DIREZIONE SISTEMI INFORMATIVI

INFN-23-03-DSI
14 febbraio 2023

IL PORTALE WEB DEGLI ACQUISTI INFN

A. Paoletti[1], C. Ciamei[1], A. Moni[1], E. Turella[1], F. Serafini[1], L. Sanelli[1], M. D'Alessandro[1], M. Gattari[1], G. Terilli[1]

[1)] *INFN-AC, Direzione Sistemi Informativi, Via E. Fermi 54, I-00044 Frascati (Roma)*

Abstract

L'oggetto di questo lavoro consiste nella descrizione delle applicazioni e dei servizi web progettati per il nuovo portale degli acquisti INFN. Sono descritti aspetti architetturali, metodologici ed organizzativi relativi al sistema nel suo insieme, rimandando a futuri lavori gli approfondimenti sulle singole componenti applicative.

DOI n. 10.15161/oar.it/76923

Published by
Laboratori Nazionali di Frascati

1 INTRODUZIONE E PREMESSE

Su richiesta del Direttore Generale, nel 2016 è stato istituito un gruppo di lavoro per la realizzazione di un'applicazione web per la gestione del ciclo acquisti nell'INFN. Al gruppo è stato chiesto di produrre un'analisi dei processi di acquisto, una revisione degli stessi e lo sviluppo di strumenti informatici a supporto.

Scopo dell'analisi, è identificare i passi, gli attori e i flussi di lavoro che sottendono alle attività di *procurement* nell'INFN. Dato che la maggioranza degli acquisti nell'Ente riguarda beni e servizi fino a 40 k€ (importo a base di gara, IVA esclusa), l'analisi – e il conseguente sviluppo dell'applicativo - considera solo questa fascia di acquisti.

Sulla base di questa analisi, si è proceduto ad una completa revisione dei processi, introducendo automatismi e intervenendo su inefficienze e inadempienze alle norme. Il risultato di questa attività è stato formalizzato in un documento di requisiti, rilasciato nel 2017, da cui è partita la progettazione e lo sviluppo, ad opera della **Direzione Sistemi Informativi, del tool presentato in questo documento**.

2 PROBLEMATICA E PANORAMICA ARCHITETTURALE

2.1 Descrizione del caso d'uso

È richiesta la creazione di un portale web per la gestione e il controllo dei processi di acquisto nell'INFN, partendo da una generica richiesta e arrivando all'emissione dell'ordine. Lo scopo di questa applicazione è molteplice: velocizzare il processo supportato, garantire tutti gli adempimenti normativi inerenti alla trasparenza e fornire un'interfaccia di controllo per il management.

Selezionati il tipo di acquisto e la natura dei beni/servizi da acquistare, il caso d'uso si ramifica in diversi scenari con specifici *workflow* e diverse tipologie di attori. È richiesto che il sistema produca tutta la **documentazione** prevista per legge e gestisca i flussi di **firma digitale**. Inoltre, il sistema deve gestire le **scritture contabili** legate al processo e adempiere agli obblighi di **trasparenza** previsti dalla legge. Nell'immagine che segue, è illustrato il flusso di un acquisto "generico" con le diverse tipologie di attori, gli stati del *workflow* e i documenti prodotti:

