Università degli Studi di Camerino

Scuola di Scienze e Tecnologie

Corso di Laurea in Informatica (Classe L-31)

TESI DI LAUREA

Scambio di flussi dati tra Banca tesoriera e Ente pubblico: Sviluppo di un web service

Laureando

Gianluca Mariani

MATRICOLA 077648

Relatore

Prof. Fausto Marcantoni

Correlatore

Dott. Mario Tartari

Anno Accademico 2012/2013

Progress is possible only if we train ourselves to think about programs without thinking of them as pieces of executable code.

Edsger W. Dijkstra

Indice

1 Introduzione

1.1 Presentazione

In questa tesi si intende presentare il lavoro svolto presso l'azienda RA Computer Spa.

Grazie a questa esperienza, durata quattro mesi, ho potuto ampliare le mie conoscenze informatiche collaborando allo sviluppo di un applicativo Java chiamato "TesoWebSign" capace di gestire l'Ordinativo Informatico Locale (O.I.L.), in linea con il D.Lgs. n. 82 del 7 marzo 2005, relativo al Codice dell'amministrazione digitale.

L'ordinativo informatico **è un'evidenza elettronica, dotata di validità amministrativa e contabile, che sostituisce, a tutti gli effetti, il documento di spesa** (mandato di pagamento) **e/o di incasso** (reversale d'incasso) **cartaceo prodotto dall'ente.**
Il conferimento della validità giuridica viene attribuito dalla firma digitale, che permette di identificare il sottoscrittore e di garantire l'integrità del documento.

L'applicativo permette agli enti, clienti delle banche che forniscono il servizio di firma digitale TesoWebSign, di accedere direttamente ai dati di loro pertinenza tramite qualsiasi computer equipaggiato con sistemi operativi Windows, Mac, Linux e di qualsiasi altro sistema in grado di collegarsi ad Internet utilizzando i browser tradizionali.

Il mio lavoro si è concentrato nello studio di un metodo per lo scambio dei flussi di dati tra la banca tesoriera e l'ente.

Di seguito sono elencati gli argomenti che verranno trattati approfonditamente:

- problematiche legislative e limiti di gestione del tesoro da parte degli enti pubblici;
- modalità di scambio dei flussi dati tra ente - banca tesoriera e requisiti di sicurezza;
- approfondimento sul metodo usato: web service, SOAP, WSDL;
- descrizione dell'applicazione finale;
- conclusioni.

1.2 English summary

In this thesis I'm going to present the work done for the company RA Computer Spa.

Thanks to this experience, lasted four months, I have been able to expand my skills collaborating on the development of a Java application called "TesoWebSign" capable of managing the "Ordinativo Informatico Locale" (O.I.L.), in line with the Decree. N. 82 of 7 March 2005 on the Digital Administration Code.

OIL is an **electronic evidence with validity administrative and accounting, which replaces, in effect, the document of expenditure** (payment order) **and/or collection** (order collection) **printed output produced by the public organizations.**
The provision of legal validity is given by the digital signature, which identifies the subscriber and ensure the integrity of the document.

The application allows organizations, customers of banks that provide the digital signature service TesoWebSign, direct access to data pertaining to them from any computer equipped with Windows, Mac, Linux or any other operating system that can connect to the Internet using traditional browsers.

My work has focused on the study of a method for the exchange of data flows between the bank treasurer and the organizations.

The following are the topics that will be treated in detail:

- legislative issues and limits management of the treasury for public organizations;
- exchange modality of data flows between public organizations - bank treasurer and safety requirements;
- discussion of used method: web services, SOAP, WSDL;
- description of the final application;
- conclusions.

1.3 RA Computer S.p.A.

Fondata nel 1980, da oltre 30 anni RA Computer è specializzata nella creazione di soluzioni software, di applicazioni web ma anche consulenza e progetti per soddisfare in modo integrato e esigenze delle Banche, Pubblica Amministrazione ed Imprese in termini di processi di pagamento.

Dal Giugno 2006 RA Computer è entrata a far parte del Gruppo SIA, gruppo leader a livello europeo sui sistemi di pagamento, ed ha arricchito la sua offerta con consulenza e servizi tecnologici in rete alle Imprese. Da maggio 2012 SIA S.p.A. diviene socio unico di RA Computer.

L'appartenenza al gruppo SIA rappresenta un'opportunità di forte crescita per il mercato di riferimento, che può contare sulla sinergia tra attori italiani in possesso di un forte know how tecnologico, elevata solidità finanziaria, eccellenza nella progettazione, innovazione e competitività.

Con un organico di più di 300 professionisti che lavorano distribuiti presso le tre sedi di Milano, Roma e Macerata, RA Computer vanta una presenza su tutte le maggiori Banche italiane, nella Pubblica Amministrazione e nelle Imprese.

La missione di RA Computer è quella di sviluppare un'offerta innovativa e competitiva nell'area dei sistemi di pagamento ed in aree ad essa associate, attraverso soluzioni, progetti e servizi per soddisfare in modo integrato le esigenze delle Istituzioni Finanziarie, Pubblica Amministrazione, Imprese e delle nuove Payment Institution.

2 Problematiche legislative

Il motivo per cui si rende necessaria la creazione di un software dedicato allo scambio di flussi dati tra una banca tesoriera e un ente è spiegato nel **Testo Unico degli Enti Locali**, di seguito sono elencate nel dettaglio le principali norme che vanno a limitare e regolamentare le funzioni amministrative di un ente locale.

2.1 Norme

[1] TESTO UNICO ENTI LOCALI

D.Lgs. n. 267/2000

PARTE II

Ordinamento finanziario e contabile

TITOLO V

Tesoreria

Capo I

Disposizioni generali

Articolo 208

Soggetti abilitati a svolgere il servizio di tesoreria.

1. Gli enti locali hanno un servizio di tesoreria che può essere affidato:

a) per i comuni capoluoghi di provincia, le province, le città metropolitane, ad una banca autorizzata a svolgere l'attività di cui all'articolo 10 del decreto legislativo 1° settembre 1993, n. 385;

b) per i comuni non capoluoghi di provincia, le comunità montane e le unioni di comuni, anche a società per azioni regolarmente costituite con capitale sociale interamente versato non inferiore a € 516.456,90, aventi per oggetto la gestione del servizio di tesoreria e la riscossione dei tributi degli enti locali e che alla data del 25 febbraio 1995, erano incaricate dello svolgimento del medesimo servizio a condizione che il capitale sociale risulti adeguato a quello minimo richiesto dalla normativa vigente per le banche di credito cooperativo;

c) altri soggetti abilitati per legge.

Articolo 209

Oggetto del servizio di tesoreria.

1. Il servizio di tesoreria consiste nel complesso di operazioni legate alla gestione finanziaria dell'ente locale e finalizzate in particolare alla riscossione delle entrate, al pagamento delle spese, alla custodia di titoli e valori ed agli adempimenti connessi previsti dalla legge, dallo statuto, dai regolamenti dell'ente o da norme pattizie.

...

...

Articolo 213

Gestione informatizzata del servizio di tesoreria

1. Qualora l'organizzazione dell'ente e del tesoriere lo consentano il servizio di tesoreria viene gestito con metodologie e criteri informatici e con l'uso di ordinativi di pagamento e di riscossione informatici, in luogo di quelli cartacei, le cui evidenze informatiche valgono a fini di documentazione, …

…

2.2 La situazione italiana

[2] Per un paese come l'Italia che sta mutando profondamente il suo assetto istituzionale, anche verso un'organizzazione di tipo federale, il ricorso alle tecnologie informatiche e della comunicazione rappresenta un fattore essenziale.
L'innovazione del sistema amministrativo passa necessariamente anche dalla possibilità di un monitoraggio puntuale e tempestivo dei flussi finanziari delle amministrazioni pubbliche.
Per questo è necessario superare l'attuale inadeguatezza conoscitiva e temporale delle rilevazioni sui flussi di cassa eliminando la disomogeneità dei dati provenienti dai differenti sistemi contabili delle amministrazioni pubbliche.

Queste necessità, coniugate con la necessità di un contenimento della spesa pubblica e di una riorganizzazione delle procedure amministrative, impongono il ricorso alle tecnologie ICT quali strumenti a supporto per la semplificazione dei processi organizzativi che portano a liberare risorse umane, finanziarie e strumentali.

Un primo passo in questo senso è stato realizzato con la costituzione del Sistema Informatizzato dei Pagamenti della Pubblica Amministrazione (SIPA) che ha permesso di integrare i pagamenti dello Stato nelle procedure interbancarie permettendo attualmente alle amministrazioni centrali, di gestire in modo completamente automatico le proprie spese.

L'infrastruttura tecnologica del SIPA, che si basa sull'interconnessione tra la Rete Unitaria della Pubblica Amministrazione (RUPA) e la Rete Nazionale Interbancaria (RNI) fornisce il modello di tesoreria telematica degli incassi e pagamenti al Sistema Informativo delle Operazioni degli Enti Pubblici (SIOPE).

La realizzazione del SIOPE passa per la creazione di un **flusso informativo** classificato secondo delle codifiche gestionali apposte sugli ordinativi emessi dalle amministrazioni locali che, tramite il coinvolgimento di tutte le banche tesoriere, permette l'aggiornamento giornaliero dell'archivio di finanza pubblica gestito dalla Banca d'Italia.

La completa, costante e sicura alimentazione del SIOPE, è necessaria anche da parte delle amministrazioni locali, regionali ed università, la totale automazione dei processi di emissione degli ordinativi di spesa e incasso attraverso la standardizzazione dei rapporti telematici tra banche tesoriere ed ente ovvero l'adozione dell'**Ordinativo Informatico Locale** (OIL).

Il servizio è rivolto a tutti gli enti e realtà della Pubblica Amministrazione, dislocati sul territorio nazionale, che hanno la necessità di sostituire i tradizionali mandati e reversali su supporto cartaceo con evidenze informatiche in grado di automatizzare il processo di gestione dei pagamenti e degli incassi.

3 Modalità di scambio dei flussi

In questi capitoli vengono presentate le caratteristiche fondamentali e i requisiti richiesti per lo scambio dei “flussi dati” nell’applicativo finale, in base alle direttive presenti nelle linee guida OIL della DigitPA (Ente nazionale per la digitalizzazione della Pubblica Amministrazione).

3.1 Linee guida

[3] Gli ordinativi informatici sono scambiati fra gli enti del comparto pubblico (in seguito denominati "PA") e le banche tesoriere o cassiere (in seguito denominate "BT") e sostituiscono quelli cartacei.
Gli ordinativi informatici sono costituiti da mandati di pagamento e reversali d'incasso. Ogni mandato di pagamento o reversale di incasso contiene una o più “disposizioni”.

Per le PA che intendono adottare l’ordinativo informatico sottoscritto con firma digitale è buona norma produrre ordinativi con un solo versante/beneficiario, non essendoci più il documento cartaceo questo non comporta dei costi aggiuntivi per la stampa e l’archiviazione.

Dall’esperienza è emerso che la gestione di ordinativi con un solo versante/beneficiario comporta una notevole semplificazione nella gestione dei documenti informatici sia per la PA che per la BT, e in particolare nelle operazioni di archiviazione, ricerca e correzione (VARIAZIONE, ANNULLO e SOSTITUZIONE).

Con l’apposizione di un’unica firma digitale può essere sottoscritto un singolo ordinativo informatico oppure più ordinativi informatici. In ogni caso, il sistema di gestione informatica dei documenti della PA (in seguito denominato anche “sistema mittente”), come definito all’art. 1 del Testo Unico approvato con decreto del Presidente della Repubblica 28 dicembre 2000 n. 445 e s.m.i. (in seguito denominato “Testo Unico”), deve rappresentare senza ambiguità gli elementi di ciascun ordinativo informatico cui la firma si riferisce.

Nel seguito, il complesso dei dati firmati (riferiti a uno o più ordinativi informatici) viene per brevità definito “flusso”.

3.2 Requisiti di sicurezza

La trasmissione telematica dei flussi tra PA e BT deve avvenire mediante qualsiasi strumento idoneo a garantire l'identità dei soggetti attori, la sicurezza degli accessi, la riservatezza delle informazioni e, più in generale, il rispetto delle regole previste nel presente regolamento.

La BT documenta e mantiene aggiornate le modalità operative per l'esecuzione degli ordinativi informatici. In particolare la BT documenta e mantiene aggiornate le specifiche tecniche e le modalità di interscambio dei dati per via telematica e per la comunicazione alla PA dell'avvenuta esecuzione degli ordinativi stessi. Inoltre la BT e la PA definiscono i requisiti relativi all'eventuale documentazione cartacea che deve essere inoltrata alla BT a supporto degli ordinativi informatici, le disposizioni per il pagamento degli ordinativi di spesa urgenti, le modalità di aggiornamento e conservazione delle informazioni.

Il processo di trasmissione di un «flusso» fra la PA e la BT si svolge secondo il seguente schema di base:

- ai fini della sottoscrizione, della ricezione e della verifica, viene gestito il «flusso»;
- ai fini dell'esecuzione (estinzione), della variazione, della sostituzione o dell'annullamento, devono essere considerati i singoli ordinativi informatici. Il trattamento e l'eventuale conseguente rendicontazione di un singolo ordinativo informatico non deve condizionare il trattamento di altri ordinativi informatici eventualmente contenuti nel medesimo flusso.

