Versione

2.5

CMDBuild®

» Workflow Manual

Novembre 2017

Author Tecnoteca srl

www.tecnoteca.com

ITA

CMDBuild ® leverages many great technologies from the open source community:
PostgreSQL, Apache, Tomcat, Eclipse, Ext JS, JasperReports, IReport, Enhydra Shark, TWE, OCS Inventory, Liferay, Alfresco, GeoServer, OpenLayers, Prefuse, Quartz, BiMserver.
We are thankful for the great contributions that led to the creation of that products.

CMDBuild ® è un prodotto di Tecnoteca S.r.l. che ne ha curato la progettazione e realizzazione, è maintainer dell'applicazione e ne ha registrato il logo.

Al progetto ha anche partecipato come committente iniziale il Comune di Udine – Servizio Sistemi Informativi e Telematici.

Il sito ufficiale di CMDBuild è http://www.cmdbuild.org

Sommario

Introduzione

CMDBuild è una applicazione Open Source finalizzata a supportare la gestione della configurazione degli oggetti e dei servizi informatici in carico al Dipartimento ICT di una organizzazione e a guidarne i processi di controllo, eventualmente secondo le “best practice” ITIL.

Gestire un Database della Configurazione (CMDB) significa mantenere aggiornata e disponibile per gli altri processi la base dati relativa agli elementi informatici utilizzati, alle loro relazioni ed alle loro modifiche nel tempo.

Con CMDBuild l'amministratore del sistema può costruire ed estendere autonomamente il proprio CMDB (da cui il nome del progetto), modellandolo su misura della propria organizzazione tramite un apposito Modulo di Amministrazione che consente di aggiungere progressivamente nuove classi di oggetti, nuovi attributi e nuove tipologie di relazioni. E' anche possibile definire filtri, “viste” e permessi di accesso ristretti a righe e colonne di ciascuna classe.

CMDBuild è in grado di fornire un completo supporto all'adozione delle “best practice” ITIL, ormai affermatesi come "standard de facto", non proprietario, per la gestione dei servizi informatici secondo criteri orientati ai processi.

Tramite un apposito sistema di gestione dei workflow è possibile definire in modo visuale, con un editor esterno, nuovi processi operanti sulle classi modellate nel database, importarli in CMDBuild ed eseguirli secondo i flussi previsti e con gli automatismi configurati.

E' disponibile un task manager integrato nell'interfaccia utente del Modulo di Amministrazione che consente di gestire in background diverse tipologie di operazioni (avvio di processi, ricezione e invio di mail, esecuzione di connettori) e di controlli sui dati del CMDB (eventi sincroni e asincroni) a fronte delle quali eseguire notifiche, avviare workflow ed eseguire script.

CMDBuild consente la stampa di report tramite il motore open source JasperReports, sia di tipo tabulare prodotti tramite un wizard interno, che di maggiore complessità ottenibili importando template disegnati tramite un apposito editor visuale esterno.

Possono essere poi definite delle dashboard costituite da grafici che mostrino in modo immediato la situazione di alcuni indicatori dello stato corrente del sitema (KPI).

Grazie all'integrazione con il diffuso sistema documentale open source Alfresco è inoltre possibile allegare documenti, immagini, video ed altre tipologie di file alle schede archiviate in CMDBuild.

E' anche possibile utilizzare funzionalità GIS per il georiferimento degli asset e la loro visualizzazione su una mappa geografica (servizi mappe esterni) e / o sulla planimetria di un ufficio (server locale GeoServer) e funzionalità BIM per la visualizzazione di modelli 3D in formato IFC.

Sono poi inclusi nel sistema un webservice SOAP ed un webservice REST, utili per implementare soluzioni di interoperabilità con architettura SOA.

CMDBuild comprende di base due framework denominati Basic Connector e Advanced Connector, che tramite il webservice SOAP sono in grado di sincronizzare le informazioni registrate nel CMDB con fonti dati esterne, ad esempio con sistemi di automatic inventory (quali lo strumento open source OCS Inventory) o con sistemi di virtualizzazione o di monitoraggio.

Un ulteriore strumento, il CMDBuild GUI Framework, consente invece tramite il webservice REST di pubblicare su portali esterni pagine web personalizzate in grado di interagire con il CMDB.

E' infine disponibile una interfaccia utente ottimizzata per strumenti “mobile” (smartphone e tablet), implementata come “app” multipiattaforma (iOS, Android) e anch'essa collegata a CMDBuild tramite il webservice REST.

I moduli di CMDBuild

Il sistema CMDBuild comprende due moduli principali:

- il Modulo di Amministrazione, dedicato alla definizione iniziale ed alle successive modifiche del modello dati e delle configurazioni di base (classi e tipologie di relazioni, utenti e permessi, dashboard, upload report e workflow, opzioni e parametri)
- il Modulo di Gestione dati, dedicato alla consultazione ed aggiornamento delle schede e delle relazioni nel sistema, alla gestione di documenti allegati, all'avanzamento dei processi, alla visualizzazione di dashboard e produzione di report

Il Modulo di Amministrazione è riservato agli utenti abilitati al ruolo di amministratore, il Modulo di Gestione è utilizzato dagli operatori addetti alla consultazione ed aggiornamento dei dati.

Documentazione disponibile

Il presente manuale è dedicato alla descrizione del sistema di Workflow compreso nell'applicazione CMDBuild, tramite cui è possibile configurare (Modulo Amministrazione) ed eseguire (Modulo Gestione) processi guidati per la gestione di attività collaborative.

Sono disponibili sul sito di CMDBuild (http://www.cmdbuild.org) ulteriori specifici manuali dedicati a:

- overview concettuale del sistema (“Overview Manual”)
- amministrazione del sistema (“Administrator Manual”)
- utilizzo del sistema (“User Manual”)
- installazione e gestione tecnica del sistema (“Technical Manual”)
- utilizzo del webservice per l'interoperabilità con sistemi esterni (“Webservice Manual”)
- utilizzo di connettori per la sincronizzazione di dati con sistemi esterni (“ConnectorsManual”)

Descrizione del sistema di workflow

Generalità

Per poter supportare le indicazioni metodologiche di ITIL il sistema CMDBuild non solamente è in grado di gestire l'aggiornamento dell'inventario degli asset e delle relazioni funzionali fra di essi, ma consente anche di definire e controllare i processi di gestione dei servizi informatici.

Un processo consiste di una sequenza di attività, svolte da operatori e/o da applicazioni informatiche, ciascuna delle quali rappresenta un'azione da svolgere all'interno del processo (nel caso specifico relativamente alla gestione degli asset informatici con criteri di qualità).

Dati il numero elevato dei processi attivabili, le peculiarità organizzative dei singoli enti e le caratteristiche di estensibilità, flessibilità ed autonomia di gestione perseguite dal progetto CMDBuild, si è scelto di implementare non una serie di processi rigidi e predefiniti, ma un sistema generico tramite il quale utenti evoluti possano disegnare ed attivare autonomamente i workflow di proprio interesse.

Nella prima parte del documento sono descritti i concetti generali ed i meccanismi di base implementati nel sistema con il refactoring di CMDBuild 2.0.

Nella seconda parte del documento viene presentato un esempio di workflow semplificato, descritto passo passo nelle sue fasi di configurazione.

Nella terza parte del documento vengono invece documentate gli strumenti tecnici disponibili per la configurazione di un workflow: definizione dei widget, descrizione delle funzioni API utilizzabili negli script tramite cui possono essere definiti gli automatismi da eseguire nell'ambito del workflow.

In appendice viene infine riportata per completezza la documentazione tecnica specifica del sistema di workflow in uso fino a CMDBuild 1.5, di cui viene mantenuta la compatibilità anche in CMDBuild 2.0 (come meglio descritto nel seguito), in attesa di dismetterlo con adeguato preavviso nelle versioni successive.

Obiettivi

Il sistema di gestione del workflow costituisce il principale valore aggiunto dell'adozione di CMDBuild, fornendo nel contempo:

- una modalità guidata di azione per tutti gli operatori di cui sarà uniformato e standardizzato il comportamento
- una garanzia di corretto aggiornamento del CMDB
- un sistema per il controllo operativo puntuale del servizio svolto
- un repository di dati relativi alle attività pregresse da cui ricavare statistiche periodiche di verifica degli SLA contrattualizzati con gli utenti

Fra i processi descritti da ITIL e configurabili in CMDBuild ricordiamo ad esempio quelli di Incident Management, Problem Management, Change Management, Configuration Management, Service Catalogue Management, ecc

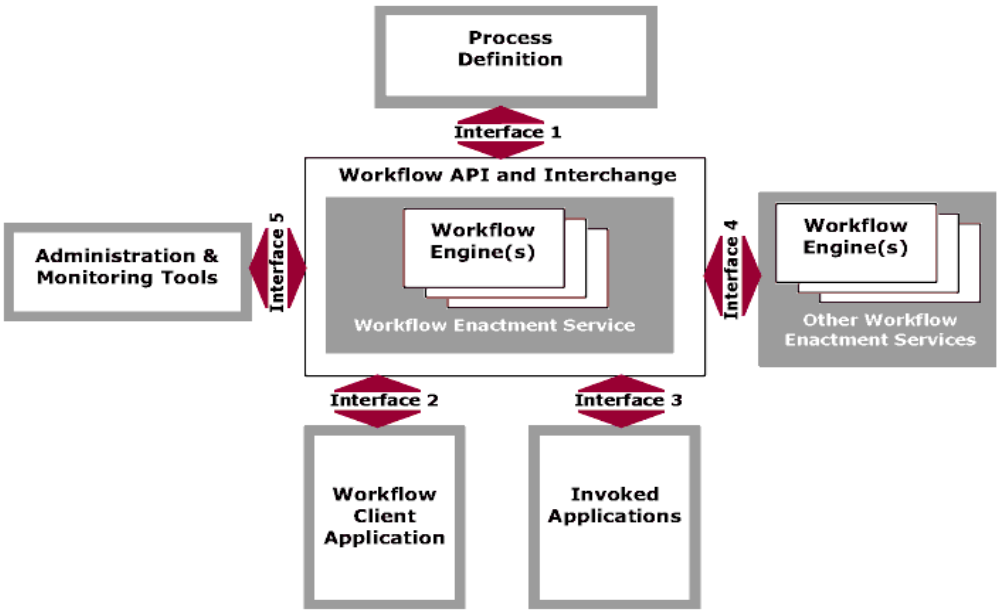
Altre tipologie di workflow possono riguardare la movimentazione di asset, l'ingresso di nuovo personale, l'attivazione di progetti di lavoro, ecc

Strumenti utilizzati

Il sistema scelto in CMDBuild per la gestione del workflow utilizza i seguenti strumenti:

- XPDL 2.0 (http://www.wfmc.org/xpdl.html) come linguaggio di definizione (standardizzato dalla WfMC, WorkFlow Management Coalition sulla base del modello sotto riportato)
- il motore open source TWS Together Workflow Server 4.4 (http://www.together.at/prod/workflow/tws), un framework estensibile che fornisce una implementazione completa e standard delle specifiche WfMC (http://www.wfmc.org/) e OMG, utilizzando al suo interno XPDL come linguaggio nativo
- l'editor visuale TWE Together Workflow Editor 4.4 (http://www.together.at/prod/workflow/twe) per il disegno del workflow e per la definizione dei meccanismi di integrazione con CMDBuild

Segue lo schema di riferimento per la gestione dei workflow secondo il modello standardizzato dal WfMC.



Terminologia

Il "vocabolario" utilizzato nel seguito del presente documento comprende i seguenti termini principali:

- con "processo" si intende una sequenza di passaggi ("attività") descritti nel sistema per svolgere una determinata azione in forma guidata e secondo regole prestabilite
- con "attività" si intende uno dei passaggi che costituiscono il flusso del processo
- con "istanza di processo" si intende una specifica attivazione di un "processo", effettuata tramite avvio e conferma del primo passaggio previsto
- con "istanza di attività" si intende una specifica attivazione di una attività, effettuata automaticamente dal sistema o manualmente da un operatore (tramite compilazione di suoi attributi e di eventuali ulteriori operazioni richieste e conferma finale)

I termini sopra indicati sono “tradotti” nel modello dati di CMDBuild secondo i seguenti criteri:

- ogni “processo” corrisponde ad una classe “specializzata”, definibile dal Modulo di Amministrazione nell'apposita voce di menu “Processi”, comprendente l’ “unione” degli attributi caratterizzanti le diverse attività costitutive
- ogni “istanza di processo” corrisponde ad una scheda della classe di tipo “processo” (attività corrente), unita alla lista delle sue versioni storicizzate (attività concluse)
- ogni “istanza di attività” corrisponde ad una scheda della classe di tipo “processo” (attività corrente) oppure ad una delle sue versioni storicizzate (attività concluse)

Ogni processo è caratterizzato da un nome, da uno o più gruppi di partecipanti, da alcune variabili e da una sequenza di attività e transizioni che lo realizzano.

Lo stato di un processo può essere:

- “attivo”, cioè fermo in una attività intermedia
- “completato”, cioè giunto alla conclusione della sua attività finale
- “abortito”, cioè chiuso in modalità anomala
- “sospeso”, mantenuto solamente per retrocompatibilità con il sistema di workflow presente fino a CMDBuild versione 1.5 inclusa

Ogni attività è caratterizzata da:

- un nome
- un esecutore, che corrispondente obbligatoriamente ad un “gruppo di utenti” ed opzionalmente ad un operatore
- un tipo: inizio processo, fine processo, attività eseguita da un operatore, attività eseguita automaticamente dal sistema
- eventuali attributi, provenienti da CMDBuild o interni al workflow, che saranno valorizzati nel corso della sua esecuzione
- eventuali widget (controlli visuali di alcune tipologie predefinite) da attivare nel corso della sua esecuzione
- uno script (nei linguaggi BeanShell, Groovy o Javascript), previsto nelle attività automatiche, tramite cui eseguire delle operazioni fra una attività utente e la successiva

Refactoring 2.0

Con la release 2.0 è stata effettuata una sostanziale revisione del sistema di workflow, con upgrade a Together Workflow Server 4.4, adozione dello standard XPDL 2.0, pieno supporto in CMDBuild al parallelismo nativo nel flusso ed importanti migliorie di performance.

Per semplificarne la scrittura si è inoltre deciso di prevedere una diversa modalità di definizione delle attività automatiche, supportando la scrittura di script ed escludendo l'utilizzo dei “tool” disponibili nelle precedenti versioni di CMDBuild.

Gli script possono essere scritti in linguaggio BeanShell, Groovy o Javascript e possono utilizzare funzioni API appositamente predisposte per la definizione di automatismi (manipolazione di variabili del processo, creazione di schede e relazioni nel CMDB, invio mail, creazione report, ecc).

L'adozione del nuovo sistema di workflow comporta la perdita della retrocompatibilità con i workflow sviluppati fino ad oggi.

Per garantire agli attuali utilizzatori di CMDBuild tempi più lunghi per la migrazione dei vecchi workflow alle nuove soluzioni adottate si è però deciso di mantenere in CMDBuild 2.0 la doppia

possibilità di lavorare (ovviamente in alternativa) sia con Together Workflow Server 2.3 (la versione utilizzata fino a CMDBuild 1.5, basata su XPDL 1.0) che con la nuova versione Together Workflow Server 4.4 (basata su XPDL 2.0).

La soluzione adottata consente quindi:

- ai nuovi utilizzatori di CMDBuild di lavorare da subito con il nuovo Together Workflow Server 4.4 e con le nuove funzionalità sviluppate nella versione 2.0 (parallelismo nativo, automatismi configurati tramite script)
- ai vecchi utilizzatori di suddividere la migrazione in due passaggi:
    1. attivare da subito la versione 2.0 per usufruire delle nuove dashboard e delle altre funzionalità implementate, mantenendo ancora attivo Together Workflow Server 2.3 (con performance già notevolmente migliorate)
    2. commutare a Together Workflow Server 4.4 solamente dopo aver testato il nuovo ambiente sull'istanza di test, una volta disponibile il tool di conversione.

E' previsto il rilascio di un tool di supporto alla migrazione dei workflow sviluppati con le precedenti versioni di CMDBuild.

Si consiglia di effettuare la migrazione in tempi non eccessivamente lunghi, dal momento che la doppia compatibilità di CMDBuild con Together Workflow Server 2.3 (XPDL 1.0) e Together Workflow Server 4.4 (XPDL 2.0) sarà mantenuta per un periodo di tempo limitato.

Modalità di implementazione

Workflow come classi particolari

I meccanismi per la gestione del workflow sono implementati in CMDBuild tramite concetti e modalità del tutto omogenei con i meccanismi già presenti nel sistema per la gestione delle schede dati.

Gli ingredienti per la gestione del workflow comprendono infatti:

- classi “speciali” di tipo “Processo” ciascuna corrispondente ad una tipologia di workflow
- attributi, corrispondenti alle informazioni presentate (in lettura o scrittura) nelle form che gestiscono l'esecuzione di ciascun singolo passaggio del processo
- relazioni con altre istanze di processo o schede standard coinvolte nel processo
- gruppi di utenti che saranno abilitati a svolgere ciascuna attività, coincidenti con i gruppi di utenti di CMDBuild
- strumenti specifici per la personalizzazione del comportamento del workflow (widget e script scritti utilizzando apposite API)

Nell'ambito degli stessi criteri di omogeneità fra classi “normali” e classi di tipo “processo”, sono stati utilizzati i seguenti accorgimenti tecnici:

- è stata creata una nuova superclasse “riservata” denominata “Activity” e contenente alcuni attributi comuni agli specifici workflow definibili, di cui tutti i workflow sono sottoclassi
- è stato utilizzato il meccanismo della “storia” per tracciare gli stati di avanzamento di un processo
- è stato mantenuto il meccanismo delle “relazioni” per creare collegamenti automatici o manuali in forma “guidata” fra una scheda dati e un'istanza di processo o fra due istanze di processo

Costruzione del flusso del processo

Gli strumenti specifici utilizzabili tramite l'editor visuale di workflow rivestono una importanza fondamentale nel consentire il disegno di processi complessi, e comprendono:

- la scelta di quali attributi posizionare su ciascuna form corrispondente ad una attività utente
- la scelta di quali widget (controlli visuali) posizionare su ciascuna form corrispondente ad una attività utente (visualizzazione o creazione o modifica di schede, visualizzazione o creazione di relazioni, selezione singola o multipla di schede, caricamento di file allegati, esecuzione di report)
- meccanismi di controllo del flusso, fra cui attività parallele e sottoprocessi
- linguaggio di scripting (BeanShell, Groovy o Javascript) per la definizione degli automatismi da eseguire fra una attività utente e la successiva
- funzioni API richiamabili negli script

Per chi fosse interessato alla documentazione degli ulteriori meccanismi utilizzati nei workflow sviluppati per le versioni di CMDBuild fino alla 1.5 inclusa (ed ancora supportati in CMDBuild 2.0 se si opta per l'utilizzo di Together Workflow Server 2.3) si rimanda alla precedente documentazione riproposta in Appendice (dedicata in particolare alla presentazione dei tool base e

del meccanismo di definizione di tool custom tramite utilizzo di appositi template).

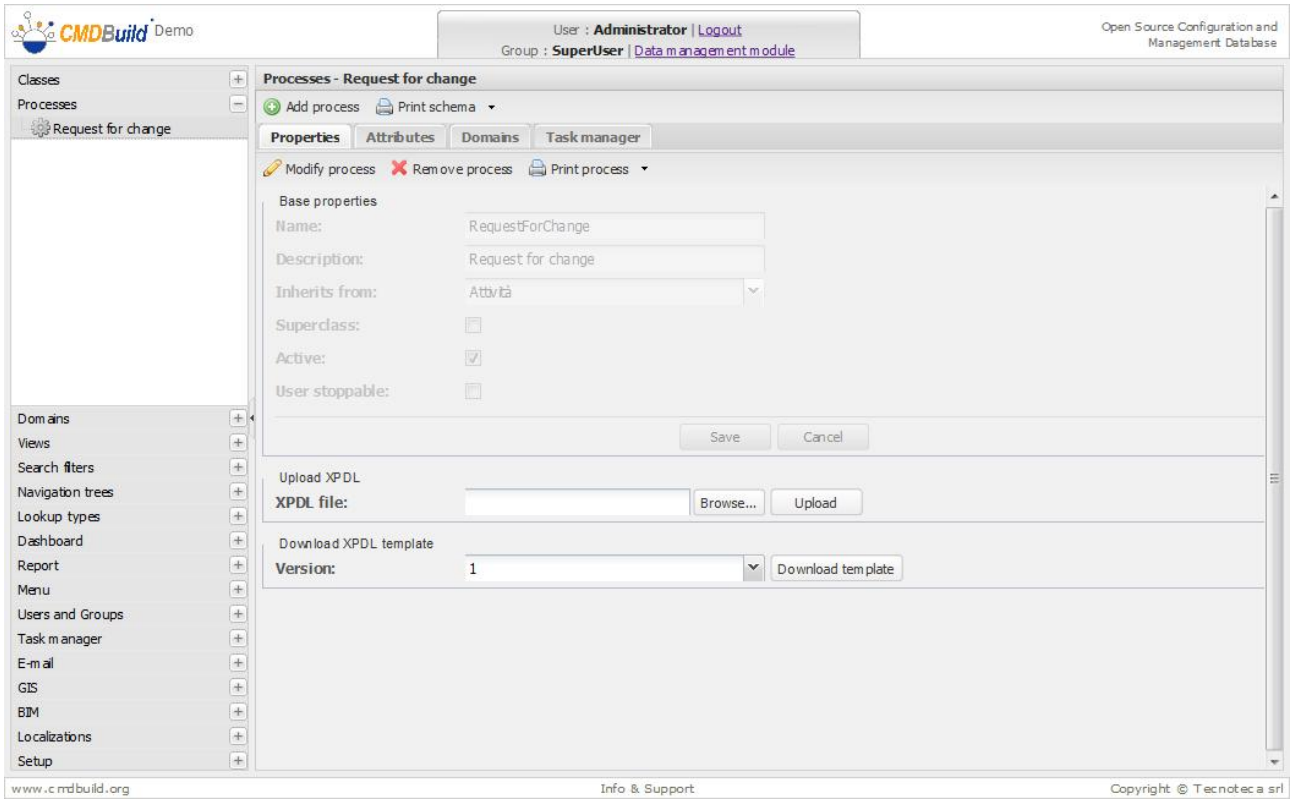
Definizione di un nuovo processo

Per la creazione di una nuova classe di tipo “Processo” si suggerisce di seguire la seguente sequenza logica di passaggi:

- analisi “sulla carta” del nuovo processo da implementare, al fine di individuare:
  - la lista dei gruppi di utenti coinvolti nel processo
  - il flusso del processo: attività utente, attività automatiche, condizioni di transizione, ecc
  - gli attributi descrittivi del processo in ciascuna delle sue attività utente, le rispettive tipologie (stringhe, interi, ecc) e la modalità di presentazione (sola lettura, anche scrittura, eventuale obbligatorietà)
  - le liste di valori predefinite necessarie per la creazione degli attributi di tipo “Lookup”
  - i domini necessari per gestire le correlazioni fra il nuovo processo ed altre classi o altri processi preesistenti (eventualmente anche utilizzati per la creazione degli attributi di tipo “Reference”)
  - i widget da configurare in ciascuna attività utente
  - gli script da configurare in ciascuna attività automatica del processo
- creazione della classe del nuovo processo, che dovrà essere definita nell'ambito della sezione “Processi” del Modulo di Amministrazione di CMDBuild, completa di:
  - gli attributi specifici individuati al passo precedente
  - i domini individuati al passo precedente
- creazione dei gruppi di utenti mancanti, che dovranno essere aggiunti tramite il Modulo di Amministrazione
- esportazione con il Modulo di Amministrazione (dal TAB “XPDL” disponibile per ogni classi di tipo “Processo”) dello “scheletro” del nuovo schema di processo, che conterrà al suo interno:
  - nome del processo
  - lista degli attributi del processo, che saranno poi posizionati nelle diverse attività utente
  - lista degli “attori” (gruppi di utenti) partecipanti al processo (cui viene aggiunto il ruolo “fittizio” denominato “Sistema” per il posizionamento delle attività automatiche)
- disegno del flusso di dettaglio del workflow tramite utilizzo dell'editor esterno TWE, con cui verrà completato lo “scheletro” esportato da CMDBuild
- salvataggio, tramite le apposite funzioni dell'editor esterno TWE del file XML (per la precisione XPDL 2.0) corrispondente al processo disegnato
- importazione in CMDBuild dello schema del processo, tramite l'apposito TAB “XPDL” disponibile nella voce di Menu “Processi” del Modulo di Amministrazione

Una volta terminate le operazioni sopra descritte il nuovo processo è pronto per poter essere utilizzato dal Modulo di Gestione di CMDBuild (Menu “Processi” o voci di tipo “processo” del Menu di Navigazione), che ne interpreterà ed eseguirà lo schema tramite il supporto del motore di workflow Together Workflow Server 4.4.