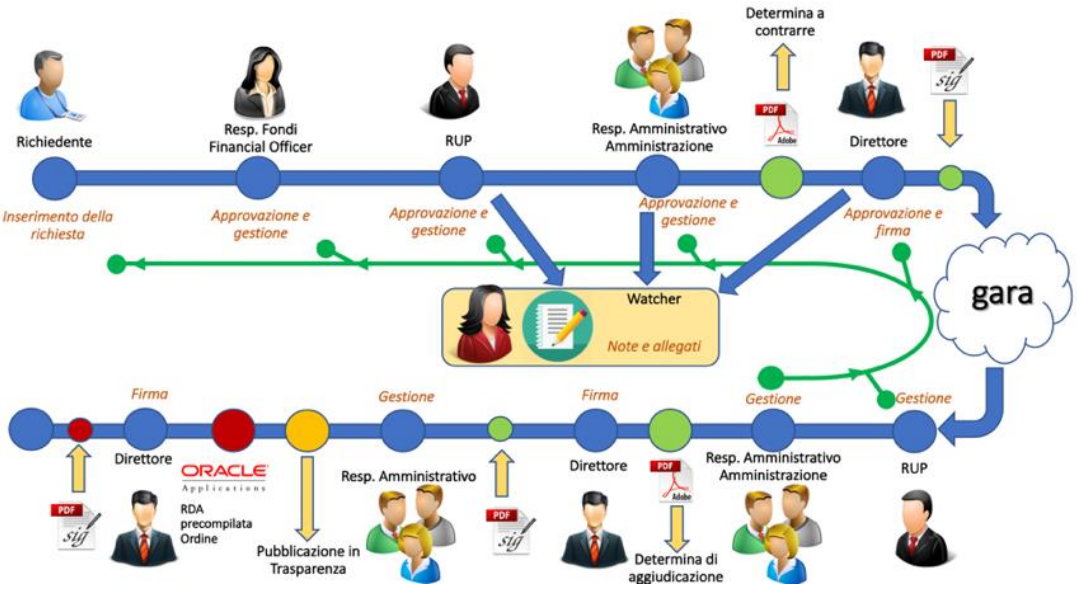

Fig. 1: Workflow del ciclo acquisti

2.2 Analisi e progettazione

Si è scelto di progettare il sistema come un insieme interoperante di moduli, prediligendo logiche architetturali distribuite (a *servizi* o *micro-servizi*) e favorendo il più possibile il riutilizzo dei componenti sviluppati. È per questo motivo che, vista la complessità e la potenziale variabilità del flusso applicativo, si è deciso di sviluppare un sistema ad hoc (**Motore di Workflow**) per governare l'iter di un processo di acquisto. Tale motore si pone nell'architettura come un servizio web interrogabile tramite *API REST*.

Per quanto riguarda l'integrazione con la contabilità, si è deciso di creare un livello intermedio (**Wrapper a Servizi**) per disaccoppiare le funzioni del sistema di contabilità dalle interfacce di accesso e rendere gli applicativi *client* indipendenti da uno specifico *vendor*. Anche questo modulo si pone come servizio web interrogabile tramite *API REST*.

Per quanto riguarda l'area documentale, si è fatto affidamento su prodotti già in uso presso l'Ente per la gestione documentale (**Alfresco Document System**) e per la gestione dei flussi di firma (**Libro Firma**). L'immagine che segue rappresenta l'architettura applicativa completa:

