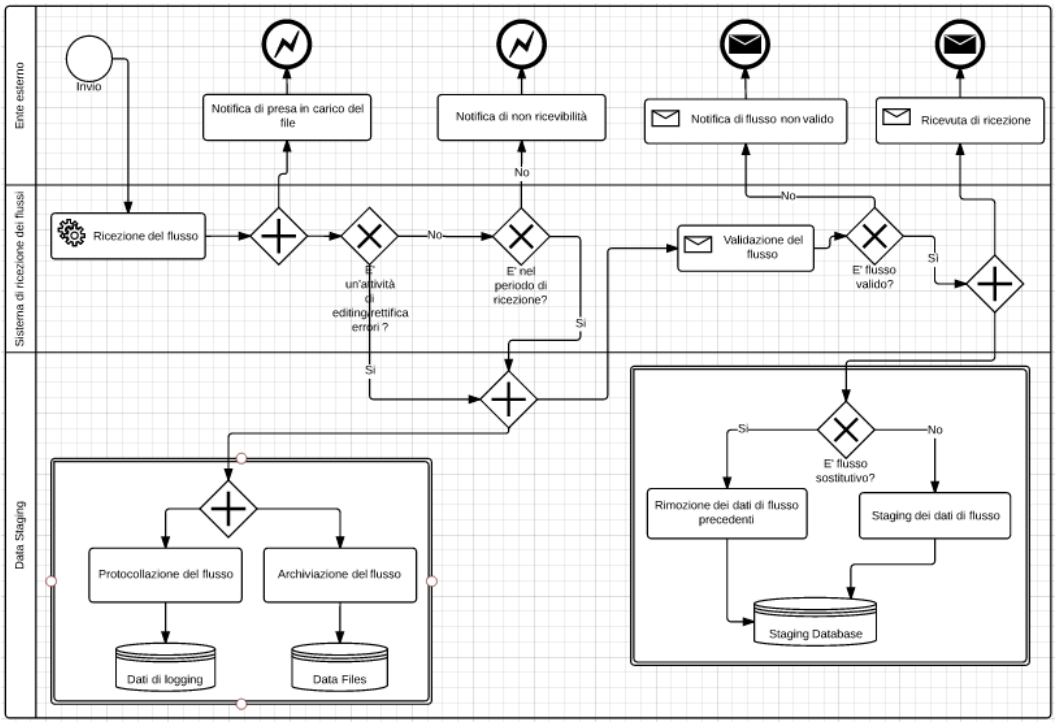

Figura 10 – Processo di ricezione di flusso informativo

Il processo di **ricezione** del flusso informativo si compone di tali eventi funzionali di processo:

- **Evento di ricezione - *file upload* - di flusso informativo**.
- **Evento di validazione di flusso informativo**.
- **Evento di editing da operatore di record del flusso informativo**.
- **Evento di validazione di record del flusso informativo.**

6.1.6.4 Evento di ricezione - *file upload* - di flusso informativo.

L'evento di ricezione è definito dall'*upload* di un *file* di *record* informativi:

- **Da operatore** - previa registrazione, attraverso un ***form* di *file upload*** reso disponibile dalla *web application* di gestione.
- **Da applicazione esterna** - attraverso la disponibilità di uno specifico ***web service* di tipo *SOAP Messages with Attachments*** (*www.w3.org/TR/SOAP-attachments*).

Nel contesto di un processo di ricezione si collocano uno o più eventi di ricezione in momenti temporali diversi compresi nell'intervallo in cui il processo di ricezione è aperto: un evento di ricezione per il flusso iniziale, ed uno o più eventi successivi per flussi *sostitutivi* dell'ultimo flusso ricevuto: **solo ove espressamente necessario si prevedono flussi — parziali — *integrativi***.

6.1.6.5 Evento di editing di alcuni record del flusso informativo

L'evento di editing è definito da operazioni di *update* e *delete* di *record* informativi specifici:

- **Da operatore** — previa registrazione, attraverso un ***form* di *editing r***eso disponibile dalla *web application* di gestione.

Nel contesto di un processo di ricezione si collocano uno o più eventi di editing in momenti temporali diversi compresi nell'intervallo in cui il processo di ricezione è aperto

6.1.6.6 Evento di validazione di flusso informativo

È l'evento di validazione di un flusso di dati in ricezione.

È differito - *asincrono* - rispetto all'evento di ricezione del flusso di dati.

Il *file* di *record* informativi verrà validato sintatticamente mediante XSD concordati che ne definiscono i vincoli:

- quali elementi ed attributi possono essere presenti, in quale relazione reciproca, quale tipo di dati può contenere (tipi di dati primitivi previsti dal formato XSD).
- definizione di nuovi tipi di dato partendo dai tipi primitivi attraverso tre possibili mecca*nis*mi:
  - *restrizione* (riduzione dell'insieme dei valori permessi);
  - *lista* (estensione ad una sequenza di valori);
  - *unione* (possibilità di scelta di un valore da differenti tipi).

L'evento di validazione di flusso informativo è schematizzato nella figura seguente:

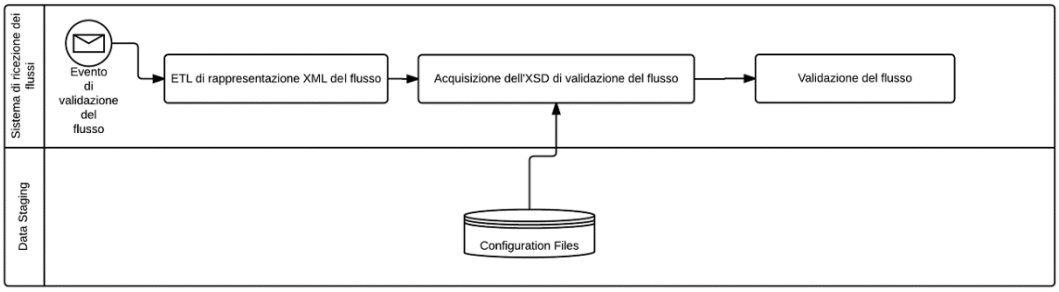

Figura 11 – Evento di validazione di flusso informativo

Il flusso informativo subirà, inoltre, un processo di validazione semantica secondo le regole esplicitate nel documento tecnico di specifica funzionale della componente flussi.

Tale regole, saranno applicabili se e solo le informazioni necessarie alla loro implementazione saranno messe a disposizione dagli enti preposti nella forma di:

- file (es: xml, txt, xls) acquisibili all'interno della soluzione
- mediante upload manuale

- condivisione di un'area predefinita.

Il processo di validazione effettuerà il controllo con gli ultimi dati disponibili all'interno della soluzione stessa.

Eventuali upload, successivi al processo di validazione, non saranno presi in considerazione salvo risottomissione manuale del processo di upload del flusso informativo (sostituzione del flusso informativo).