Le operazioni descritte possono essere eseguite anche più volte a fronte della necessità di modifica di un processo già importato, con l'unica avvertenza che le modifiche saranno recepite solo dalle nuove istanze del processo che verranno avviate.



Avvio ed avanzamento di un processo

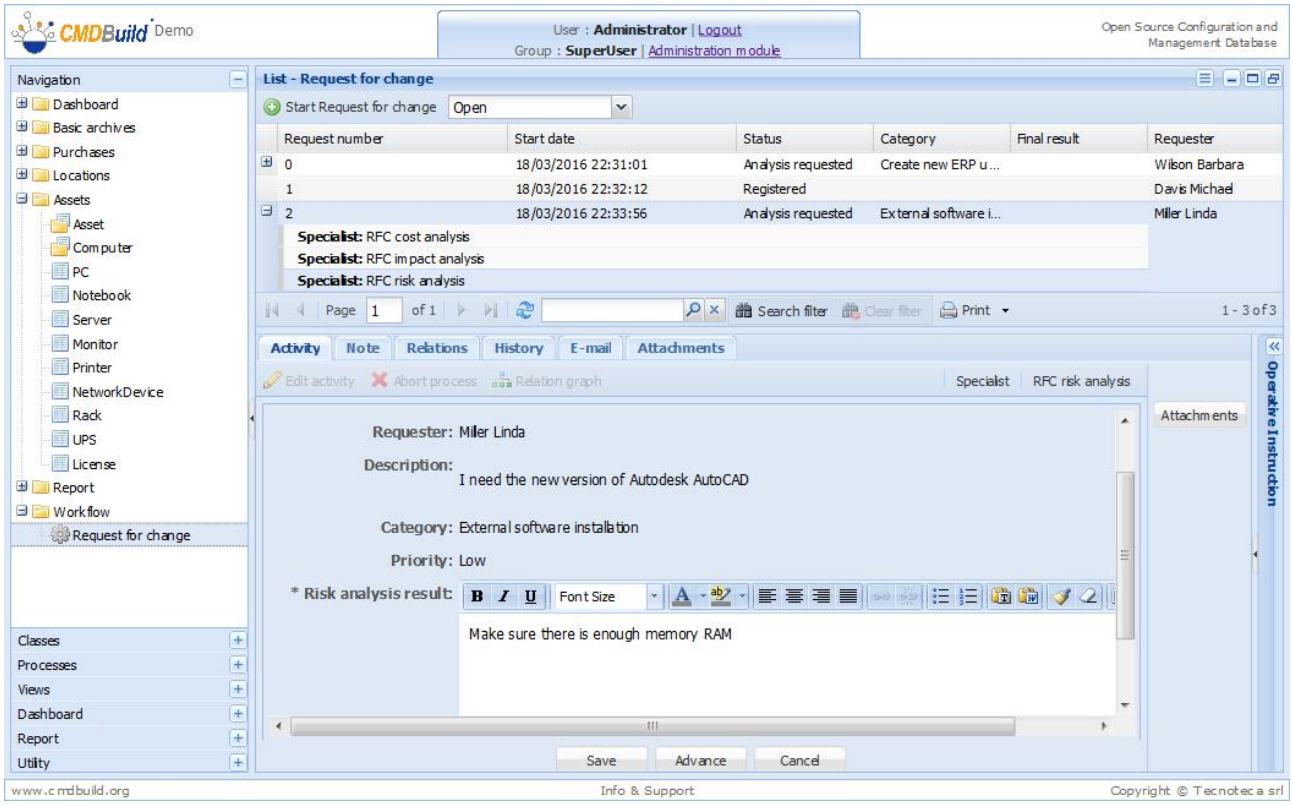
L'applicazione CMDBuild comprende nel Modulo Gestione la possibilità di interpretare, tramite il supporto del motore TWS Together Workflow Server, i processi disegnati esternamente con TWE Together Workflow Editor e poi importati tramite il Modulo di Amministrazione.

Sempre con l'obiettivo di mantenere la massima coerenza con le funzionalità di CMDBuild dedicate alla gestione delle schede degli item gestiti nel sistema, l'interfaccia utente del Modulo Gestione è stata progettata in modo omogeneo con quella utilizzata per le normali "classi" di dati:

- è disponibile una apposita voce di menu "Processi" omogenea con la voce "Schede dati" (oppure possono essere inseriti elementi di tipo "processo" nel menu di "Navigazione", assieme agli elementi di tipo "Schede dati" o a report e dashboard
- la gestione dei processi riprende le gestioni standard già presenti per le normali schede dati: "Lista", "Scheda", "Dettagli", "Note", "Relazioni", "Storia", "Allegati"
- nel TAB "Lista" di uno specifico processo sono visualizzate le istanze delle attività in cui l'utente è coinvolto (perché partecipa a quell'attività o ha partecipato ad attività precedenti di quel processo) con:
  - filtri per stato (avviato, completato, sospeso)
  - area dati con visualizzazione tabellare delle informazioni (il nome del processo, il nome dell'attività, la descrizione della richiesta, lo stato del processo e gli ulteriori attributi definiti come "display base" nel Modulo di Amministrazione), rese "cliccabili" per l'accesso alla scheda di gestione di quell'attività
  - eventuali evidenze di attività parallele in corso per quell'istanza di processo
  - pulsanti per creare una nuova attività o per operare su quella scelta

- nel TAB “Scheda” è possibile visualizzare o compilare gli attributi previsti per quell’istanza di attività del processo (l'accesso in scrittura o sola lettura è impostabile tramite l'editor TWE) oppure effettuare eventuali ulteriori operazioni tramite gli appositi widget (controlli visuali) configurati con l'editor TWE
- nel TAB “Note” è possibile visualizzare o inserire annotazioni sull’istanza di attività
- nel TAB “Relazioni” è possibile visualizzare o inserire relazioni fra l’istanza dell’attività e istanze di altre classi (“schede”)
- nel TAB “Storia” è possibile visualizzare le versioni precedenti di quell’istanza di attività (istanze già eseguite)

La lista delle attività da eseguire viene presentata in alto nella successiva form esemplificativa, mentre lo svolgimento di un'attività viene effettuato compilando la scheda presentata in basso.



Essendo i workflow un caso particolare di classi, anche nella form di gestione dei workflow sono presenti in alto a destra i pulsanti di controllo per portare a pieno schermo la zona superiore o inferiore della form.

Interazione del workflow con strumenti esterni

Generalità

In alcuni casi può essere richiesto che un processo (ad esempio una nuova richiesta di Helpdesk) venga avviato da un utente non informatico (ad esempio l'utilizzatore dell'oggetto o servizio IT), non sufficientemente esperto per utilizzare l'interfaccia standard dell'applicazione CMDBuild.

Tale necessità può essere risolta utilizzando il CMDBuild GUI Framework, come descritto al paragrafo successivo.

Avvio processo da portale intranet tramite il CMDBuild GUI Framework

Il GUI Framework è un ambiente di programmazione sviluppato in ambiente javascript / Jquery, utilizzabile per implementare una interfaccia utente semplificata tramite cui utenti non informatici possono interagire con l'applicazione CMDBuild.

Il GUI Framework fornisce le seguenti caratteristiche principali:

- è attivabile in portali basati su tecnologie diverse
- consente una libertà pressochè illimitata nella progettazione del layout grafico, definibile tramite un descrittore XML e con possibilità di intervenire sul foglio stile CSS
- garantisce tempi ridotti di configurazione grazie a funzioni predefinite (logiche di comunicazione, di autenticazione, ecc) ed a soluzioni grafiche native (form, grid, pulsanti di upload ed altri widget)
- si autoadatta alle form di avanzamento di workflow disegnate tramite l'editor visuale TWE
- interagisce con CMDBuild tramite il webservice REST
- è in grado di raccogliere dati da database di altre applicazioni permettendo quindi la gestione di soluzioni miste

Un esempio di implementazione basata sul CMDBuild GUI Framework è quello del portale Self-Service facente parte della versione preconfigurata CMDBuild READY2USE.

Il portale Self-Service di CMDBuild READY2USE consente agli utenti non informatici di interagire con il personale IT per segnalare le proprie necessità e rimanere poi aggiornati sulle attività di risoluzione.

Ogni utente accederà al portale previa autenticazione locale o collegata al repository Active Directory aziendale.

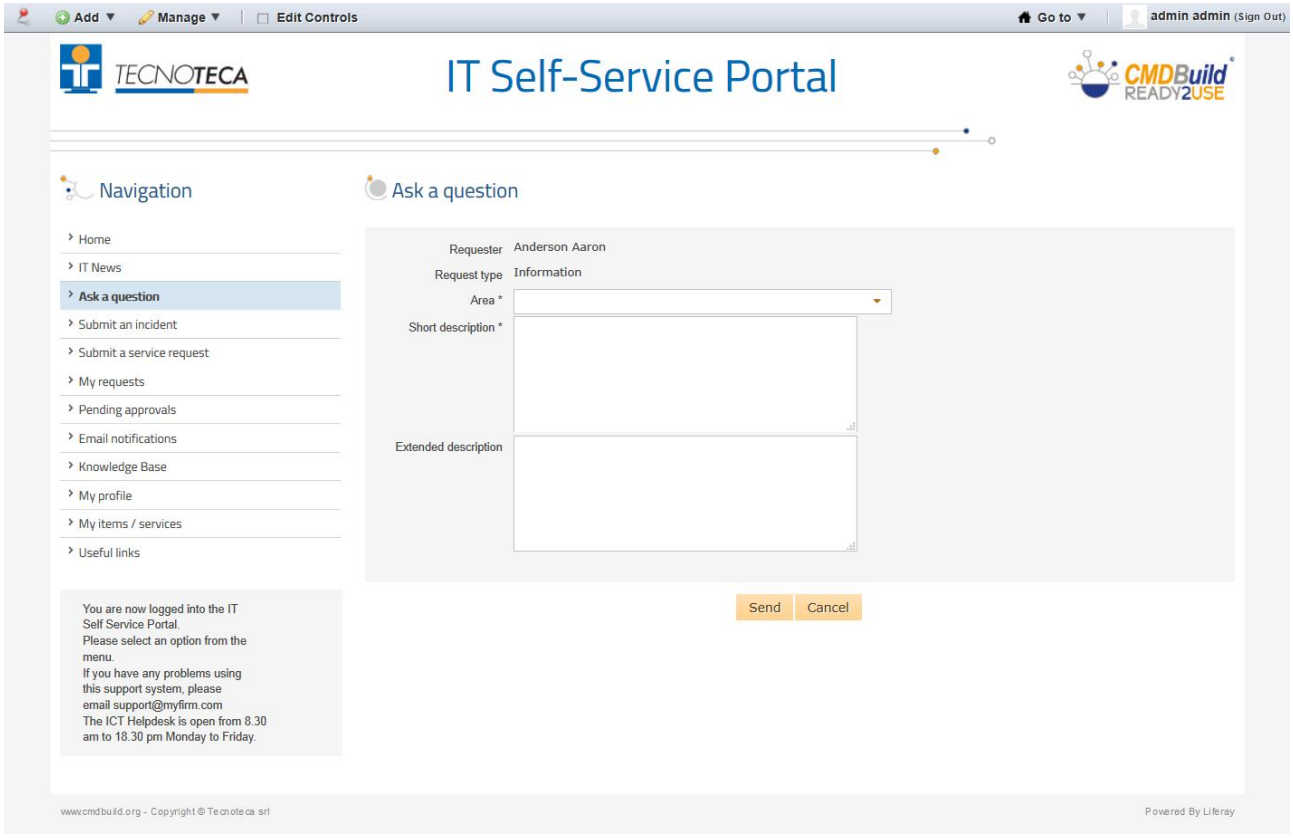
La home page del portale include:

- un menu completo, sulla sinistra
- un accesso veloce alle funzionalità principali, in alto al centro
- le notizie IT più recenti
- lo stato di avanzamento delle ultime richieste inoltrate

CMDBuild propone una implementazione del portale funzionante come portlet nel portale open source Liferay. Il CMDBuild GUI Framework utilizzato è però attivabile in portali basati su tecnologie diverse, in quanto sviluppato in ambiente javascript / JQuery. E' quindi possibile richiedere implementazioni personalizzate del portale self-service funzionanti su portali diversi.

Il portale Self-Service di CMDBuild READY2USE implementa le seguenti funzionalità:

- pubblicazione di notizie IT
- richiesta di informazioni tecniche
- apertura di una segnalazione di guasto IT
- richiesta di un servizio IT, scelto dal catalogo dei servizi
- consultazione dello stato di avanzamento delle proprie richieste
- approvazione delle richieste di autorizzazione di competenza dell'utente corrente
- FAQ
- riepilogo delle email di notifica ricevute
- profilo dell'utente collegato
- lista degli asset e dei servizi assegnati all'utente collegato
- link utili



L'utilizzo del CMDBuild GUI Framework non è l'unica alternativa possibile.

E' anche possibile implementare da zero delle interfacce web esterne, nel linguaggio di programmazione preferito ed interagendo con CMDBuild direttamente tramite i suoi webservice REST e SOAP.

Si tratta però di una soluzione meno efficace rispetto al riutilizzo del GUI Framework già disponibile.

Esempio di configurazione di un nuovo processo

Generalità

Il processo scelto per descrivere i diversi passaggi necessari per la sua configurazione è un processo semplificato di Richiesta di Modifica (Request for Change o RfC).

Si sottolinea che si tratta di un processo estremamente semplificato e modellato esclusivamente a fini didattici, proposto per comprenderne le modalità di configurazione e non per un utilizzo effettivo in ambiente di produzione.

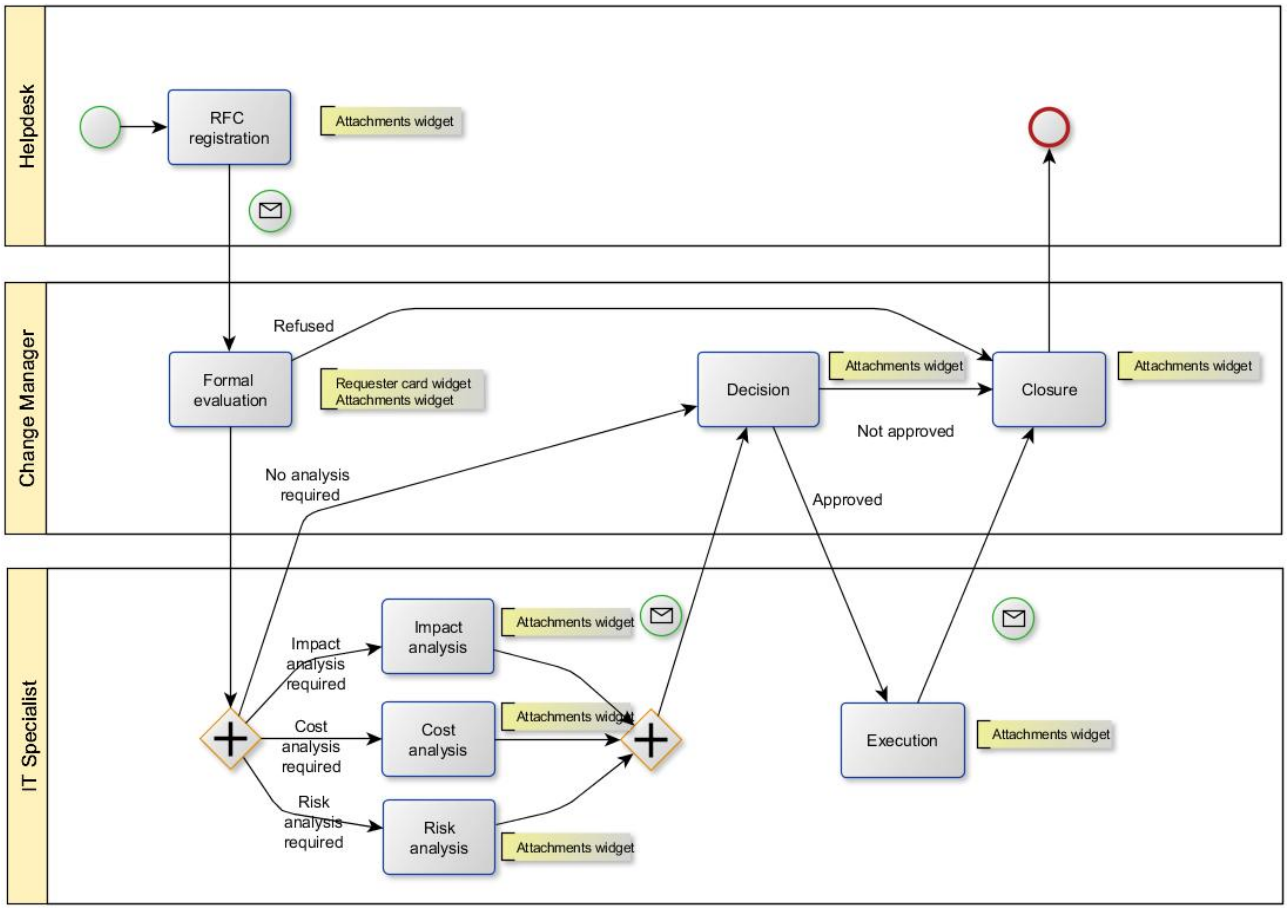
Il processo di esempio, completo della definizione in CMDBuild e del flusso XPDL disegnato con TWE, è disponibile nel database demo fornito assieme a CMDBuild.

Descrizione del processo RfC di esempio

Gli attori del processo sono i gruppi di utenti:

- Helpdesk, che esegue la registrazione iniziale della richiesta pervenuta da un utente
- Change Manager, responsabile delle modifiche apportate agli asset IT aziendali
- Specialista IT, coinvolto per la produzione di documenti di analisi e per la esecuzione della modifica

Segue uno schema logico del processo:

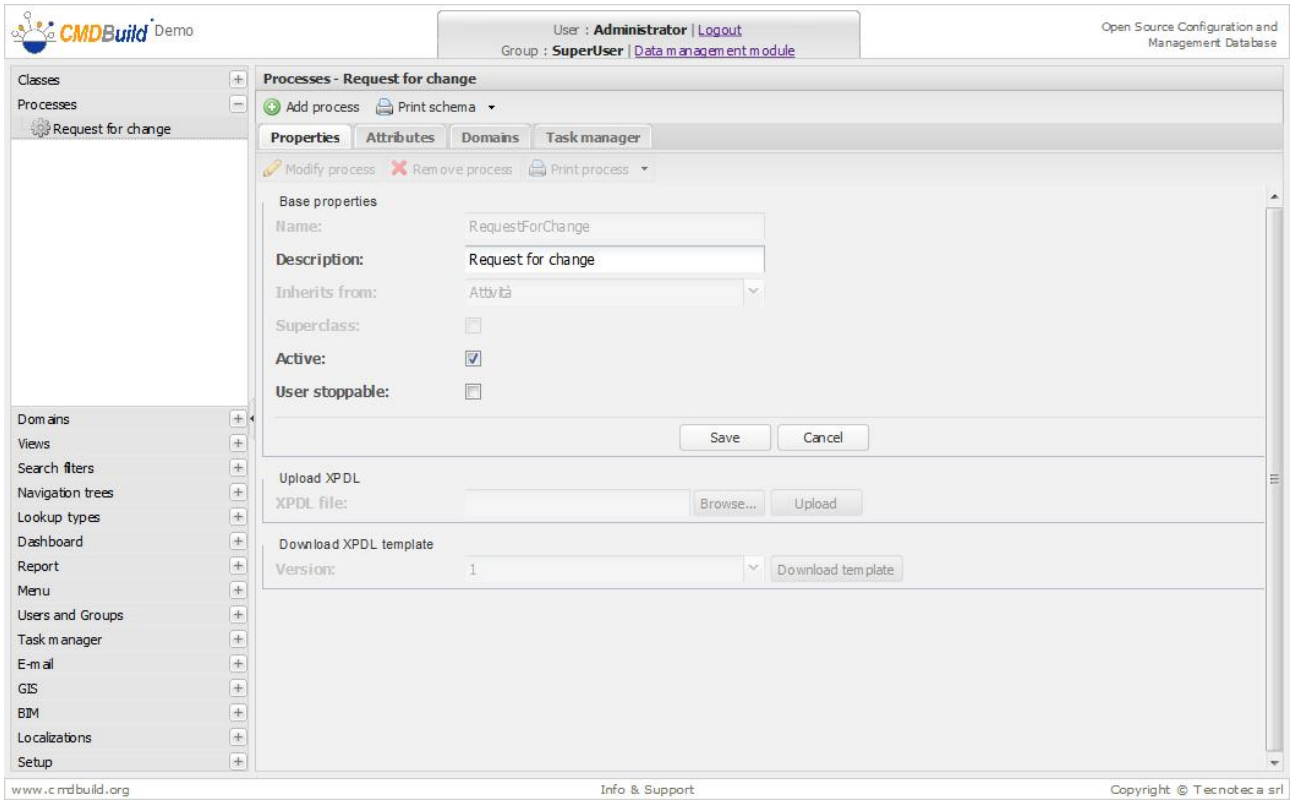


Il processo prevede le seguenti attività:

- registrazione della RfC
- valutazione degli aspetti formali della richiesta e:
  - chiusura diretta se la RfC non è accettabile
  - passaggio alla fase decisionale se non sono richieste attività di analisi
  - richiesta di esecuzione di una o più tipologie di analisi, fra analisi di impatto, analisi di costo, analisi del rischio
- esecuzione delle tipologie di analisi richieste (analisi di impatto, di costo, del rischio)
- decisione del Change Manager, con eventuale chiusura diretta se la RfC non è approvata
- esecuzione della RfC da parte di uno specialista IT, se la RfC è stata approvata
- chiusura finale

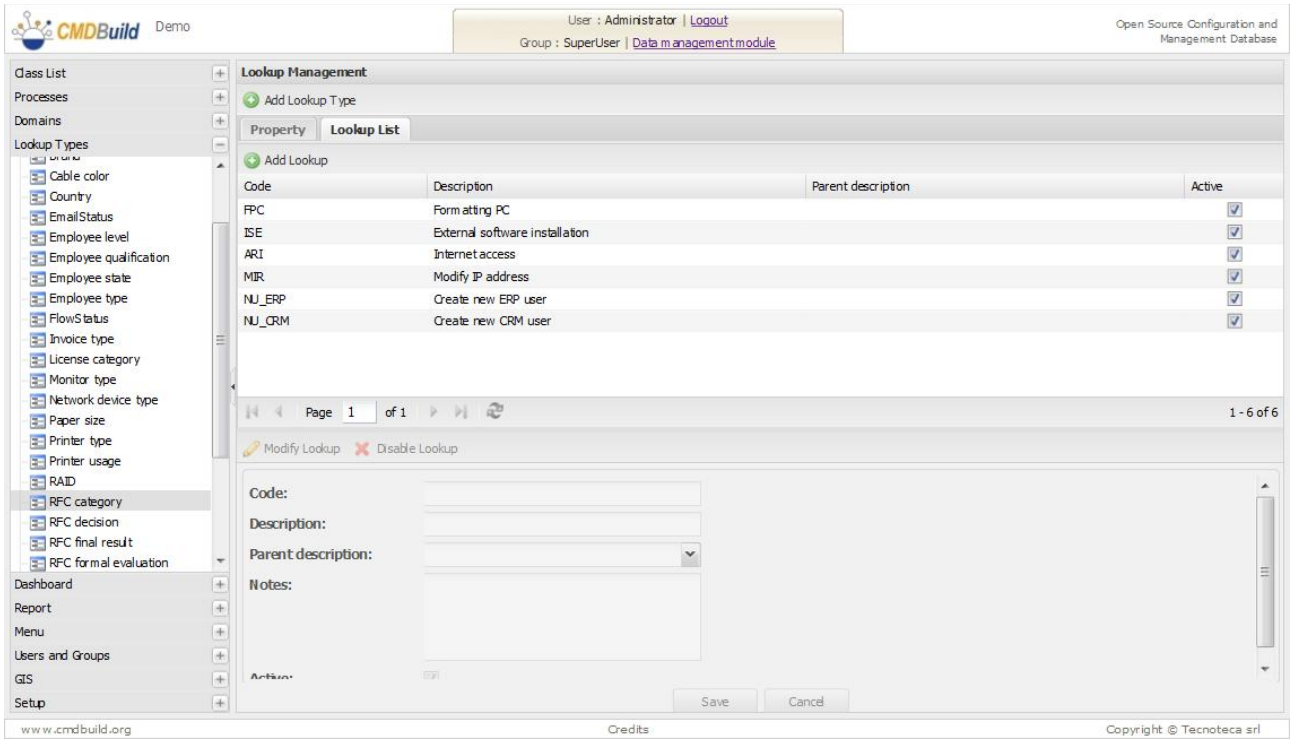
Fase 1 – Creazione oggetti in CMDBuild

Per la gestione del workflow viene creato tramite il Modulo di Amministrazione, alla voce Processi del Menu, il processo RequestForChange:

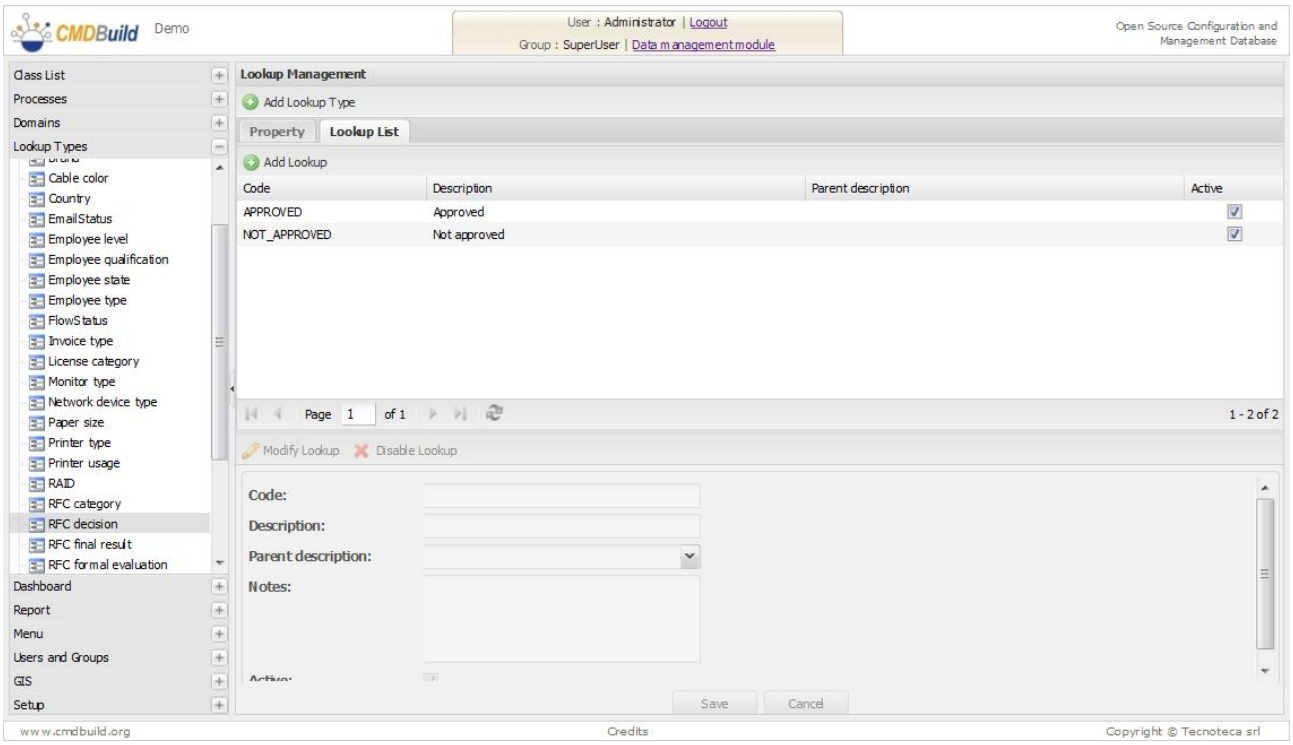


Alcuni degli attributi previsti nel processo sono di tipo Lookup e richiedono quindi la preventiva definizione delle liste corrispondenti, come evidenziato agli screenshot successivi.

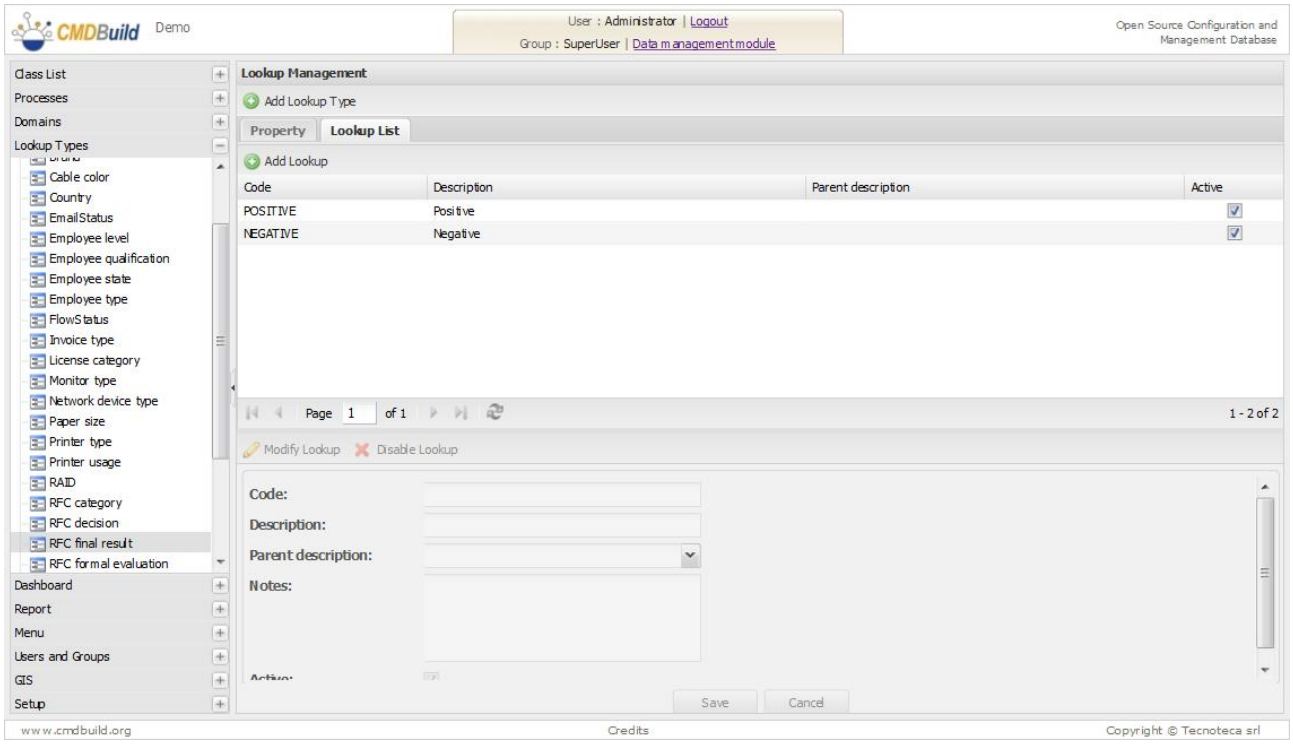
Lookup RFC category (collegata poi all'attributo “Category” del processo)



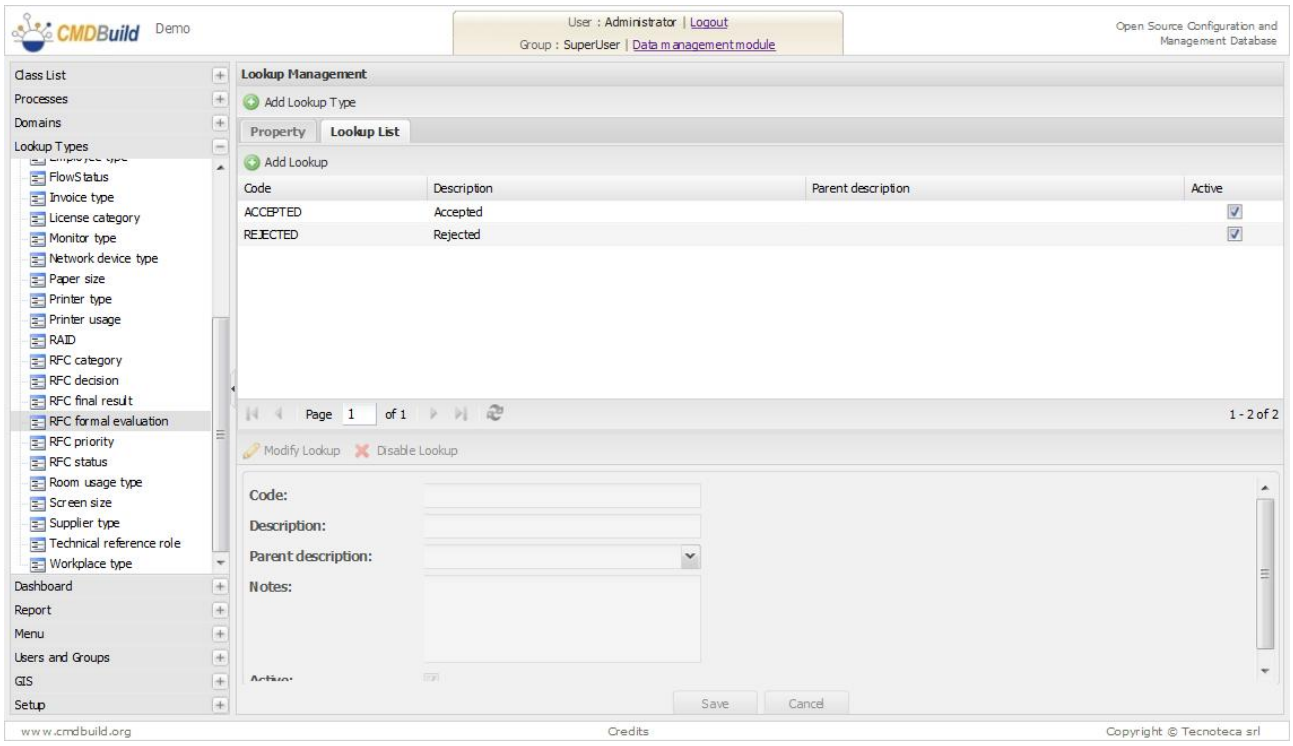
Lookup RFC decision (collegata poi all'attributo “Decision” del processo)



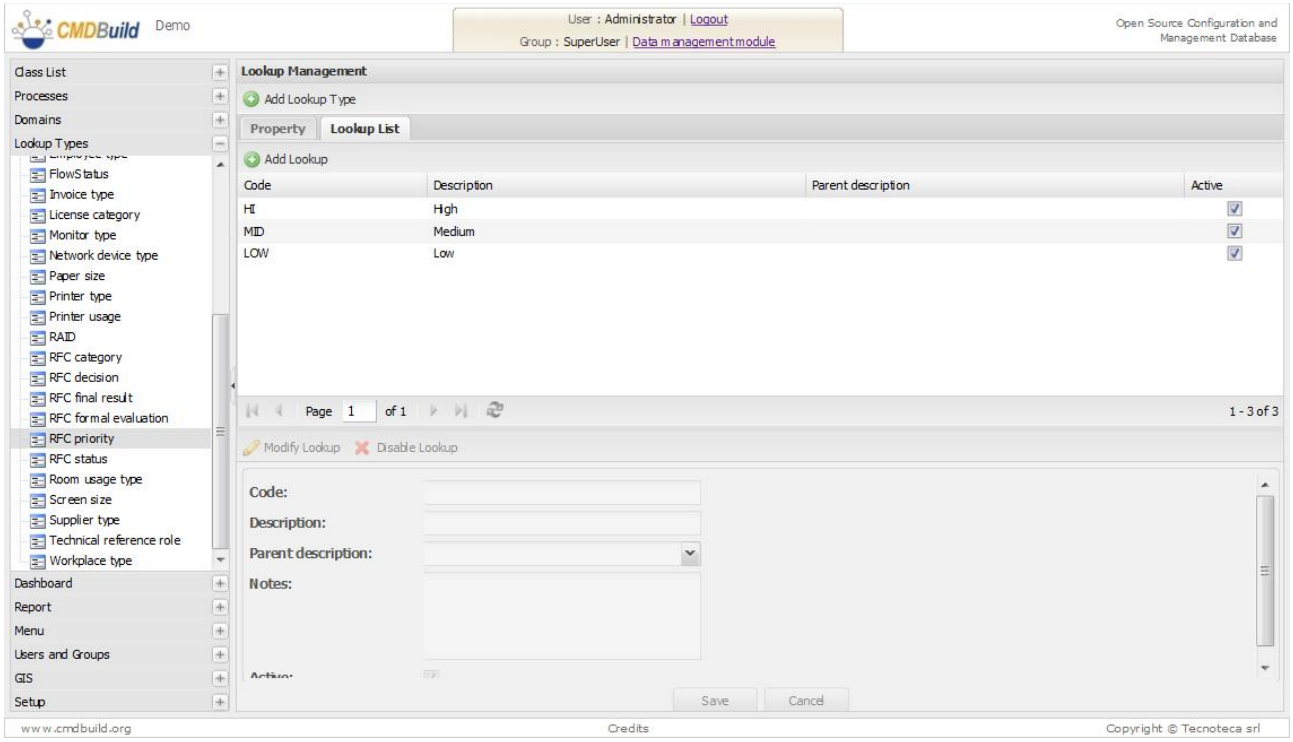
Lookup RFC final result (collegata poi all'attributo “FinalResult” del processo)



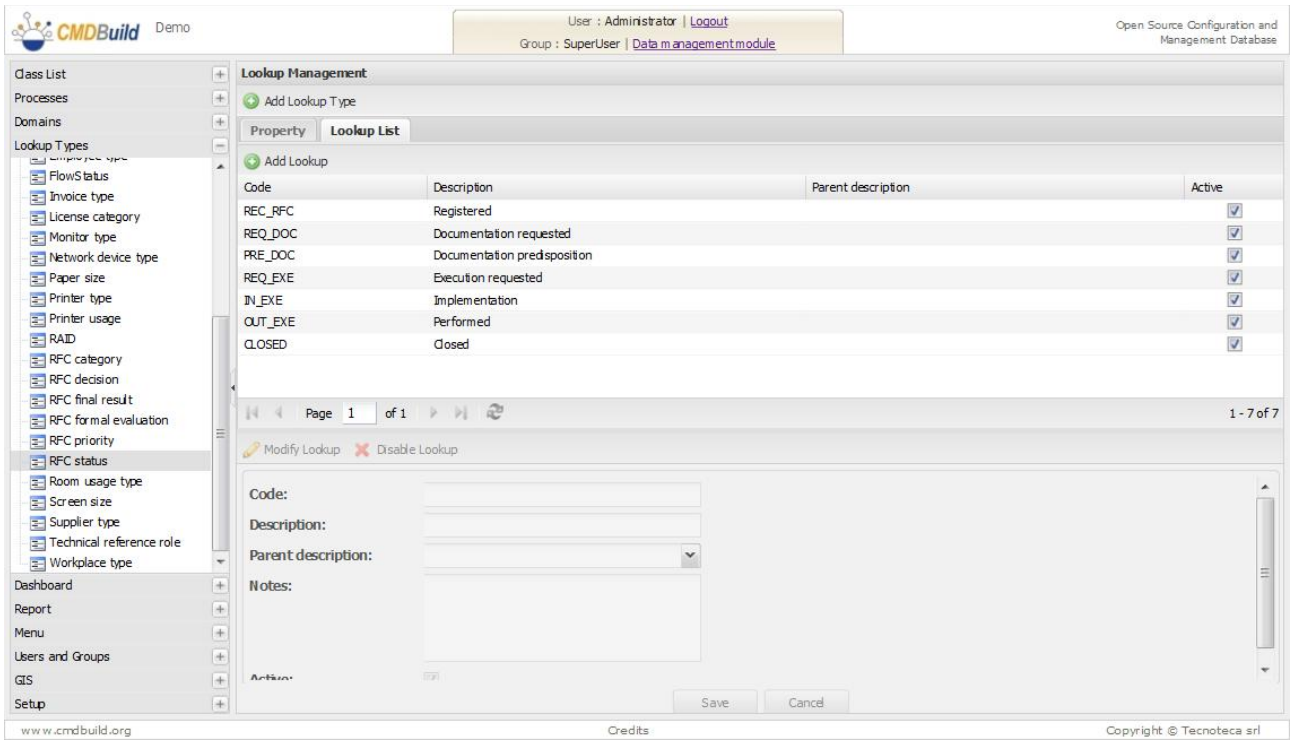
Lookup RFC formal evaluation (collegata poi all'attributo “FormalEvaluation” del processo)



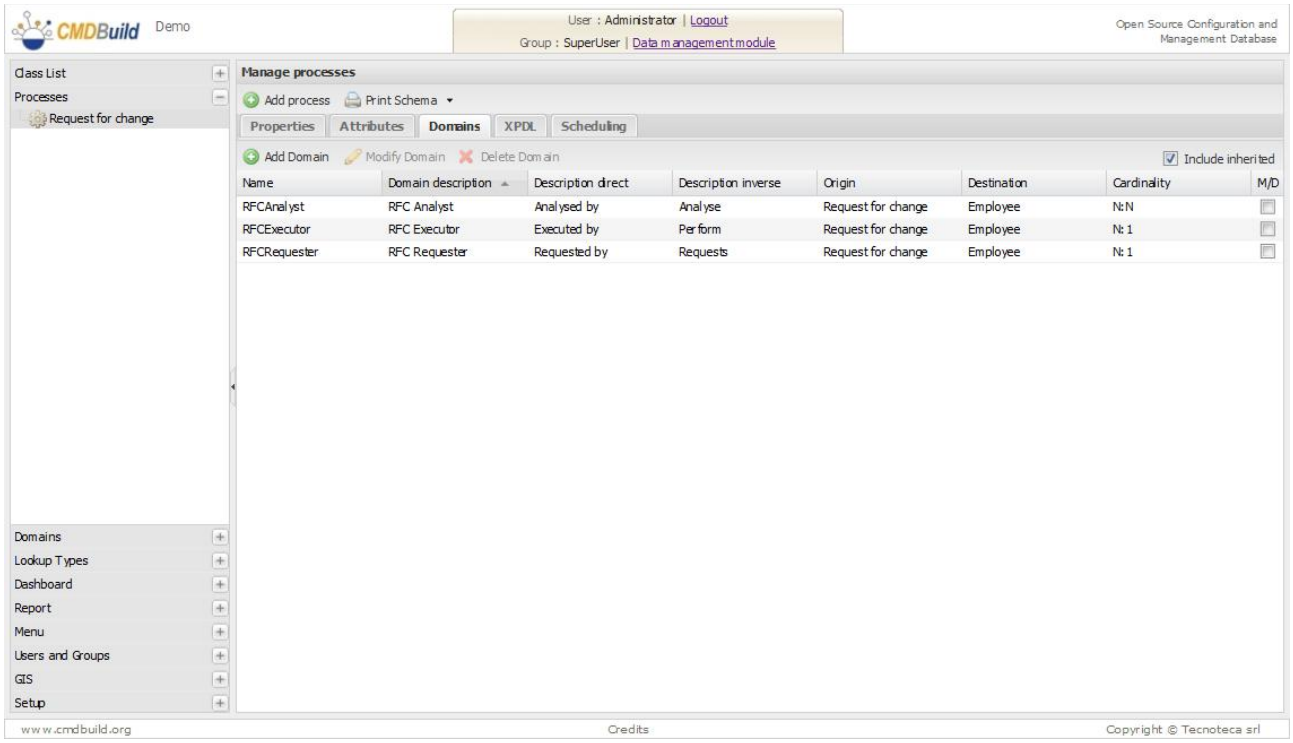
Lookup RFC priority (collegata poi all'attributo “RFCPriority” del processo)



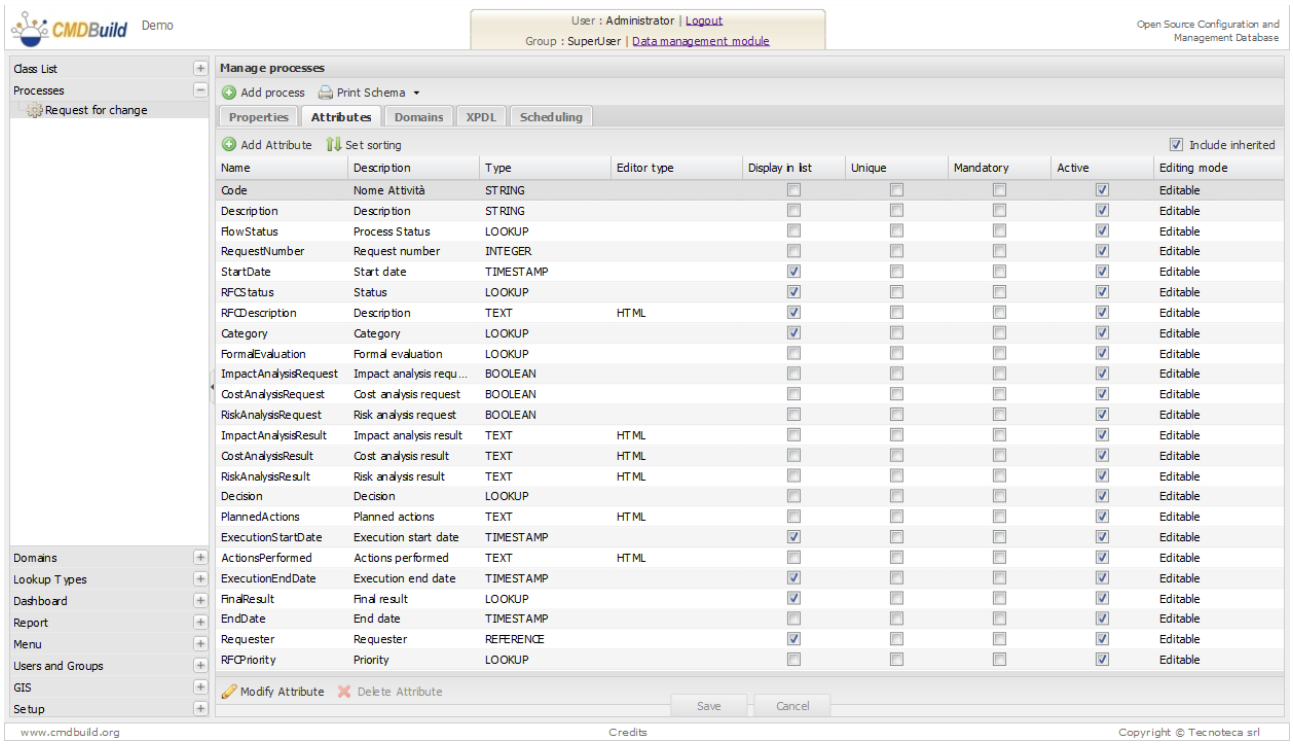
Lookup RFC status (collegata poi all'attributo “RFCStatus” del processo)