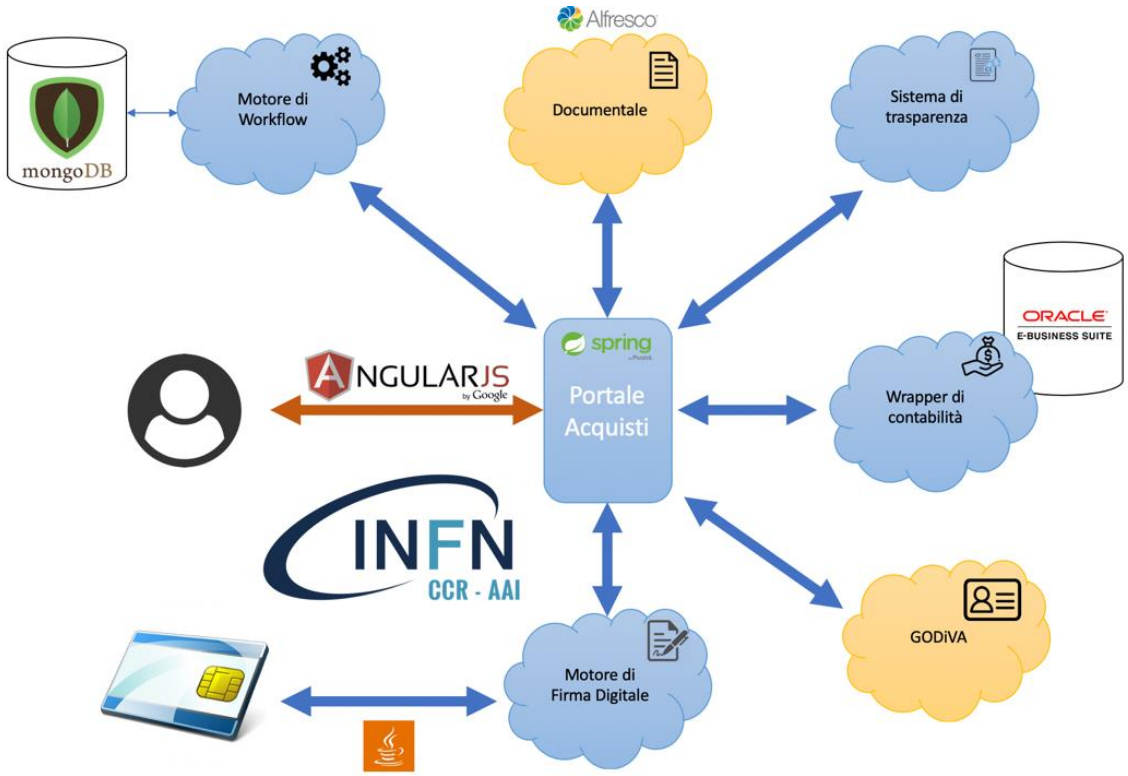

Fig. 2: Architettura applicativa del ciclo acquisti

3 PORTALE ACQUISTI

3.1 Descrizione

Il centro del sistema è costituito da un'applicazione web sviluppata in *Java2EE* con *framework* **Spring MVC** e **AngularJS**. L'applicazione prevede accessi autenticati a specifiche figure e permette, secondo i livelli di accesso, di inserire, visualizzare e modificare le pratiche relative agli acquisti. È disponibile un'interfaccia di *backend* per la gestione e l'approvazione

delle pratiche, ad uso delle amministrazioni e di tutte le figure approvative previste dal *workflow*. L'autenticazione e l'autorizzazione sono gestite tramite **INFN-AAI** e flussi **OAuth2**.

L'applicazione è strutturata su due livelli. *Client*: responsabile della logica e dell'interfaccia; *server*: responsabile dell'accesso a dati e servizi. Questa separazione, sempre più comune, permette di disaccoppiare i dati, la logica e la visualizzazione, trasformando il *server* in un gestore di *datamart* e agevolando lo sviluppo di *client* multipiattaforma.

3.2 Modello dei dati

Il modello dei dati è condiviso fra *client* e *server* e ogni operazione *CRUD* sulle entità del modello è formalizzata all'interno di servizi web, permettendo di interrogare il *server* anche da sistemi esterni (previa autenticazione) per fornire ad esempio statistiche aggiornate secondo specifiche *query* di ricerca.

L'interazione fra *client* e *server* avviene tramite *API REST*, con codifica delle informazioni in *JSON*. La struttura di questo linguaggio permette di rappresentare in maniera molto snella le collezioni e gli oggetti complessi, semplificando i processi di de/serializzazione di oggetti JAVA (gestiti lato *server*) e oggetti JavaScript (gestiti lato *client*).

3.3 Componente server

Il server è strutturato come un provider di dati e di funzionalità atomiche sugli stessi, esponendo delle *API REST* autenticate. È sviluppato usando **Spring MVC**, un comune framework **Java2EE** ed è strutturato in specifiche classi denominate Componenti (Controller, Servizi, Business Object, DAO…)

I *Controller* sono i principali *entry-point* applicativi. Sono mappati su specifiche URL e si occupano di gestire la logica di funzionamento di ogni API. Sono responsabili di rispondere in prima istanza alle interrogazioni del *client*, controllare le autorizzazioni per l'accesso, fare piccole elaborazioni, accedere a dati e servizi e fornire una risposta.

I *Servizi* forniscono funzionalità comuni a più Controller e permettono ad es. di connettersi a servizi esterni come il sistema documentale, il Motore di Workflow, GODiVA ecc.

I *Data Access Object* permettono di accedere alle basi di dati e, tramite *Business Object*, fornire le informazioni di dominio ai *Controller*. Modellano le operazioni CRUD a basso livello e si preoccupano di interagire con i database.

3.4 Componente client

La componente client è sviluppata in **Angular JS**, un *framework single-page* JavaScript creato da Google. Il client è strutturato in componenti definite *controller* e *servizi* che si occupano di gestire la logica, il *routing* e l'interazione con il *server*.

Ogni controller è deputato alla gestione e al popolamento di una o più viste HTML e ha lo scopo di gestire il flusso applicativo. L'interazione con il server avviene tramite dei *servizi*, speciali classi JavaScript composte da metodi asincroni per l'accesso alle API lato server. Tali metodi sono richiamati dal *controller* a seguito di eventi significativi come azioni dell'utente o specifiche temporizzazioni, permettendo di aggiornare il set di dati e ridisegnare la pagina.