6.1.6.7 Evento di validazione di *record* del flusso informativo

La validazione di uno specifico *record* del flusso informativo è differita — a*sincrono* — rispetto all'evento di *editing* da operatore del *record*.

Alla fine del processo di validazione, il sistema provvederà a notificare all'utente lo stato della validazione effettuata sul record modificato.

6.1.6.8 Processo di trasmissione

È il processo di predisposizione del flusso informativo in trasmissione, e di *file downloading* di tale flusso.

Sarà possibile attivare il processo di trasmissione sono fronte di flussi di input privi di errori. Potrà comunque esservi un unico evento di trasmissione del flusso. Qualora si voglia ritrasmettere lo stesso, sarà necessaria un'operazione di "sblocco" flussi pubblicati da parte di un operatore autorizzato.

Il processo di trasmissione dei flussi informativi è schematizzato nella figura seguente:

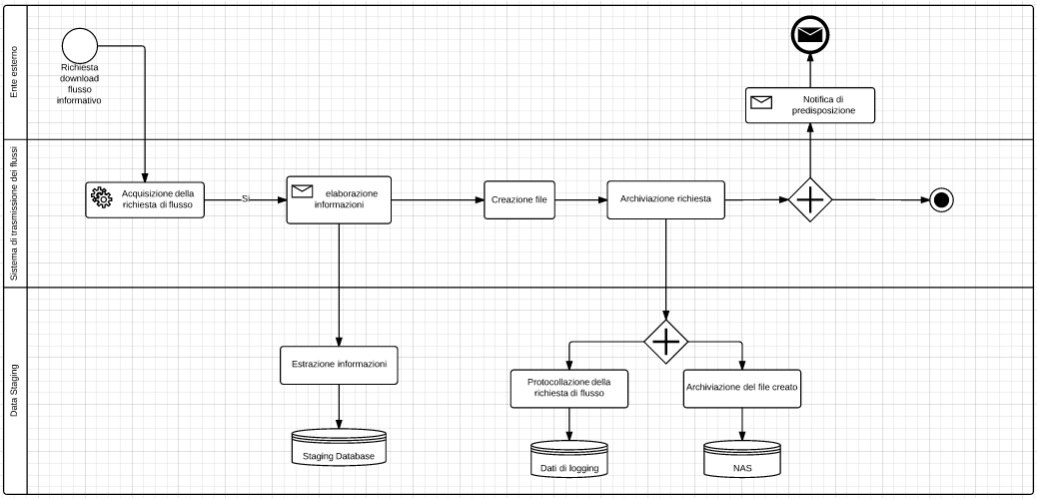

Figura 12 – Processo di trasmissione di flusso informativo

Il processo di **trasmissione** del flusso informativo è definito da tali eventi funzionali di processo:

- **Evento di predisposizione di flusso informativo**.

- **Evento di trasmissione - *file download* - di flusso informativo**.

6.1.6.9 Evento di predisposizione di flusso informativo

Attraverso specifica procedura *ETL* si produrrà il *file* di *record* informativi in trasmissione.

6.1.6.10 Evento di trasmissione - *file download* - di flusso informativo.

L'evento di trasmissione è definito dal *download* di un *file* di *record* informativi:

- **Da operatore** - attraverso un ***form*** **di** ***file download*** reso disponibile dal portale di gestione.
- **Da applicazione esterna** - attraverso la disponibilità di uno specifico ***web service*** **di tipo** ***SOAP Messages with Attachments*** (*www.w3.org/TR/SOAP-attachments*).

6.1.6.11 Ruoli di utenza

Gestione Flussi comporterà la definizione - nel contesto di *Identity Management* del portale dei servizi - dei **ruoli di utenza** di seguito indicati:

- Ruolo ***operatore in ricezione*** (*Azienda*) - autorizzato all'invio ed alla modifica di un flusso in ricezione.
- Ruolo ***operatore in trasmissione*** *(Dipartimento)* - autorizzato all'acquisizione di un flusso in trasmissione.
- Ruolo ***amministratore*** - Tale ruolo avrà la possibilità di monitorare tutte operazioni effettuate a Sistema ed ha entrambe le funzionalità dei ruoli "operatore in ricezione" ed "operatore in trasmissione".

7. Identificazione, autenticazione ed autorizzazione dei sistemi fruitori dei servizi

Obiettivo del capitolo è quello di illustrare i meccanismi e le specifiche con cui il sistema Sinfonia implementa i meccanismi di identificazione, autenticazione ed autorizzazione dei sistemi applicativi cooperanti che inoltrano richieste di servizio in modalità web services.

Nel seguito si intende per Sistema Fruitore un sistema applicativo che fruisce dei servizi di cooperazione di Sinfonia, esposti attraverso dall'ESB, che funge da Sistema Erogatore. Vengono considerati sistemi fruitori, alla stregua di qualunque altro sistema applicativo cooperante, le stesse componenti applicative ed i sistemi infrastrutturali di Sinfonia per le richieste di servizio che questi inoltrano verso le altre componenti applicative di Sinfonia.

L'identificazione, l'autenticazione e l'autorizzazione del Sistema Fruitore, per ciascun servizio di cooperazione esposto, sono implementati dall'ESB Tibco.

7.1. Il processo complessivo

Di seguito è illustrato il processo complessivo di interazione tra sistemi cooperanti relativi al processo di identificazione, autenticazione e autorizzazione nel caso di invocazione dei servizi esposti da Sinfonia da sistemi fruitori esterni al sistema Sinfonia.

1. Il Sistema Erogatore riceve un messaggio SOAP contenente il certificato X.509v3 del Sistema Fruitore in conformità alla specifica ”X.509 Certificate Token Profile” fornito da WS-Security. L’integrità del messaggio è garantita dalla firma digitale apposta con certificato X.509v3.
2. Il Sistema Erogatore verifica l’integrità sintattica del messaggio verificando la rispondenza all’xsd. In particolare viene estratto dal messaggio di input il token X.509v3, rappresentante il Sistema Fruitore. Successivamente si applicano tutti i controlli necessari a verificare la validità del certificato. Ottenuta la validazione del certificato, il Sistema Erogatore verifica che il common-name del certificato X.509v3 sia tra quelli autorizzati ad interagire con il Sistema Erogatore.
3. Il Sistema Erogatore, superati tutti i controlli del passo precedente ed in conformità con quanto definito nei successivi paragrafi relativi all’autorizzazione per i servizi di cooperazione, provvede ad erogare il servizio richiesto.
4. Il Sistema Erogatore accerta la conformità della richiesta applicativa rispetto alle eventuali politiche di sicurezza aggiuntive specifiche del servizio applicativo richiesto.
5. Il messaggio SOAP di response inviato al Sistema Fruitore soddisferà, analogamente al messaggio SOAP di request, la specifica “X.509 Certificate Token Profile” fornito WS-Security.