I formati adottati devono possedere almeno i seguenti requisiti:

- consentire, nei diversi ambiti di applicazione e per le diverse tipologie di trattazione, l'archiviazione, la leggibilità, l'interoperabilità e l'interscambio dei «flussi»;
- la non alterabilità dei «flussi» durante le fasi di accesso e conservazione;
- la possibilità di effettuare operazioni di ricerca tramite indici di classificazione o di archiviazione;
- l'immutabilità del contenuto e della sua struttura. A tale fine i «flussi» non devono contenere macroistruzioni o codice eseguibile, tali da attivare funzionalità che possano modificarne nel tempo la struttura o il contenuto.

Al fine di garantire il rispetto del requisito di interoperabilità si prevede una rappresentazione in formato XML del flusso contenente gli ordinativi informatici e dei messaggi di ritorno. Dovranno essere definite strutture che ne consentano la validazione sia presso la PA (all'atto della generazione) e sia presso la BT (all'atto della verifica formale) e viceversa.

Le strutture rappresentano lo standard a cui le PA e le BT devono attenersi; il rigoroso rispetto dello standard è indispensabile per garantire l'interoperabilità.
Per la visualizzazione dei flussi, devono essere adottate soluzioni che presentino le informazioni in modo fedele alla struttura.

4 Tecnologie utilizzate

4.1 Web service

4.1.1 Cos'è un web service

[4][5][6] Un web service è un componente applicativo. È possibile definirlo come un sistema software in grado di mettersi al servizio di un'applicazione comunicando su di una medesima rete tramite il protocollo HTTP, questo consente quindi alle applicazioni che vi si collegano di usufruire delle funzioni che mette a disposizione.
Un web service comunica tramite protocolli e standard definiti "aperti" e quindi sempre a disposizione degli sviluppatori, ha inoltre un'altra caratteristica molto particolare utile al suo scopo: è auto-contenuto ed auto-descrittivo.

Un servizio web è in grado di offrire un'interfaccia software assieme alla descrizione delle sue caratteristiche, ovvero è in grado di farci sapere quali funzioni mette a disposizione (senza bisogno di conoscerle a priori) e ci permette inoltre di capire come vanno utilizzate, è possibile stabilire le operazioni che fornisce ed iniziare immediatamente ad usarle, in quanto ogni operazione ha una sua descrizione comprendente i parametri che si aspetta di ricevere, quelli che restituirà ed il tipo di entrambi.

Il funzionamento dei web service è molto semplice. Essi utilizzano il protocollo HTTP per mettersi in comunicazione con l'applicazione che intende usufruire delle loro funzioni, dopodiché starà all'applicazione sfruttare i servizi web a proprio piacimento.

Oltre ad HTTP però, i servizi web utilizzano molti altri standard web, tutti basati su XML, tra cui:

- **XML Schema**
- **WSDL** (Web Services Description Language)
- **SOAP** (Simple Object Access Protocol)

XML può essere utilizzato correttamente tra piattaforme differenti (Linux, Windows, Mac) e differenti linguaggi di programmazione, è inoltre in grado di esprimere messaggi e funzioni anche molto complesse e garantisce che tutti i dati scambiati possano essere utilizzati ad entrambi i capi della connessione.

4.1.2 Perché utilizzare i web service

Questa è una lista dei principali motivi per utilizzare o sviluppare un web service:

- i web service permettono l'interoperabilità tra diverse applicazioni software e su diverse piattaforme hardware/software;
- utilizzano un formato dei dati di tipo testuale, quindi più comprensibile e più facile da utilizzare per gli sviluppatori (esclusi ovviamente i trasferimenti di dati di tipo binario);
- normalmente, essendo basati sul protocollo HTTP, non richiedono modifiche alle regole di sicurezza utilizzate come filtro dai firewall;
- sono semplici da utilizzare e possono essere combinati l'uno con l'altro (indipendentemente da chi li fornisce e da dove vengono resi disponibili) per formare servizi "integrati" e complessi;
- permettono di riutilizzare applicazioni già sviluppate;
- fintanto che l'interfaccia rimane costante, le modifiche effettuate ai servizi rimangono trasparenti;
- i *servizi web* sono in grado di pubblicare le loro funzioni e di scambiare dati con il resto del mondo;
- tutte le informazioni vengono scambiate attraverso protocolli "aperti".

4.1.3 Problematiche dei web service

Come ogni tecnologia anche i web service presentano alcuni problemi:

- **Le performance**. I web service presentano performance drasticamente inferiori rispetto ad altri metodi di comunicazione utilizzabili in rete. Questo svantaggio è legato alla natura stessa dei *servizi web*. Essendo basati su XML ogni trasferimento di dati richiede l'inserimento di un notevole numero di dati supplementari (i tag XML) indispensabili per la descrizione dell'operazione. Inoltre tutti i dati inviati richiedono di essere prima codificati e poi decodificati ai capi della connessione. Queste due caratteristiche dei web service li rendono poco adatti a flussi di dati intensi o dove la velocità dell'applicazione rappresenti un fattore critico.
- **Il protocollo base, HTTP**. Quando si sviluppa un web service è necessario tener conto del protocollo di base. È quindi indispensabile disporre di un'applicazione terza che gestisca le richieste HTTP oppure è necessario includerla direttamente nel codice del nostro programma qualora si desideri la sua totale indipendenza. Va detto comunque che generalmente il codice che implementa un web service viene fatto eseguire da un Web server (es. Apache) tramite CGI (per es. con Python) o tramite appositi moduli (vedi PHP). Eseguendo il codice del web service attraverso un server web la gestione di HTTP è immediatamente assicurata.

4.2 SOA

Nel mondo eterogeneo del Web nasce la necessità di avere un modo “standard” per esporre un software su una rete attraverso un’interfaccia comprensibile da tutte quelle aziende che, volendo far interagire le proprie applicazioni con quelle di altre compagnie, riconoscono tale standard.
È così che sono introdotti i web service e viene adottata l’Architettura Orientata ai Servizi (SOA).

La Service-Oriented Architecture è un’architettura concettuale che non fa riferimento a nessuna particolare implementazione. Essa pone delle specifiche condizioni che i componenti del sistema devono rispettare e caratteristiche che tale sistema deve necessariamente avere.

I web service sono invece una nuova tecnologia, che si basa su standard e che fornisce un facile metodo di interfacciamento tra le applicazioni. Questa tecnologia non è legata all’architettura SOA ma ha molti punti di contatto con essa e sistemi che utilizzano i web service, sfruttando tutte le loro potenzialità, implementano esattamente un’architettura di tale tipo.

Le principali caratteristiche che un servizio SOA deve avere sono:

- essere autocontenuto e modulare;
- essere definito da un’interfaccia ed indipendente dall’implementazione;
- essere ricercabile e recuperabile dinamicamente;
- essere debolmente accoppiato con altri servizi (loosely coupled);
- essere reso disponibile sulla rete attraverso la pubblicazione della sua interfaccia (in un Service Directory o Service Registry) ed accessibile in modo trasparente rispetto alla sua allocazione;
- fornire un’interfaccia possibilmente a “grana grossa” (coarse-grained): deve mettere a disposizione un basso numero di operazioni;
- essere realizzato in modo tale da permetterne la composizione con altri.

4.3 XML

XML (eXtensible Markup Language) è un linguaggio di markup, ovvero un linguaggio marcatore basato su un meccanismo sintattico che consente di definire e controllare il significato degli elementi contenuti in un documento o in un testo.

Il compito di un documento XML è memorizzare i dati all’interno di una struttura gerarchica che rappresenti le relazioni esistenti fra di essi, senza curarsi minimamente della loro rappresentazione visuale. Tali dati possono poi essere visualizzati in molti modi differenti, a seconda del caso, come ad esempio una semplice pagina HTML.

4.4 SOAP

La tecnologia web service è essenzialmente basata sul messaggio e quindi era necessario adottare un insieme di regole per poter inoltrare una richiesta client e costruire quindi un messaggio che possa essere capito dal server. Il server di conseguenza utilizzerà lo stesso formato per la risposta. Si è voluto scegliere un metodo standard per rappresentare i dati scambiati e la scelta non poteva che ricadere su XML. Il trasporto di questi dati in formato XML è affidato al protocollo HTTP. Dalla combinazione di XML e HTTP nasce proprio il protocollo SOAP (Simple Object Access Protocol), il protocollo standard per i web service.

Come precedentemente spiegato, SOAP è basato su XML. In effetti, un messaggio SOAP non è altro che un documento XML che descrive una richiesta di elaborazione o il risultato di una elaborazione, dove le informazioni contenute nella richiesta e nella risposta vengono serializzate secondo un formato prestabilito, utilizzando XML come strumento che garantisce l'indipendenza dalla piattaforma.

Il protocollo SOAP può essere informalmente distinto dall'equazione: XML + HTTP = SOAP. In maniera più formale, come definito nelle sue specifiche, questo protocollo è costituito da tre parti che sono state pensate per essere funzionalmente ortogonali, permettendo così una maggior semplicità, ottenuta attraverso la modularizzazione. Le tre parti sono:

- *SOAP envelope*: definisce una struttura per descrivere cosa c'è nel messaggio, come deve essere processato e se esso è opzionale o obbligatorio;
- *SOAP encoding rule:* un meccanismo di serializzazione che può essere usato per scambiare istanze di tipi di dati definiti dalle applicazioni;
- *SOAP RPC rapresentation:* una covenzione utilizzata per rappresentare le chiamate e le risposte alle procedure remore (RPC).

L'architettura abbastanza semplice è la seguente [Figura 1]:

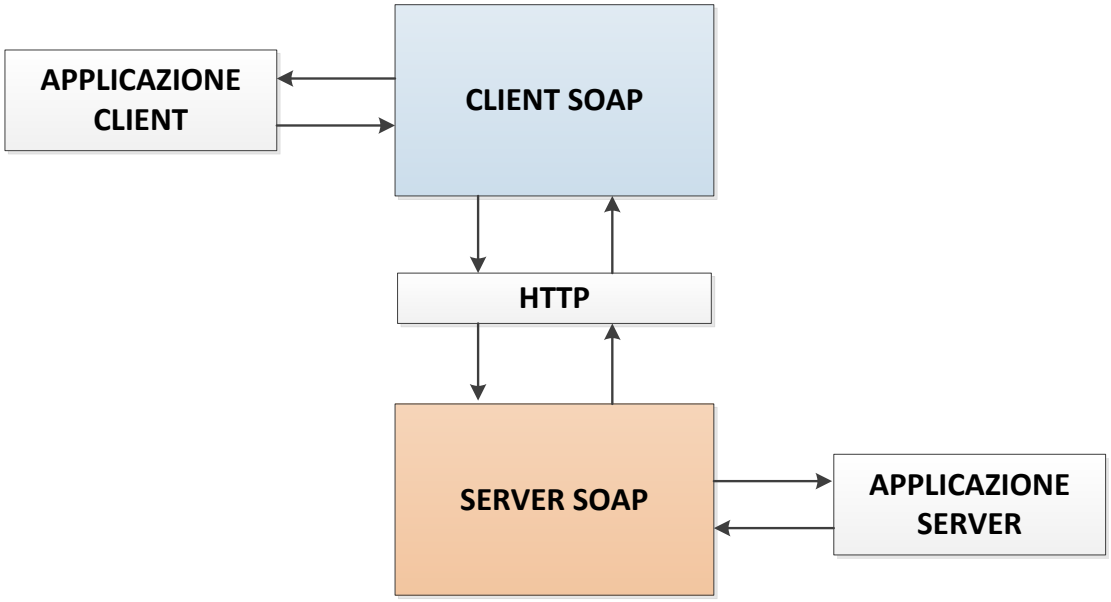


Figura 1 - Architettura SOAP

Il protocollo SOAP è adatto a supportare un'architettura client-server: i dati richiesti ed elaborati fra client e server sono organizzati in messaggi SOAP e vengono trasportati attraverso il protocollo HTTP o un altro protocollo di trasporto. I messaggi SOAP sono fondamentalmente trasmissioni unidirezionali da un mittente ad un destinatario ma sono spesso combinati per implementare modelli del tipo richiesta/risposta.

SOAP consiste di tre parti: una busta, che definisce un framework, per la descrizione del contenuto del messaggio e della modalità di elaborazione (SOAP envelope construct), un insieme di regole di codifica per l'espressione di istanze di tipi di dati definiti dalle applicazioni (SOAP encoding) ed una convenzione per la rappresentazione di chiamate e risposte di una procedura remota (SOAP RPC).