3.5 User Experience e Layout

La componente visuale è gestita tramite *template* HTML5/CSS3 e librerie JS/CSS come **Bootstrap**. Tali librerie permettono di comporre dei layout coerenti e aderenti alle *best practice*

in materia di *User Experience* e *Interaction Design*[1], rendendo estremamente fruibili i contenuti e semplificando le attività dell'utente.

Ove possibile, si è cercato di ridurre il numero di "click" necessari per completare un'operazione, semplificando l'interfaccia, il layout e le barre di navigazione, in modo da far capire dove ci si trovi (*Where am I?*) cosa si può fare (*Where can I go?*) e cosa è necessario fare (*What should I do?*), nel più breve tempo possibile.

Particolare attenzione è posta sulla *consistency* dell'interfaccia, cercando di ridurre ogni ambiguità e ogni fonte di confusione per l'utente, riducendo il carico cognitivo e l'affidamento sulla memoria a breve termine: se un dato può fornirlo il sistema, non deve fornirlo l'utente, ricordandosi un'informazione già inserita in passato. Inoltre, si è cercato di avvicinare quanto più possibile il "linguaggio" dell'applicazione al linguaggio dell'utente, con messaggi, richieste e menu a tendina composti da termini ed espressioni appartenenti al comune gergo lavorativo.

Infine, si è puntato molto sulla messaggistica e sulla gestione degli errori, elemento imprescindibile per qualsiasi applicativo complesso (e distribuito): l'utente deve sempre essere informato su ciò che sta facendo il sistema, evitando di cedere a comportamenti determinati dall'incertezza o dalla paura di commettere errori. Sulla base di questi requisiti, si è strutturato un layout secondo il principio *dell'F-Shaped Pattern*[2], con una struttura a menu sulla sinistra, una barra in alto con le informazioni utente, e una seconda barra con le operazioni contestuali (campi di ricerca, pulsanti di navigazione interna).

---

[1] https://www.interaction-design.org/literature/topics/design-principles

[2] https://www.nngroup.com/articles/f-shaped-pattern-reading-web-content-discovered/

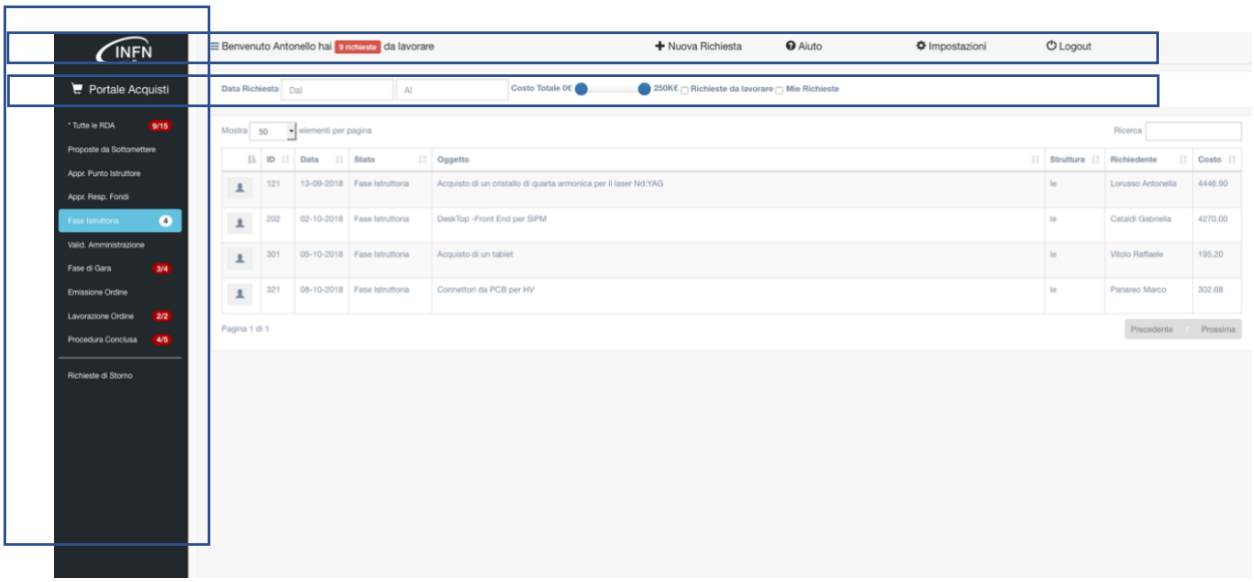

Fig. 3: Lista di richieste da lavorare

Le attività richieste all'utente sono evidenziate in rosso ("9 richieste da lavorare") e sono raggiungibili con un click. In bianco, invece, è mostrato il numero di richieste che l'utente può visualizzare, variabile in base al livello di accesso. Selezionando una fase, e successivamente una richiesta, si accede alla pagina di dettaglio della stessa, da cui l'utente può intervenire inserendo informazioni, messaggi e documenti e (secondo i livelli di accesso) dare la propria approvazione per proseguire nel workflow.