Nel caso l’interazione attraverso servizi web avvenga tra componenti applicative o infrastrutturali interne al sistema Sinfonia, al fine di ridurre l’overhead generato dall’applicazione delle policy di sicurezza previste dalla WS-Security, viene adottato un modello di sicurezza semplificato, bassato sull’utilizzo di SSL (Secure Socket Layer) in modalità “mutual authentication” sufficiente a garantire, oltre all’autenticazione dell’erogatore, anche l’autenticazione e l’autorizzazione del fruitore.

7.2. Identificazione ed autenticazione dei Sistemi Fruitori

L’identificazione e l’autenticazione del Sistema Fruitore esterno a Sinfonia è basata sull’utilizzo del certificato X.509v3 di autenticazione del Sistema Fruitore, secondo lo standard Web Services Security X.509 Certificate Token Profile (http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-x509TokenProfile.pdf).

L’identificazione del sistema fruitore deve individuare in modo univoco e certo lo specifico sistema che sta invocando il servizio esposto. Ciò significa che due installazioni distinte di uno stesso prodotto software, anche nello stesso dominio organizzativo ovvero anche sullo stesso sistema server fisico, sono identificate tramite due identità differenti e, quindi, tramite due distinti certificati di autenticazione X.509v3.

Ne consegue che ogni sistema fruitore deve essere dotato di un certificato X.509v3 il cui commonname deve essere censito nell’anagrafe dei certificati X.509v3 associati ai sistemi fruitori autorizzati ad interagire con il sistema erogatore.

Come da standard WS-Security, il certificato di autenticazione X.509v3 sarà utilizzato per la firma di parti del messaggio SOAP, in particolare dovranno essere firmate le seguenti porzioni di messaggio

- il tag <Timestamp>, previsto nell’header del messaggio SOAP;
- il tag <To>, previsto nell’header del messaggio SOAP;

- il tag <Action>, previsto nell'header del messaggio SOAP;
- il tag <MessageID>, previsto nell'header del messaggio SOAP;
- il tag <ReplyTo>, previsto nell'header del messaggio SOAP;
- il tag <Body>.

7.3. Integrità del messaggio

L'integrità del messaggio SOAP associato all'invocazione di un web service assicura che i messaggi non siano intercettati e alterati durante lo scambio fra Sistema Fruitore e Sistema Erogatore.

L'integrità delle parti fondamentali del messaggio è garantita sottoponendo a processo di firma:

- il tag <Timestamp>, previsto nell'header del messaggio SOAP;
- il tag <To>, previsto nell'header del messaggio SOAP;
- il tag <Action>, previsto nell'header del messaggio SOAP;
- il tag <MessageID>, previsto nell'header del messaggio SOAP;
- il tag <ReplyTo>, previsto nell'header del messaggio SOAP;
- il tag <AttributiAutorizzativi>, presente nell'header del messaggio SOAP;
- il tag <Body>.

7.4. Non ripudio del messaggio

Il non ripudio di un messaggio trasmesso dal Sistema Fruitore al Sistema Erogatore è garantito dall'autenticazione che una firma è in grado di offrire.

Infatti, l'univocità della firma digitale applicata ad un messaggio impedisce che il proprietario della firma disconosca le informazioni contenute nel messaggio firmato.

Il non ripudio del messaggio è garantito dall'applicazione della firma digitale da parte del Sistema Fruitore al messaggio SOAP-Request per:

- il tag <Timestamp>, previsto nell'header del messaggio SOAP;
- il tag <To>, previsto nell'header del messaggio SOAP;
- il tag <Action>, previsto nell'header del messaggio SOAP;
- il tag <MessageID>, previsto nell'header del messaggio SOAP;
- il tag <ReplyTo>, previsto nell'header del messaggio SOAP;
- il tag <AttributiAutorizzativi>, presente nell'header del messaggio SOAP;
- il tag <Body>.

7.5. Mantenimento delle informazioni della richiesta di servizio

Allo scopo di mantenere le informazioni relative alla richiesta del servizio, in maniera funzionale a dimostrare a terzi la legittimità dell'operato di Sinfonia, ove necessario, saranno memorizzati nel sistema i dati salienti dei messaggi scambiati tra Sinfonia ed i sistemi fruitori.

In considerazione delle ripercussioni che tale scelta può avere sul sistema in termini di occupazione dei volumi e di ulteriore carico transazionale, la definizione dei servizi per i quali saranno mantenute le suddette informazioni sarà effettuata congiuntamente con il committente.

7.6. Riservatezza del messaggio

La riservatezza del messaggio SOAP deve garantire che i dati trasmessi non siano alterati durante lo scambio e non siano interpretabili da alcuno con l'eccezione di chi ha il permesso di accedervi.

Lo strumento per garantire la riservatezza del messaggio è l'utilizzo di SSL (Secure Socket Layer), che permette di creare un canale protetto per lo scambio di dati tra due Sistemi.

Tutti i servizi di Sinfonia esposti come web services standard attraverso l'ESB sono fruibili su protocollo SOAP su HTTPS.

7.7. Firma dei messaggi di risposta

Per garantire integrità, non ripudio e riservatezza dei messaggi di risposta, nelle SOAP-Response saranno firmati, utilizzando il certificato X509v3 di Sinfonia, i tag:

- <Timestamp>
- il tag <To>, previsto nell'header del messaggio SOAP;
- il tag <Action>, previsto nell'header del messaggio SOAP;
- il tag <MessageID>, previsto nell'header del messaggio SOAP;
- il tag <RelatesTo>, previsto nell'header del messaggio SOAP;
- il tag <Body>.

7.8. Autorizzazione del Sistema Fruitore

Il Sistema Erogatore dopo aver autenticato e identificato il Sistema Fruitore verifica che quest'ultimo sia abilitato all'invocazione dei web services esposti dal Sistema Erogatore.

Il controllo si sostanzia nel verificare che il Sistema Fruitore abbia l'abilitazione ad interrogare i web services esposti dal Sistema Erogatore.

8. I Servizi Infrastrutturali

Al funzionamento del sistema Sinfonia contribuiranno i seguenti servizi infrastrutturali, che verranno dettagliati nel seguito del capitolo:

1. **Tibco**, Enterprise Message Bus
2. **Intalio**, Business Process Management System
3. **PEC**, posta elettronica certificata
4. **Firma Digitale**
5. **WSO2** Service Governance Registry

Prima di fornire una panoramica di ciascuno di questi strumenti software, è necessario chiarire il loro ruolo nell'architettura generale del sistema, cioè specificare come si relazionino a vicenda in modo da contribuire al funzionamento complessivo del sistema Sinfonia.