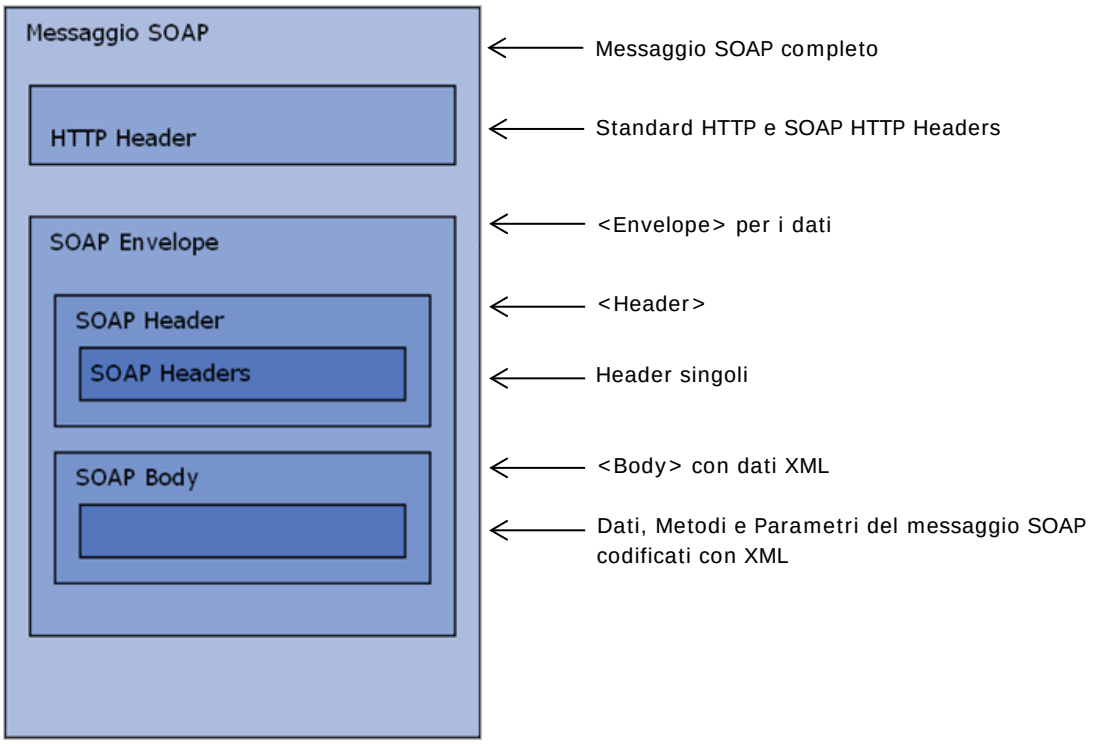


Figura 2 - Struttura di un messaggio SOAP

Un messaggio SOAP, come mostrato in figura 2, è un documento XML che contiene i seguenti elementi:

- **Envelope**, identifica il documento come un messaggio SOAP.
- Un elemento **Header** opzionale, contenete informazioni specifiche per l'applicazione, che permette di definire alcuni messaggi, anche con diversi destinatari nel caso il messaggio dovesse attraversare più punti di arrivo.
- **Body** è un elemento indispensabile che contiene le informazioni scambiate dalle richieste/risposte.
- **Fault** è un elemento opzionale che fornisce informazioni riguardo ad eventuali errori manifestati durante la lettura del messaggio.

Il Document Type Definition (DTD) di un documento XML costituisce l'insieme di regole cui il documento deve sottostare, fornendo una sorta di grammatica.

Un Namespace è un nome fittizio, un alias, che si è soliti assegnare ad un percorso remoto di un file, tipicamente un file di schema che viene legato al documento XML.
I namespace permettono di definire dei nomi in modo universale. Un namespace è una collezione di nomi, identificati da un Universal Resource Identifier (URI), utilizzati in XML come tipi di elementi o nomi di attributi.

Le regole principali per realizzare un messaggio SOAP sono le seguenti:

- deve essere codificato con XML;
- deve utilizzare il SOAP Envelope namespace;
- deve utilizzare il SOAP Encoding namespace;
- non deve contenere il collegamento ad un DTD e non deve contenere istruzioni per processare XML.

4.5 WSDL

Una volta definito il modo in cui comunicare è necessario sapere cosa mettere a disposizione e cosa è possibile rendere disponibile con un web service.
Se si può utilizzare un metodo di un oggetto, allora è necessario sapere: come si chiama il metodo, che parametri accetta, di che tipo e come va interpretato il risultato ottenuto.

È proprio questo il compito del file **WSDL** (Web Service Description Language), un file in formato XML che descrive il web service in termini di metodi richiamabili, nomi dei parametri, tipi, …
È una sorta di contratto tra client e server, che si accordano in questo modo sulle informazioni da scambiarsi.

WSDL ha quindi come obiettivo la descrizione dei servizi Web esposti da un server. In particolare un documento WSDL è un file XML che presenta al suo interno diverse sezioni:

- la definizione del servizio *(service)*: informazioni su un servizio esposto;
- i collegamenti *(binding)*: descrive il protocollo supportato, le operazioni consentite ed i relativi input e output;
- le operazioni *(portType)*: descrive l'insieme delle operazioni supportate e ognuna delle quali costituita dai messaggi di richiesta e di risposta;
- i messaggi *(message)*: contiene i parametri di richiesta e di risposta del servizio;
- i tipi (t*ypes)*: contiene le definizioni dei tipi di dati utilizzati in ingresso ed in uscita con la descrizione della loro struttura.

5 Sviluppo di un web service

5.1 I ruoli fondamentali

Il modello dei web service, da un punto di vista orientato ai servizi, è basato su tre ruoli fondamentali e sulle necessarie interazioni ciascun ruolo.

I tre ruoli [Figura 3] sono:

- **Service Provider** – È una piattaforma che fornisce l'accesso al servizio. Per esporre un'applicazione come un servizio, il provider deve fornire un accesso basato su un protocollo standard e una descrizione standard del servizio (meta data).
- **Service requester o user** – In generale, è un'applicazione che invoca o avvia un'interazione con un servizio. Il service requester e il service provider devono condividere le stesse modalità di accesso e interazione col servizio.
- **Service registry** – È un registro presso il quale i service provider possono pubblicare il servizio e i service requester possono trovare i servizi desiderati. Il service registry deve fornire una tassonomia consistente dei web service per facilitarne la scoperta, e una descrizione dettagliata dell'azienda che fornisce il servizio e del servizio stesso.

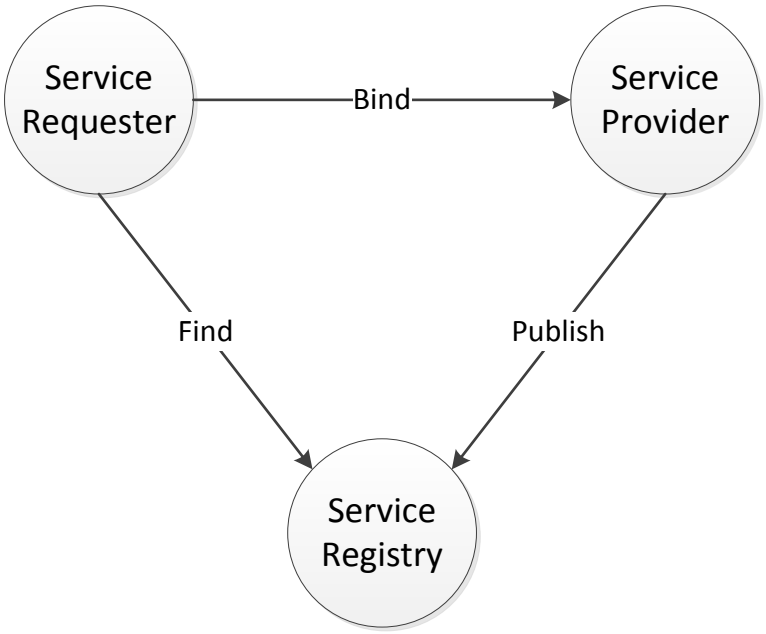


Figura 3 - Ruoli e interazioni

Le operazioni che consentono l'interazione tra i tre componenti del modello sono:

- **Publish** – Il service provider invia la descrizione del servizio, che ospita, al service registry. Questo classifica il servizio e lo pubblica in maniera tale che un service requester possa trovarlo.
- **Find** – Il service requester interroga il service registry per ottenere la descrizione del servizio richiesto.
- **Bind** – Il service requester utilizza i dettagli presente nella descrizione del servizio per contattare il service provider mediante il quale interagisce con l'implementazione del servizio.

5.2 Pubblicazione e utilizzo del servizio web

XML Schema, WSDL e SOAP sono i tre standard principali che bisogna assolutamente conoscere per comprendere nel dettaglio il funzionamento di un servizio web.

XML Schema serve per definire qual è la costruzione legale di un documento XML.
Questa è la lista delle principali funzioni di XML Schema:

- definire gli elementi (tag) che possono apparire in un documento;
- definire gli attributi che possono apparire in un elemento;
- definire quali elementi devono essere inserirli in altri elementi (detti 'figli');
- definire il numero degli elementi figli;
- definire quando un elemento dev'essere vuoto o può contenere testo, elementi, oppure entrambi;
- definire il tipo per ogni elemento e per gli attributi (intero, stringa, …);
- definire i valori di default per elementi ed attributi.

Esistono sostanzialmente due tipi di elementi:

- gli elementi semplici non possono contenere altri elementi e avere attributi. Possono contenere solo testo;
- gli elementi complessi possono contenere testo, altri elementi e attributi in qualsiasi combinazione.

[7] La forza di un web service è la standardizzazione e la comunicazione che avviene grazie allo scambio di informazioni in forma testuale.
Questo, da una parte limita le performance per l'overhead di elaborazione della semantica dei messaggi, ma dall'altra consente un utilizzo obliquo da tutte le piattaforme software.
Inoltre, un altro punto di forza è quello legato alla sicurezza: i messaggi testuali viaggiano su protocolli e porte "aperte" ai firewall.

Di seguito, figura 4, viene rappresentata l'architettura di un web service, il processo di pubblicazione (da parte di chi crea il servizio) e di utilizzo (da parte di chi lo vuole utilizzare).

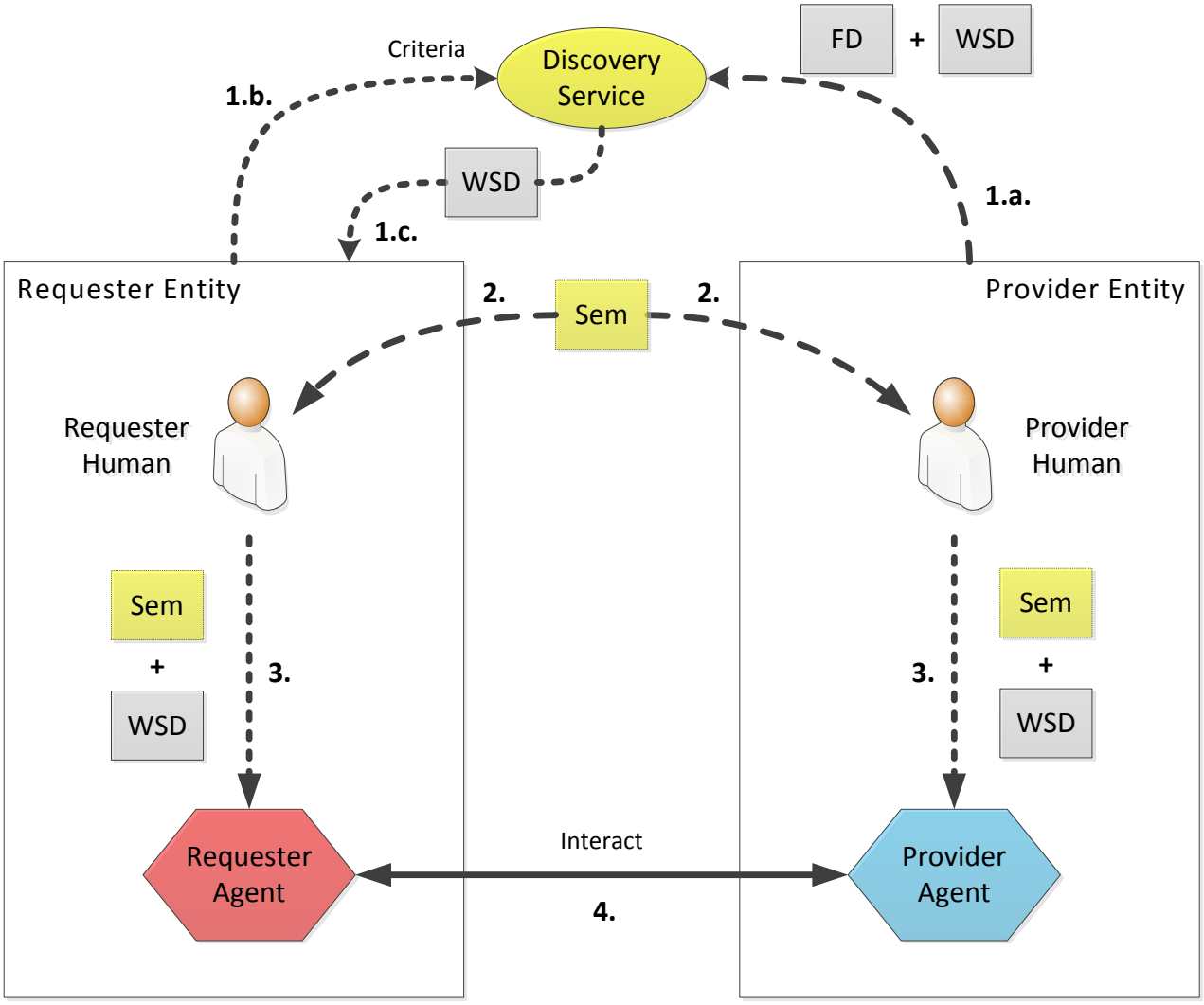


Figura 4 - Architettura di un web service

1. Le entità Requester e Provider si "scoprono" tra loro
   a. Il provider (chi crea il web service) crea il servizio la cui descrizione (nome dei metodi, parametri attesi, parametri di ritorno, ecc) è affidata ad un documento WSDL, cioè un formato standard XML. La pubblicazione del servizio viene affidata ad un registro (terzo rispetto le parti).
   b. Il client interroga il registro per scoprire (fase di discovery) i servizi di cui ha bisogno.
   c. Nel momento in cui la ricerca ha esito positivo, chiede al registro il documento WSDL, che definisce la struttura del servizio, e quindi indirizzi e modi sul come interrogarlo.