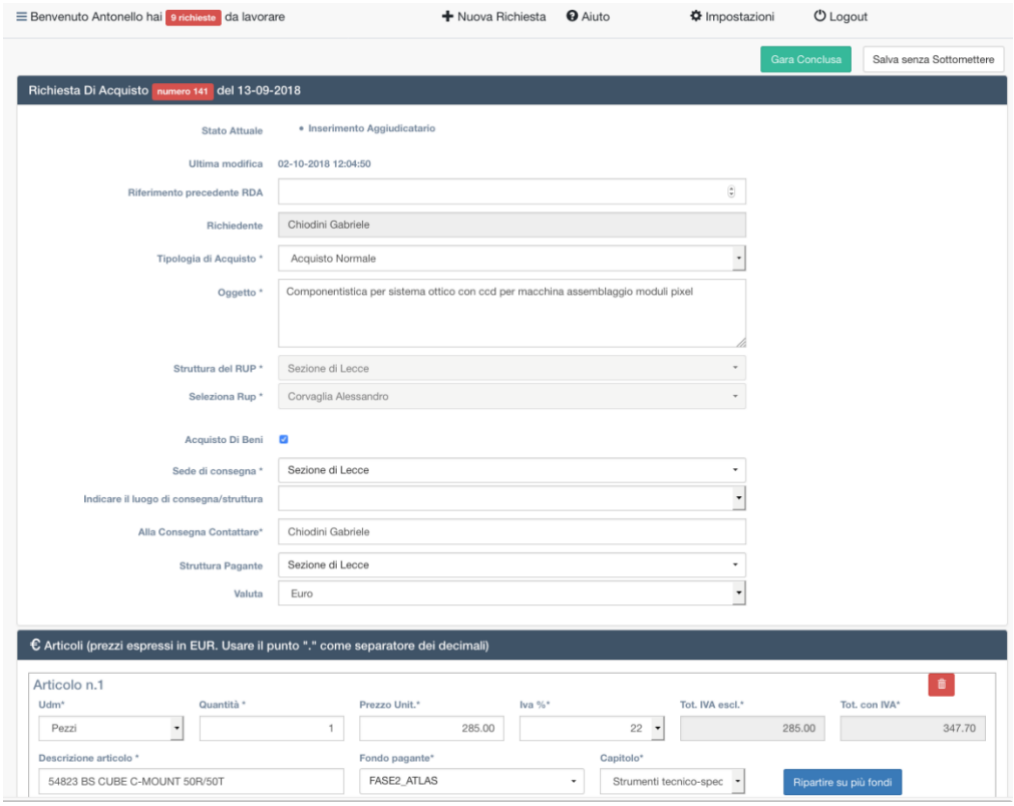

Fig. 4: Dettaglio di una richiesta

3.6 Gestione del flusso applicativo

L'applicazione interagisce con il Motore di Workflow sia per gestire il flusso della richiesta sia per comporre il layout: in base allo stato della richiesta e all'utente autenticato, il motore invia all'applicazione la lista di campi da compilare e i prossimi stati verso cui l'utente autenticato può far avanzare la richiesta. Successivamente, l'applicazione compone la form di inserimento e mostra una barra di navigazione personalizzata.

4 MOTORE DI WORKFLOW

Il motore di workflow è un servizio web sviluppato in **Java2EE** con *framework* **Spring MVC**. È composto da una parte *server* per l'accesso ai servizi e da una parte *client* per la gestione dei flussi di lavoro (*workflow*).

4.1 Descrizione di un workflow

Un workflow è una rappresentazione di un processo amministrativo in forma di grafo orientato. Esso mappa le fasi del processo sui nodi del grafo e le transizioni sugli archi. Rappresenta i possibili percorsi di una pratica amministrativa, definendone gli stati, le figure autorizzative e le condizioni da verificare per passare da uno stato all'altro.