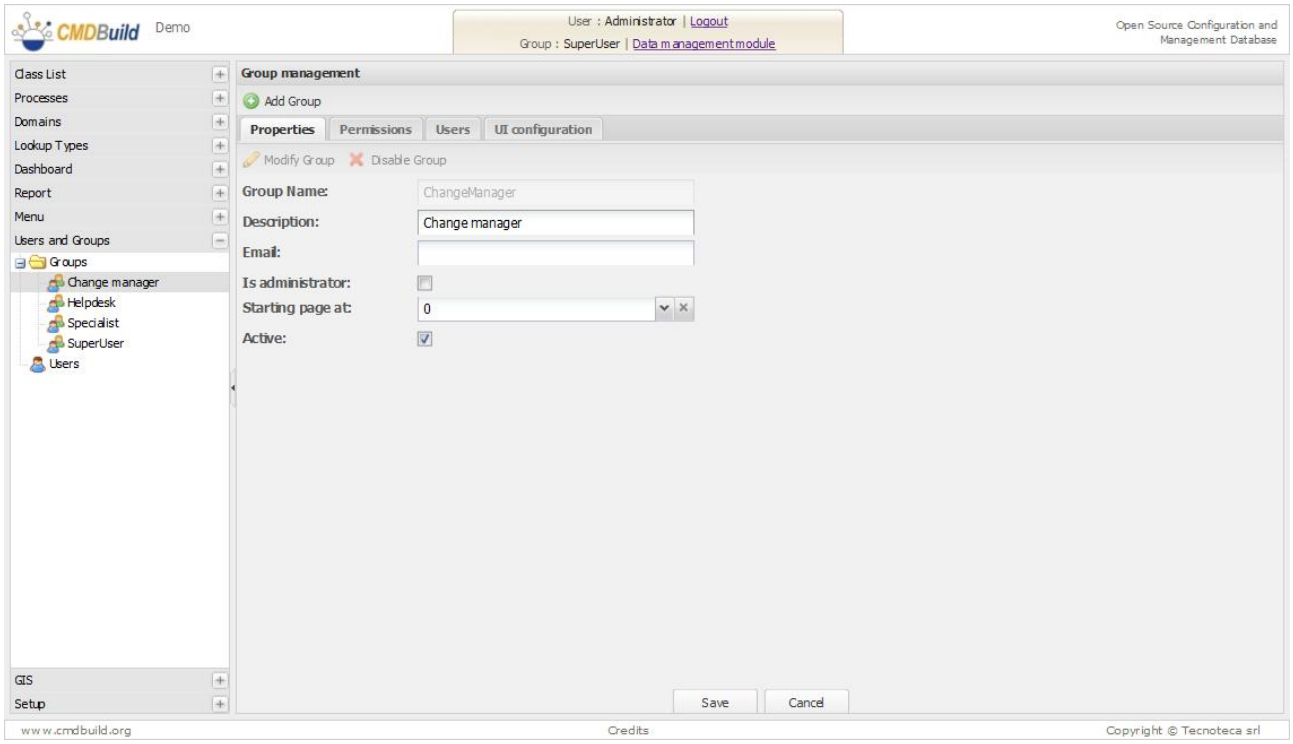
Per poter definire nel processo gli attributi “Requestor” come foreign key sulla classe “Employee” e le relazioni con il Change Manager e con gli specialisti IT che rispettivamente valutano ed eseguono la RfC vengono creati i seguenti “domini”:



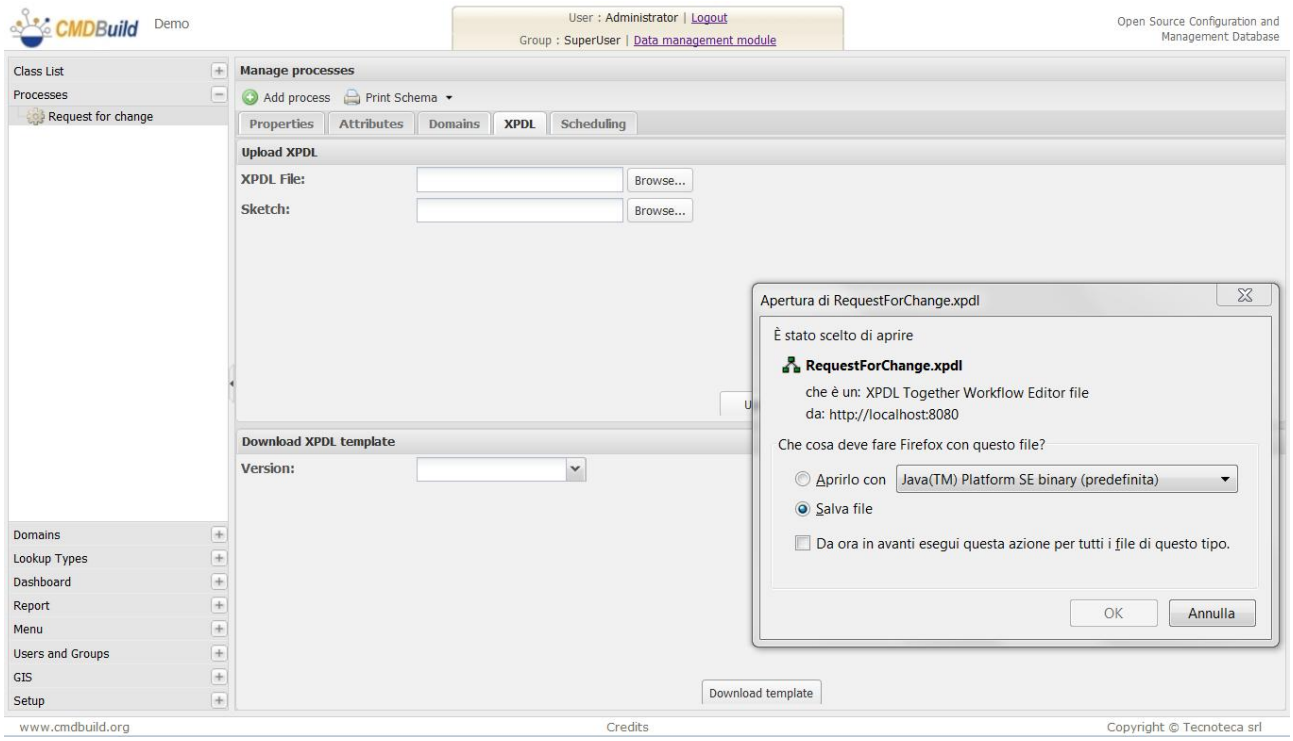
A questo punto possono essere finalmente creati gli attributi del processo:



Da ultimo devono essere creati I gruppi di utenti coinvolti nel workflow:



A questo punto può essere esportato lo “scheletro” XPDL prodotto da CMDBuild, che sarà utilizzato con l'editor visuale TWE per disegnare il flusso di dettaglio del processo stesso:



Il file XPDL prodotto conterrà solamente i dati generali al momento disponibili:

- il nome del processo
- la lista degli attributi non riservati presenti nella classe di gestione del processo
- la lista dei ruoli definiti nel sistema

Tali dati costituiranno il punto di partenza delle attività svolte poi tramite l'editor TWE, nel cui ambito saranno poi arricchiti di tutti gli aspetti relativi al flusso specifico del processo.

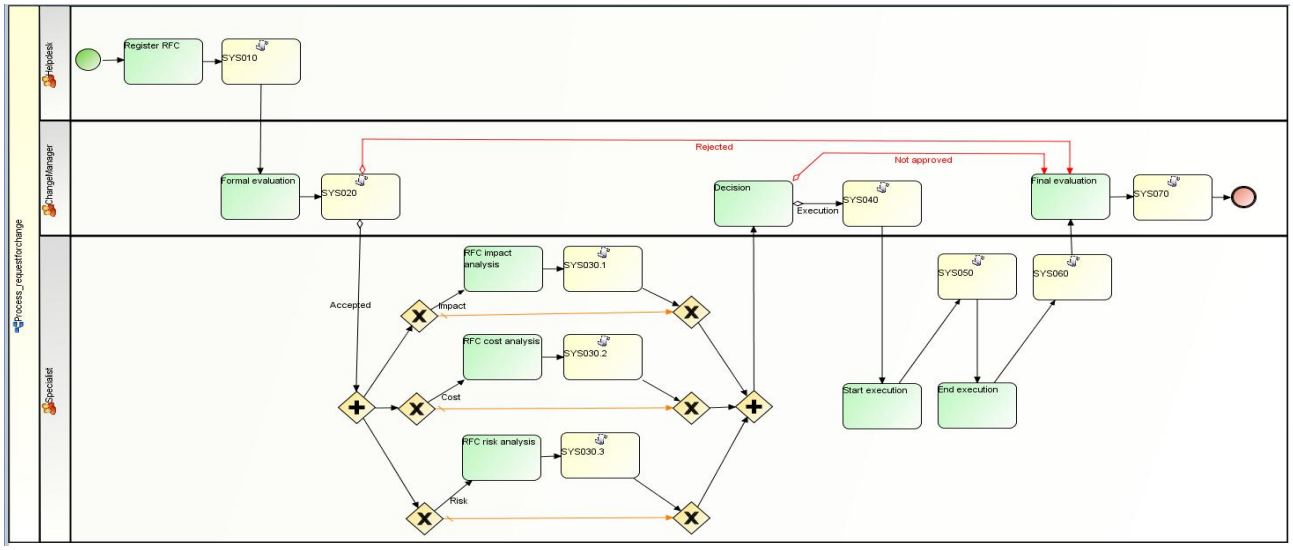
Fase 2 – Configurazione del flusso con TWE

Attraverso l'editor TWE vengono eseguite le seguenti operazioni:

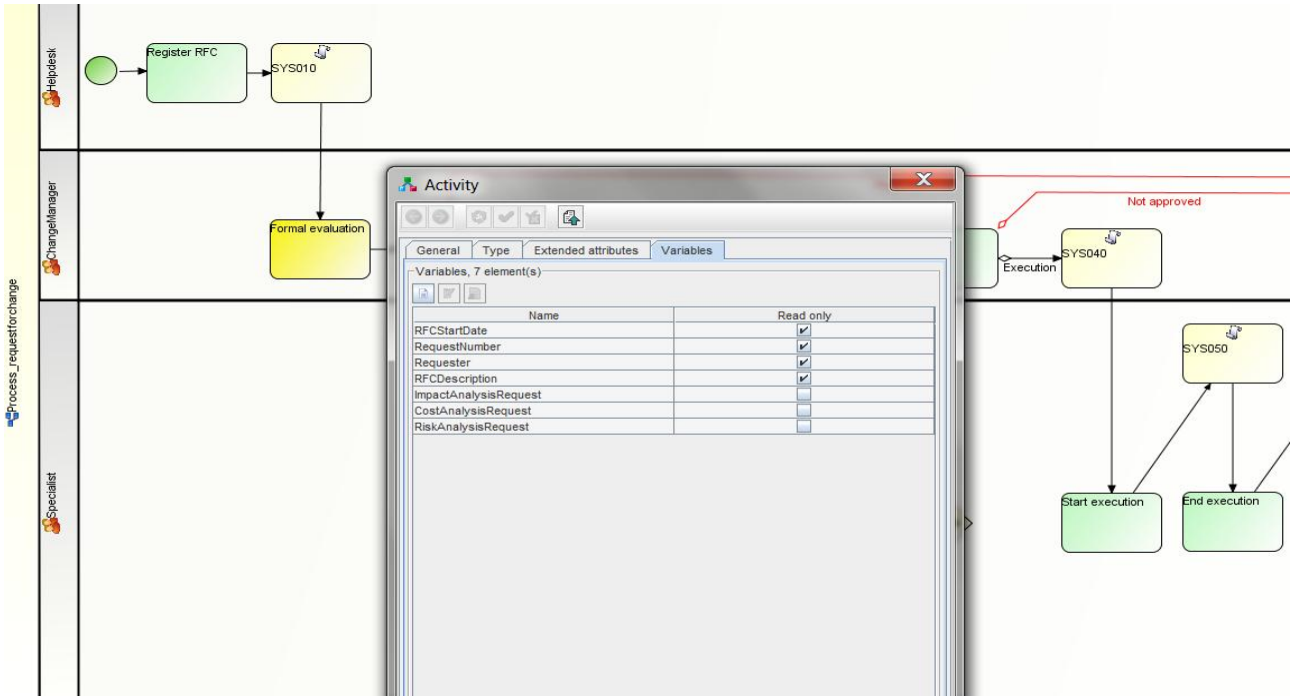
- disegno del flusso, con posizionamento delle attività delle diverse tipologie previste (inizio processo, fine processo, attività utente, attività automatiche, attività di routing per la gestione del parallelismo) e collegamento delle stesse in base alle tipologie di transizioni previste
- completamento delle attività utente, specificando quali attributi del processo dovranno essere mostrati nella form corrispondente a quell'attività (con indicazione se in sola lettura o anche in scrittura) e quali widget dovranno essere resi disponibili nella stessa form (con indicazione dei parametri previsti da ciascuno)
- completamento delle attività automatiche, scrivendo lo script che implementa gli automatismi richiesti in quell'attività (tramite utilizzo delle API messe a disposizione a tale scopo)
- completamento delle transizioni fra attività, specificando i criteri sulla cui base il flusso dovrà percorrere una transizione o un'altra, nel caso di scelte condizionate

Seguono alcuni screenshot descrittivi delle attività sopra elencate.

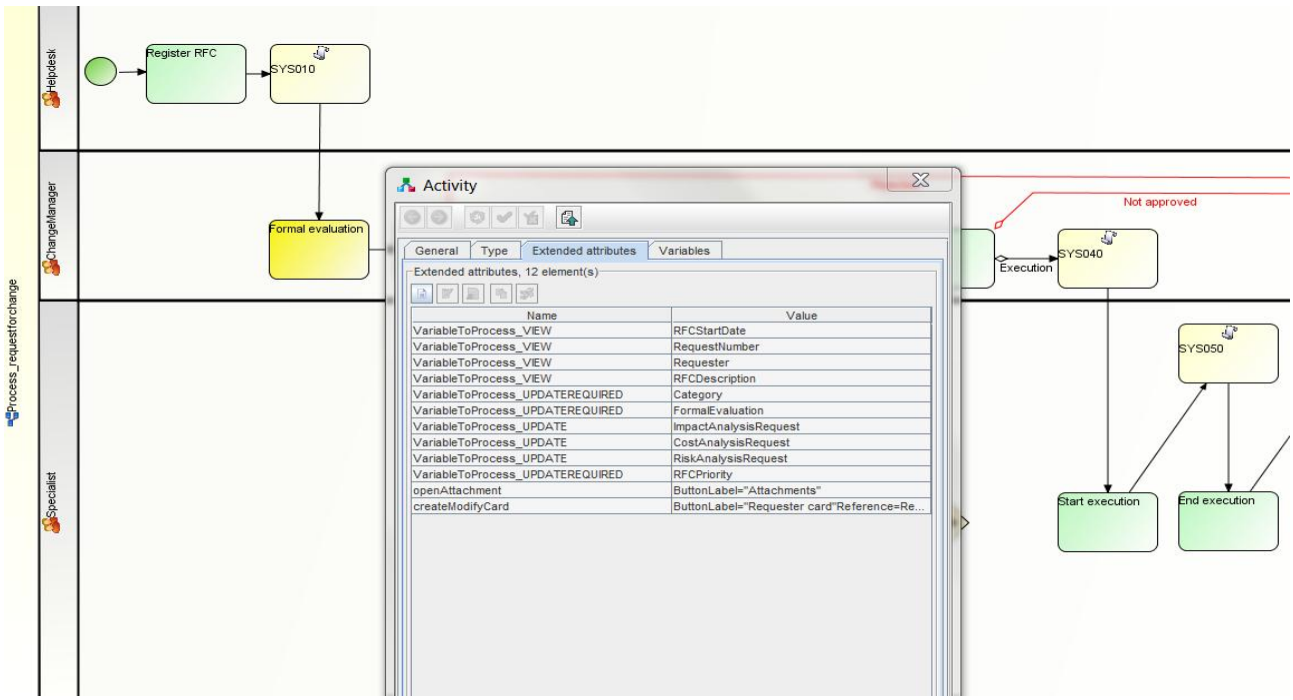
<u>Disegno complessivo del flusso</u>:



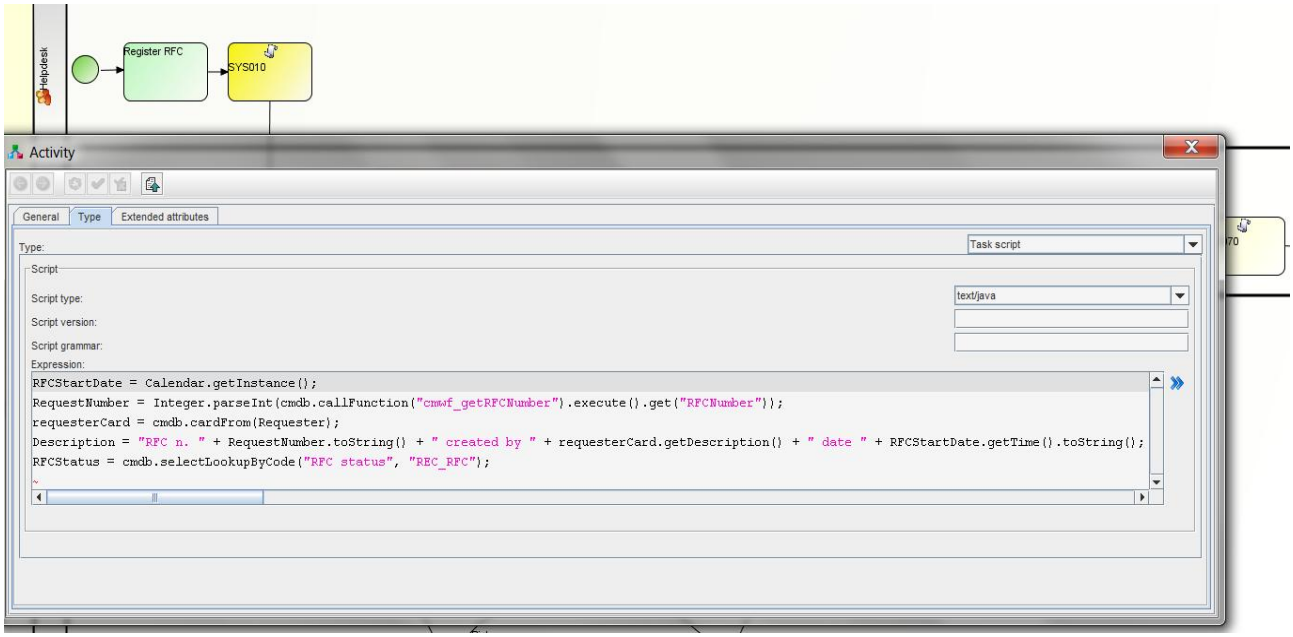
Attività utente - TAB “Variables”, da utilizzare per la scelta degli attributi che devono comparire nella form (con eventuale indicazione della modalità di sola lettura):



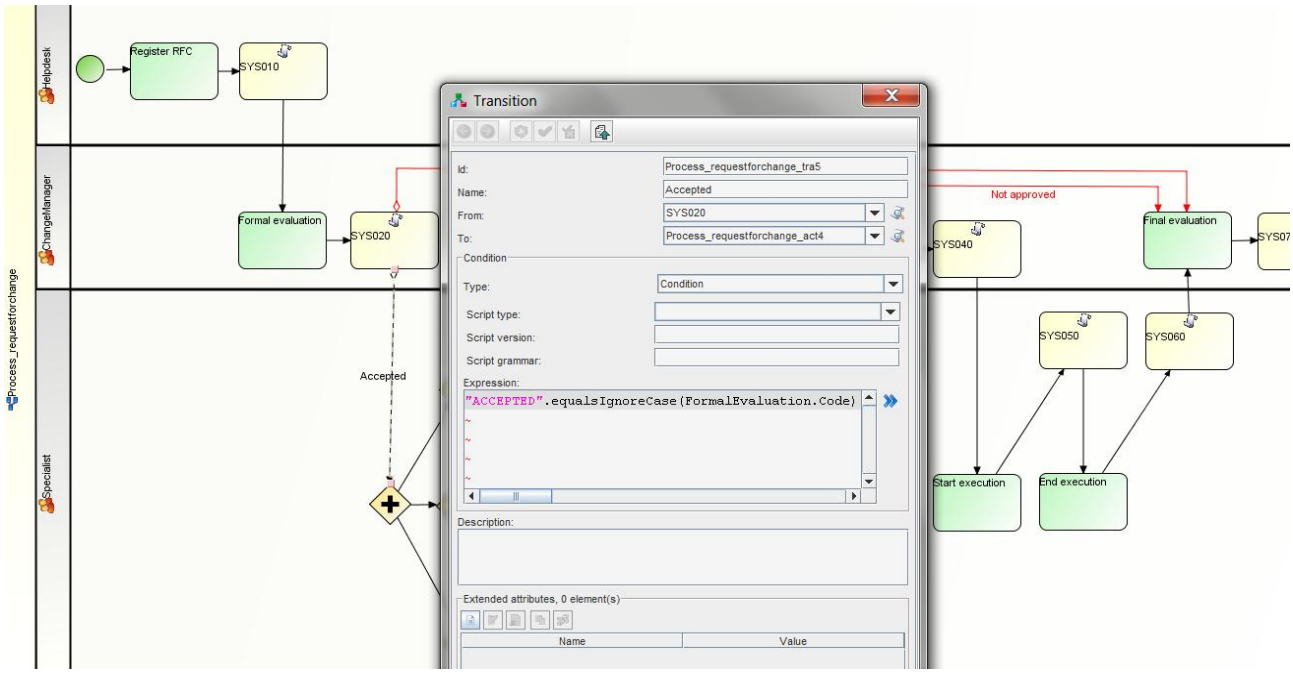
Attività utente - TAB “Extended Attributes”, da utilizzare per indicare gli attributi obbligatori (“UPDATEREQUIRED”) e per richiedere l'inserimento nella form di uno o più widget (nell'esempio openAttachments per gli allegati e createModifyCard per consultare la scheda del richiedente)



Attività automatica - TAB “Type”, da utilizzare per scrivere lo script che implementa gli automatismi previsti (nell'esempio l'attività SYS010 esegue l'impostazione automatica della data di sistema, l'attribuzione automatica di un numero progressivo univoco, la costruzione di una descrizione significativa, l'impostazione del nuovo stato raggiunto dal processo).



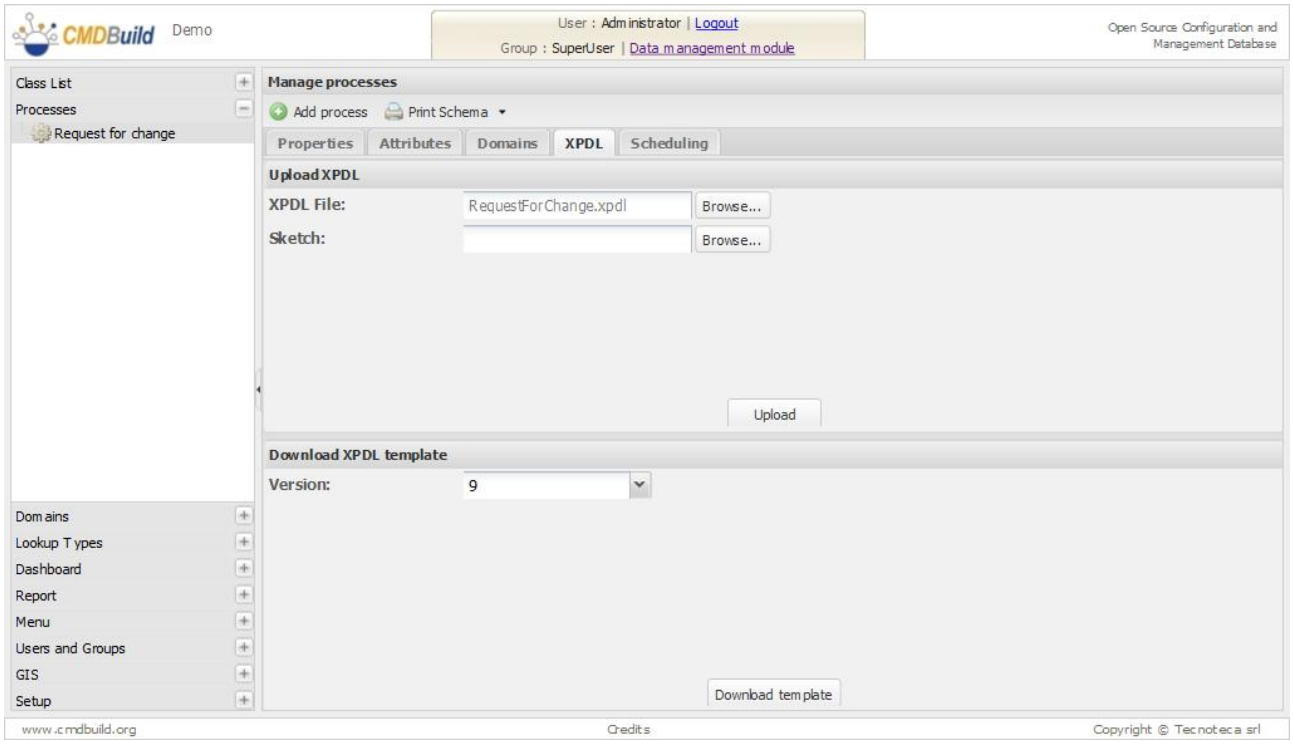
Transizione, da utilizzare per collegare due attività, in forma condizionata o meno (nell'esempio è prevista una condizione e si riferisce alla scelta di accettazione formale della RfC)



Fase 3 – Importazione in CMDBuild del file XPDL risultante

Una volta completata la configurazione del flusso del processo in TWE si dovrà caricare in CMDBuild il file XPDL corrispondente.

Il flusso di un processo può essere modificato successivamente anche più volte, semplicemente esportando la versione ultima da CMDBuild, editandola con TWE e reimportandola in CMDBuild. Va tenuto presente che la nuova versione sarà utilizzata per i nuovi processi che verranno avviati, mentre i processi già in corso proseguiranno ciascuno con la versione XPDL valida nel momento in cui sono nati.



