Analisi tecnica

Architettura del nuovo contesto applicativo Sacer WS

Sommario

Redatto da	Engineering / Francesco Fioravanti
Verificato da	
Approvato da	
Data	29/12/2017
Pagine	8
Nome File	ArchitetturaSacerWS.odt

Storia del documento

Versione	Data	Autore	Cambiamenti apportati
1.0			Prima versione

Riferimenti

Nella tabella seguente sono riportati i documenti e/o le risorse di riferimento utili alla comprensione dei requisiti (es. architetture tecnologiche, specifiche, norme, etc..

Numero	Riferimento	Reperibile in

1 Obiettivo del documento

Il presente documento presenta una proposta architetturale finalizzata a separare il contesto Sacer in due applicativi distinti, uno dedicato alle funzioni online di amministrazione e monitoraggio, e l'altro che esponga tutti i principali web service REST, in particolare quelli "ad alto traffico" quali quelli di versamento.

Scopo di questa evoluzione è quello di mantenere la continuità di servizio da parte del componente più critico del sistema (i ws RESTdi versamento) anche durante gli aggiornamenti dei moduli online, mediamente più frequenti.

2 Descrizione del sistema attuale

L'attuale sistema è un applicativo monolitico per quanto riguarda le interfacce esposte (online e web service) ma dal punto di vista architetturale è costituito da un insieme di moduli, alcuni dei quali dotati di cicli di sviluppo e versione proprii e riutilizzati all'interno di altri contesti applicativi.

2.1 descrizione dei moduli del sistema

I principali moduli software dell'attuale sistema Sacer sono:

Sacer-web	Definisce le interfacce dell'applicativo (pagine web e endpoint dei web service) , la logica di navigazione tra le pagine e parte della logica di business per il frontend online. Questo modulo ha la stessa versione dell'applicativo Sacer e viene sviluppato e rilasciato come parte di esso
Sacer-ejb	Definisce la logica di business dell'applicativo, sia per l'online che per i web service. Questo modulo ha la stessa versione dell'applicativo Sacer e viene sviluppato e rilasciato come parte di esso
Sacer-slg	Definisce un insieme di classi generate in automatico a supporto dell'interfaccia utente del sistema gestita tramite il framework "Spagolite" Questo modulo ha la stessa versione dell'applicativo Sacer e viene sviluppato e

	rilasciato come parte di esso
Sacer-jpa	Definisce la "persistence unit" dell'applicativo cioè l'insieme delle classi Java che rappresenta il database sottostante secondo l'approccio ORM (Object-Relational Mapping). Queste classi vengono prodotte in modo semi-automatico: un software genera le classi "grezze" a partire dalle varie tabelle del database che poi devono venire completate a mano. Questo modulo ha la stessa versione dell'applicativo Sacer e viene sviluppato e rilasciato come parte di esso
SpagoFat	Ex "spagolite", implementa la maggior parte del codice condiviso di definizione dell'interfaccia utente, oltre al codice di autenticazione e di gestione del log di audit. Questo modulo ha un ciclo di sviluppo separato da quello di Sacer, ha una versione propria ed è usato anche da altri applicativi.
Sacer-xml	Contiene le classi Java che implementano le strutture dati definite dai vari XSD usati dall'applicativo, in particolare sono usate per serializzare i metadati gestiti dai web service. Queste classi vengono usate con libreria Castor oppure da JAXB. Questo modulo ha un ciclo di sviluppo separato da quello di Sacer, ha una versione propria ed è usato anche da altri applicativi.
Idp-rdbms-login	Implementa il modulo di login usato dall'IDP Shibboleth. Viene usato anche da Sacer, come libreria, per la funzione di disattivazione automatica degli utenti dopo un numero prefissato di errori di login. Questo modulo ha un ciclo di sviluppo separato da quello di Sacer, ha una versione propria ed è usato anche da altri applicativi.
cryptolibrary	La libreria crittografica di Sacer, usata per la verifica delle firme digitali; anche questo modulo ha un ciclo di sviluppo separato da quello di Sacer ed ha una versione propria.

3 Requisiti

Come accennato all'inizio, il requisito principale di evoluzione dell'architettura è quello di mantenere l'operatività dei web service e degli utenti versatori del sistema anche durante il deploy della parte online di Sacer e di limitare l'interruzione del servizio solo ai rilasci del modulo deputato a gestire i web service, che si prevedono meno frequenti.

Per evitare di dover fornire a tutti gli utenti versatori un nuovo URL verso cui indirizzare i vari software di versamento già sviluppati, è essenziale che il nuovo contesto applicativo possa venire esposto mantenendo inalterati gli URL dei vari web service (sullo stile di https://parer.regione.emilia-romagna.it/sacer/<nomews>).

Un requisito emerso durante le prime fasi di progettazione è quello di avere un versionamento ed un ciclo di sviluppo autonomo per il modulo "sacer-jpa" (che definisce le classi-entity JPA che rappresentano le tabelle del database). Questo consentirebbe un maggiore accoppiamento tra questo modulo e il rilascio degli aggiornamenti allo schema del database durante le fasi di sviluppo e deploy; sarebbe cioè possibile individuare con un unico numero di versione sia il modulo che gli script di allineamento del DB. Sacer in questo caso verrebbe sempre distribuito con l'ultima versione consolidata di questo modulo, coincidente con l'ultima versione stabile del DB – tipicamente quella presente nell'ambiente di Produzione.