2. Il passaggio seguente è un accordo tra le parti, il requester e il provider devono concordare sulla semantica (Sem) dell'interazione desiderata.

3-4. L'ultimo passaggio è la messa in produzione del sistema. Il client crea uno strato software (request agent) a partire dal WSDL che interagisce con lo strato software (provider agent, cioè il servizio vero e proprio) fornitore del servizio. Lo scambio di messaggi avviene utilizzando un altro protocollo XML, SOAP (Simple Object Access Protocol), anch'esso standard.
La richiesta (in forma testuale) viene scompattata dal suo involucro SOAP, portata in forma binaria ed elaborata secondo la logica del servizio. La risposta, a sua volta viene portata in forma testuale, convogliata in un messaggio SOAP ed inoltrata (generalmente su protocollo HTTP) verso il mittente.

La cosa interessante, dal punto di vista dello sviluppatore è che i meccanismi di comunicazione (messaggi SOAP) vengono creati in maniera semplice a partire dal documento WSDL. Infatti, chi si occupa di sviluppo (lato client o lato server) dovrà preoccuparsi di creare la logica del servizio e definire il WSDL del servizio stesso. La creazione dell'infrastruttura di comunicazione sarà automatizzata dalla presenza di opportuni strumenti di sviluppo.

5.3 Sviluppo in Java

Realizzare un web service in J2EE non è una procedura particolarmente complicata. Grazie alla presenza di strumenti che supportano questa operazione un web container come Tomcat può esporre in maniera piuttosto semplice un servizio sotto forma di web service.

Il funzionamento di un web service in ambiente J2EE è illustrato nella figura 5.

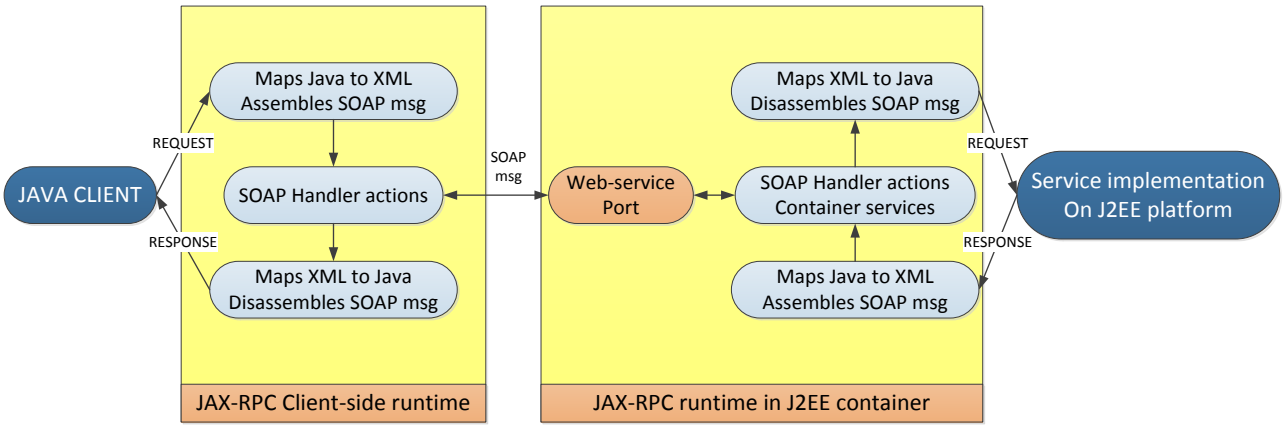


Figura 5 - Comunicazione tra client e web service

Si focalizzi ora l'attenzione sulla parte sinistra (lato server) dell'immagine.
L'implementazione del servizio è indipendente dal modo in cui esso verrà erogato, quindi potrà essere prevista semplicemente una classe java con relativi metodi. Il descrittore WSDL definisce in che modo si dovrà accedere alla classe che fornisce il servizio. Tutto quello che sta in mezzo (trasformazione del messaggio SOAP in comando java e viceversa) è a carico del middleware, che creerà l'opportuna infrastruttura.
Infatti, compito dello sviluppatore sarà: creare il servizio, definire il descrittore ed installare il tutto sulla macchina server.

Lato client, la procedura sarà analoga. Sarà sufficiente conoscere l'indirizzo del descrittore WSDL e costruire attorno ad esso i giusti messaggi SOAP (lo farà sempre uno strato di middleware).

Tralasciando la procedura di pubblicazione di un servizio, quello che serve alla creazione è la definizione della logica (anche una semplice classe), un descrittore WSDL e l'installazione su un web container predisposto a pubblicare web service. Il middleware si preoccuperà di configurare opportunamente il servizio per essere accessibile.

Per quattro semplici motivi: performance, scalabilità, sicurezza, affidabilità. Cose che una semplice classe java, da sola, non garantisce.

Per lo sviluppo di web service che rispondano direttamente dal container (strato web application) è possibile citare il progetto Apache Axis, questo framework permette lo sviluppo di un web service a partire dal descrittore WSDL oppure a partire dalla logica definita in una classe. In pratica la logica del servizio risiede a livello di web application, quindi gestita dal container.

Lo sviluppo di web service su tecnologia J2EE è molto più interessante, proprio perché permette di avere a disposizione tutti i servizi offerti dall'application server. Il punto di logica applicativa di un web service è uno stateless session bean. È intuitivo comprendere il perché: una richiesta ad un web service è per sua natura una richiesta senza stato.

In generale, il fatto di poter usufruire di tutti i servizi messi a disposizione dall'application server garantisce ottime performance al servizio creato. La gestione delle transazioni, l'eventuale accesso a sorgenti dati, le politiche di sicurezza, l'eventuale scalabilità, sono tutte caratteristiche curate dall'application server. In molti casi reali, soprattutto parlando di web service, acceduti da diverse organizzazioni distribuite geograficamente, questo è quanto meno necessario.

L'unico potenziale problema è legato al modo con il quale l'application server crea il middleware che gestisce lo scambio messaggi. In realtà, vista l'importanza dell'argomento, la stessa Sun ha promosso una serie di iniziative per definire degli standard sulla produzione dei web service. In particolare, la presenza del Java Web Services Developer Pack (JWSDP) permette la creazione di package indipendenti dalla piattaforma su cui verranno installati.

5.4 Il flusso dati

È possibile sintetizzare in uno schema, vedi figura 6, i vari passaggi del flusso dati:

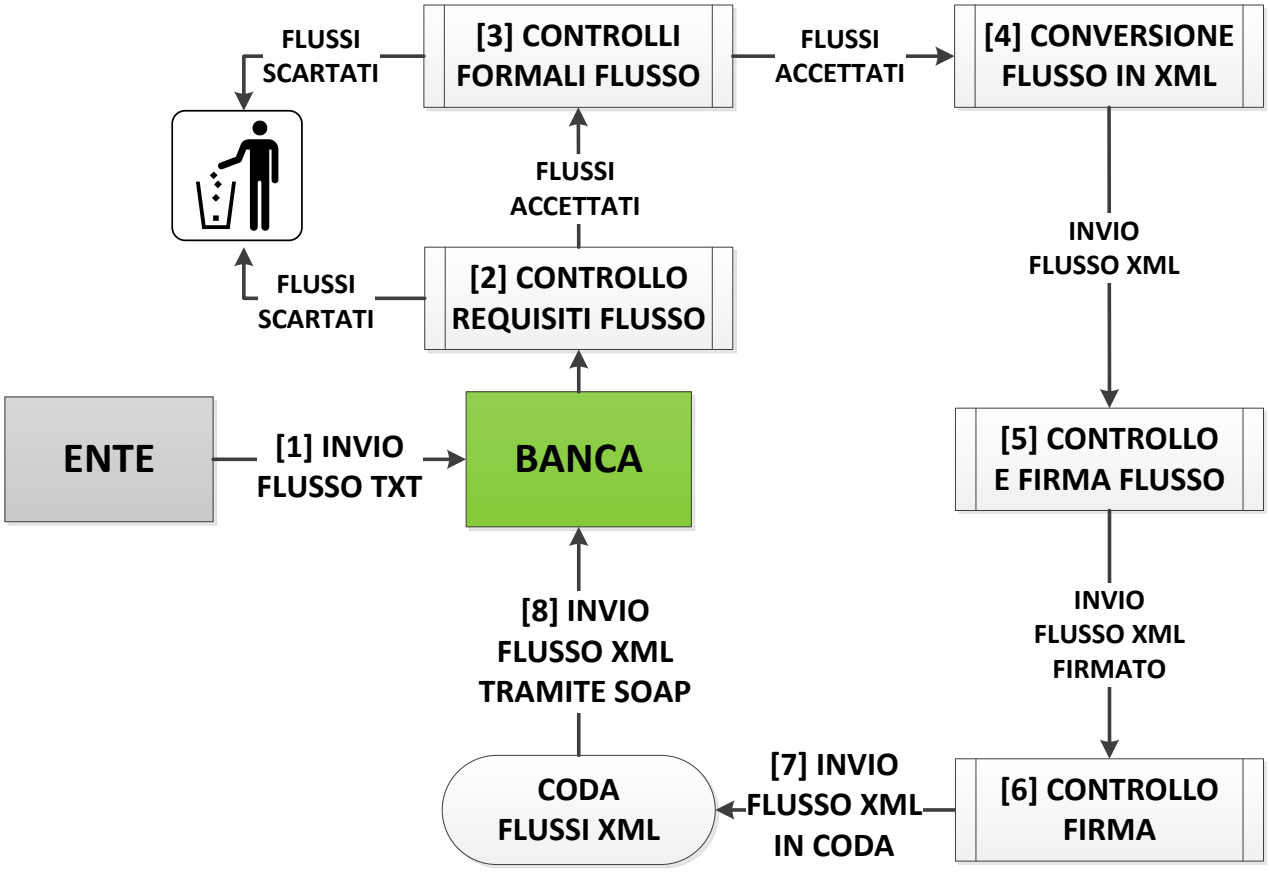


Figura 6 - Percorso del flusso dati

1. l'ente invia il flusso TXT alla banca;
2. la banca esegue un controllo dei requisiti di base del flusso e lo rifiuta se non coerente con gli attributi concordati (formato, lunghezza, separatori utilizzati, …);
3. la banca esegue controlli formali sul flusso;
4. la banca normalizza e traduce il flusso in formato XML (in base agli standard previsti per l'OIL);
5. l'ente controlla gli ordinativi tradotti, li firma ed autorizza il tesoriere ad eseguirli;
6. l'applicazione a questo punto controlla l'autenticità, la validità e i poteri di firma (ossia se chi ha firmato era autorizzato a farlo);
7. se tutti i controlli hanno esito positivo il flusso viene inserito in coda con altri XML in attesa della registrazione;
8. l'XML in coda viene inviato alla banca tramite il protocollo SOAP e registrato, in attesa di esecuzione.