Dal punto di vista, dell'architettura orientata ai servizi (Service Oriented Architecture, SOA), gli strumenti software coinvolti sono essenzialmente il Business Process Management System (BPMS) Intalio, l'Enterprise Service Bus (ESB) Tibco e il Service Registry WSO2:

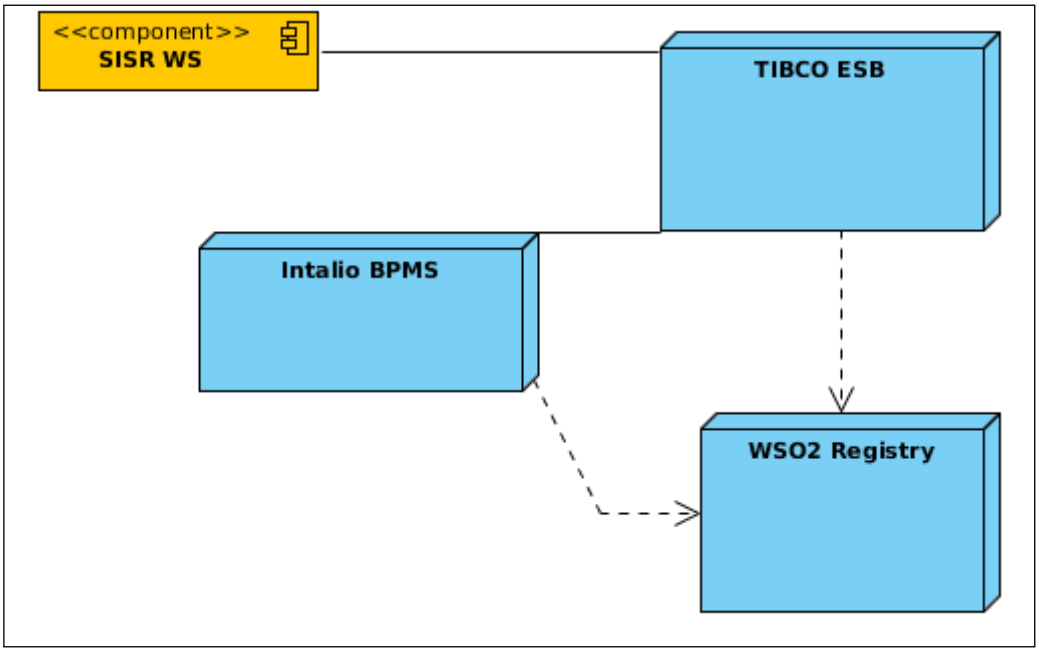

Figura 13 - Architettura SOA - Strumenti Software

I restanti due componenti infrastrutturali sono la Firma Digitale e la Posta Elettronica Certificata, che si integrano nativamente con i vari applicativi di Sinfonia.

8.1. Tibco

TIBCO Active Matrix è la piattaforma per costruire e dispiegare applicazioni SOA e comprende:

- il componente *Active Matrix Service Bus*, che include i Container SOAP, Mediation e Adapter e che servirà le Composite Applications (CA) che fungeranno da mediation routes tra sistemi diversi
- il componente *Tibco EMS* (Enterprise Message Service), basato su standard JMS
- il componente *Tibco Administrator*, che permette l'amministrazione dell'intera architettura SOA

L'interazione tra Tibco Active Matrix Service Bus e Tibco EMS permette la gestione di situazioni anche potenzialmente molto complesse, dove i connettori nativi tipicamente orientati ai servizi di Active Matrix (SOAP e REST) si possono integrare con il componente di messaggistica per garantire una QOS di livello elevato. Inoltre, la presenza dell'EMS permette di bufferizzare i messaggi per gestire carichi di lavoro anche importanti, oltre a garantire la possibilità di recapitare messaggi anche a subscriber momentaneamente non disponibili.

La soluzione proposta consentirà quindi di integrare i sistemi utilizzando qualunque Enterprise Integration Pattern (EIP), consentendo la realizzazione delle CA seguendo le best practices attuali.

Il template di processo ESB che consentirà l'integrazione tra due sistemi è quindi schematizzabile in questo modo

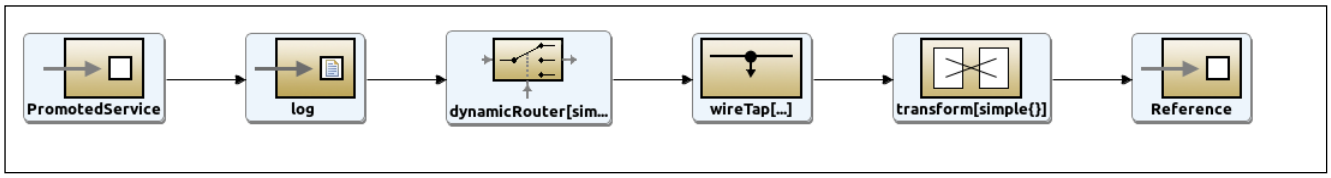

Figura 14 - TIBCO

Laddove:

- l'ESB promuove un riferimento al Web Service Destinazione, fungendo da proxy per quel servizio
  - il messaggio in ingresso viene loggato (Log)
  - del messaggio in ingresso viene calcolata l'eventuale destinazione sulla base dei dati al suo interno contenuti (Content-Based Routing)
  - viene effettuato un monitoring del canale (Wire Tap)
  - il messaggio in ingresso viene eventualmente trasformato (Message Transformation)
  - il messaggio viene recapitato alla destinazione, da cui si leggerà l'eventuale Acknowledge (sulla base dell'ACK restituito, il paradigma di scambio potrà essere In-Out o In-Only).

È importante notare che nello schema proposto è facilmente modificabile la QOS semplicemente inserendo un enpoint JMS (rappresentato, quindi da un messaggio consegnato su Tibco EMS) prima del promoted service, e/o prima della reference, in modo da garantire la consegna in ottemperanza della qualità del servizio concordata.