Un'applicazione a supporto di un processo può demandare al motore di workflow la gestione del ciclo di vita di una pratica, preoccupandosi solo delle informazioni di dominio strettamente legate al processo gestito. Le informazioni sull'iter della pratica saranno gestite dal motore di workflow utilizzando un oggetto chiamato *istanza di workflow*, che accompagnerà la pratica durante il suo ciclo di vita.

Tale istanza si configura come una sorta di "puntatore" allo stato attuale del processo ed è totalmente indipendente da altre istanze di uno stesso processo. Ogni istanza è dotata di un contesto, rappresentato da coppie chiave/valore, aggiornate a turno dall'applicazione e dal motore di workflow, il quale è anche responsabile di aggiornare il puntatore allo stato attuale man mano che la relativa pratica prosegue nel suo iter.

Ogni passaggio di stato è mappato su un link del grafo ed è sottoposto a specifiche condizioni affinché sia attivabile. Dette condizioni sono codificate in forma di espressioni *booleane* e sono basate su uno o più elementi fra: contesto dell'istanza (valorizzazione di una specifica chiave); specifiche autorizzazioni dell'utente autenticato (ruolo posseduto); informazioni sullo stato del flusso (terminato/in corso).

Il passaggio di stato viene richiesto dall'applicazione titolare della pratica, indicando il link da attivare, fra quelli possibili. Il motore controlla le condizioni di attivazione del link e, se verificate, aggiorna lo stato corrente dell'istanza con il nodo raggiunto dal link richiesto.

4.2 Interfaccia web di progettazione del modello di workflow

È disponibile un'interfaccia web per la progettazione di un workflow e per definire le informazioni a corredo su link e nodi