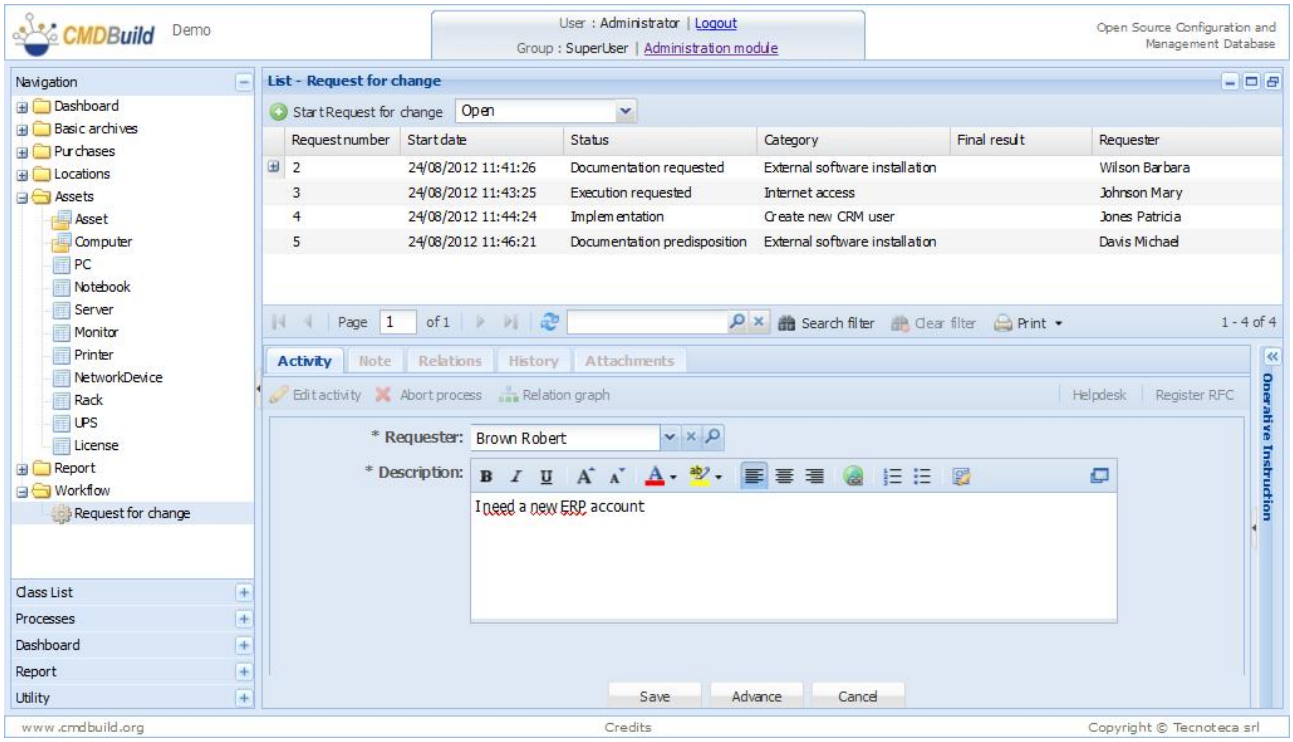
Fase 4 – Esecuzione del processo da CMDBuild

Il workflow importato in CMDBuild è ora disponibile per essere utilizzato dai gruppi di operatori previsti.

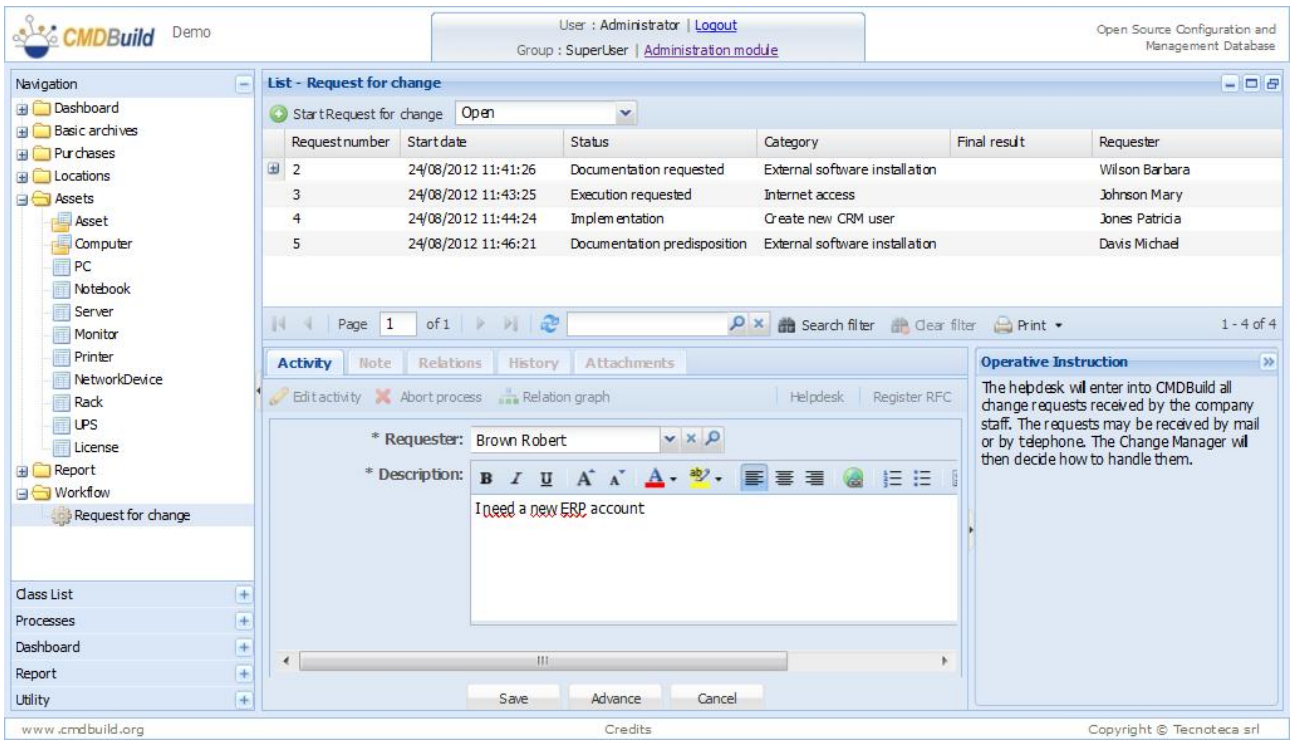
Nel caso dell'esempio il workflow di gestione delle RfC dovrà essere avviato da un operatore del gruppo Helpdesk, valutato da un operatore del gruppo Change Manager, analizzato ed eseguito da un operatore del gruppo Specialisti IT. Va considerato a tale proposito che gli operatori del gruppo SuperUser possono “impersonare” qualunque altro gruppo definito in CMDBuild.

Posizionandosi sulla gestione del processo RfC il sistema presenta le RfC aperte (o nello stato selezionato agendo sulla lista in alto: aperte, sospese, completate, abortite, tutte).

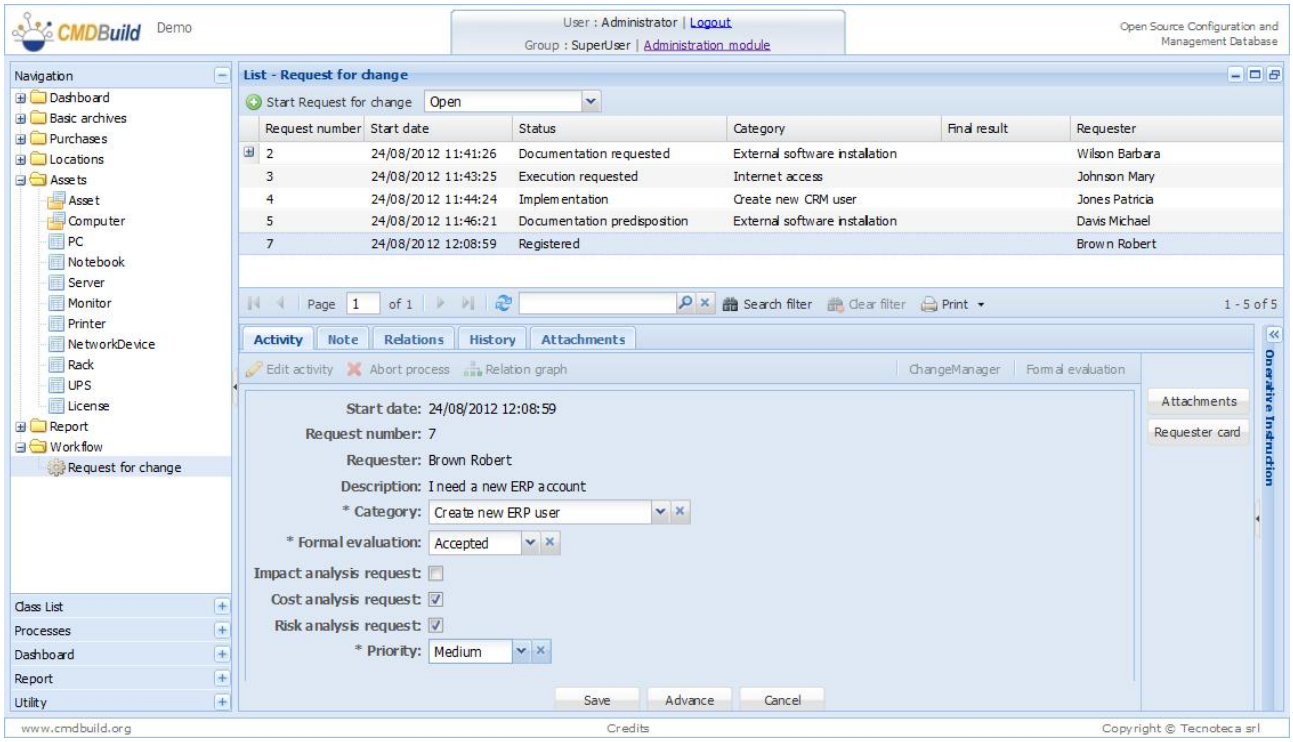
Tramite il pulsante “Start Request for Change” l'Helpdesk potrà registrare una nuova richiesta.



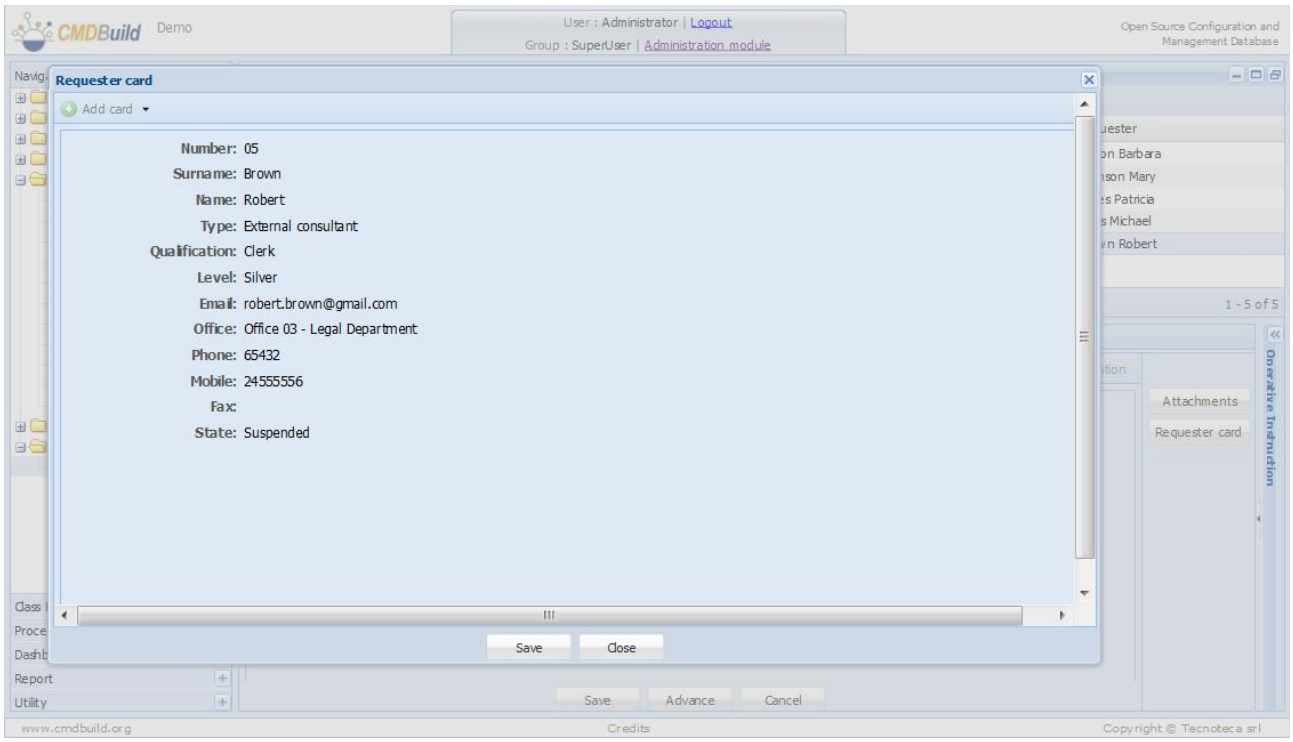
Eventualmente prima di compilare la form l'operatore può consultare le istruzioni operative associabili ad ogni attività utente (impostabile con TWE compilando il campo “Description” nel TAB “General” dell'attività).



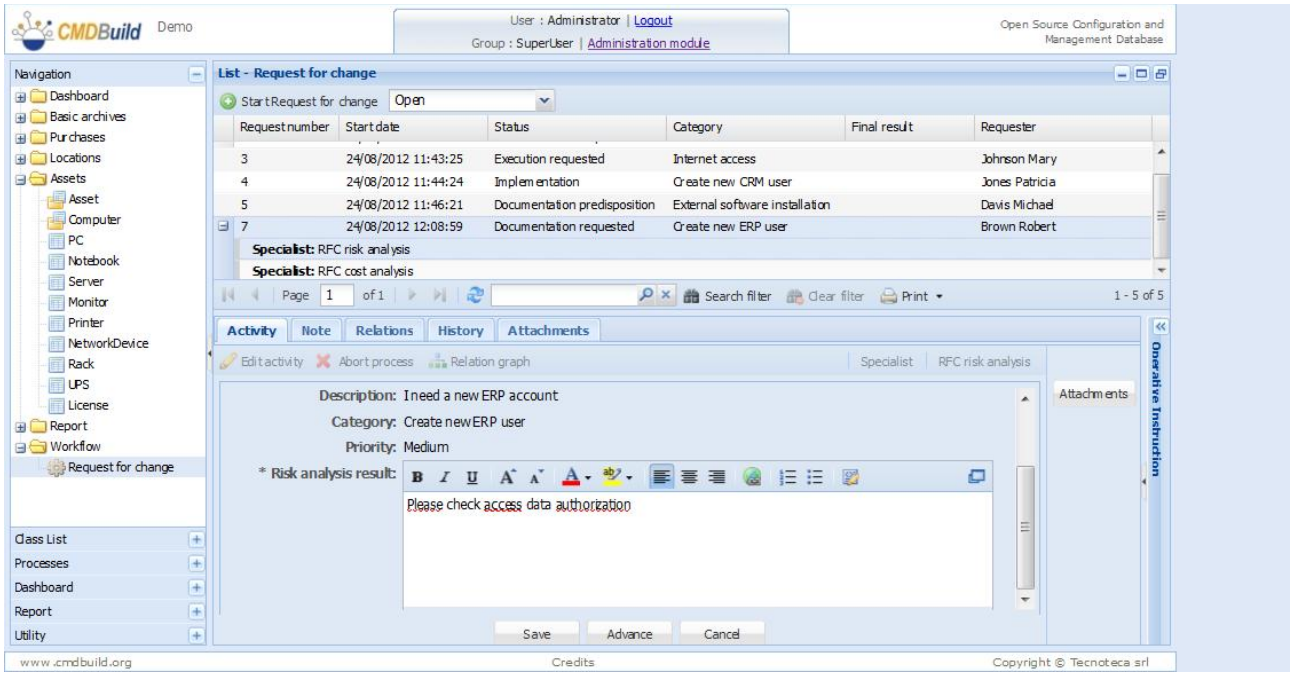
Confermando l'avanzamento al passo successivo l'attività viene presa in carico dal Change Manager che, sempre nel nostro esempio semplificato, dovrà compilare le seguenti informazioni:

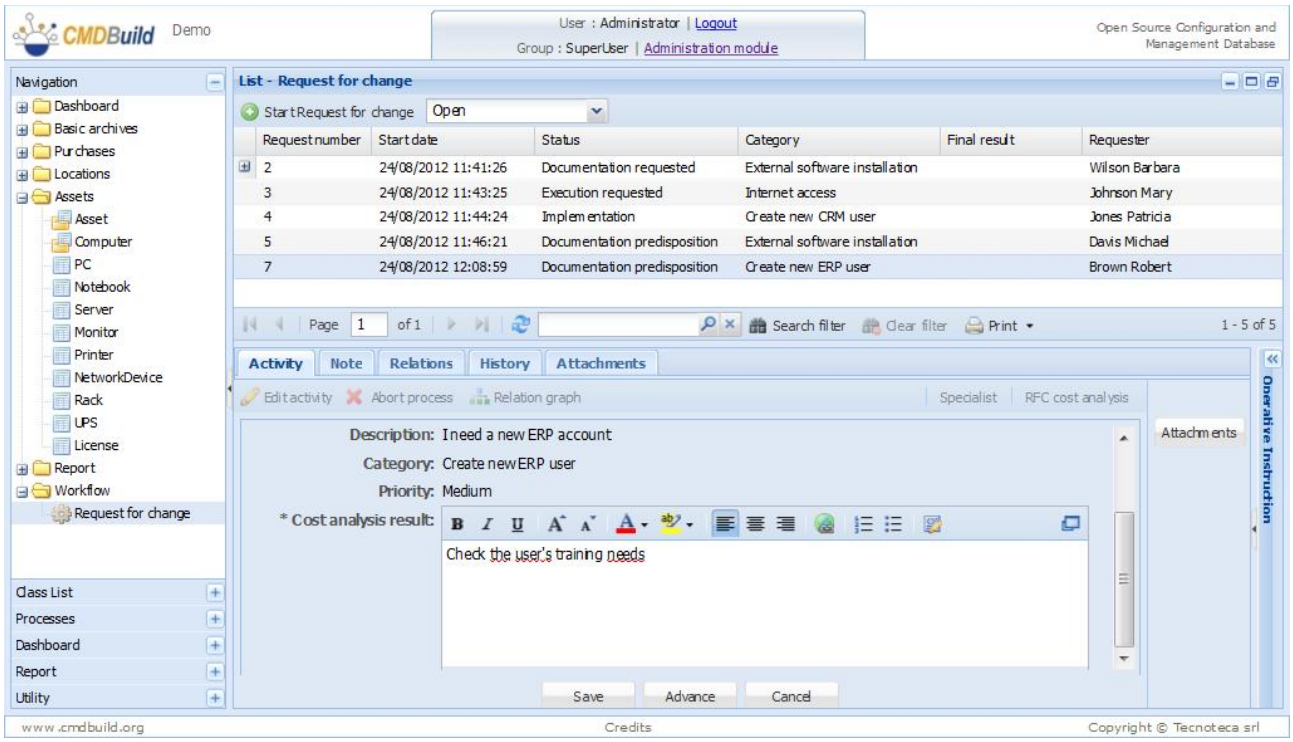


Nell'esempio è stato previsto in questo passaggio il possibile utilizzo dei widget di caricamento allegati e di consultazione della scheda completa del richiedente:

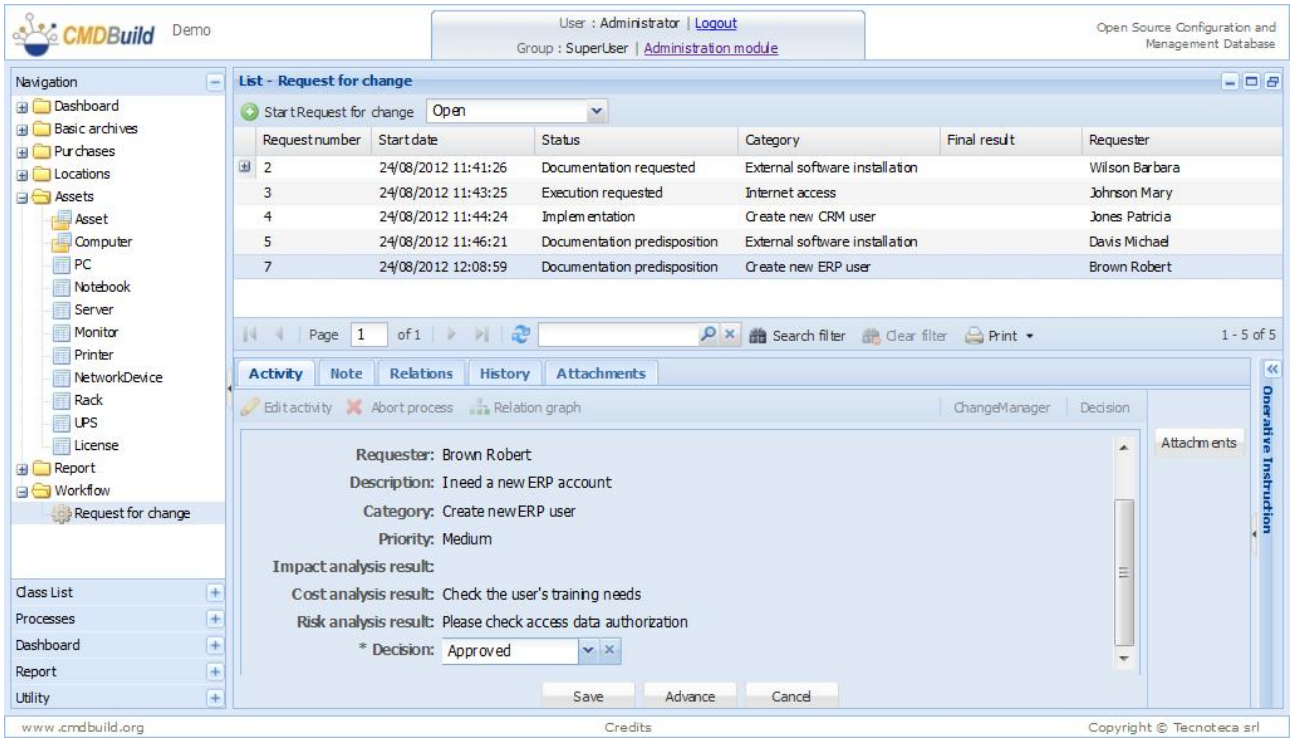


Il Change Manager ha richiesto nel nostro esempio due tipologie di analisi, per cui il workflow passa in carico agli Specialisti IT che, in parallelo (sfruttando quindi una delle nuove funzionalità implementate in CMDBuild 2.0), possono eseguire le loro analisi (di rischio e di costo rispettivamente) e riportarne i risultati.