Nell'ambito dell'architettura proposta, Tibco si dovrà porre come punto centrale di scambio di messaggi, indipendentemente dal protocollo utilizzato: da questo punto di vista, il disegno seguente sintetizza la possibilità, offerta dal Service Bus, di tutte le comunicazioni all'interno del sistema, indipendentemente dal protocollo scelto per lo scambio di messaggi:

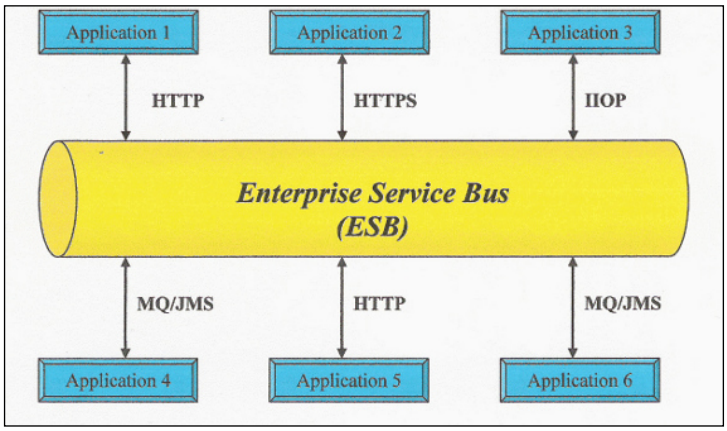

Figura 15 - Enterprise Service Bus

8.2. Intalio

La suite Intalio BPM è il un prodotto open source di Business Process Management, conforme allo standard BPM 2.0 e nativamente integrato con un motore di Business Rules (con la possibilità di impostare le regole direttamente nell'ambiente di designer). La piattaforma è costruita e basata attorno al modellatore standard BPMN STP di Eclipse e un motore Apache ODE BPEL, entrambi personalizzati da Intalio. Fornisce tutte le componenti necessarie per la progettazione, implementazione e gestione dei processi aziendali più complessi.

Intalio si interfaccerà sia con Tibco, ossia l'Enterprise Service Bus, sia direttamente con gli applicativi che potranno essere fonte di movimentazione dei processi di business. Tali applicativi, ovviamente, potranno anche essere notificati dell'avanzamento dei medesimi processi.

Intalio è utilizzato classicamente come orchestratore di processi SOA. Nel momento in cui un'integrazione di sistemi è realizzabile chiamando diversi servizi di diversi attori, e alla base di questa integrazione c'è una logica di business, tale integrazione è facilmente realizzabile attraverso un flusso di business rappresentato da uno schema BPM. Lo schema BPM permette essenzialmente di modellare decisioni di business semplicemente disegnando un diagramma, e consente ai sistemi coinvolti di contribuire al processo di business in accordo con l'orchestrazione modellata ed eseguita del server Intalio.

Lo schema seguente rappresenta graficamente il ruolo di Intalio nell'incapsulare la logica di business, oltre alla sua stretta integrazione con Tibco nel portare a termine l'orchestrazione dei servizi, sia interni che di partner esterni. Si noti come la logica di processo sia completamente all'interno di Intalio, mentre la parte di integrazione con altri servizi sia di esclusiva competenza di Tibco.

Altra cosa importante da notare di questo diagramma è che l'unica interfaccia di Intalio verso i servizi di cui è orchestratore è, in realtà, quella verso il sistema Tibco, che si incarica di fare da proxy per i servizi collaboranti e mantiene al suo interno le mediation routes verso i servizi stessi.

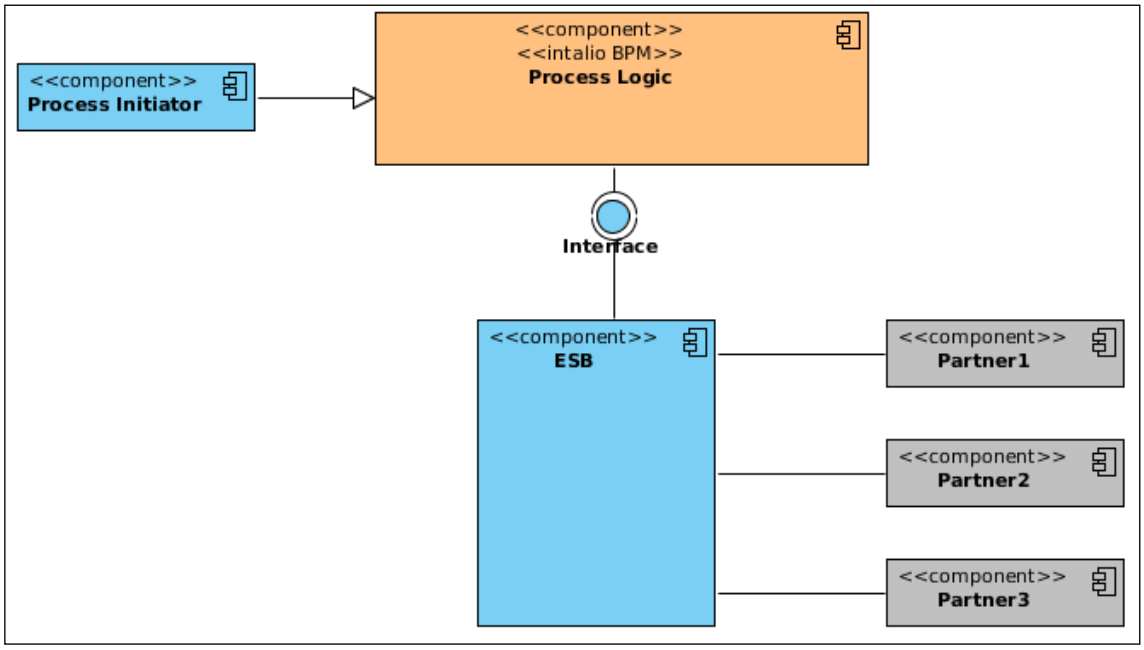

Figura 16 - Intalio

8.3. PEC

Il servizio infrastrutturale di PEC consente a tutti gli applicativi di Sinfonia l'invio e la ricezione di posta elettronica certificata.

La **Posta Elettronica Certificata** (PEC) è il sistema che consente di inviare e-mail con **valore legale equiparato ad una raccomandata con ricevuta di ritorno**, come stabilito dalla vigente normativa (DPR 11 Febbraio 2005 n.68).

Inoltre, il sistema di Posta Certificata, grazie ai **protocolli di sicurezza** utilizzati, è in grado di **garantire la certezza del contenuto** non rendendo possibili modifiche al messaggio, sia per quanto riguarda i contenuti che eventuali allegati.

La Posta Elettronica Certificata garantisce - in caso di contenzioso - l'opponibilità a terzi del messaggio.

Il termine "Certificata" si riferisce al fatto che il gestore del servizio rilascia al mittente una **ricevuta** che costituisce **prova legale** dell'avvenuta spedizione del messaggio ed eventuali allegati. Allo stesso modo, il gestore della casella PEC del destinatario invia al mittente la **ricevuta di avvenuta consegna.**

I gestori certificano quindi con le proprie "ricevute" che il messaggio:

- È stato spedito
- È stato consegnato
- Non è stato alterato

8.3.1 Il modulo di gestione PEC: J-Communicator

Il modulo J-Communicator è la componente trasversale che consente di gestire la messaggistica multicanale, quale l'invio/ricezione di mail, sms, fax; è predisposto per gestire anche l'invio e la ricezione di Posta Elettronica Certificata ed è in grado di elaborare migliaia di messaggi di PEC al giorno.

Il sistema è stato testato ed attivo con sistemi di PEC forniti ad es. da InfoCert, Actalis, PosteCert, ecc.