In figura 7 viene mostrato uno schema di flusso più dettagliato riguardante la procedura complessiva dell'Ordinativo Informatico Locale:

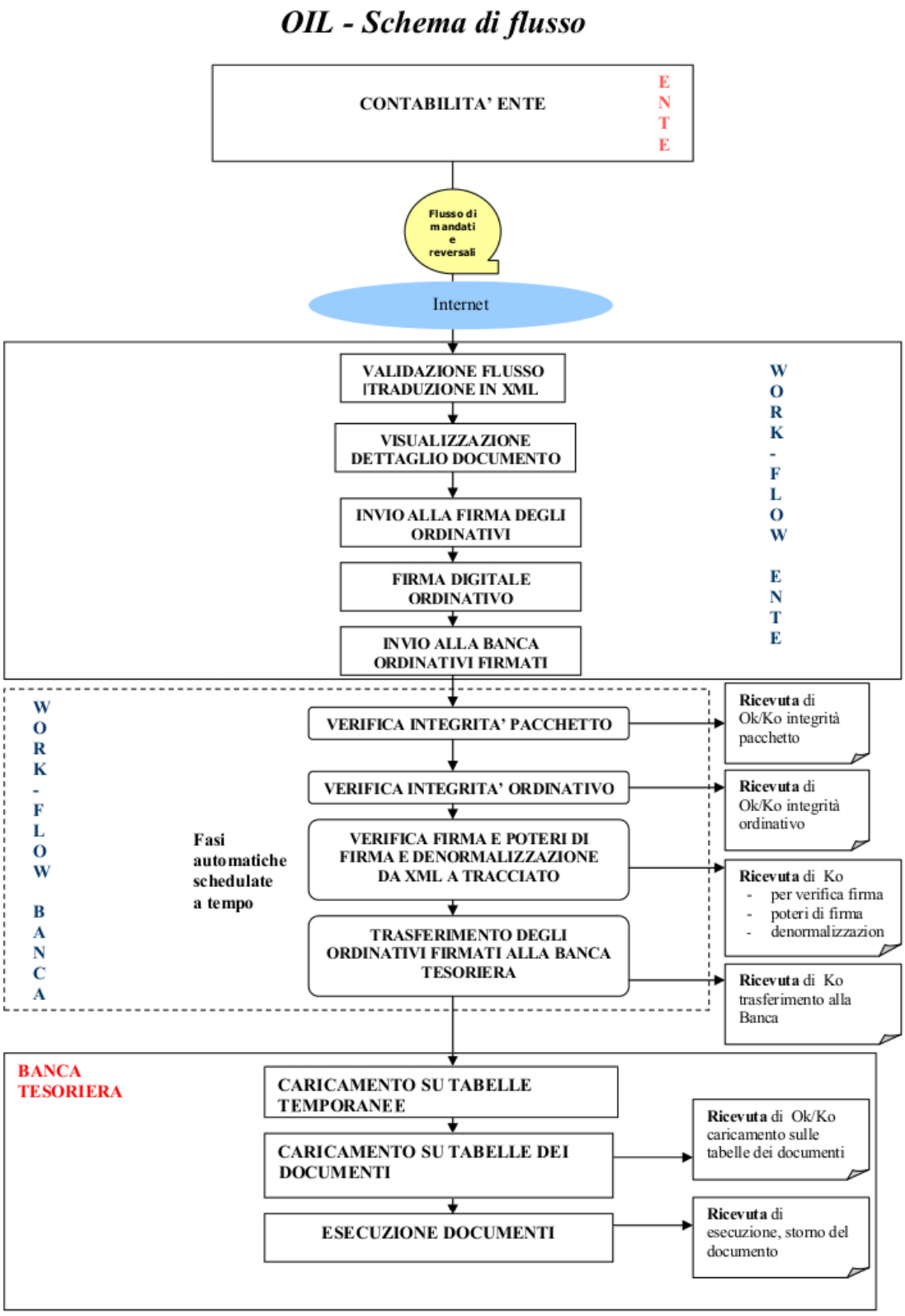


Figura 7 - Schema di flusso OIL

6 Strumenti utilizzati

6.1 Eclipse

Eclipse è un ambiente di sviluppo integrato multi-linguaggio e multipiattaforma.
Può essere utilizzato per la produzione di software di vario genere, si passa infatti da un completo IDE per il linguaggio Java (JDT, "Java Development Tools") a un ambiente di sviluppo per il linguaggio C++ (CDT, "C/C++ Development Tools") e a plug-in che permettono di gestire XML, Javascript, PHP e persino di progettare graficamente una GUI per un'applicazione JAVA (Eclipse VE, "Visual Editor"), rendendo di fatto Eclipse un ambiente RAD.

6.2 Tomcat v5.5

Tomcat è un software open source la cui funzionalità principale è quella di Web Application Server, ovvero la capacità di gestire e supportare le pagine JSP e le servlet. Nelle ultime versioni però sono state aggiunte funzionalità tra cui quella di supporto ai web service attraverso l'utilizzo del framework ApacheAxis.

Catalina è il contenitore di servlet Java di Tomcat. Catalina implementa le specifiche di Sun Microsystems per le servlets Java e le "JavaServer Pages (JSP, Pagine JavaServer)

6.3 Apache HTTP Server

Apache HTTP Server è un webserver per il protocollo HTTP, disegnato per poter girare come un processo standalone ed è un prodotto open source distribuito con il codice sorgente.
Caratteristica fondamentale è che è portabile su tutte le versioni di Unix, Windows, ecc.

6.4 Apache Axis

[8][9] Axis è un framework per realizzare sistemi di integrazione basati su SOAP. Il prerequisito fondamentale per l'installazione di Axis è la configurazione di un servlet container come Tomcat.
I web service sono stati creati con l'IDE Eclipse opportunamente aggiornato col plugin WTP (Web Tools Platform) che permettere l'utilizzo di Axis con Eclipse.

Le caratteristiche più importanti del framework sono:

- implementazione SOAP 1.1/1.2;
- supporto JWS (Java Web Services): permette un facile e immediato deploy dei WS
- supporto serializzazione/de-serializzazione;
- implementazione WSDL;
- utility WSDL2Java e Java2WSDL;
- Soap Monitor e TCP Monitor: due applicazioni scritte in Java che permettono di monitorare il traffico SOAP;
- possibilità di usare tre differenti metodi per l'invocazione dei WS: Dynamic Invocation Interface, Stub generato dal WSDL e Dynamic Proxy.

I componenti di Axis sono quindi: un engine per l'elaborazione dei messaggi SOAP, handler per la gestione dei messaggi, vari meccanismi di deploy, serializzazione/de-serializzazione, configurazione e di trasporto dei messaggi.

È sufficiente sviluppare la propria classe in Java, testarla, cambiare l'estensione da .java e .jws e il WS è pronto. Axis lo tratterà in maniera simile ad una JSP ossia lo compilerà e si occuperà dell'interfacciamento con SOAP mediante la conversione automatica SOAP-JAVA.

Tale metodo ha ovviamente il pregio della semplicità a scapito della flessibilità: è infatti impossibile decidere quali metodi esporre e specificare conversioni specifiche tra SOAP e JAVA. Un metodo più flessibile è l'uso dei file WSDD (Web Service Deployment Descriptor), introdotti da Axis, ma per ``il primo'' WS l'uso del JWS è più che adeguato. Un WS accessibile è quindi una classe Java che espone i propri metodi pubblici e non-static.

6.5 JAX-RPC

La tecnologia utilizzata per l'invocazione dei web service di questo progetto è JAX-RPC (Java API for XML-Based Remote Procedure Call) che permette l'invocazione dei web service, impostando in modo semplice e veloce i parametri di tale invocazione.
Questa tecnologia mette a disposizione inoltre il supporto per il mapping dei tipi da XML a Java e viceversa.

JAX-RPC, che necessita di SOAP, fornisce il supporto per il modello di scambio di messaggi SOAP. Inoltre SAAJ (SOAP Attachment API for Java), utilizzato da JAX-RPC, fornisce una API Java per la costruzione e la manipolazione di messaggi SOAP con allegati.

È possibile trasferire, all'interno di allegati al messaggio SOAP, vari tipi di documento, come ad esempio documenti XML o immagini.
JAX-RPC permette di utilizzare diverse tecniche per invocare i web service, a seconda delle esigenze e del livello di dinamicità che si vuole ottenere.
Axis implementa JAX-RPC ed è compilato nel JAR file axis.jar, che implementa appunto le API JAR-RPC dichiarate nei file JAR jaxrpc.jar e saaj.jar.
Tutti i file e le librerie di cui necessita Axis sono contenute in un package, axis.war, che può essere copiato in un servlet container.

In Axis agli handler viene passato un oggetto MessageContext definito da Axis che aggiunge funzionalità rispetto a quello delle jax-rpc. L'oggetto MessageContext contiene tutte le informazioni rilevanti sul servizio al quale l'operazione correntemente invocata e processata dall'handler appartiene. È infatti possibile oltre ad ottenere il messaggio SOAP di richiesta recuperare informazioni sul nome del servizio, il nome dell'operazione, una serie di parametri specificati dall'utente al momento della pubblicazione dell'handler e altro ancora.

6.6 IBM DB2

IBM DB2 è un Relational Database Management System (RDBMS) della IBM.
Attualmente, DB2 e Oracle si contendono il primo posto nel mercato dei DBMS.
DB2 risulta il primo database ibrido, con modello relazionale e XML.
Questo favorisce la gestione di applicazioni che interagiscono con documenti XML, permettendo ad esempio, l'interrogazione diretta del database tramite XQuery.

7 Funzionamento dell’applicazione finale

7.1 TesoWebSign 5.0

7.1.1 Presentazione

[10] Il **TesoWebSign 5.0** è l’applicativo sul quale è stato integrato lo scambio di flussi dati, in questo capitolo verranno descritte le sue funzioni principali e più nello specifico, la parte riguardante lo scambio di flussi tramite web service utilizzabile dall’utente finale.

Per cominciare, è possibile distinguere tre differenti tipologie di utenza:

- amministratore di Banca: è abilitato ad utilizzare tutte le funzioni disponibili nell’applicazione TesoWebSign per configurare/monitorare/supportare tutti gli enti/utenti di tutte le filiali della Banca;
- amministratore di Ente: può effettuare tutte le operazioni possibili abilitate dalla Banca, compresa la creazione di nuovi utenti dell’Ente con profilo/abilitazioni uguali o inferiori ai propri;
- utente Ente: può effettuare le operazioni a cui è stato abilitato e solo relativamente all’ente di appartenenza.

L’utente finale può accedere al servizio “TesoWebSign” tramite la funzione di autenticazione, ogni utente deve possedere una “user-id” e una “password” attraverso i quali si accede tramite la pagina di logon dell’applicazione.
La fase di logon prevede un preventivo controllo sull’abilitazione all’accesso in procedura ed alle funzioni associate al profilo dell’utente.
Successivamente viene verificato che l’utente abbia almeno un ente associato e quindi viene caricato il menù dell’utente.

7.1.2 Il programma

La pagina successiva al logon è riportata in figura 8 e funge come esempio per lo standard utilizzato in tutte le pagine del sito

Figura 8 - Pagina iniziale

Il menù è personalizzato per ogni profilo utente, quindi ad ogni definizione di utente, si deve stabilire a quale profilo appartiene e quindi quali sono le voci del menù cui egli può accedere.
Questa accortezza è fondamentale per non permettere ad un utente di generare altri utenti con maggiori diritti rispetto a se stesso.

Il prodotto TesoWebSign è composto da tre sezioni:

- Sezione informativa: applicazione web che permette all'utente di interrogare le posizioni contabili del servizio di tesoreria.
- Sezione firma: sezione per il controllo e la gestione dell'apposizione della firma.
- Sezione dispositiva: per l'upload (automatico o manuale) dei flussi dalla contabilità dell'Ente al sistema per il controllo e l'apposizione della firma sugli ordinativi e il successivo invio degli ordinativi firmati alla tesoreria aver apposto la firma (o le firme in caso di più firmatari).

Tralasciando le molte funzioni implementate, ci concentreremo sull'argomento della tesi: l'invio dei flussi dati dalla contabilità dell'Ente in modalità "manuale".
Esiste infatti anche una modalità "automatica" che prevede che la contabilità depositi il flusso in una cartella ed un batch schedulato a tempo lo prelevi automaticamente inserendo il flusso nella tabella appropriata al quale però non ho lavorato personalmente.

7.1.3 Lo scambio dei flussi dati

Alla funzione si accede selezionando la voce Importa Flussi [Figura 9] dal menu.

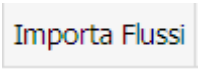


Figura 9 - Voce del menu

Si presenta la pagina Web di invio dei flussi [Figura 10] suddivisa in tre sezioni da utilizzare:

- tipologia flusso da inviare (menù relativo all'ente),
- nome del flusso da inviare,
- tasto funzione invia flusso.