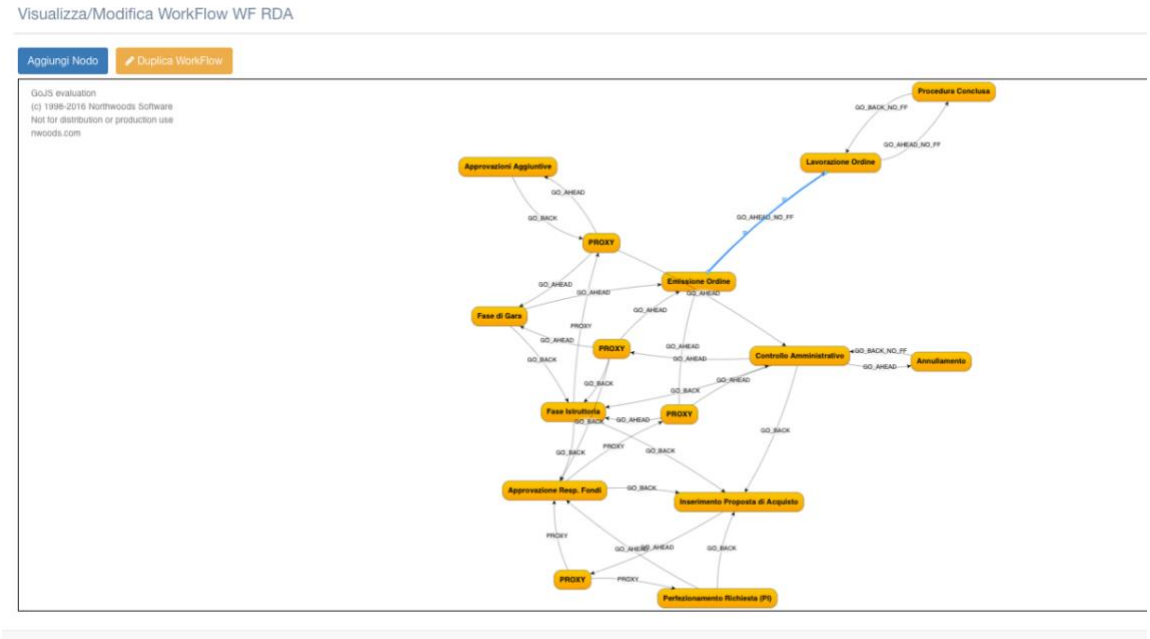

Fig. 5: Rappresentazione del workflow del ciclo acquisti

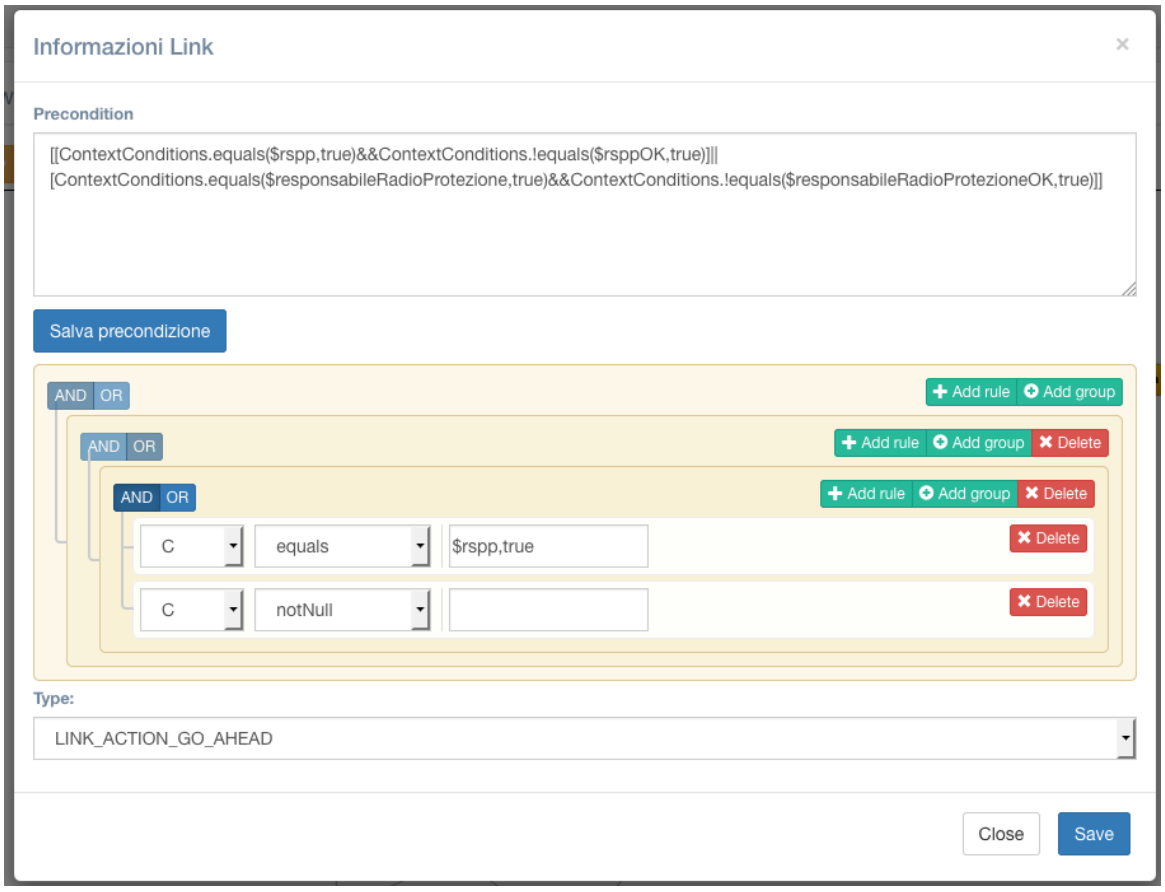

Fig. 6: Gestione delle precondizioni di attivazione di un link

4.3 Utilizzo da parte delle applicazioni

Un'applicazione può richiedere (tramite API) un'istanza di un modello di workflow, ricevendo dal motore un riferimento alfanumerico all'istanza appena creata. Tale riferimento sarà utilizzato nelle successive comunicazioni fra applicazione e motore, ad esempio, per chiedere di passare allo stato successivo.

5 WRAPPER A SERVIZI PER LA CONTABILITÀ

Il Wrapper a Servizi per la contabilità è un servizio web sviluppato in **Java2EE** con framework **Spring MVC**. Espone delle *API REST* per l'accesso alle funzionalità del sistema contabile, mascherandone la complessità e le interfacce.

La versione attualmente installata del sistema contabile presenta limitazioni in termini di interfacce e per questioni di modularità, flessibilità e manutenibilità, si è deciso di creare un livello software per tradurre le chiamate *PL/SQL* in *API REST* e permettere, da un lato, di fornire una modalità di interrogazione più moderna e, dall'altro, di non legare le applicazioni a uno specifico *vendor*/versione, anche in vista di future evoluzioni del sistema contabile.