8.4. Firma Digitale

Il sistema di firma digitale fornito è integrato nativamente con il sistema di gestione documentale, consentendo di firmare e verificare qualsiasi documento. Nello specifico, le SmartCard utilizzate saranno emesse da Aruba e verranno consegnate assieme a un kit di firma digitale.

Il Kit per Firma Digitale sarà composto da:

- dispositivo sicuro di generazione delle Firme (Smart Card)
- lettore di Smart Card
- software di Firma e Verifica

Installato il Kit sul proprio computer, attraverso il Software di Firma sarà possibile selezionare il documento elettronico da sottoporre a Firma Digitale e, previa attivazione di un account, alla Marcatura Temporale.

Al momento della Firma del documento, il software chiederà l'inserimento del codice di protezione del dispositivo (PIN) che - se correttamente inserito - procederà con la creazione del file firmato digitalmente.

Il file firmato assumerà l'estensione .p7m che si sommerà all'estensione del file originario. Pertanto firmando un documento .txt, al termine del processo di Firma Digitale il documento assumerà l'estensione .txt.p7m che rappresenta una busta informatica (CADES o PADES).

Tale busta incorpora al suo interno il documento originario, il Certificato del sottoscrittore ed un Hash del documento firmato con il Certificato del sottoscrittore.

Tali componenti consentiranno, in fase di verifica della Firma da parte del destinatario del documento firmato, di accertare che:

- il documento non sia stato modificato dopo la Firma
- il Certificato del sottoscrittore sia garantito da una Autorità di Certificazione (CA) inclusa nell'Elenco Pubblico dei Certificatori
- il Certificato del sottoscrittore non sia scaduto
- il Certificato del sottoscrittore non sia stato sospeso o revocato

Se tutte le verifiche daranno esito positivo, il documento sottoscritto digitalmente potrà essere considerato valido a tutti gli effetti di legge.

8.4.1 Il modulo di firma digitale: J-Sign

Il modulo di firma J-Sign consente la firma e la verifica via web di qualsiasi documento digitale.

Il modulo è realizzato interamente in java ed è portabile su qualsiasi piattaforma (sistema operativo/browser), si interfaccia con qualsiasi smart card/token degli enti certificatori accreditati mediante le consolidate librerie PKCS#11.

Il modulo consente sia la firma nel formato P7M che nel formato nativo PDF, inoltre è già conforme alla recente normativa (Determinazione Commissariale 28 luglio 2010 che modifica la Deliberazione CNIPA n. 45/2009) di fatto è già compatibile con il formato CAdES e con l'algoritmo di hashing SHA-256.

Il modulo è integrato nel sistema documentale e verrà utilizzato in tutti i processi di gestione dei documenti digitali per cui l'Ente vorrà adottare la firma digitale.

Il modulo sarà adottato ad es. nel processo di Protocollazione, negli Atti (Determinazioni, Decreti, Delibere...), nella gestione della Fatturazione Elettronica e in tutti i processi documentali in cui l'Ente vorrà adottare la firma digitale.

Il modulo può essere utilizzato anche da sistemi esterni al documentale e integrato mediante l'invocazione di servizi web e la redirezione verso il servizio web di firma digitale.

8.5. WSO2

Con il termine Service Registry, in ambito SOA, ci si riferisce ad un sistema che contiene tutte le informazioni necessarie (come URL e modalità di accesso) al reperimento di tutti i servizi disponibili in esso registrati.

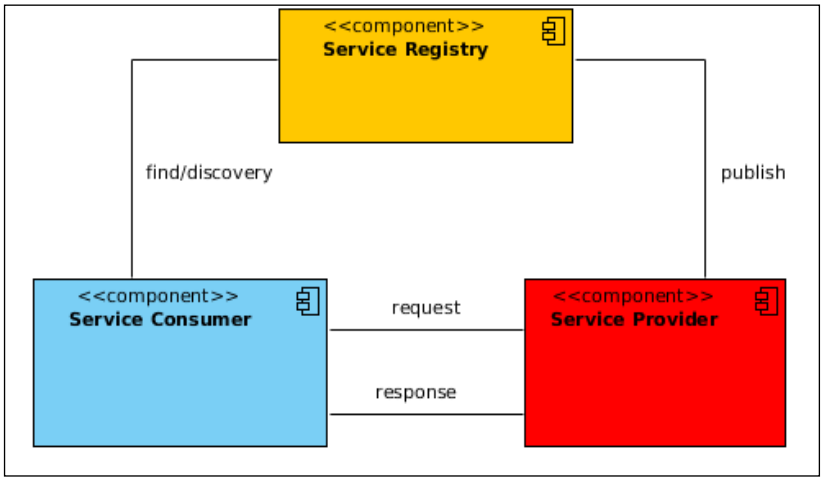

Figura 17 – Service Registry

Come si evince dallo schema sopra riportato, il vantaggio di un service registry è quello di disaccoppiare il service provider dal service consumer dal punto di vista della ricerca di un servizio e dal punto di vista della sua pubblicazione. Il prodotto scelto per espletare questa funzione all'interno dell'architettura SEC-SISR è WSO2 Governance Registry: si tratta di un prodotto 100% open source (licenza Apache 2) che consente di memorizzare, catalogare indicizzare e gestire i metadati dell'architettura SOA in un modo semplice e scalabile.

8.6. Deployment

Le componenti infrastrutturali Intalio BPM, Tibco EBS/EMS, OpenLDAP e Liferay descritte nei precedenti paragrafi sono ospitate ciascuna in un apposito nodo virtuale dell'infrastruttura di deployment di Sinfonia. Su ciascuno di questi nodi è presente la distribuzione Linux CentOS 6.6.

Allegato 1 - Package della presentation logic con le relative responsabilità

secsisr-<sistema>-<sigla area>-**web-condivisi.jar**	
Prefisso del Package	**Descrizione**
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi	Contiene le classi comuni al Web-tier per i diversi sistemi
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi.cache	Contiene le classi che implementano la memorizzazione, nell'oggetto Session o in cache, dei dati utili all'interazione del caso d'uso.
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi.command	Contiene le classi che implementano la gestione del pattern command
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi.contesto	Contiene e classi del web tier che implementano la gestione del contesto applicativo
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi.controller	Contiene le classi del web tier che implementano funzionalità condivise da tutti i controller
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi.filter	Contiene le classi del web tier che implementano la gestione dei filter applicativi
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi.interfaces	Contiene i package dedicati alle interfacce del web tier
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi.listener	Contiene le classi del web tier che implementano la gestione dei listener applicativi
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi.report	Contiene le classi del web tier che implementano la gestione dei report birt
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi.supporto	Contiene le classi del web tier che implementano la gestione di supporto applicativo (ad esempio logging)
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi.ws	Contiene le classi del web tier che implementano la gestione di client per services jaxws
it.exprivia.secsisr.<sistema>.<sigla area>.web.util	Contiene le classi del web tier che implementano funzionalità aggiuntive di utilità

secsisr-<sistema>-<sigla area>-**delegate.jar**	
Prefisso del Package	**Descrizione**
it.exprivia.secsisr.<sistema>.<sigla area>.delegate	Contiene le classi delegate condivise da tutte le classi del Web-tier verso lo strato EJB-Tier
it.exprivia.secsisr.<sistema>.<sigla area>.delegate.common	Contiene le classi di supporto condivise da tutte le classi delegate che implementano il pattern business delegate verso lo strato ejb
it.exprivia.secsisr.<sistema>.<sigla area>.delegate.<entita>	Contiene le classi che implementano il pattern business delegate verso lo strato ejb per la specifica entità di business