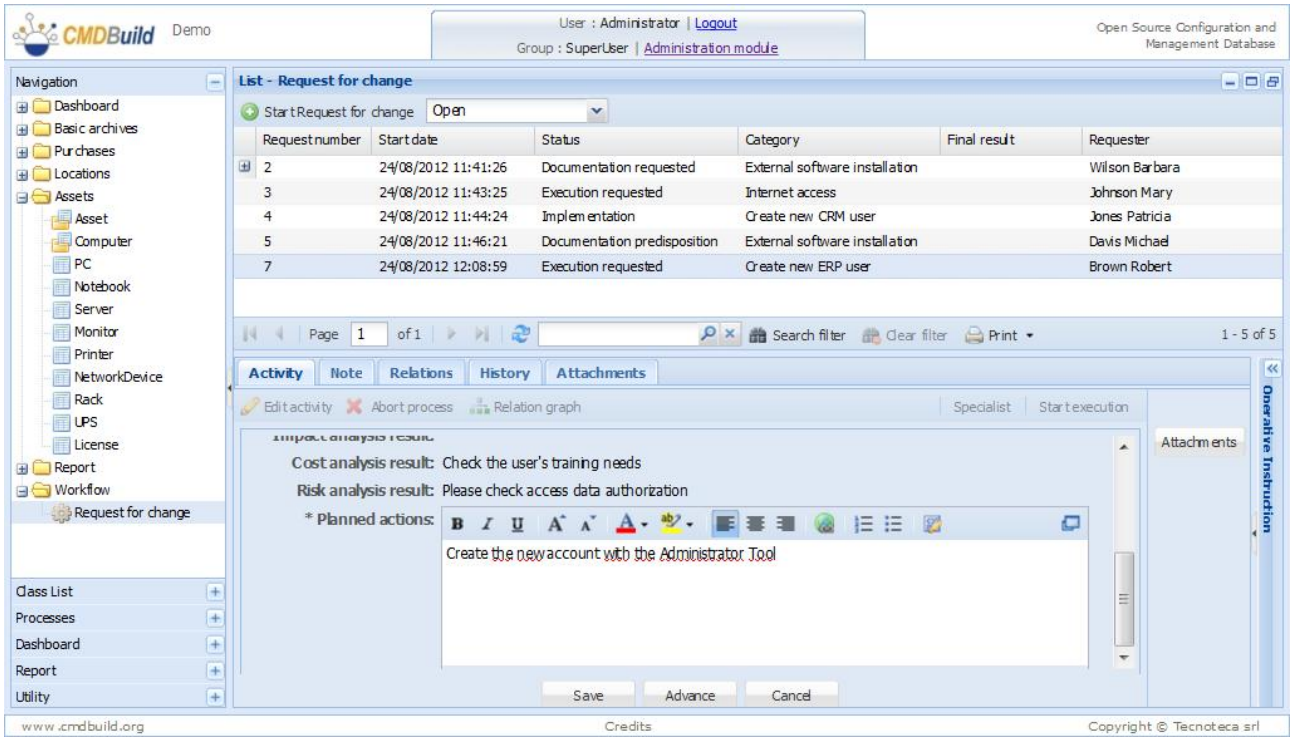


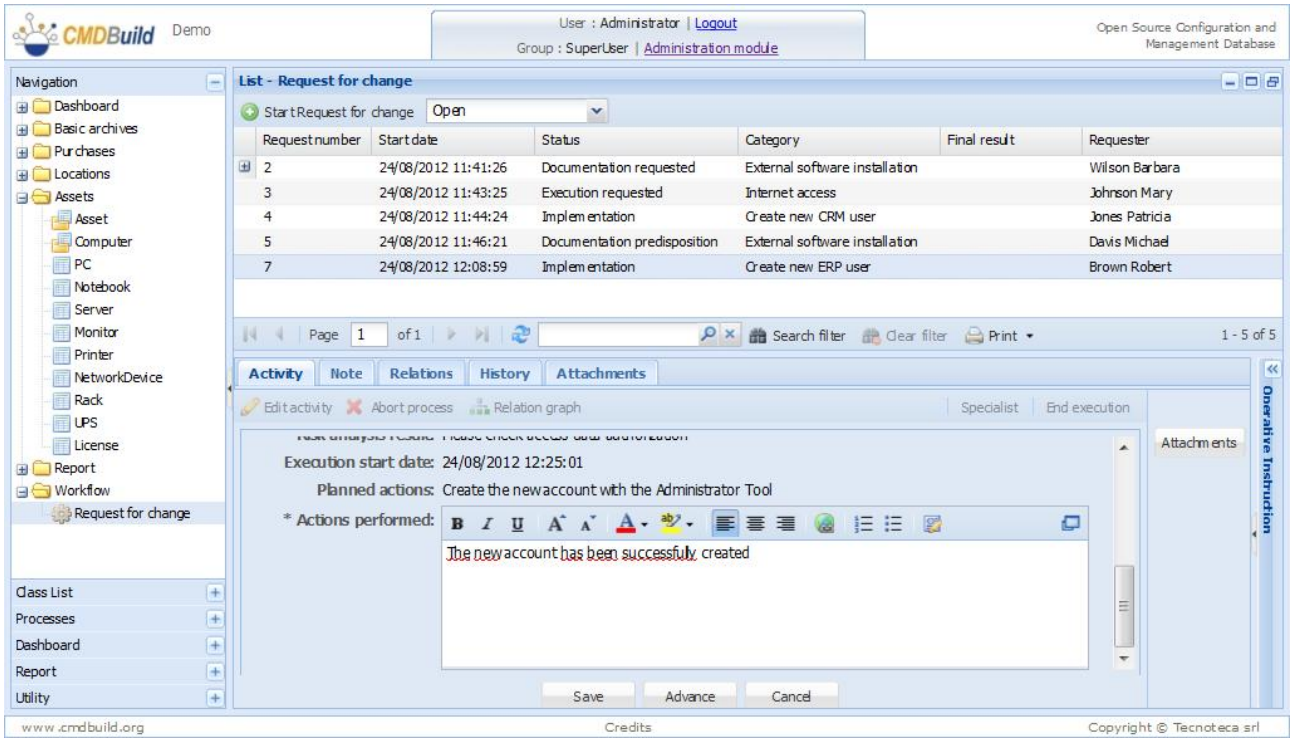


Il Change Manager dispone ora degli esiti degli approfondimenti richiesti e può prendere la sua decisione.

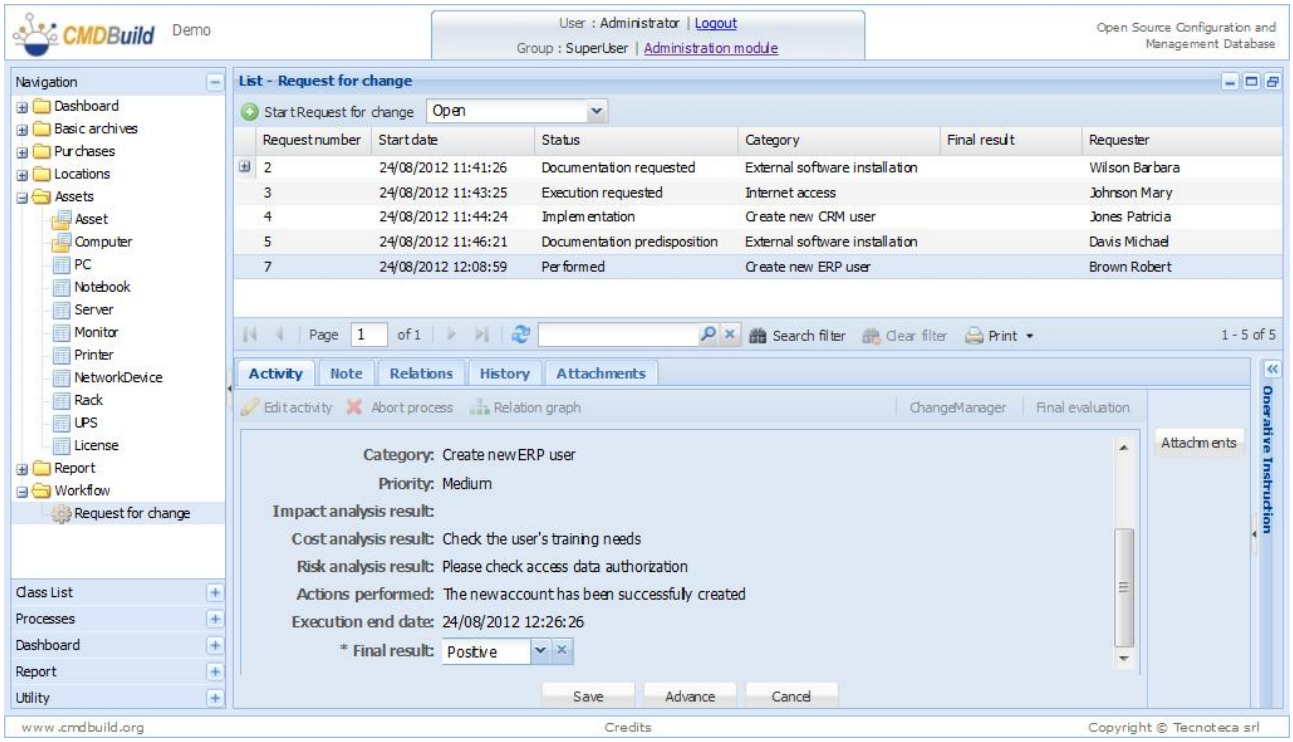


Se la decisione è positiva, sulla base del flusso disegnato con TWE, viene richiesto agli Specialisti IT di eseguire l'attività della RfC. L'esecuzione prevede la presa in carico iniziale della richiesta con indicazione delle attività da svolgere e la registrazione finale delle attività effettivamente svolte.

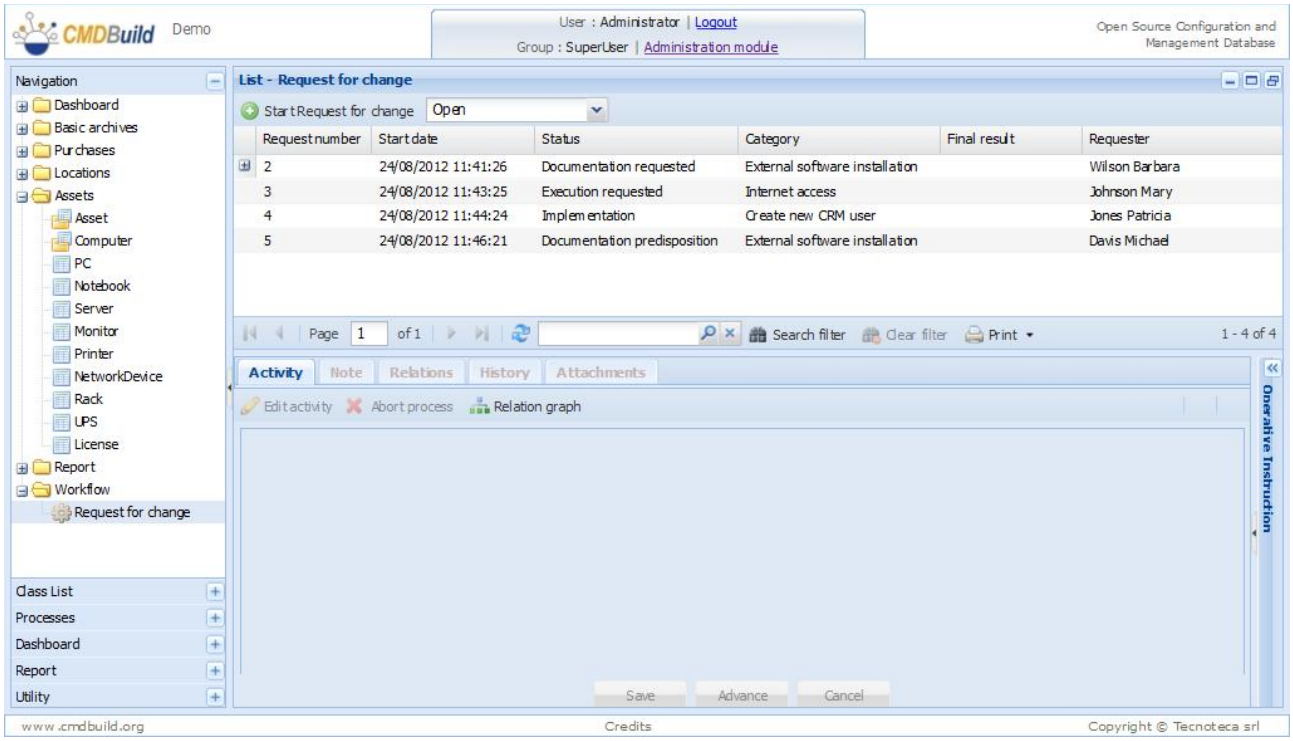




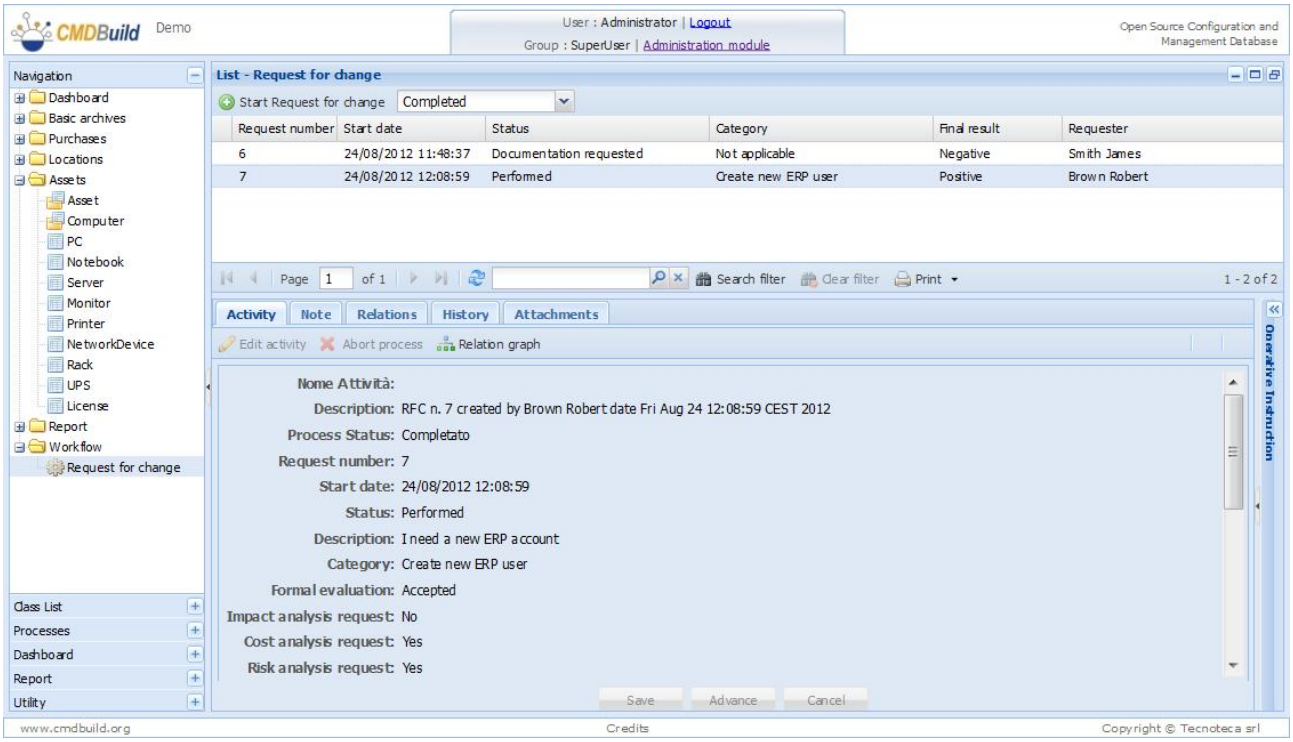
Come ultima operazione il Change Manager chiude la RfC indicando un esito positivo.

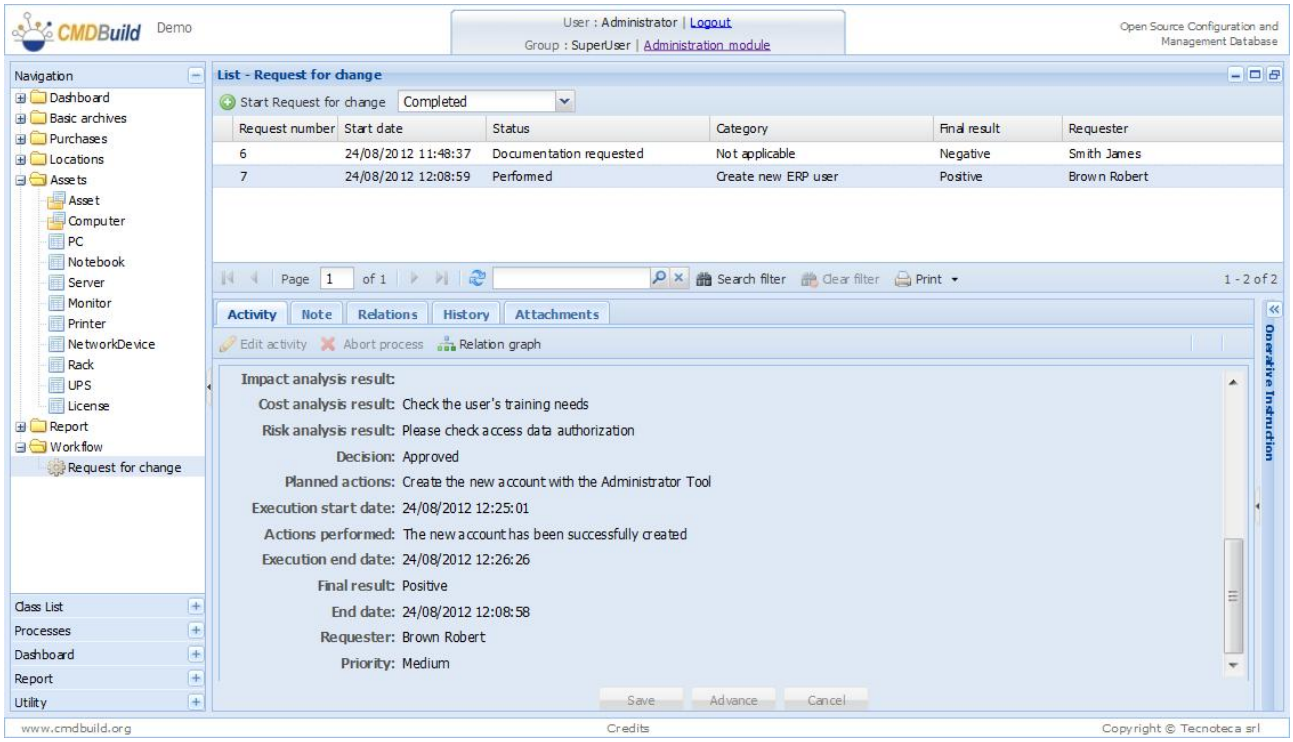


A questo punto la RfC su cui abbiamo lavorato (numero 7) non compare più nella lista delle RfC aperte.

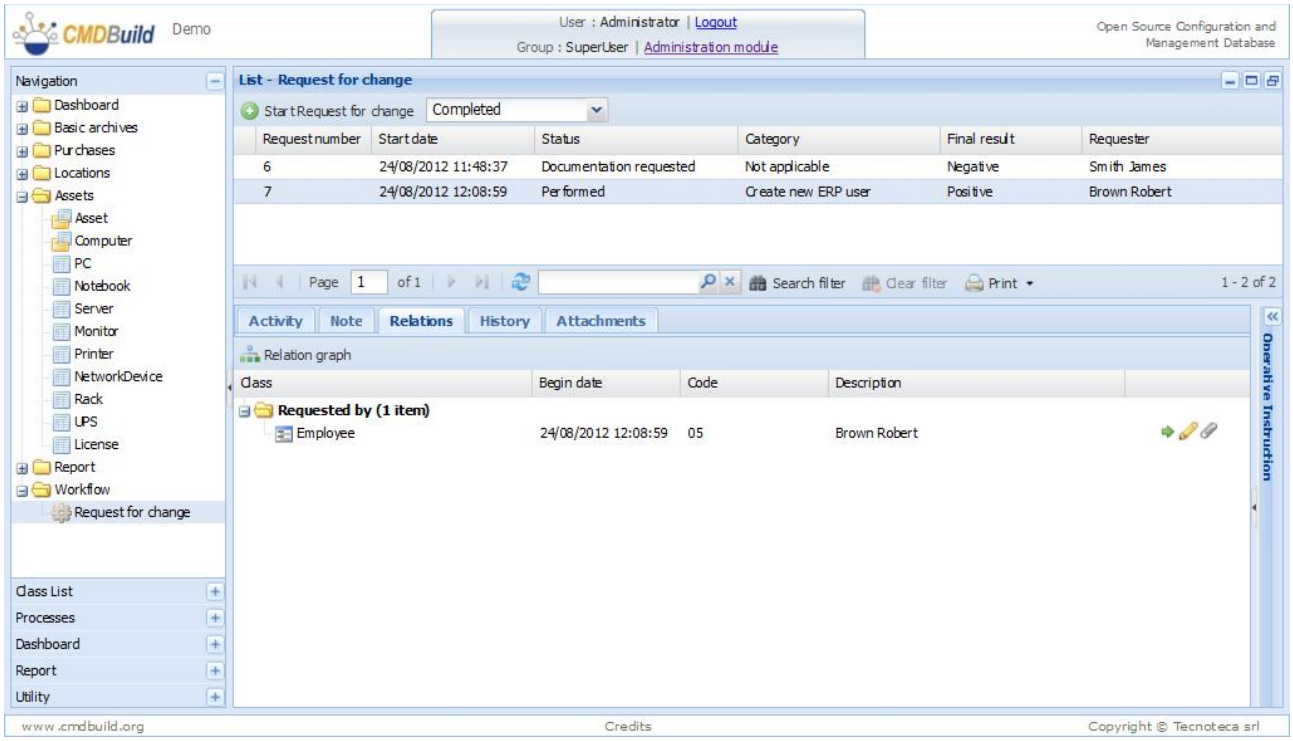


E' però consultabile con tutte le sue informazioni nella lista delle RfC completate (selezionabile tramite l'apposito controllo in alto sulla form).

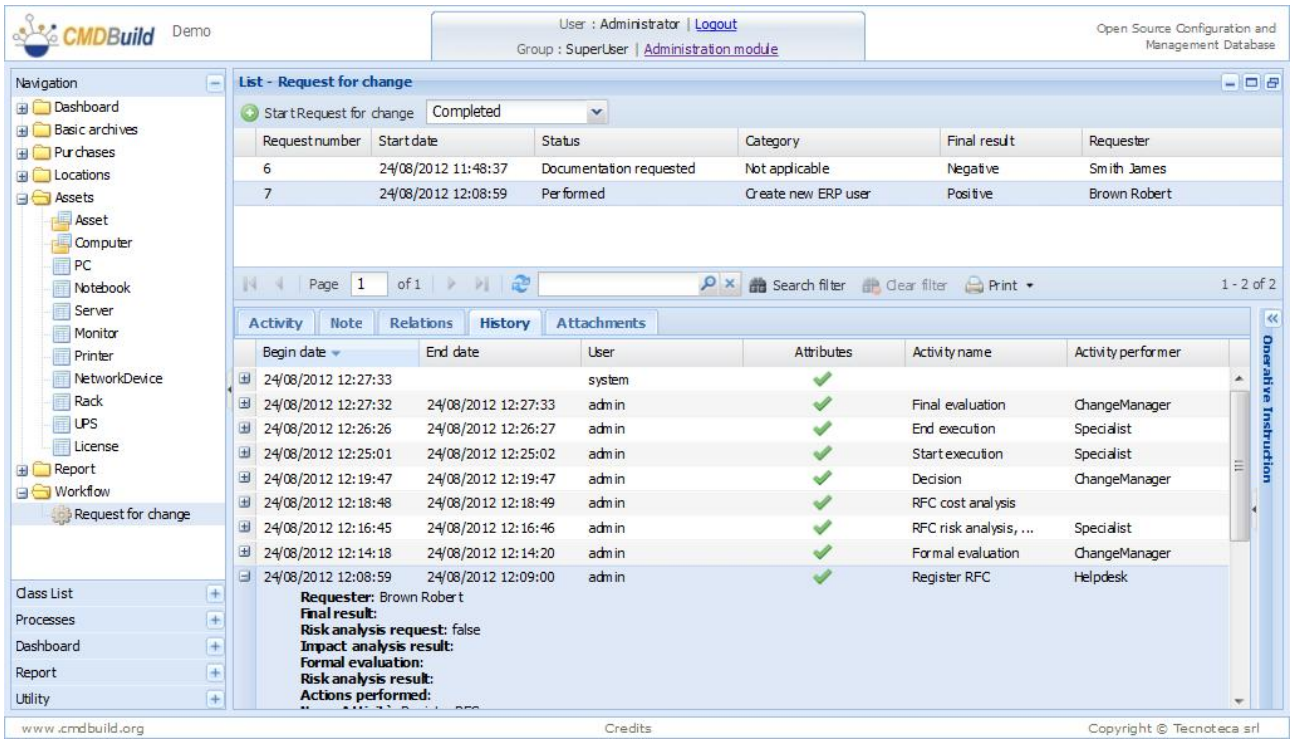




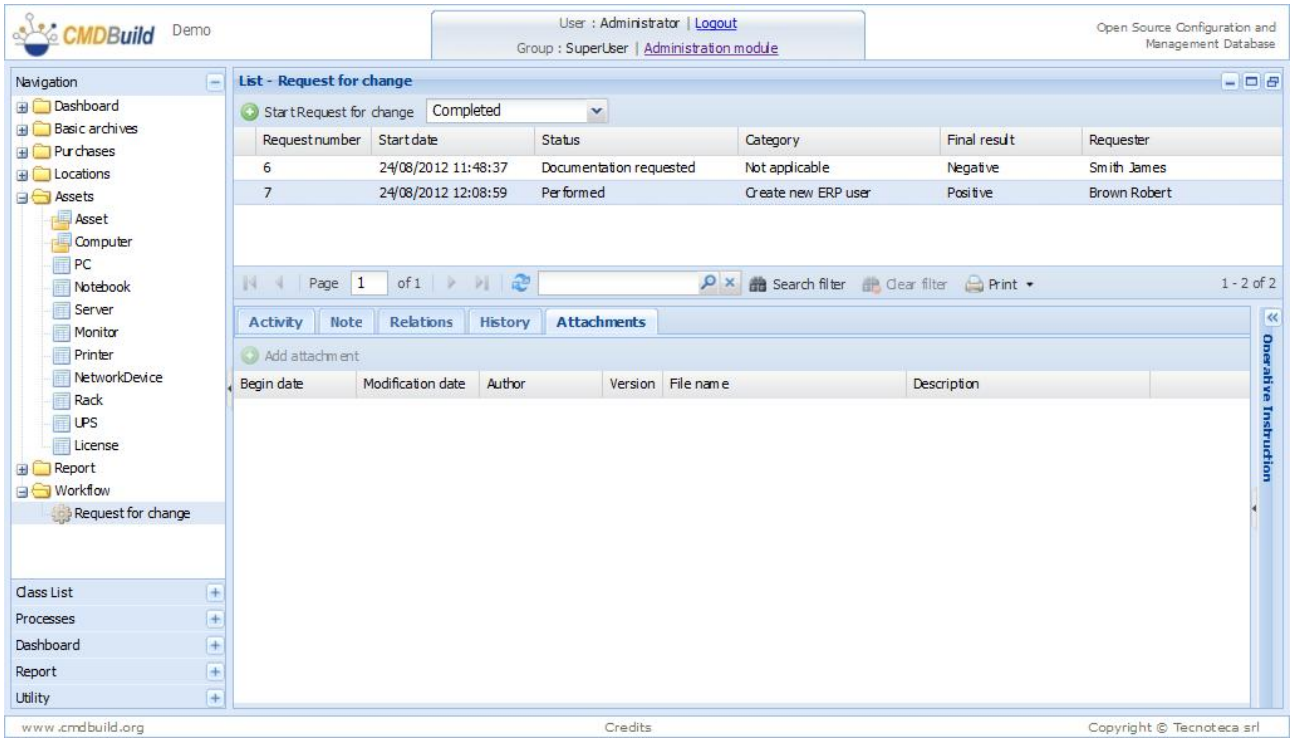
Oltre alle informazioni base possono essere consultate le relazioni configurate con quell'istanza del processo RfC (TAB Relazioni).