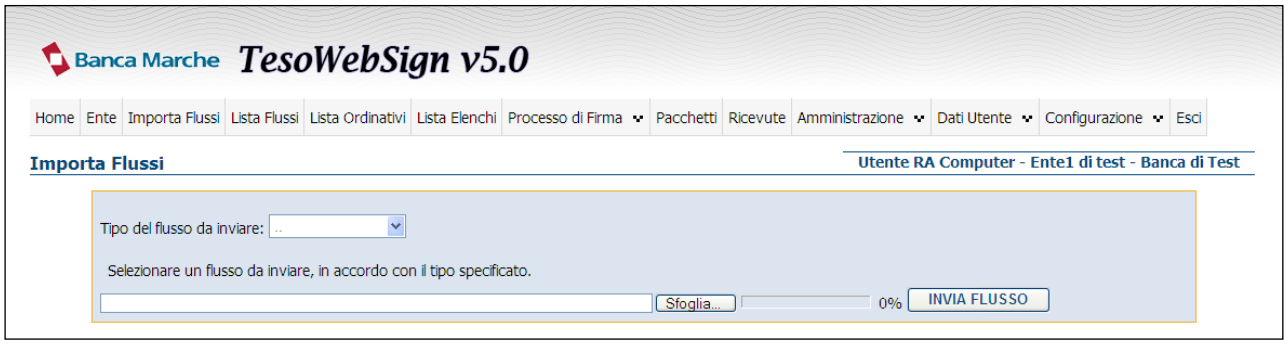


Figura 10 - Sezione Importa Flussi

Ogni ente avrà una serie di possibili tipologie di flusso da inviare, definite in fase di configurazione ente. Sulla base dei flussi abilitati per l'ente, si compone la combo-box [Figura 11]

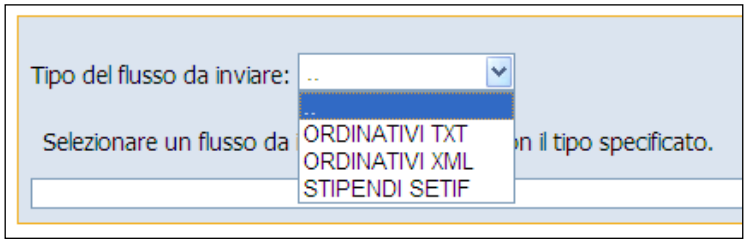


Figura 11 - Tipo di flusso

Una volta scelta la tipologia di flusso da inviare si sceglie il flusso dal sistema dell'utente [Figura 12]

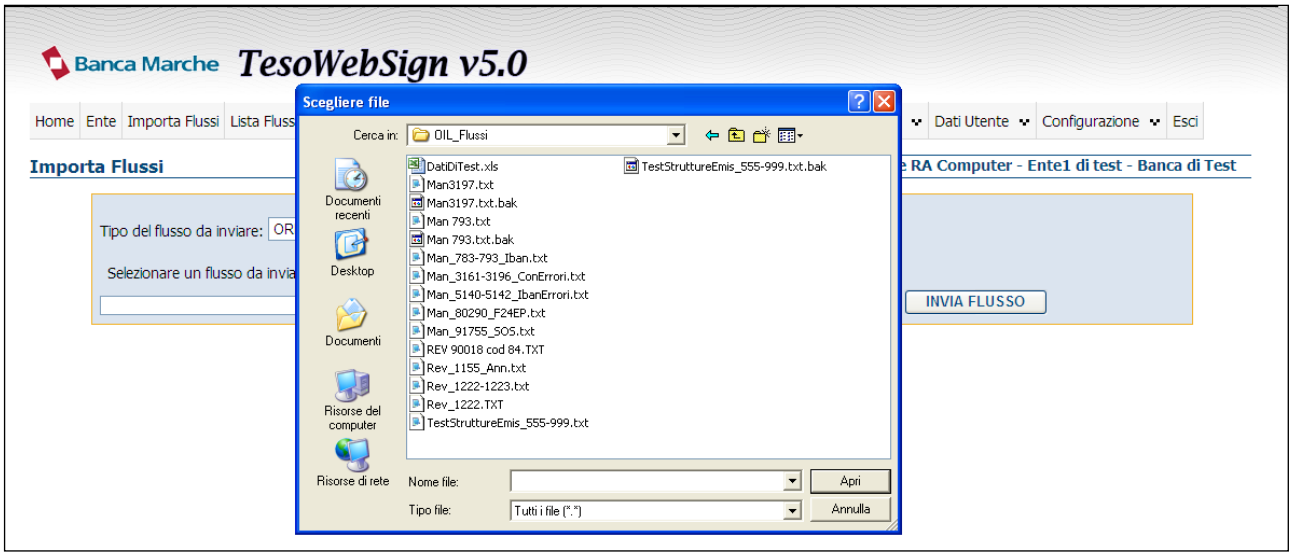


Figura 12 - Scelta file dal sistema dell'utente

Si otterrà una schermata che riassume le opzioni utente: il tipo di flusso selezionato nella combo Box ed il suo nome riportato nella riga sottostante [Figura 13].

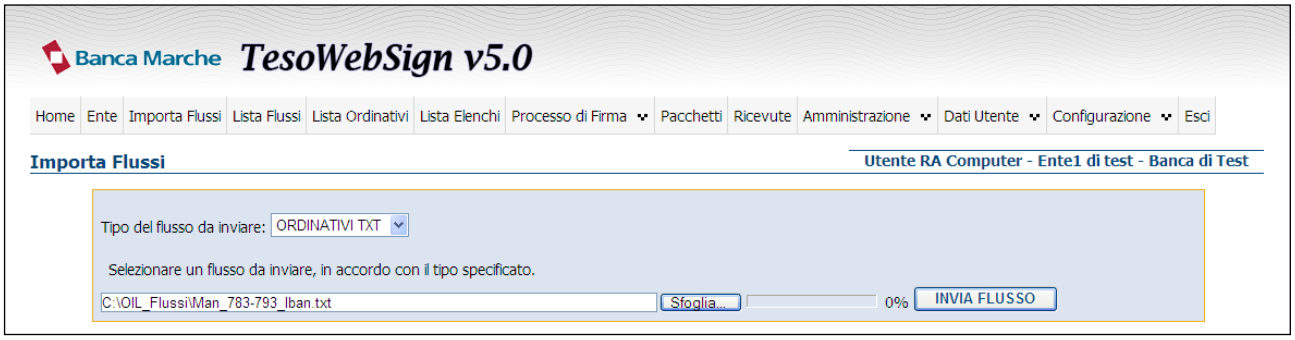


Figura 13 - Schermata di riassunto

Scegliendo il tasto funzione "INVIA FLUSSO" [Figura 14] si esegue così il batch interattivo che si occupa di:

- effettuare l'upload del file,
- controllare la congruenza dello stesso con la tipologia scelta,
- normalizzare i singoli documenti,
- validarli,
- presentare una videata riassuntiva delle operazioni effettuate.

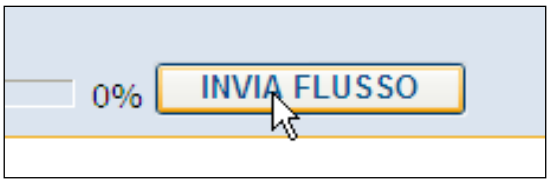


Figura 14 - Pulsante

Al termine del caricamento viene mostrato un messaggio [Figura 15]

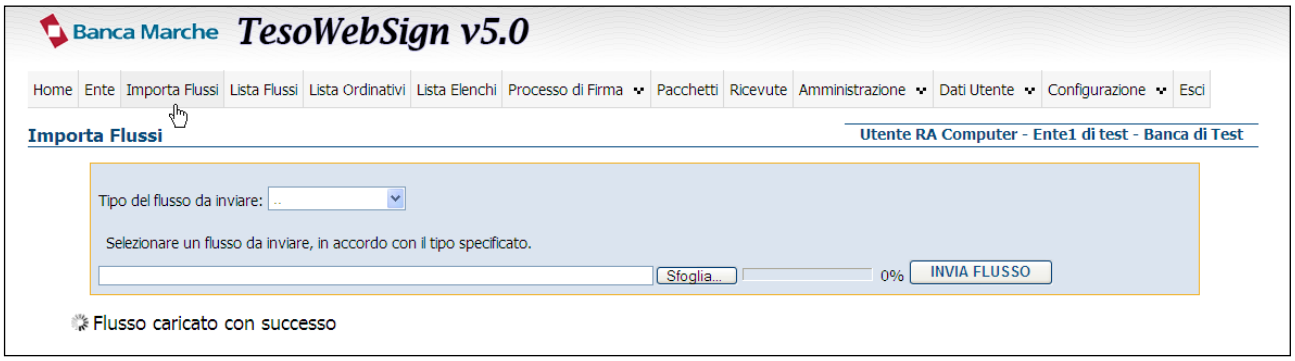


Figura 15 - Messaggio positivo

Se viene riconosciuto che il flusso allegato è della tipologia scelta, parte l'elaborazione dei singoli documenti [Figura 16].

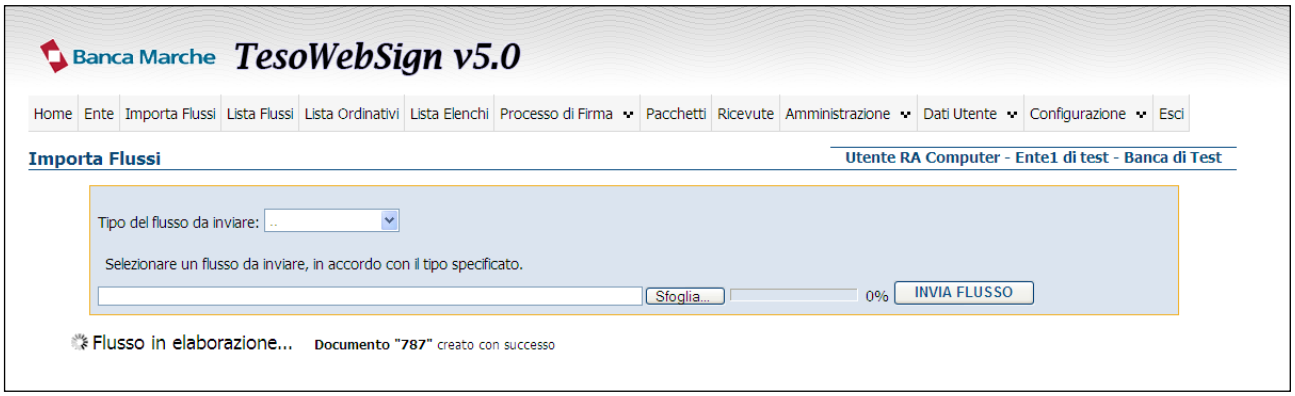


Figura 16 - Elaborazione

Viene poi eseguita una serie di controlli [Figura 17].

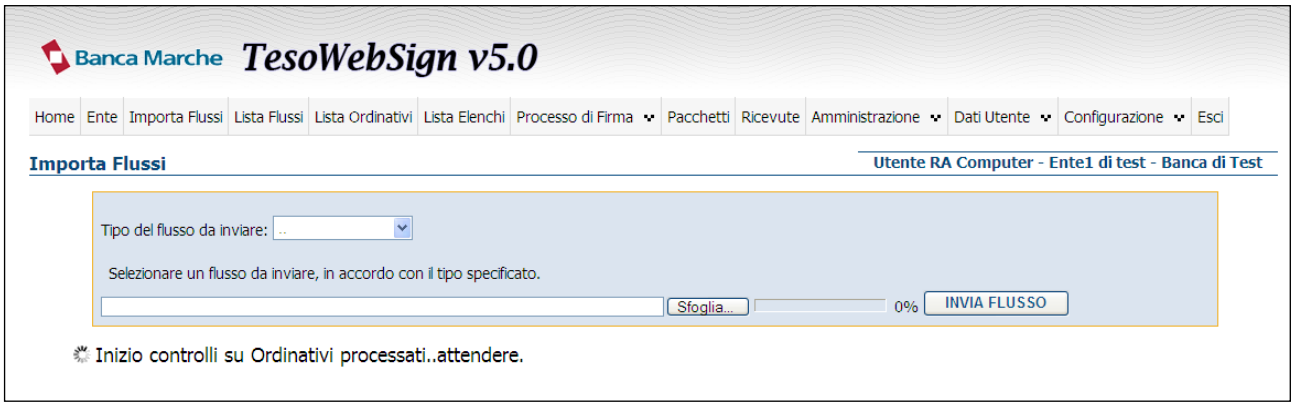


Figura 17 - Inizio controlli

Al termine dell'elaborazione si avrà una videata riassuntiva dell'operazione effettuata [Figura 18] indicante se:

- ci sono documenti non congrui con la tipologia del flusso inviato,
- il numero di documenti riconosciuti,
- il numero di documenti validi,
- il numero di documenti errati.