Le funzionalità al momento gestite riguardano l'ambito dei *preimpegni*, degli *storni* e degli *ordini*, esponendo delle API autenticate per la creazione e la consultazione.

6 STRUMENTI E METODOLOGIE DI SVILUPPO E DEPLOYMENT

Per la gestione del ciclo di vita del software si è fatto riferimento alle ultime *best practice* in tema di sviluppo e *deployment*, orientate alla più ampia solidità e, soprattutto, all'automazione dei processi di rilascio.

Le modifiche al codice sono tracciate tramite repository GIT e ogni azione viene validata in un'ottica di *early failing*, in modo da individuare quanto prima le potenziali falle e impedire il rilascio nelle fasi successive.

In particolare, a seguito di tali controlli, è possibile avviare processi di *Continuous Integration* e *Continuous Delivery* attraverso la generazione di pacchetti rilasciabili sull'architettura finale che ospiterà il servizio. In questo processo è possibile introdurre fasi automatizzate di gestione delle dipendenze, usando ad es. *Gradle*, *maven*, *composer* e simili.

7 FORMAZIONE, TEST, ROLLOUT E ASSISTENZA

Per l'attivazione dei *tool* si è deciso di scaglionare le diverse strutture in maniera da rodare man mano il sistema e aiutare l'utenza nell'uso delle nuove interfacce.

In particolare, per ogni nuova struttura attivata, si è proceduto con corsi di formazione mirati alle diverse figure, basati su lezioni teoriche, esercizi pratici e test sull'interfaccia, volti a far emergere criticità (bug e difficoltà nell'uso) prima dell'effettivo rilascio in produzione.

Questa modalità ha permesso di tenere un contatto più stretto con l'utenza e, soprattutto, ha permesso al servizio di assistenza di gestire in maniera più puntuale le richieste, orientandosi su specifiche casistiche. L'utilizzo del *ticketing*, combinato con sistemi di *kanban*, ha reso misurabile questa attività, ponendo le basi per analisi di urgenza/opportunità nel caso di interventi sul software.

8 CONCLUSIONI E SVILUPPI FUTURI

A metà 2020 abbiamo attivato il sistema in quasi tutte le strutture, con circa 6000 richieste evase a partire da fine 2018. Questa esperienza è stata un'importante lezione sotto diversi aspetti, poiché ha rappresentato un primo tentativo di sviluppo utilizzando logiche di *dev-ops*. È stato, inoltre, il primo rilascio basato su tecnologia **Spring/Angular**, con netta separazione

delle componenti *frontend* e *backend* e, infine, è stato un primo tentativo di utilizzo massivo della distribuzione di funzionalità su sistemi a *microservizi*.

L'evoluzione delle architetture, dei linguaggi e dei framework, pone continuamente il problema dell'obsolescenza di pratiche, linguaggi e paradigmi e non si esclude un futuro passaggio da Angular JS ad Angular 6, con la conseguente riprogettazione dell'interfaccia client, e soprattutto, il passaggio dalla *Paranoid Auth* a tecnologie più standard basate su **OAuth2**.

Per quanto riguarda il **motore di workflow**, si prevede nel medio/lungo termine un passaggio a sistemi più standardizzati, basati su linguaggi di definizione dei processi (**Business Process Modeling Notation**) che permettano di disegnare e modificare agevolmente un *workflow*.

Occorre inoltre estendere il caso d'uso a tutte le procedure di procurement in essere nell'Ente, compresi i lavori e gli acquisti sopra ai 40 k€. Per fare ciò occorre ripartire dall'analisi coinvolgendo la Direzione Affari Finanza e Controllo, competente a produrre i template di documentazione necessaria e governare il processo amministrativo, e a seguire un gruppo di esperti di dominio (*key users*) per l'analisi funzionale e la fase di test successiva allo sviluppo.

Alcuni adeguamenti sono comunque stati fatti nel frattempo, principalmente per consentire di gestire tutti gli acquisti effettuati dalle Strutture almeno come *repository* della documentazione di gara da inviare in conservazione documentale.

Infine, per quanto riguarda i processi di firma digitale, si attende nel breve termine il rilascio di un sistema commerciale di gestione dei flussi di firma, già integrato nel ciclo acquisti, che permetterà l'abbandono delle *Smart Card* e l'adozione di sistemi di firma remota.