Può anche essere consultata la sequenza completa delle attività di avanzamento del processo (TAB Storia).



Caricando degli allegati nel corso del processo (tramite utilizzo dell'apposito widget) possono essere consultati i documenti eventualmente disponibili (TAB Allegati).



Widget utilizzabili nelle attività utente del workflow

Lista widget

CMDBuild rende disponibili alcuni widget (controlli visuali), posizionati nella parte destra delle form che gestiscono l'avanzamento del processo attraverso le attività previste.

Dal punto di vista grafico tali controlli sono rappresentati sotto forma di pulsanti caratterizzati dalla label specificata in fase di definizione.

Dal punto di vista della configurazione vanno definiti sotto forma di “Extended attribute” (previsti nello standard XPDL) utilizzando l'editor TWE.

Nel presente documento sono riferiti come tipi di dati sia i tipi primitivi (integer, string, date, float, boolean) che i tipi complessi aggiunti nei workflow di CMDBuild (lookup = id + type + description, lookups = array di lookup, reference = id + idclass + description, references = array di reference).

Controllo visuale	Descrizione	Parametri	Note
manageRelation	Presenta la lista selezionabile delle schede in relazione con la scheda specificata secondo il dominio specificato	Input: DomainName *string* ClassName *string* ObjId *integer* ButtonLabel *string* EnabledFunctions *character array* Required *integer* IsDirect *stringa* oppure Input: DomainName *string* ObjRef *reference* ButtonLabel *string* EnabledFunctions *character array* Required *integer* Output: CheckArray *references*	EnabledFunctions è un array di valori booleani che abilita varie funzionalità secondo il seguente criterio posizionale: - collega elemento - aggiungi e collega elemento - abilitazione check selezione - abilitazione radio button selezione - modifica relazione - scollega elemento - modifica elemento - cancella elemento Il parametro Required = 1 va indicato solo se è obbligatoria la selezione di almeno un elemento IsDirect può assumere i valori “true” o “false”
linkCards	Presenta la lista paginata selezionabile di tutte le schede appartenenti ad una data classe, con possibile visualizzazione su mappa geografica	Input: ClassName *string* ButtonLabel *string* SingleSelect *integer* NoSelect *integer* Required *integer* Filter *string* DefaultSelection *string* AllowCardEditing *integer*	Il parametro SingleSelect = 1 va indicato solo se va consentita la selezione di una unica riga (radio-button anziché checkbox) Il parametro NoSelect = 1 disabilita la selezione di righe (né radio button né checkbox) Il parametro Required = 1 rende obbligatoria la selezione di almeno una riga Il parametro Filter è di tipo espressione CQL (CMDBuild query language) Esempio: Filter = “from Persona where Id =

		DisableGridFilterTog gler *boolean* Map *string* StartMapWithLatitude *integer* StartMapWithLongitu de *integer* StartMapWithZoom *integer* Metadata *string* MetadataOutput *string* Output: CheckArray *references [metadataOutput] text*	{client:Cliente.Id}" Il parametro opzionale DefaultSelection specifica la query CQL usata per la selezione automatica al momento dell'apertura del widget Il parametro opzionale AllowCardEditing = 1 aggiunge una icona per la modifica della card Il parametro opzionale DisableGridFilterToggler = "true" nasconde il pulsante "Disabilita filtro" Il parametro opzionale Map abilita la visualizzazione della mappa (se impostato = 'enabled') I parametri relativi alla modalità di presentazione iniziale della mappa sono opzionali La variabile Metadata accetta come unico valore (in attesa di future estensioni) la stringa 'point:POINT'. La variabile MetadataOutput accetta come unico valore la stringa '_metadataOutput' che rappresenta il nome della variabile in uscita. Entrambi servono a gestire la selezione di un unico punto su una poligonale già presente. Le coordinate del punto verranno ritornate nella variabile metadataOutput nel formato WGS84. Un possibile esempio: punto:POINT(5847010.6684071 1438393.2786558)
createModifyCard	Presenta in modifica la scheda specificata (se ObjId è specificato) oppure consente la creazione di una nuova scheda nella classe specificata	Input: ClassName *string* ButtonLabel *string* ReadOnly integer oppure Input: Reference *reference* ButtonLabel *string* *ReadOnly integer* oppure Input: ClassName *string* ObjId *integer* ButtonLabel *string* *ReadOnly integer*	Esempio: ClassName='Utente' ObjId=client:Richiedente ButtonLabel = 'Crea o modifica Utente' Richiedente Nota: il prefisso "client:" è necessario per accedere ad una variabile prima che il workflow sia avanzato al passo successivo ReadOnly=1 presenta la card in sola lettura

		Output: Reference *reference*	
createReport		Input: ReportType *string* ReportCode *string* ButtonLabel *string* ForcePDF *integer* ForceCSV *integer* Parametro-1 Parametro-2 ... Parametro-n Output: ReportURL *string*	ReportType può al momento assumere solo il valore 'custom' ReportCode corrisponde all'attributo “Code” del report nella tabella “Report” ForcePDF forza l'output in formato PDF ForceCSV forza l'output in formato CSV Parametro-1 ... Parametro-n rappresentano i parametri di lancio previsti dal report
manageEmail	Permette di produrre tramite template o scrivere email libere che verranno inviate all'avanzamento del processo.	Input: ButtonLabel *string* ToAddresses *string* CCAddresses *string* Subject *string* Content *string* *Assignments string* ReadOnly *boolean*	Alla richiesta di visualizzazione delle email viene controllata la casella di posta per nuove email I parametri ToAddresses, CcAddresses, Subject e Content sono "string template" in cui possono essere inclusi "tag" per la "sostituzione" di variabili (ulteriori informazioni sono presenti al paragrafo successivo) E' richiesta la configurazione dei parametri di invio mail nel file *workflow.conf* .
openNote	Visualizza la pagina comprendente l'editor HTML per l'inserimento di note	Input: ButtonLabel *string*	Non utilizzabile nella prima attività di un processo
openAttachment	Visualizza la pagina predisposta per il caricamento di file da allegare al processo corrente	Input: ButtonLabel *string*	Non utilizzabile nella prima attività di un processo
calendar	Visualizza il calendario con riportate le date scelte	Input: ButtonLabel *string* ClassName *string* Filter *string* EventStartDate *date* EventEndDate *date* EventTitle *string*	ClassName è la classe da cui prelevare le date da mostrare sul calendario, con eventuale filtro (opzionale ma con precedenza su ClassName). Il parametro EventEndDate è opzionale. EventTitle indica l'attributo da cui prelevare il testo da riportare sul calendario per ogni data.
presetFromCard	Popola l'activity corrente con i dati recuperati da una card selezionata.	Input: ButtonLabel *string* ClassName *string* Filter *string* AttributeMapping *string*	ClassName il nome della classe, alternativo a Filter che invece è una espressione CQL. AttributeMapping è una stringa nella forma 'a1=c1,a2=c2' e indica come mappare attributi dell'activity con quelli della card. La virgola separa gli assegnamenti.

webService	Visualizza il risultato di una chiamata a Web Service (attualmente solo SOAP) come una griglia. È possibile selezionare delle righe di questa griglia, ed ottenere la loro serializzazione XML come output del widget.	Input: ButtonLabel *string* EndPoint *string* Method *string* NameSpacePrefix *string* NameSpaceURI *string* NodesToUseAsRows *string* NodesToUseAsColumns *string* SingleSelect='true' Mandatory='true' ReadOnly='true' Parametri *string* OutputSeparator *string* Output: Variabile di output *string*	EndPoint=URL del servizio Method=Nome del metodo NameSpacePrefix=prefisso del namespace' (opzionale) NameSpaceURI='URI del name space' (opzionale) NodesToUseAsRows= I nomi degli elementi (separati da virgole senza spazi) della risposta da visualizzare nella griglia NodesToUseAsColumns=I nomi degli elementi (separati da virgole senza spazi) della risposta da usare come colonne della griglia. Parametri della chiamata (opzionale) = parametri eventuali previsti dal Web Service. Variabile di output (opzionale) che verrà valorizzata con la serializzazione dell'XML corrispondente ai nodi selezionati. Se di tipo stringa, allora va specificato anche il separatore. OutputSeparator (opzionale)= carattere che verrà usato per separare i risultati. Se assente verranno restituiti come array di stringhe.
startWorkflow	Consente di avviare un workflow secondo due modalità: 1) configurazione letta direttamente da widget 2) configurazione letta da una tabella “di appoggio”	1) Input: ButtonLabel *string* WorkflowCode *string* oppure 2) Input: ButtonLabel *string* FilterType *string* Filter *string* Output: processRef *ReferenceType*	1) WorkflowCode nome del processo da avviare 2) FilterType attualmente supporta solo “cql” Filter il filtro cql da utilizzare per selezionare una serie di card da una tabella di CMDBuild. Il risultato del filtro deve essere l'elenco dei nomi dei processi da poter avviare dallo widget stesso.
grid	Consente di gestire una griglia di righe (aggiungendo, rimuovendo e/o modificandone le righe)	Input: ClassName string ButtonLabel string CardSeparator string AttributeSeparator *string*	ClassName rappresenta il nome della classe su cui si vuole operare CardSeparator separatore fra le diverse card inserite (default ";") AttributeSeparator separatore fra gli

		KeyValueSeparator *string* PresetsType="function" Presets *string* Output: Variabile di output *string*	attributi della stessa card(default "&") KeyValueSeparator separatore fra un attributo e il suo valore (default "==") La variabile di output sarà quindi un'unica stringa contenente la serializzazione dei dati inseriti, separati dai caratteri sopracitati PresetType e Preset servono per precaricare dei valori nella griglia che poi potranno essere modificati direttamente dall'utente. Se si vuole utilizzare come input una variabile o l'output di un'altra grid basta specificare semplicemente la chiave Preset=InputString dove InputString è una stringa formattata esattamente come l'output della grid. Se si vuole precaricare la griglia a partire da una funzione (contenente come nomi di colonna gli stessi campi della classe di riferimento) bisognerà specificare PresetsType="function" e Presets=”wf_function_name” dove ”wf_function_name” è il nome di una stored procedure nel database definita secondo i criteri usati per la creazione delle dashboard. Eventuali parametri vanno specificati di seguito nella forma: Param1="value1" (parametro di input della funzione) Param2="value2" (parametro di input della funzione) E' possibile inoltre precaricare dei valori nella griglia tramite l'apposito pulsante nel widget chiamato “Importa da CSV”. Ovviamente il file deve rispettare le convenzioni definite per l'importazione dei file CSV in CMDBuild. Sarà altresì possibile specificare il separatore, e la modalità con cui importare I dati (Sostituisci o Aggiungi)
customForm	Consente di gestire una form o una griglia di righe (aggiungendo, rimuovendo e/o modificandone le righe)	Input: ButtonLabel *string* ModelType *"[form|class|function]"* Layout *"[grid|form]"* DataType *[raw_json|raw_text|function]* ReadOnly *"[true|false]"* Required *"[true|false]"* AddDisabled *"[true|false]"* DeleteDisabled *"[true|false]"*	La struttura della custom form può essere definita a partire da: form - array di oggetti JSON class - attributi di una classe function - parametri di input di una funzione Il layout può essere form (come se fosse una card di CMDBuild) o una serie di righe. Il dati del widget possono essere inizializzati a partire da: raw_json - array di oggetti JSON raw_text – stringhe di testo opportunamente strutturate

		ImportDisabled *"[true|false]"* ModifyDisabled *"[true|false]"* SerializationType *"[json|text]"* KeyValueSeparator *string* AttributesSeparator *string* RowsSeparator *string* Output: Variabile di output *string*	function – i valori di output di una funzione I dati possono essere serializzati come tipo testo (si veda il widget grid) o come tipo json.
navigationTree	Consente di selezionare una o più schede dati tramite una interfaccia basata su un albero di navigazione (sottoinsieme del grafo dei domini) preconfigurato	Input: NavigationTreeName string ButtonLabel string Output: CheckArray *references*	NavigationTreeName rappresenta il nome dell'albero da visualizzare
adminStart	Nel caso di processo con più attività di start differenziate per gruppo individua l'attività da svolgere per l'utente amministratore		Non prevede parametri né di input né di output E' un “extended attribute”, e non un widget (non ha interfaccia utente), ma viene descritto in questa sezione essendo configurato nello stesso modo dei widget.

Informazioni ulteriori per l'utilizzo degli “string template” nel tool manageEmail

Il tool *manageEmail* permette di scrivere mail che verranno inviate all'avanzamento del processo. Alla richiesta di visualizzazione delle email, per visualizzare la griglia, viene controllata la casella di posta per nuove email.

parametri di input	• *string* ButtonLabel • uno o più blocchi di definizione delle mail ◦ *string template* ToAddresses: indirizzi di destinazione ◦ *string template* CcAddresses: indirizzi copia carbone ◦ *string template* Subject: oggetto della mail ◦ *string template* Content: corpo della mail (HTML) ◦ *string template* Condition: espressione javascript la cui valutazione determina se la mail va generata o meno • altri parametri opzionali contenenti query o espressioni javascript • *flag* ReadOnly: email in sola visualizzazione
parametri di output	nessuno

Il *flag* di sola lettura è un valore inteso come booleano; sono considerati *true* un valore booleano (del processo) un valore intero positivo o una stringa non vuota

Gli *string template* sono delle stringhe in cui viene eseguita la sostituzione delle variabili, nella forma {namespace:localname}, che vengono interpretate in modo diverso a seconda del namespace (se il namespace è omesso, di default viene usato "server").

client:name **client**:name.Id **client**:name.Description	Variabile *name* del form; per lookup e reference è necessario specificare tramite la notazione puntata se si vuole l'*Id* o la *Description*
server:name	Variabile *name* del processo al passo precedente
xa:name	Variabile *name* della definizione dell'extended attribute, espansa come template ad esclusione delle variabili con namespace *js* e *cql*
user:id **user**:name	ID e nome dell'utente connesso
group:id **group**:name	ID e nome del gruppo connesso
js:name	Variabile *name* della definizione dell'extended attribute interpretata a sua volta come un template e valutata come codice javascript
cql:name.field	Variabile *name* della definizione dell'extended attribute interpretata a sua volta come un template e valutata eseguendo una query CQL, di cui viene preso il campo identificato da *field*

I blocchi di definizione delle mail possono essere scritti in due forme:

ToAddresses="**…**"
CcAddresses="..."
Subject="..."
Content="..."

oppure (nel caso si voglia specificare più di una mail):

ToAddresses1="**…**"
CcAddresses1="..."
Subject1="..."
Content1="..."
ToAddresses2="..."
CcAddresses2="..."
Subject2="..."
Content2="..."
...

Esempio 1

ToAddresses="pippo@pluto.it"
Subject="**{cql:QueryRichiedente.Description}** - **{client:Richiesta}**"
QueryRichiedente="select Description,Email,Ufficio from Dipendente where Id = **{cql:SillyQuery.Id}**"
SillyQuery="select Id from Dipendente where Id=**{client:Richiedente}**"

Address: L'indirizzo di destinazione è completato staticamente con la stringa pippo@pluto.it

Body: Il corpo del messaggio è vuoto

Subject:

- {cql:QueryRichiedente.Description} viene sostituito con il campo Description della prima card del risultato della query scritta nella variabile QueryRichiedente dell'extended attribute

- {cql:SillyQuery.Id} in QueryRichiedente viene a sua volta sostituita con il campo Id della card ritornata dalla query SillyQuery (sono infatti supportate query annidate) a cui è stato prima sostituito {client:Richiedente} con il valore preso dal form
- {client:Richiesta} di viene completato con il valore del form

Esempio 2

```
...
Content="Il richiedente, {js:JoinJS}, appartenente all'ufficio {cql:QueryRichiedente.Ufficio_value} richiede:<br
/><br />{server:Richiesta}"
JoinJS="{js:FirstJS}+"#"+{js:SecondJS}"
FirstJS="{cql:QueryRichiedente.Description}.slice(0,{xa:SplitLength})"
SecondJS="{cql:QueryRichiedente.Description}.slice({xa:SplitLength})"
SplitLength=2
QueryRichiedente="select Description,Email,Ufficio from Dipendente where Id = {Richiedente}"
```

Questo è un esempio decisamente più complesso.

In body sono presenti tre variabili da sostituire:

- {js:JoinJS} valuta la variabile dell'extended attribute come un'espressione javascript, separando con # le due variabili FirstJS e SecondJS valutate sempre tramite javascript
- {js:FirstJS} e {js:SecondJS} a loro volta contengono sia una variabile presa da un field della query CQL QueryRichiedente sia una variabile statica presa da quelle dell'extended attribute
- {cql:QueryRichiedente...} a sua volta contiene un riferimento ad una variabile lato server di nome Richiedente
- {cql:QueryRichiedente.Ufficio_value} ha la particolarità di utilizzare la descrizione del reference Ufficio invece che il suo ID (che sarebbe stato semplicemente Ufficio)
- {server:Richiesta} prende sempre una variabile lato server (come Richiedente), ma dichiarando il namespace

API utilizzabili nelle attività automatiche del workflow

CMDBuild rende disponibili alcune API (Application Programming Interface) utilizzabili nelle attività automatiche del workflow per la scrittura di script con cui implementare comportamenti personalizzati (manipolazione di variabili del processo, creazione di schede e relazioni nel CMDB, invio mail, creazione report, ecc).

- La condizione per l'invio dell'email è sempre verificata in quanto {xa:SplitLength} è costante e l'espressione javascript è sempre vera.

Generalità

Parole chiave

Processo
ProcessId: int Id del processo corrente
ProcessClass: String nome della Classe del processo corrente
ProcessCode: String ProcessInstanceId univoco del processo corrente

Esecutore
_CurrentUser: ReferenceType reference allo User che ha svolto l'ultima attività del processo corrente
_CurrentGroup: ReferenceType reference al Role che ha svolto l'ultima attività del processo corrente

API
cmdb indentifica le funzioni native di CMDBuild

Gestione degli oggetti di CMDBuild

Interessano i tipi di dati specifici di CMDBuild, per altri tipi di dati (interi, stringhe, date, float) sono utilizzabili tutti i metodi di manipolazione offerti dal linguaggio Java.

ReferenceType

Metodi
getId(): int restituisce l'id del Reference
getDescription(): String

restituisce la descrizione del Reference

LookupType

Metodi
getId(): int restituisce l'id della Lookup
getType(): String restituisce il tipo di Lookup
getDescription(): String restituisce la descrizione della Lookup
getCode(): String restituisce il codice della Lookup

CardDescriptor

Metodi
getClassName(): String restituisce il nome della Classe per una variabile di tipo CardDescriptor
getId(): int restituisce il nome dell'Id per una variabile di tipo CardDescriptor
equals(CardDescriptor cardDescriptor): boolean confronta la variabile di tipo CardDescriptor con quella specificata

Card

Metodi
getCode(): String restituisce il Code per una variabile di tipo Card
getDescription(): String restituisce la Description per una variabile di tipo Card
has(String name): boolean verifica la presenza dell'attrbiuto specificato nella variabile di tipo Card
hasAttribute(String name): boolean verifica la presenza dell'attrbiuto specificato nella variabile di tipo Card
get(String name): Object restituisce il valore dell'attrbiuto specificato della variabile di tipo Card
getAttributeNames(): Set<String> restituisce la lista degli attributi della variabile di tipo Card
getAttributes(): Map<String, Object> restituisce la lista degli attributi e relativi valori della variabile di tipo Card. I valori ritornati dalla funzione rispettano i tipi di CDMBuild (ReferenceType, LookupType, Date, Integer, ...)