Figura 18 - Flusso con documenti riconosciuti

Nel caso in cui nel flusso ci fossero documenti non riconosciuti [Figura 19], si avrà:

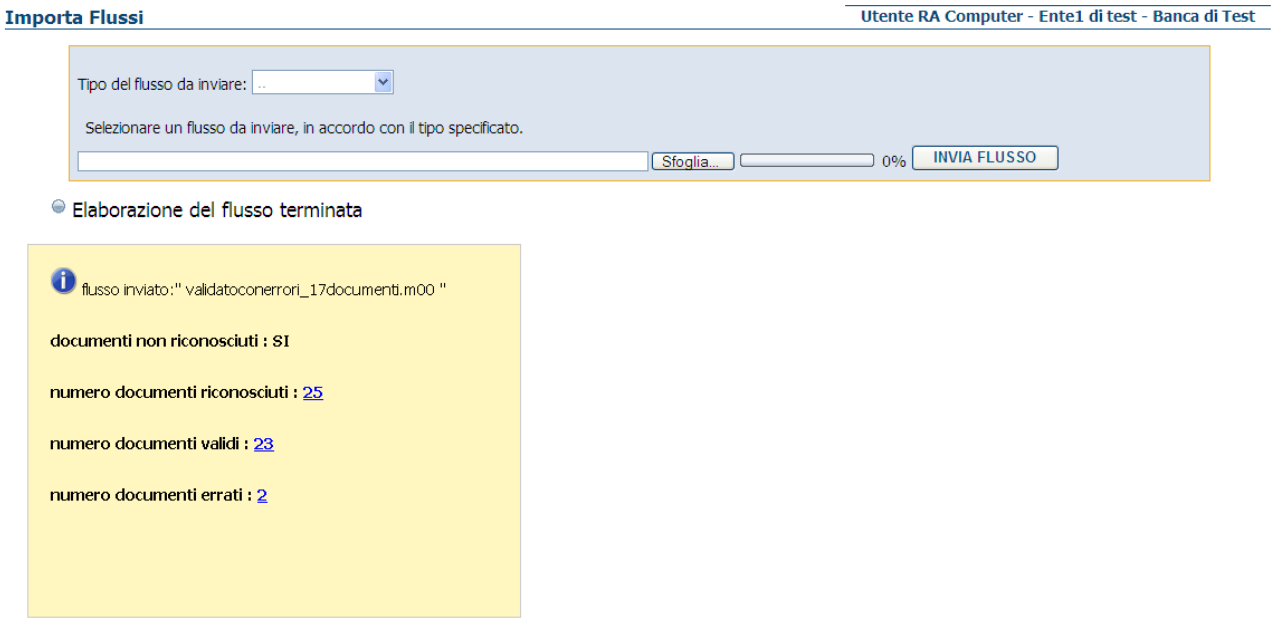


Figura 19 - Flusso con documenti non riconosciuti

7.1.4 Gestione degli errori

Naturalmente è stato implementato anche un metodo per la gestione degli errori:

- Tipo di flusso non selezionato [Figura 20]

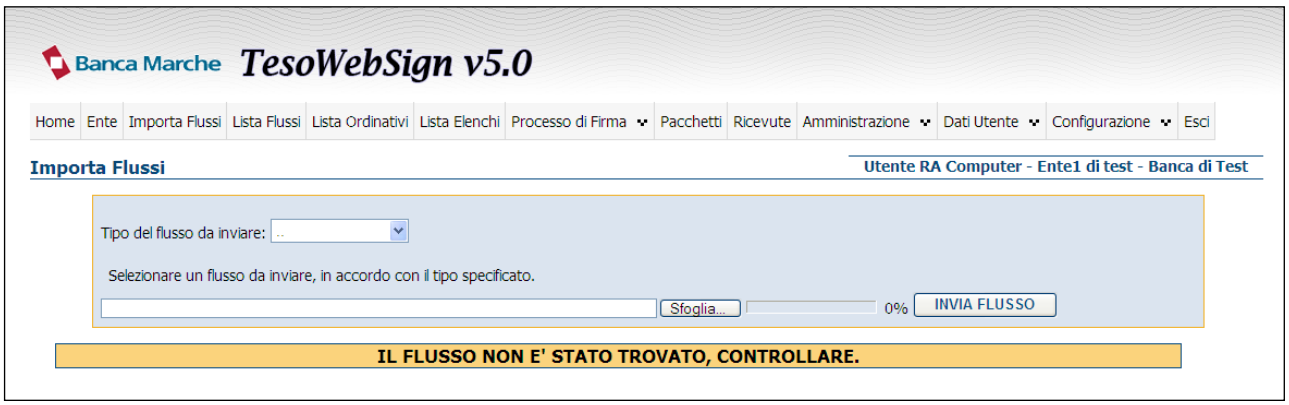


Figura 20 - Flusso non selezionato

❖ Nome del flusso non specificato [Figura 21]

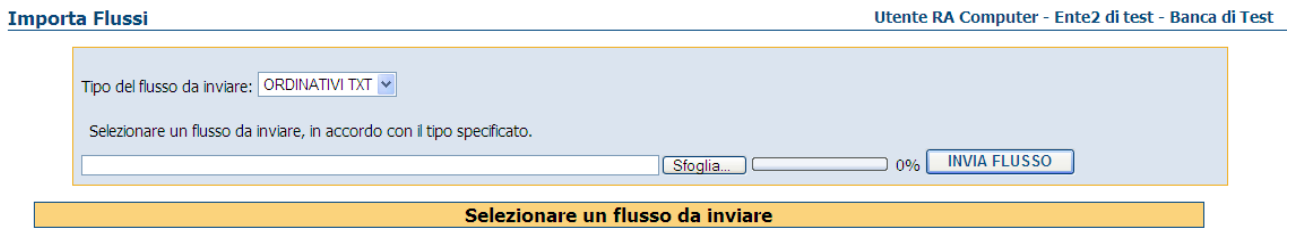


Figura 21 - Flusso non specificato

❖ Tipo di flusso non coerente con il tipo flusso scelto [Figura 22]

Figura 22 - Flusso non coerente

❖ Errore generico, non compreso tra quelli previsti in fase di analisi [Figura 23]

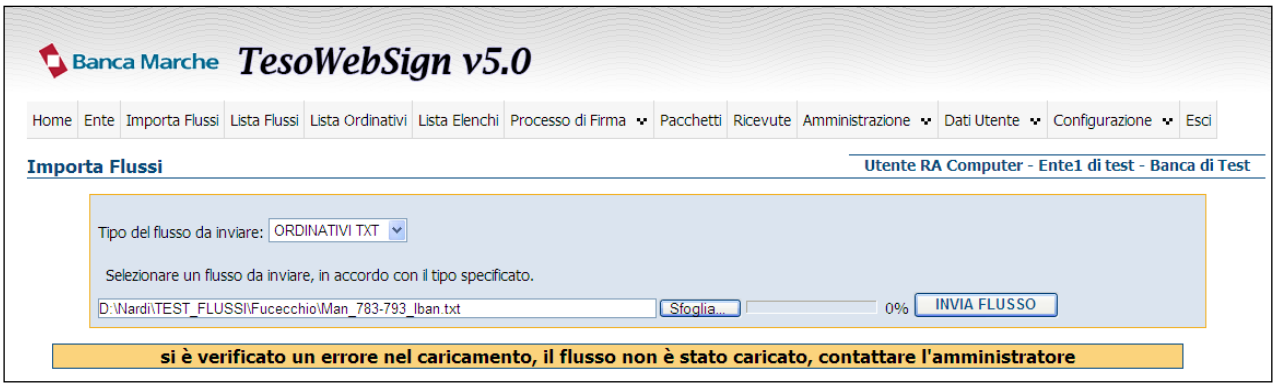


Figura 23 - Errore generico

È stato implementato il riconoscimento di un errore anche mentre il batch gira ed analizza i singoli ordinativi [Figura 24].

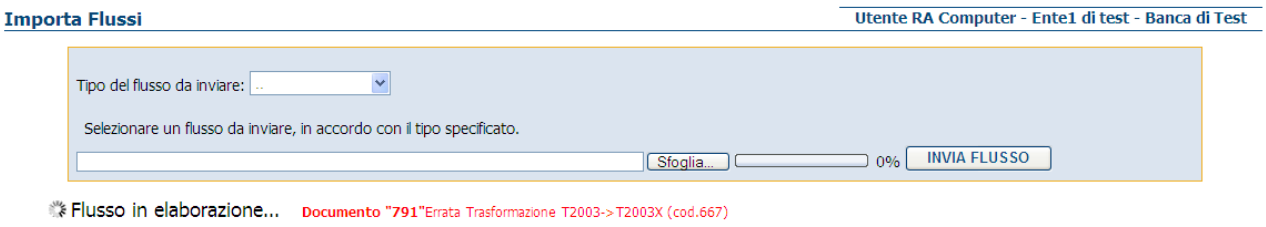


Figura 24 - Errore specifico

7.2 Meccaniche del programma

Il meccanismo realizzato prevede per l'utente un'unica fase che va dallo scarico del flusso sul server alla successiva fase di elaborazione fino ad ottenere gli ordinativi in formato XML o i messaggi di errore legati ad un loro eventuale scarto.

Sono previste due funzioni che agiscono in cascata. Con la prima funzione, l'utente sposta il flusso dalla propria macchina al server. Con la seconda funzione (richiamata dopo che la prima ha terminato correttamente il suo lavoro), il flusso presente sul server viene "normalizzato" e trasformato nella configurazione firmabile (XML). La fase di normalizzazione è gestita con un apposito thread lanciato dalla prima funzione. Poiché possono arrivare nel server più flussi contemporaneamente, occorre gestire un meccanismo che permetta di limitare il numero di thread in esecuzione in contemporanea. Ogni thread, terminato il lavoro sul proprio flusso, va a controllare se in archivio esistono flussi non associati a thread di normalizzazione e, in questo caso, provvede ad "acquisire il flusso" e a normalizzarlo.

Indipendentemente dal formato originale, ogni documento del flusso verrà trasformato in un documento in formato XML avente gli stessi tag previsti per i beneficiari (o versanti) di un mandato (o reversale).

Parlando di Ordinativo Informatico, i formati originali accettati saranno:

- tracciato TXT (standard RA Computer),
- tracciato TXT proprietario,
- tracciato XML (Standard RA Computer),
- tracciato XML proprietario.

è evidente che l'utilizzo di tracciati proprietari prevede un lavoro di analisi congiunta per la normalizzazione e la traduzione verso l'XML standard previsto dalle circolari tecniche dell'ABI sull'argomento approvate dal DigitPA (ex CNIPA).

L'applicazione prevede anche la possibilità di importare anche altre tipologie di documenti/flussi (p.es. Elenchi di disposizioni). Il processo è del tutto analogo, ma in questa sede ne evitiamo la trattazione.

Il flusso originale degli ordinativi viene quindi archiviato nel database tramite un apposita procedura, ad ogni flusso inviato dall'ente corrisponde una riga nella tabella Flussi [Figura 25] se il processo di upload va a buon fine.

Flussi : Tabella

Nome campo	Tipo dati	Descrizione
idFlusso	Contatore	
idEnte	Numerico	
idFlussoEnte	Testo	Riferimento univoco dell'Ente - Codice di riconciliazione
utente	Testo	
timeStampFlusso	Data/ora	
idTipoFlusso	Numerico	
docNonRoconosciuti	Testo	(SI/NO/ALL)
numDocRiconosciuti	Numerico	
numDocValidi	Numerico	
numDocErrati	Numerico	
flussoOriginario	Memo	Flusso originale dell'ente
flussoElaborato	Memo	Contiene il T2003X

Figura 25 - Tabella Flussi

Il campo idTipoFlusso punta alla tabella tipiFlusso [Figura 26].

tipiFlusso : Tabella

Nome campo	Tipo dati
idTipoFlusso	Numerico
dscrTipoFlusso	Testo
descrBreve	Testo

Figura 26 - Tabella tipiFlusso

Nella tabella tipiFlusso si trova la descrizione e la categoria di ogni flusso [Figura 27].

tipiFlusso : Tabella

idTipoFlusso	dscrTipoFlusso	descrBreve	categoriaProgrammaElaborazione
1	Mandati TXT	MAN	macro
2	Reversali TXT	REV	macro
3	Ordinativi XML	XML	XSLT
4	Flussi stipendi TXT	STI	subTXT
5	Flussi stipendi SETIF	STF	subTXT
6	Flusso beneficiari TXT	BEN	subTXT
7	Flusso contributi TXT	CON	subTXT

Figura 27 - Dati nella tabella tipiFlusso

Il flusso importato e memorizzato nella tabella Flussi viene immediatamente elaborato da un thread di servizio.
Il thread che elabora il flusso deve avere un solo dato di input: il solo idFlusso del flusso da prendere in carico. Sarà a suo totale carico reperire una connessione al database contenente la tabella Flussi e recuperare i dati del flusso necessari per la successiva fase di normalizzazione.

La prima attività del thread è quella di individuare la tipologia del flusso da elaborare per richiamare l'opportuno programma di normalizzazione.
Le macro categorie dei flussi elaborabili, al momento, sono tre: flussi di mandati e reversali in formato testo, flussi di mandati e reversali in formato XML, flussi di elenchi.
Quindi, il thread di normalizzazione, per prima cosa recupera i dati relativi al flusso da normalizzare, quindi lancia l'opportuno programma di normalizzazione.

In definitiva sono state sviluppate tante "linee di normalizzazione" quante sono le varie tipologie di flussi che possono arrivare: una linea utilizza una macro di normalizzazione con successiva traduzione in formato XML, una linea tratta i flussi di tipo SETIF e li trasforma in XML, una linea tratta i flussi XML con XSLT,…

Le varie linee lavorano indipendentemente le une dalle altre ed è possibile aggiungerne delle altre senza intaccare il funzionamento di quelle già implementate. È il thread che, in funzione dei dati del flusso da trattare, sceglie l'opportuna linea di normalizzazione.