Dal momento che questa revisione architetturale avviene contestualmente allo sviluppo del nuovo web service di versamento fascicoli si richiede che questa separazione avvenga "per gradi", sviluppando il nuovo servizio direttamente nel nuovo contesto e spostando in una fase successiva al rilascio, tutti gli altri.

4 Criticità esposte

La richiesta di effettuare la migrazione dei web service tra i due contesti applicativi in fasi successive, porta necessariamente ad una situazione in cui alcuni servizi saranno implementati da Sacer ed altri dal nuovo contesto applicativo. Questa situazione è destinata a rimanere anche al termine del processo di migrazione, dal momento che alcuni servizi – tra cui quello di monitoraggio per Zabbix – devono risiedere nel contesto di Sacer.

Poiché la maggior parte dei servizi REST di Sacer è basata su un insieme comune di classi di trasferimento dati (DTO) e di funzioni utilità, questa evoluzione potrebbe portare ad una duplicazione di questo insieme di classi che dovranno essere presenti in entrambi i contesti, con ovvie difficoltà di manutenzione.

5 Nuova architettura

La nuova architettura del sistema prevede un nuovo contesto applicativo (sacerws) composto da un modulo di interfaccia utente minimale destinato ad implementare gli endpoint dei servizi, servito da una logica di business implementata tramite un modulo EJB in modo analogo a quanto avviene per Sacer.

Il nuovo modulo sacerws-ejb, destinato a contenere le classi comuni a tutti i web service oltre all'implementazione del servizio di versamento fascicolo – a cui seguiranno gli altri – viene ottenuto separando queste classi dal modulo sacer-ejb di Sacer.

Per ridurre al minimo i problemi di duplicazione, tutte le classi java che vengono inserite nel modulo sacerws-ejb vengono rimosse dal modulo EJB di Sacer; il progetto Sacer acquisisce quindi sacerws-ejb come dipendenza esterna, in modo analogo a quanto avviene col modulo sacer-xml.

Il modulo sacer-jpa viene separato in un progetto autonomo, con un versionamento autonomo che diviene dipendenza sia di Sacer che del nuovo progetto Sacerws.

Quest'ultima evoluzione apre la possibilità di avere diverse versioni di sacer-jpa distribuite con sacer e sacerws, relative a diverse generazioni dello schema del database. Questo non dovrebbe costituire un problema purché le eventuali modifiche alle tabelle, recepite da Sacer, non impattino il funzionamento dei web service, che possono così continuare ad operare con la vecchia versione del modello ORM.

5.1 Descrizione dei moduli del sistema

I principali moduli software del nuovo SacerWs sono

Sacerws-web	Definisce gli endpoint dei ws Questo modulo ha la stessa versione dell'applicativo SacerWS e viene sviluppato e rilasciato come parte di esso
Sacerws-ejb	Definisce la logica di business per i web service ed implementa le classi comuni a tutti i ws. Questo modulo ha la stessa versione dell'applicativo SacerWS e viene sviluppato

	e rilasciato come parte di esso. Questo stesso modulo è incluso come dipendenza in Sacer.
Sacer-jpa	Definisce la “persistence unit” dell’applicativo cioè l’insieme delle classi Java che rappresenta il database sottostante secondo l’approccio ORM (Object-Relational Mapping). Queste classi vengono prodotte in modo semi-automatico: un software genera le classi “grezze” a partire dalle varie tabelle del database che poi devono venire completate a mano. Questo modulo ha un ciclo di sviluppo separato da quello di SacerWS, ha una versione propria ed è usato anche da Sacer.
SpagoFat (alcuni moduli)	Ex “spagolite”, implementa la maggior parte del codice condiviso di definizione dell’interfaccia utente, oltre al codice di autenticazione e di gestione del log di audit. Questo modulo ha un ciclo di sviluppo separato da quello di SacerWS, ha una versione propria ed è usato anche da altri applicativi.
Sacer-xml	Contiene le classi Java che implementano le strutture dati definite dai vari XSD usati dall’applicativo, in particolare sono usate per serializzare i metadati gestiti dai web service. Queste classi vengono usate con libreria Castor oppure da JAXB. Questo modulo ha un ciclo di sviluppo separato da quello di SacerWS, ha una versione propria ed è usato anche da altri applicativi.
Idp-rdbms-login	Implementa il modulo di login usato dall’IDP Shibboleth. Viene usato anche da Sacer, come libreria, per la funzione di disattivazione automatica degli utenti dopo un numero prefissato di errori di login. Questo modulo ha un ciclo di sviluppo separato da quello di SacerWS, ha una versione propria ed è usato anche da altri applicativi.
cryptolibrary	La libreria crittografica di Sacer, usata per la verifica delle firme digitali; anche questo modulo ha un ciclo di sviluppo separato da quello di SacerWS ed ha una versione propria.

6 Diagramma a blocchi della nuova architettura