Secsisr-<sistema>-<sigla area>**-web.jar**	
Prefisso del Package	**Descrizione**
it.exprivia.secsisr.<sistema>.<sigla area>.web.command	Dedicato alle classi che contengono la porzione di codice che effettua un'azione complessa
it.exprivia.secsisr.<sistema>.<sigla area>.web.controller	Dedicato ai Controller dei casi d'uso web
it.exprivia.secsisr.<sistema>.<sigla area>.web.service	Dedicato ai Service dei casi d'uso web
it.exprivia.secsisr.<sistema>.<sigla area>.web.validator	Dedicato ai servizi di validazione deputati ai controlli su dati delle pagine web dei casi d'uso web

Allegato 2 - Package per la realizzazione dei casi d'uso dei sistemi con le relative responsabilità

Nome	Descrizione
it.exprivia.secsisr.<sistema>.<sigla area>.web.validator.<Entita>Validator	Contiene i metodi che effettuano controlli sui dati di una specifica entità utilizzati delle pagine web e dal relativo controller dell'entità.
it.exprivia.secsisr.<sistema>.<sigla area>.web.controller.<Entita>Controller	Il Controller del pattern MVC, motore del page flow per la relativa entità di business dalla quale prende il nome.
it.exprivia.secsisr.<sistema>.<sigla area>.web.service.<Entita>Service	Classe service spring per il Controller dell'entità di riferimento che ne alleggerisce l'implementazione rendendo più strutturato, modulare e manutenibile il codice.
it.exprivia.secsisr.<sistema>.<sigla area>.web.command.<DescrizioneAzioneEntita>Command	Classe di supporto al Controller del caso d'uso corrispondente. Alleggerisce l'implementazione del Controller rendendo più modulare il codice. L'azione compiuta è strettamente legata al caso d'uso della specifica entità di business. it.exprivia.secsisr.<sistema>.<sigla area>.
it.exprivia.secsisr.<sistema>.<sigla area>.delegate.<entita>.<Entita>DelegateImpl	Il Model del pattern MVC ed elemento disaccoppiante fra la logica di business nello strato EJB ed il Controller secondo quanto dettato dal pattern Business Delegate. I metodi definiti sono it.exprivia.secsisr.<sistema>.<sigla area>.
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi.controller.web.MainController	Superclasse di tutti i controller web dei casi d'uso che raccoglie l'implementazione dei metodi a questi comuni.
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi.controller.web.MainControllerAjax	Superclasse di tutti i controller web dei casi d'uso con funzionalità asincrone Ajax che raccoglie l'implementazione dei metodi a questi comuni.
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi.controller.rest.RestCommon	Superclasse di tutti i controller rest dei casi d'uso che raccoglie l'implementazione dei metodi a questi comuni.
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi.controller.web.Ordina	Classe comparator di supporto per effettuare una comparazione di stringhe sulla base di una relazione d'ordine.
it.exprivia.secsisr.<sistema>.<sigla	Superclasse di tutti i validator dei casi d'uso che

area>.web.condivisi.controller.web.MainValidator	raccoglie l'implementazione dei metodi a questi comuni.
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi.cache.Cache	Classe per l'utilizzo dello strato di cache jboss infinispan utile per rendere l'applicazione scalabile e clusterizzabile a seconda delle esigenze di performance richieste e del carico elaborativo che il sistema deve sopportare. Nasconde tutti i dettagli implementativi della cache e consente di migliorare le performance.
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi.command.Command	Interfaccia da implementare per definire l'azione che lo specifico comando deve eseguire.
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi.command.HelperCommandExecutor	Classe di utilità che consente l'esecuzione immediata e sequenziale di comandi.
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi.command.OrdinaElencoCommand	Classe comparator di supporto per effettuare una comparazione di collection generiche tipizzate a runtime, sulla base di una relazione d'ordine.
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi.contento.GestoreContesto	Classe di utilità per la gestione e la memorizzazione dei dati del context applicativo.
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi.listener.SessionListener	Classe di gestione degli eventi associati al ciclo di vita della sessione web che esegue opportune operazioni al verificarsi di specifici eventi di sessione.
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi.listener.WebServerListener	Classe di gestione degli eventi associati al ciclo di vita del context applicativo che esegue opportune operazioni al verificarsi di specifici eventi del context.
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi.listener.TokenInfo	Classe utile per la memorizzazione e il tracciamento delle informazioni del client rest che effettua richieste json al server.
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi.supporto.AppRepositorySelector	Classe di utilità per la gestione del log applicativo implementato attraverso log4j.
Java Server Page Caso Uso	La View del pattern MVC. E' la pagina costruita dinamicamente sul server ed inviata al browser client. Vedi documentazione sulla tecnologia Java Server Page in J2EE.
org.springframework.web.servlet.DispatcherServlet	Servlet che si occupa di smistare tutte le richieste (POST, GET, ecc.) ai vari handlers, quindi funge fa Front Controller. La DispatcherServlet, essendo una servlet a tutti gli

	effetti, deve essere mappata nel file di configurazione “web.xml”. Vedi documentazione API spring-framework-3.2.3.
org.springframework.web.servlet.mvc.multiaction.MultiActionController	Rappresenta un controller generico di Spring che gestisce azioni multiple. Vedi documentazione API spring-framework-3.2.3.
org.springframework.web.servlet.view.InternalResourceViewResolver	Ogni Controller restituisce il nome logico di una view che viene risolto da questa classe.
org.springframework.web.servlet.handler.SimpleUrlHandlerMapping	Questa classe si occupa di mappare tutti gli URL ed il corrispondente Controller per gestire la richiesta di esecuzione di un flusso.
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi.listener.TokenInfo	Classe utile per la memorizzazione e il tracciamento delle informazioni del client rest che effettua richieste json al server.
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi.filter.TimerFilter	Classe filtro che consente al server di verificare quanto tempo il server impiega per smaltire una richiesta effettuata da un client.