Tutti i flussi arrivati e che appartengono alla categoria "macro" [Figura 28] subiscono due processi: normalizzazione e traduzione [Figura 29], seguiti dalla fase di salvataggio [Figura 30].

tipiFlusso : Tabella

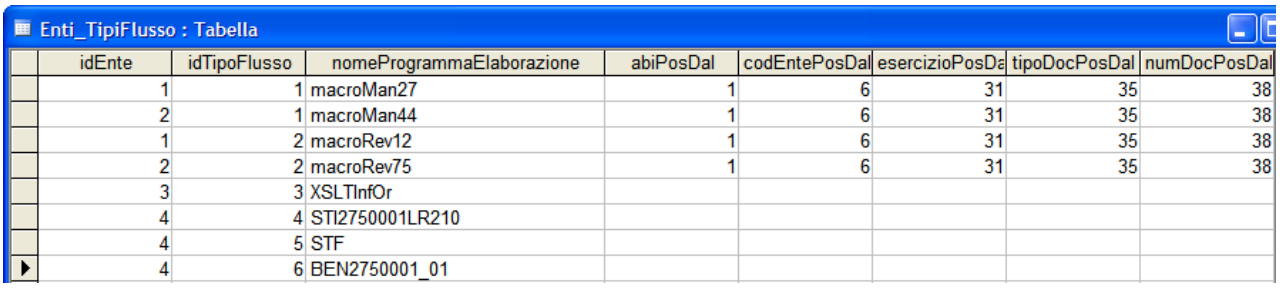
idTipoFlusso	dscrTipoFlusso	descrBreve	categoriaProgrammaElaborazione
1	Mandati TXT	MAN	macro
2	Reversali TXT	REV	macro
3	Ordinativi XML	XML	XSLT
4	Flussi stipendi TXT	STI	subTXT
5	Flussi stipendi SETIF	STF	subTXT
6	Flusso beneficiari TXT	BEN	subTXT
7	Flusso contributi TXT	CON	subTXT

Figura 28 - Dati di categoria "macro"

Enti : Tabella

idEnte	abiBanca	codEnteBanca	descrEnte	flags
1	06160	1150200	Comune di Fucecchio	000011100011
2	06160	1070050	Comune di Campi Bisenzio	000011100011
3	06160	8590001	Comune di Firenze	000011100011
4	06055	2750001	Regione Marche	000011100011

Figura 29 - Dati nella tabella Enti



Enti_TipiFlusso : Tabella

idEnte	idTipoFlusso	nomeProgrammaElaborazione	abiPosDal	codEntePosDal	esercizioPosDa	tipoDocPosDal	numDocPosDal
1	1	macroMan27	1	6	31	35	38
2	1	macroMan44	1	6	31	35	38
1	2	macroRev12	1	6	31	35	38
2	2	macroRev75	1	6	31	35	38
3	3	XSLTInfOr					
4	4	STI2750001LR210					
4	5	STF					
4	6	BEN2750001_01					

Figura 30 - Dati nella tabella Enti_TipiFlusso

Per esempio, dalle tabelle precedenti si evince che il comune di Fucecchio può inviare flussi di mandati e reversali in formato testo che saranno elaborati dalle macro di normalizzazione siglate con “Man27” per i flussi contenenti mandati e “Rev12” per i flussi contenenti reversali.

Il flusso, inizialmente, subisce una fase di pre-elaborazione: una macro di normalizzazione trasformerà il flusso in un formato standard RA interno, noto come T2003X.

Per chiarire meglio il concetto, una macro è una classe java che legge in input un flusso non standard e lo trasforma in uno elaborabile direttamente dal traduttore. Può essere vista come un “solutore” che trasforma direttamente il flusso TXT nel flusso XML standard ABI

7.3 Fase di traduzione

Dopo la normalizzazione in formato T2003X, questa tipologia di flussi subisce una fase di “traduzione” che, documento per documento, trasforma i record del T2003X in formato XML.

Si leggono i record dal T2003X di un documento, lo si trasforma (se possibile) nel formato XML, si salva il documento nella tabella Documenti, si aggiorna il “Documento elaborato” sul form dell’utente, si esegue lo stesso ciclo per i documenti successivi del file T2003X, alla fine, si aggiorna l’ultima check box del report (con i dati risultanti) e si invia il messaggio di “FINE ELABORAZIONE”.

7.4 Flusso di esempio

I seguenti dati sono stati opportunamente modificati in quanto riconducibili ad un ente reale. Il file è in formato SOR, ovvero un TXT inviato dall'ente e ordinato (SORt).

7.4.1 Flusso di base

20130212.M01.T2000.sor

00001003030720130000018000004400000000120220130000000046990500000000000000000030307010000000N000000000000000
0000000 0000000INSM
00001003030720130000018000004400000007000 UE0001 00000000000000000000000
00000000 a00xxxx
INSM
00001003030720130000018000004400000011000000000000000000030307010000000004699050
53150220130000000002500000000007000000000001000000N05175700482 00000000acque spa servizi idrici via garigliano
1 50053EMPOLI CA07606340000000359391 29V000consumi acque periodo 10/11/2012 - 10/12/2012 - Fattura
Acquisto - 01-2013-0001234 - 03/01/ 000321000INSM
00001003030720756400001800000440000001 4C2013 - 2013
INSM
00001003030720130000018000004400000014IIT89V0760114000000035935519
INSM
00001003030720130000018000004400000018000000100000000000000000002315 000000004699050
INSM

7.4.2 Flusso tradotto

20130212.M01.TE2003X.sor

00001852945700000000000000000002013MAN00000000440000000INS000000000181572013000001812022013QUARANTASEIMIL
ANOVECENTONOVANTA/50
00000000461234100000000000000000000000000000003030701 0000000N a00xxxx
EMISSIONE_MANDATO 00000 0 00000000 0000000000000UE0001 0
00000000000000000000 0000000
00001003030700000000000000000002013MAN00000000440000001INS0010000001 acque spa servizi idrici via
garigliano 1 50053EMPOLI FI 00000000 0000000000
00000 00000 00000000
05175700482 05175700482 05175700482
00001094720450000000000000000002013MAN00000000476000001INS0020000001
000000004699050 0010000000N000consumi acque perido 10/11/2012 - 10/12/2012 - Fattura Acquisto - 01-2013-0000920 -
03/01/025000000000 0700000000072125022013000000000000000000000000000
LIBERA FRUTTIFERA ESENTE BOLLO BONIFICO BANCARIO E POSTALE ITALIANOA CARICO
BENEFICIARIO 0000000 000000000000
00003453030700000000000000000002013MAN00000000440000001INS0050000001C2013 - 2013
00003453030700000000000000000002013MAN00000000440000001INS0050000001IIT00V0712108000000033435674
00001003030700000000000000000002013MAN00000000440000001INS0080000001000000000000000002005
000000004779050

7.4.3 Flusso XML

20130212.M01.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<?xml-stylesheet type="text/xsl" href=""?>
- <elenco_ordinativi xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation=""
xsi:schemaLocation="http://www.w3.org/2000/09/xmldsig#
http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd">
<id_elenco_ordinativi>xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx</id_elenco_ordinativi>
<data_ora_generazione>2013-02-12T12:25:03</data_ora_generazione>
- <flusso_ordinativi xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<codice_ABI_BT>DUMMY</codice_ABI_BT>
<identificativo_flusso>00000000000000000</identificativo_flusso>
<data_ora_creazione_flusso>DUMMY</data_ora_creazione_flusso>
<codice_ente>DUMMY</codice_ente>
<descrizione_ente>DUMMY</descrizione_ente>
<codice_Istituto_BT>00001</codice_Istituto_BT>
<codice_ente_BT>00xxxxxx</codice_ente_BT>
<esercizio>2013</esercizio>
<numero_distinta>00000xx</numero_distinta>
<data_distinta>2013-02-12</data_distinta>
- <mandato>
<tipo_operazione>INSERIMENTO</tipo_operazione>
<numero_mandato>00000000xx</numero_mandato>
<data_mandato>2013-02-12</data_mandato>
<importo_mandato>46990.50</importo_mandato>
<conto_evidenza>xxxxxxx</conto_evidenza>
<responsabile_provvedimento>a00xxxx</responsabile_provvedimento>
- <informazioni_beneficiario>
<progressivo_beneficiario>0000001</progressivo_beneficiario>
<importo_beneficiario>46990.50</importo_beneficiario>
<tipo_pagamento>BONIFICO BANCARIO E POSTALE</tipo_pagamento>
<data_scadenza_pagamento>2013-02-15</data_scadenza_pagamento>
<destinazione>LIBERA</destinazione>
<tipo_contabilita_ente_ricevente>FRUTTIFERA</tipo_contabilita_ente_ricevente>
- <classificazione>
<codice_cgu>2315</codice_cgu>
<importo>46990.50</importo>
</classificazione>
- <bollo>
<assoggettamento_bollo>ESENTE BOLLO</assoggettamento_bollo>
</bollo>
- <spese>
```

<soggetto_destinatario_delle_spese>**A CARICO BENEFICIARIO**</soggetto_destinatario_delle_spese>
</spese>
- <beneficiario>
<anagrafica_beneficiario>**ente di prova**</anagrafica_beneficiario>
<indirizzo_beneficiario>**via prova 1**</indirizzo_beneficiario>
<cap_beneficiario>**00010**</cap_beneficiario>
<localita_beneficiario>**ROMA RM**</localita_beneficiario>
<partita_iva_beneficiario>**xxxxxxxxxx**</partita_iva_beneficiario>
</beneficiario>
- <piazzatura>
<tipologia_pagamento_estero>**IBAN**</tipologia_pagamento_estero>
<iban>**IT89Vxxxxxxxxxxxxxxxxxxxxxxxxx**</iban>
</piazzatura>
<causale>**consumi acque periodo 10/11/2012 - 10/12/2012 - Fattura Acquisto - 01-2013-0000021 - 03/01/2013 - 2013**</causale>
- <informazioni_aggiuntive>
<lingua>**ITALIANO**</lingua>
</informazioni_aggiuntive>
- <dati_a_disposizione_ente_beneficiario>
<importo_netto_dovuto>**46990.50**</importo_netto_dovuto>
<importo_lordo_dovuto>**46990.50**</importo_lordo_dovuto>
<totale_ritenute_beneficiario>**0.00**</totale_ritenute_beneficiario>
</dati_a_disposizione_ente_beneficiario>
</informazioni_beneficiario>
- <dati_a_disposizione_ente_mandato>
<riga />
- <dati_ente>
<dipartimento>**UE0001**</dipartimento>
<denominazione_struttura>**DUMMY**</denominazione_struttura>
- <impegno_accertamento>
<numero>**0**</numero>
</impegno_accertamento>
<importo_prelevabile_in_contanti>**0.00**</importo_prelevabile_in_contanti>
- <decreto>
<numero>**00000**</numero>
</decreto>
- <provvedimento>
<numero>**0**</numero>
</provvedimento>
<caricato_da>**a00xxxx**</caricato_da>
</dati_ente>
</dati_a_disposizione_ente_mandato>
</mandato>
</flusso_ordinativi>
</elenco_ordinativi>

8 Conclusioni

Aver collaborato allo sviluppo di web service e al perfezionamento di un programma come TesoWebSign in un'azienda seria e innovativa come la Ra Computer è stata un'esperienza costruttiva.
Ho potuto sperimentare un vero ambiente di lavoro interagendo con una realtà lavorativa sconosciuta, i rapporti all'interno dell'azienda sono stati agevoli, c'è stata un'integrazione immediata nella realtà del personale, cosa che ha influenzato positivamente la convivenza in azienda e di conseguenza il lavoro svolto. L'apprendimento di tecnologie all'inizio sconosciute attraverso strumenti quali SDK, siti web autorevoli e manuali si è rilevato proficuo.

Le conoscenze maturate nel corso di laurea, come la programmazione Object-Oriented, le nozioni teoriche ricevute dai corsi di Ingegneria del Software e Reti di Elaboratori sono state essenziali per lo svolgimento dello stage e conseguentemente alla realizzazione di questa tesi.

L'applicazione TesoWebSign è tutt'oggi in fase di miglioramento, le sue funzionalità vengono ampliate e corrette a seconda delle richieste dei clienti dell'azienda, soggetti pubblici e privati tra cui istituti di credito come Banca Marche, FriulAdria, CariParma, Banca Carige, …

9 Bibliografia

[1] Testo unico delle leggi sull'ordinamento degli enti locali – D.Lgs. 18 agosto 2000, n. 267
http://www.camera.it/parlam/leggi/deleghe/00267dl.htm

[2] Quaderno CNIPA n° 29, 2006 – DigitPA
[3] Linee Guida OIL, 2011 – DigitPA
http://www.digitpa.gov.it

[4] Giulio Turetta – Guida Web Service, 2006
http://www.html.it/guide/guida-web-service/

[5] Gianni Nitti – Costruire applicazioni basate su Web Service, 2001
http://www.aspitalia.com/articoli/webservice.aspx

[6] David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris, David Orchard – Web Services Architecture, 2004
http://www.w3.org/TR/ws-arch/

[7] Davide Quaglia – Comunicazioni in rete tramite Java
http://www.di.univr.it/documenti/OccorrenzaIns/matdid/matdid301370.pdf

[8] Alessandro Gabrielli – Analisi, Studio e Sperimentazione di Tecnologie Web Service, 2009
http://www.dis.uniroma1.it/~degiacom/didattica/semingsoft/seminari-studenti/09-01-09 - SIS - Alessandro Gabrielli/Documentazione.doc

[9] Prof. Alfredo De Santis – Web Services, 2002
http://www.di.unisa.it/~ads/corso-security/www/CORSO-0203/piattaformesviluppowireless/webservices.htm

[10] TesoWebSign5.0 – Manuale utente