Allegato 3 - Package che implementano l'EJB-tier con le relative responsabilità

Secsisr-<sistema>-<sigla area>-ejb-condivisi.jar	
Nome	**Descrizione**
it.exprivia.secsisr.<sistema>.<sigla area>.ejb.condivisi.MainEJB3	Superclasse di tutti gli EJB che raccoglie i metodi che hanno implementazione comune.
it.exprivia.secsisr.<sistema>.<sigla area>.ejb.condivisi.SqlUtil	Classe di utilità che consente di accedere a tutte le stringhe di query di CRUD esternalizzate dal codice della logica applicativa.
it.exprivia.secsisr.<sistema>.<sigla area>.vo.common.Pattern	Classe contenente i pattern utilizzati dalla logica di business

secsisr-<sistema>-<sigla area>-ejb.jar	
Nome	**Descrizione**
it.exprivia.secsisr.<sistema>.<sigla area>.ejb.<entita>.<Entita>Crud	Classe di supporto all'EJB per le letture e le query transazionali di dati della specifica entità dal DB.
it.exprivia.secsisr.<sistema>.<sigla area>.ejb.<entita>.<Entita>Interface	Superclasse delle interfacce Local e Remote. Contiene sia i servizi di business esposti all'esterno dell'EJB container tramite lookup via JNDI che quelli fruibili localmente nell'EJB container tramite lookup via JNDI
it.exprivia.secsisr.<sistema>.<sigla area>.ejb.<entita>.<Entita>Remote	Interfaccia remota dell'EJB della specifica entità.
it.exprivia.secsisr.<sistema>.<sigla area>.ejb.<entita>.<Entita>Local	Interfaccia locale dell'EJB della specifica entità.
it.exprivia.secsisr.<sistema>.<sigla area>.ejb.<entita>.<Entita>Impl	Implementazione della logica di business del session bean stateless della specifica entità. Espone i servizi di business tramite lookup via JNDI.
it.exprivia.secsisr.<sistema>.<sigla area>.entity.<Entita>	Classe entity per una *entità* applicativa. Tali classi mappano le tavole del DB
it.exprivia.secsisr.<sistema>.<sigla	Classe value object condivisa che

area>.enums.<TipoEnumeratore>	rappresenta una entità di tipo enumerativo
it.exprivia.secsisr.<sistema>.<sigla area>.validators.Check<TipoValidatore>	Classe di Annotation che abilita la verifica di validità del tipo
it.exprivia.secsisr.<sistema>.<sigla area>.validators.Check<TipoValidatore>Validator	Classe di validazione preposta alla verifica di validità dello stato della specifica entità

Allegato 4 - Package che implementano le classi di reporting

Prefisso del Package	Descrizione
it.exprivia.secsisr.<sistema>.<sigla area>.batch	Contiene i package comuni ai processi batch
it.exprivia.secsisr.<sistema>.<sigla area>.batch.common	Contiene classi e package condivisi da tutti i processi batch.
it.exprivia.secsisr.<sistema>.<sigla area>.batch.jobs	Contiene le classi che rappresentano i jobs dei processi batch.
it.exprivia.secsisr.<sistema>.<sigla area>.batch.steps	Contiene le classi che rappresentano i gli step di Lettura, Processazione e Scrittura (ItemReader, ItemProcessor, ItemWriter)
it.exprivia.secsisr.<sistema>.<sigla area>.batch.tasklet	Contiene le classi che rappresentano i tasklet ovvero le scomposizioni di steps in unità più elementari per una più efficiente organizzazione.
it.exprivia.secsisr.<sistema>.<sigla area>.batch.chunks	Contiene le classi che rappresentano i chunk ovvero le scomposizioni di steps in unità più elementari per una più efficiente organizzazione.
it.exprivia.secsisr.<sistema>.<sigla area>.batch.bean	Contiene le classi di trasporto POJO.

Allegato 5 - Package che contengono le classi che hanno la responsabilità della gestione dei log applicativi

Nome	Descrizione
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi.contesto.GestoreContesto	Gestisce i dati di contesto dell'applicazione web.
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi.servlet.MainServlet	Servlet astratta e quindi non istanziabile, superclasse di tutte le servlet dei casi d'uso e ne raccoglie l'implementazione dei metodi comuni.
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi.controller.MainController	Superclasse di tutti i controller dei casi d'uso e ne raccoglie l'implementazione dei metodi a questi comuni.
it.exprivia.secsisr.<sistema>.<sigla area>.web.area.casouso.CasoUsoController	Il Controller del pattern MVC, motore del page flow per il caso d'uso da cui prende il nome.
it.exprivia.secsisr.<sistema>.<sigla area>.web.condivisi.servlet.ServletHelper	Classe astratta e quindi non istanziabile, superclasse di tutte gli helper dei casi d'uso e ne raccoglie l'implementazione dei metodi comuni.
it.exprivia.secsisr.<sistema>.<sigla area>.web.componente.servlet.casouso.SvltCasoUso	Il Controller del pattern MVC. E' la servlet motore di page flow del caso d'uso da cui prende il nome.
it.exprivia.secsisr.<sistema>.<sigla area>.web.componente.servlet.casouso.SvltCasoUsoHelper	Classe di supporto alla servlet del caso d'uso corrispondente. Alleggerisce l'implementazione della servlet rendendo più modulare il codice.
log4J	log4J è una libreria Java sviluppata dalla Apache Software Foundation che permette di mettere a punto un sistema di logging per tenere sotto controllo il comportamento di una applicazione.

Allegato 6 - Package che contengono le classi che hanno la responsabilità della gestione dei parametri di configurazione delle aree applicative

Nome	Descrizione
it.exprivia.secsisr.<sistema>.<sigla area>.batch.areaapplicativa.batchcasouso.BatchCasoUso	Classe contenente il metodo “main” per l’avvio di un processo batch.
it.exprivia.secsisr.<sistema>.<sigla area>.web.proxy.parametroconfigurazione.DelegatoParametroConfigurazione.java	Il Model del pattern MVC ed elemento disaccoppiante fra la logica di business nello strato EJB e la servlet secondo quanto dettato dal pattern Business Delegate. Inoltre disaccoppia la logica di business dalla logica di accesso ai dati, implementata nel Data Access Object (DAO).
it.exprivia.secsisr.<sistema>.<sigla area>.ejb.entita.dao.ParametroConfigurazioneDao	Data Access Object (DAO) per la relativa entità, incapsula l'implementazione delle query SQL definite nelle interfacce che implementa.
it.exprivia.secsisr.<sistema>.<sigla area>.ejb.entita.bean.ParametroConfigurazioneBean	Session Ejb di facciata (facade): espone i servizi di business all'esterno tramite lookup via JNDI. Usa gli EJB local.
it.exprivia.secsisr.<sistema>.<sigla area>.ejb.entita.bean.ParametroConfigurazioneEJB	Session Ejb locale (local): espone i servizi di business agli Ejb di facciata tramite lookup locale al container.