Attachments

Metodi
fetch(): Iterable<AttachmentDescriptor> restituisce la lista degli allegati della card o del processo istanziato
upload(Attachment... attachments):void allega i documenti alla card o al processo istanziato
upload(String name, String description, String category, String url):void crea un allegato con nome, descrizione e categoria specificate a partire dal file con la URL specificata e lo allega alla card o al processo istanziato
selectByName(String... names): SelectedAttachments resituisce gli allegati della card o del processo istanziato con il nome specificato
selectAll(): SelectedAttachments restituisce tutti gli allegati della card o del processo istanziato

AttachmentDescriptor

Metodi
getName(): String restituisce il nome dell'allegato
getDescription(): String restituisce la descrizione dell'allegato
getCategory(): String restituisce la categoria dell'allegato

Attachment

Metodi
getUrl(): String restituisce la URL del file

DownloadedReport

Metodi
getUrl(): String restituisce l'URL locale in cui è stato salvato il report
equals(DownloadedReport downloadedReport): boolean confronta la variabile di tipo DownloadedReport con quella specificata

Metodi di accesso al CMDBuild

NewCard

Costruttori

newCard(String className): NewCard istanzia una nuova Card da creare in CMDBuild nella Classe specificata

Modificatori
withCode(String value): NewCard aggiunge il Code alla nuova card da creare in CMDBuild
withDescription(String value): NewCard aggiunge la Description alla nuova card da creare in CMDBuild
with(String name, Object value): NewCard aggiunge il valore specificato per l'attributo specificato alla nuova card da creare in CMDBuild
withAttribute(String name, Object value): NewCard aggiunge il valore specificato per l'attributo specificato alla nuova card da creare in CMDBuild

Azioni
create(): CardDescriptor crea la nuova card in CMDBuild settando gli attributi precedentemente definiti

Esempio:

```
/*
 * Creazione di una nuova card nella classe "Employee" avente i
 * seguenti attributi:
 *  "Code"       =    "T1000"
 *  "Name"       =    "James"
 *  "Surname"    =    "Hetfield"
 */
cdNewEmployee = cmdb.newCard("Employee")
.withCode("T1000")
.with("Name", "James")
.withAttribute("Surname", "Hetfield")
.create();
```

ExistingCard

Costruttori
existingCard(String className, int id): ExistingCard istanzia una Card esistente nella Classe specificata avente l'Id specificato per interrogare CMDBuild
existingCard(CardDescriptor cardDescriptor): ExistingCard istanzia una Card esistente indicata dal CardDescriptor specificato per interrogare

CMDBuild

Modificatori
withCode(String value): ExistingCard imposta il Code per la Card da richiedere a CMDBuild
withDescription(String value): ExistingCard imposta la Description per la Card da richiedere a CMDBuild
with(String name, Object value): ExistingCard imposta l'attirbuto specificato con il valore specificato per la Card da richiedere a CMDBuild
withAttribute(String name, Object value): ExistingCard imposta l'attrbiuto specificato con il valore specificato per la Card da richiedere a CMDBuild
withAttachment(String url, String name, String category, String description): ExistingCard allega un documento (indicato tramite url locale del server) alla card selezionata impostando il nome del file, la categoria e la descrizione
attachments(): ExistingCard permette di accedere agli allegati della card selezionata
selectAll(): ExistingCard permette la selezione di tutti i documenti della card selezionata
selectByName(String name1, String name2, ...):ExistingCard permette la selezione di tutti i documenti della card selezionata

Azioni
update() aggiorna la Card in CMDBuild impostando gli attrbituti precedentemente indicati con i valori specificati
delete() cancella (cancellazione logica) la Card da CMDBuild Se è stato usato il modificatore attachments, cancella solamente i file selezionati
fetch(): Card richiede la Card a CMDBuild con gli attributi precedentemente indicati. Se non sono stati utilizzati modificatori allora viene richiesta l'intera Card (con tutti gli attributi) fetch(): Iterable<AttachmentDescriptor> Se è stato usato il modificatore attachments, il metodo restituisce la lista di allegati della card
upload(Attachment attachment, Attachment attachment2,..) da usare in presenza del modificatore attachments: allega alla card uno o più file

upload(Attachment attachment, String description, String category, String url) da usare in presenza del modificatore attachments: allega alla card un singolo file con descrizione e categoria indicate
download(): Iterable<Attachment> Se è stato usato il modificatore attachments, il metodo restituisce gli allegati della card selezionati
copyTo() Se è stato usato il modificatore attachments, il metodo copia un allegato selezionato della card in una destinazione specificata
moveTo() Se è stato usato il modificatore attachments, il metodo sposta un allegato selezionato della card in una destinazione specificata

Esempi:

```
/*
 * Modifica della card precedente creata nella classe “Emplyee”
 * impostando i seguenti attributi:
 *  “Phone”      =     “754-3010”
 *  “Email”      =     “j.hetfield@somemail.com”
 */
cmdb.existingCard(cdNewEmplyee)
.with(“Phone”, “754-3010”)
.withAttribute(“Email”, “j.hetfield@somemail.com”)
.update();

/*
 * Cancellazione (logica) della card precedente creata nella classe
 * “Emplyee”
 */
cmdb.existingCard(cdNewEmplyee)
.delete();


/*
 * Cancellazione dell'allegato alla card precedentemente
 * creata nella classe “Employee”
 */

Iterable <AttachmentDescriptor> attachments =
cmdb.existingCard(cdNewEmplyee)
.attachments()
.fetch();
```

```
/*
 * Cancellazione dell'allegato alla card precedentemente
 * creata nella classe “Employee”
 */
cmdb.existingCard(cdNewEmplyee)
.attachments()
.selectByName(String[]{"attachment-name"})
.delete();
```

NewProcessInstance

Costruttori
newProcessInstance(String className): NewProcessInstance istanzia una nuova istanza di processo da creare in CMDBuild per il processo specificato

Modificatori
withDescription(String value): NewProcessInstance aggiunge la Description alla nuova card da creare in CMDBuild
with(String name, Object value): NewProcessInstance aggiunge il valore specificato per l'attributo specificato al nuovo processo da creare in CMDBuild
withAttribute(String name, Object value): NewProcessInstance aggiunge il valore specificato per l'attributo specificato al nuovo processo da creare in CMDBuild

Azioni
start(): ProcessInstanceDescriptor crea il nuovo processo in CMDBuild settando gli attributi precedentemente definiti, e non avanza
startAndAdvance(): ProcessInstanceDescriptor crea il nuovo processo in CMDBuild settando gli attributi precedentemente definiti, e avanza al passaggio successivo

Esempio:

```
/*
 * Creazione di una nuova card nella classe “RequestForChange”
 * avente i seguenti attributi
 *  “Requester”  = “James Hetfield”
 *  “RFCExtendedDescription” = “My printer is broken”
 */
pidNewRequestForChange =
```

```
cmdb.newProcessInstance("RequestForChange")
.with("Requester", "James Hetfield")
.withAttribute("RFCExtendedDescription", "My printer is broken")
.startAndAdvance();
```

ExistingProcessInstance

Costruttori
existingProcessInstance(String processClassName, int processId): ExistingProcessInstance istanzia un'istanza di processo esistente nella classe di processo specificata avente l'Id specificato

Modificatori
with(String name, Object value): ExistingProcessInstance imposta l'attributo specificato con il valore specificato per l'istanza di processo
withAttribute (String name, Object value): ExistingProcessInstance imposta l'attributo specificato con il valore specificato per l'istanza di processo
withDescription(String value): ExistingProcessInstance imposta l'attributo specificato con il valore specificato per l'istanza di processo
attachments(): Attachments consente di accedere agli allegati dell'istanza di processo

Azioni
abort(): void abortisce l'istanza di processo
advance(): void avanza l'istanza di processo
resume(): void risveglia l'istanza di processo sospesa
suspend(): void sospende l'istanza di processo aperta
update(): void aggiorna l'istanza di processo

Esempio:

```
/*
* Aggiornamento dell'istanza di processo nella classe "Request
* for change" con Id = pid modificandone il richiedente e
* avanzamento del processo al passo successivo
```

```
*/

cmdb.existingProcessInstance("RequestForChange",  pid)
.with("Requester", cdNewEmployee.getId())
.advance();
```

NewRelation

Costruttori
newRelation(String domainName): ExistingProcessInstance istanzia una nuova relazione da aggiungere in CMDBuild nel Dominio specificato

Modificatori
withCard1(String className, int cardId): NewRelation imposta la card al lato sorgente della relazione
withCard2(String className, int cardId): NewRelation imposta la card al lato destinazione della relazione

Azioni
create() crea la nuova relazione in CMDBuild tra le Card indicate nel Dominio specificato

Esempio:

```
/*
 * Creazione di una nuova relazione nel dominio "AssetAssegnee"
 * tra una card della classe "Asset" selezionata
 * attraverso l'attributo di tipo Reference "Item" e
 * la card precedentemente creata nella classe "Emplyee"
 */
cmdb.newRelation("AssetAssegnee")
.withCard1("Employee", cdNewEmployee.getId())
.withCard2("Asset", Item.getId())
.create();
```

ExistingRelation

Costruttori
existingRelation(String domainName): ExistingRelation istanzia una relazione esistente in CMDBuild nel Dominio specificato

Modificatori

withCard1(String className, int cardId): ExistingRelation imposta l'IdClass e l'ObjId della Card dal lato sorgente della relazione
withCard2(String className, int cardId): ExistingRelation imposta l'IdClass e l'ObjId della Card dal lato destinazione della relazione

Azioni
delete() cancella (cancellazione logica) la relazione esistente in CMDBuild tra le Card indicate nel Dominio specificato

Esempio:

```
/*
 * Cancellazione della relazione sul dominio "AssetAssegnee"
 * tra le card indicate in precedenza
 */
cmdb.existingRelation("AssetAssegnee")
.withCard1("Employee", cdNewEmployee.getId())
.withCard2("Asset", Item.getId())
.delete();
```

QueryClass

Costruttori
queryClass(String className): QueryClass istanzia una query per interrogare la classe specificata in CMDBuild

Modificatori
withCode(String value): QueryClass imposta il Code della Card per il filtro da utilizzare per interrogare CMDBuild
withDescription(String value): QueryClass imposta la Description della Card per il filtro da utilizzare per interrogare CMDBuild
with(String name, Object value): QueryClass imposta il valore per l'attributo specificato della Card per il filtro da utilizzare per interrogare CMDBuild
withAttribute(String name, Object value): QueryClass imposta il valore per l'attributo specificato della Card per il filtro da utilizzare per interrogare CMDBuild

Azioni
fetch(): List<Card> eseque la query di ricerca su CMDBuild sulla Classe specificata e restituisce la lista

delle Card che rispettano il filtro impostato precedentemente

Esempio:

```
/*
 * Elenco delle card della classe “Employee” che hanno
 * l'attributo “State” impostato ad 'Active'
 */
Employees = cmdb.queryClass(“Employee”)
.with(“State”, “Active”)
.fetch();
```

CallFunction

Costruttori
callFunction(String functionName): CallFunction istanzia una chiamata ad una stored procedure precedentemente definita in PostgreSQL

Modificatori
with(String name, Object value): CallFunction imposta il valore del parametro di input specificato per la stored procedure

Azioni
execute(): Map<String, String> esegue la stored procedure e restituisce la lista dei parametri di output con i relativi valori

Esempio:

```
/*
 * Chiamata della stored procedure PostgreSQL
 * “cmwf_getImpact”(IN “DeliveryDate” date, IN “Cost” integer,
 * OUT “Impact” character varying)
 * che calcola il livello di impatto (attributo di
 * processo “Impact”) di una attività su una scala “Alto”,
 * “Medio” e “Basso” dati in input la data di prevista
 * consegna (attributo di processo “ExpectedDeliveryDate”) ed
 * il costo (attributo “ManHoursCost”) espresso in ore/uomo
 */
spResultSet = cmdb.callFunction(“cmwf_getImpact”)
.with(“DeliveryDate”, ExpectedDeliveryDate.getTime())
.with(“Cost”, ManHoursCost)
```

```
    .execute();
    Impact = spResultSet.get("Impact")
```

Nota: le funzioni SQL da chiamare devono essere definite secondo gli standard di CMDBuild. Per la loro definizione si veda l'Administrator Manual, sezione TAB Grafici, paragrafo "Definizione della sorgente dati (Funzione PostgreSQL)".

QueryRelations

Costruttori
queryRelations(CardDescriptor cardDescriptor): ActiveQueryRelations istanzia una query per richiedere a CMDBuild le Card in relazione con quella specificata
queryRelations(String className, int id): ActiveQueryRelations istanzia una query per richiedere a CMDBuild le Card in relazione con quella specificata da className ed id

Modificatori
withDomain(String domainName): ActiveQueryRelations imposta il Dominio su cui eseguire la query

Azioni
fetch(): List<CardDescriptor> esegue la query su CMDBuild utilizzando i parametri definiti precedentemene, restituisce la lista delle Card collegate

Esempio:

```
/*
 * Elenco degli "Asset" collegati alla card "Employee" indicata
 * dal CardDescriptor cdNewEmployee creata in precedenza,
 * attraverso la relazione sul dominio "AssetAssegnee"
 */
assets = cmdb.queryRelation(cdNewEmployee)
.withDomain("AssetAssegnee")
.fetch();
```

CreateReport

Costruttori
createReport(String title, String format): CreateReport istanzia la creazione del Report nel formato specificato (pdf, csv) avente il Titolo specificato

Modificatori
with(String name, Object value): CreateReport imposta il valore del parametro di input specificato per il Report

Azioni
download(): DownloadedReport genera il Report indicato utilizzando i paramteri definiti precedentemente

Esempio:

```
/*
 * Genera il Report "DismissedAssets" che mostra l'elenco
 * degli Asset dismessi
 */
newReport = cmdb.createReport("Assigned assets to")
.download();
```

NewMail

Costruttori
newMail(): NewMail istanzia una nuova mail da inviare

Modificatori
withFrom(String from): NewMail imposta il mittente della mail da inviare
withTo(String to): NewMail imposta i destinatari della mail da inviare
withCc(String cc): NewMail imposta i destinatari in copia conoscenza della mail da inviare
withBcc(String bcc): NewMail imposta i destinatari in copia conoscenza nascosta della mail da inviare
withSubject(String subject): NewMail imposta l'oggetto della mail da inviare
withContent(String content): NewMail imposta il testo della mail da inviare
withContentType(String contentType): NewMail imposta il MimeType del contenuto della mail da inviare, i valori ammessi sono "text/html" o "text/plain". Se non specificato il valore di dafault è "text/plain"
withAttachment(URL url): NewMail imposta l'url di un documento da allegare alla mail

withAsynchronousSend(bool boolean): NewMail
invia la mail in modo asincrono rispetto allo script; in tal modo si eviteranno possibili problemi di timeout, ma non sarà più possibile intervenire in caso di eccezione nell'invio della mail stessa

Azioni
send() esegue l'invio della mail utlizzando le impostazioni precedentemente definite

Esempio:

```
/*
 * Invio di una nuova email
 */
cmdb.newMail()
.withFrom("fromaddress@somemail.com")
.withTo("toaddress@somemail.com")
.withCc("ccaddress@somemail.com")
.withSubject("Mail subject")
.withContent("Mail content")
.send();
```

NewMailQueue

Costruttori
newMailQueue(): NewMailQueue istanzia una nuova coda di email

Metodi
newMail(): QueueableNewMail aggiunge una nuova email alla coda
sendAll(): void invia tutte le email della coda

```
/*
 * Invio di una nuova email
*/
cmdb.newMailQueue()
.newMail()
.withFrom("fromaddress@somemail.com")
.withTo("toaddress@somemail.com")
```

```
.withCc("ccaddress@somemail.com")
.withSubject("Mail subject")
.withContent("Mail content")
.add()
.sendAll();
```

Metodi per la conversione di tipi

ReferenceType

Metodi
referenceTypeFrom(Card card): ReferenceType restituisce l'oggetto ReferenceType relativo alla Card specificata
referenceTypeFrom(CardDescriptor cardDescriptor): ReferenceType restituisce l'oggetto ReferenceType relativo al CardDescriptor specificato
referenceTypeFrom(int id): ReferenceType restituisce l'oggetto ReferenceType relativo alla carda avente l'Id specificato

Esempio:

```
/*
 * Impostare l'attributo di processo "Requester" di tipo
 * Reference dato il CardDescriptor "cdNewEmployee"
 * creato precedentemente
 */
Requester = cmdb.referenceTypeFrom(cdNewEmployee);
```

LookupType

Metodi
selectLookupById(int id): LookupType restituisce l'oggetto LookupType avente l'Id pecificato
selectLookupByCode(String type, String code): LookupType restituisce l'oggetto LookupType avente Type e Code specificati
selectLookupByDescription(String type, String description): LookupType restituisce l'oggetto LookupType avente Type e Description specificati

Esempio:

```
/* Impostare l'attributo di processo "State" di tipo Lookup avente:
 *  "Type" =     "Employee state"
 *  "Code" =     "ACTIVE"
 */
State = cmdb.selectLookupByCode("Employee state", "ACTIVE");
```

CardDescriptor

Metodi
cardDescriptorFrom(ReferenceType reference): CardDescriptor restituisce il CardDescriptor della card specificata attraverso l'oggetto ReferenceType specificato

Esempio:

```
/*
 * Ottenere il CardDescriptor relativo all'attributo di
 * processo "Requester" tipo Reference
*/
cdSelectedEmployee = cmdb.cardDescriptorFrom(Requester);
```

Card

Metodi
cardFrom(ReferenceType reference): Card restituisce l'oggetto Card della card specificata attraverso l'oggetto ReferenceType specificato

Esempio:

```
/*
 * Ottenere la Card completa relatvo all'attributo di
 * processo "Requester" tipo Reference
 */
selectedEmployee = cmdb.cardFrom(Requester);
```

Appendice: Documentazione per utilizzo TWS 2.3

Avvertenza

Alla presente appendice viene riportata per completezza la documentazione tecnica specifica del sistema di workflow in uso fino a CMDBuild 1.5, di cui viene mantenuta la compatibilità anche in CMDBuild 2.0, in attesa di dismetterlo non appena possibile.

Si ricorda che CMDBuild 2.0 offre la doppia possibilità di lavorare (ovviamente in alternativa) sia con Together Workflow Server 2.3 (la versione utilizzata fino a CMDBuild 1.5, basata su XPDL 1.0) che con la nuova versione Together Workflow Server 4.4 (basata su XPDL 2.0).

Si consiglia di effettuare la migrazione in tempi brevi, dal momento che tale doppia compatibilità sarà mantenuta per un periodo di tempo limitato.

Metodi automatici utilizzabili nel workflow

Per l'utilizzo con Together Workflow Server 2.3 (superato dal sistema basato su Together Workflow Server 4.4) CMDBuild rende disponibili alcuni metodi (“tool”) utilizzabili all'interno dei “tool activity” (attività automatiche) per eseguire varie tipologie di operazioni:

- metodi per la manipolazione di variabili: conversione fra tipologie di dati, concatenazione di stringhe, ecc
- metodi per il controllo del flusso: iteratore, sospensione processo, ravvio processo
- metodi di accesso al CMDB: creazione nuova scheda, lettura o modifica attributo, creazione relazione, ecc
- metodi esterni: invio mail, lettura ora di sistema, ecc

Metodi per la manipolazione di variabili

Tool	Descrizione	Parametri input	Parametri output	Note
addDays	Aggiunge alla data specificata il numero di giorni indicato	InputDate *date* days *integer*	OutputDate *date*	
boolToString	Converte una variabile booleana in stringa	InputBool *boolean*	OutputString *string*	
boolCopy	Copia il valore di una variabile boolean in un'altra variabile boolean	From *boolean*	To *boolean*	
clearIterator	Azzera l'iteratore	RefArray *references* HasNext *boolean* *CurrentIndex* *integer*	RefArray *references* HasNext *boolean* CurrentIndex *integer*	RefArray viene impostato a null, CurrentIndex viene impostato a 0 e HasNext a false
clearLookup	Azzera il valore di	Lookup *lookup*		Viene impostato a -1 il

	una variabile Lookup			valore dell'attributo “Id”
clearReference	Azzera il valore di una variabile Reference	Ref *reference*		Viene impostato a -1 il valore dell'attributo “Id”
concat concat3 / concat4 / ... / concat8	Concatena due o più stringhe	InputString1 *string* InputString2 *string* ... InputStringn *string*	OutputString *string*	
createReferenceObj	Crea una variabile di tipo reference e la inizializza	ClassName *string* ObjId *integer* Description *string*	OutRef *reference*	La variabile viene inizializzata con i valori letti dagli attributi “ClassName”, “ObjId” e “Description” della scheda dati specificata
dateToString	Converte una variabile data in stringa	InputDate *date*	OutputString *string*	
floatToString	Converte una variabile float in stringa	InputFloat *float*	OutputString *string*	
floatCopy	Copia il valore di una variabile float in un'altra variabile float	From *float*	To *float*	
getReferenceId	Estrae l'attributo “Id” da una variabile di tipo reference	Ref *reference*	CardId *integer*	
getReferenceClassId	Estrae l'attributo “ClassId” da una variabile di tipo reference	Ref *reference*	ClassId *integer*	
getLookupDescription	Estrae l'attributo “Description” da una variabile di tipo lookup	Lookup *lookup*	Description *string*	
getLookupId	Estrae l'attributo “Id” da una variabile di tipo lookup	Lookup *lookup*	Id *Integer*	
getLookupCode	Estrae l'attributo “Code” da una variabile di tipo lookup	Lookup *lookup*	Code *String*	
getReferenceDescription	Estrae l'attributo “Description” da una variabile di tipo reference	Ref *reference*	Description *string*	

getReferenceFrom Array	Estrae dall'array specificato il Reference di indice specificato	RefArray *references* *Index integer*	OutRef *reference*	Se l'array è nullo o l'indice è maggiore della sua dimensione restituisce “null”
intToString	Converte una variabile integer in stringa	InputInt *integer*	OutputString *string*	
intCopy	Copia il valore di una variabile integer in un'altra variabile integer	From *integer*	To *integer*	
lookupToString	Converte il campo “Id” della variabile lookup in stringa	InputLookup *lookup*	OutputString *string*	
nextInt	Incrementa la variabile di tipo integer specificata	InputInt *integer*	InputInt *integer*	
referenceToString	Converte il campo “Id” della variabile reference in stringa	InputReference *reference*	OutputString *string*	
stringToDate	Converte una variabile stringa in data	InputString *string*	OutputDate *date*	Accetta in input i formati YY/mm/dd oppure YY/mm/dd HH:mm:ss
stringCopy	Copia il valore di una variabile stringa in un'altra variabile stringa	From *string*	To *string*	
dateCopy	Copia il valore di una variabile date in un'altra variabile date	From *date*	To *date*	
stringToBool	Converte una variabile stringa in un valore booleano	From *string*	To *boolean*	Accetta in input le stringhe true oppure false
stringToInt	Converte una variabile stringa in un intero	From *string*	To *integer*	Accetta in input la rappresentazione sotto forma di stringa di un numero intero
stringToFloat	Converte una variabile stringa in un reale	From *string*	To *float*	Accetta in input la rappresentazione sotto forma di stringa di un numero reale

Metodi per il controllo del flusso

Tool	**Descrizione**	**Parametri input**	**Parametri output**	**Note**
nextRef	Incrementa	RefArray	HasNext	RefArray è un array di

	l'iteratore su un array di reference	*references* CurrentIndex *integer*	*boolean* CurrentIndex *integer* CurrentValue *reference*	reference, CurrentValue è il reference corrispondente all'indice corrente
resetIterator	Resetta l'iteratore	RefArray *references*	HasNext *boolean* CurrentIndex *integer*	CurrentIndex viene impostato a 0, HasNext vale true se l'array è non vuoto
resumeProcess	Riavvia il processo specificato	ProcessInstanceId *string* *Complete integer*		Il processo specificato deve essere in stato “Sospeso” Se “Complete” assume il valore 1 il processo avanza anche allo step successivo
suspendProcess	Sospende il processo specificato	ProcessInstanceId *string*		Si può utilizzare la costante “CURRENT” per indicare il processo corrente Il processo viene sospeso nel momento immediatamente precedente alla successiva attività manuale
voidApp	Tool nullo			

Metodi di accesso al CMDB

Tool	Descrizione	Parametri input	Parametri output	Note
createCard	Crea una nuova scheda e restituisce l'”Id”	ClassName *string* CardCode *string* CardDescription *string*	CardId *integer*	Il metodo valorizza solo gli attributi base “Code” e “Description” Per valorizzare gli altri si deve utilizzare il tool updateAttribute o definire un metatool di tipo createCard
createCardRef	Crea una nuova scheda e restituisce il reference	ClassName *string* CardCode *string* CardDescription *string*	CardReference *reference*	Il metodo valorizza solo gli attributi base “Code” e “Description” Per valorizzare gli altri si deve utilizzare il tool updateAttribute o definire un metatool di tipo createCard
createRelation	Crea una relazione fra due schede	DomainName *string* ClassName1 *string* ClassName2 *string* ObjId1 *integer* ObjId2 *integer*	Done *boolean*	
createRelation1Ref	Crea una relazione fra due schede di	DomainName *string*	Done *boolean*	

	cui la prima indicata tramite reference	ObjReference1 *reference* ClassName2 *string* ObjId2 *integer*		
createRelation2Ref	Crea una relazione fra due schede di cui la seconda indicata tramite reference	DomainName *string* ClassName1 *string* ObjId1 *integer* ObjReference2 *reference*	Done *boolean*	
createRelationRefs	Crea una relazione fra due schede entrambe indicate tramite reference	DomainName *string* ObjReference1 *reference* ObjReference2 *reference*	Done *boolean*	
deleteRelation	Elimina una relazione tra due schede	DomainName *string* ClassName1 *string* ClassName2 *string* ObjId1 *integer* ObjId2 *integer*	Done *boolean*	
deleteRelationByReference	Elimina una relazione fra due schede entrambe indicate tramite reference	DomainName *string* ObjReference1 *reference* ObjReference2 *reference*	Done *boolean*	
selectAttribute	Legge un attributo della scheda specificata	ClassName *string* AttributeName *string* ObjId *integer*	AttributeValue *string*	Il valore restituito è sempre rappresentato tramite stringa
selectAttributeFromReference	Legge un attributo della scheda specificata, indicata tramite reference	ObjReference *reference* AttributeName *string*	AttributeValue *string*	Il valore restituito è sempre rappresentato tramite stringa
selectLookup	Legge la descrizione di una voce Lookup, indicata tramite tipo e “Id”	Type *string* LookupId *integer*	LookupDescription *string*	
selectLookupById	Restituisce una voce Lookup, indicata tramite “Id”	LookupId *integer*	Lookup *lookup*	
selectLookupByTypeDesc	Restituisce una voce Lookup, indicata tramite tipo e descrizione	Type *string* Description *string*	Lookup *lookup*	
selectLookupByTypeCode	Restituisce una voce Lookup, indicata tramite tipo e code	Type *string* Code *string*	Lookup *lookup*	

selectReferenceByCode	Restituisce un oggetto reference corrispondente alla scheda indicata tramite codice	ClassName *string* Code *string*	OutRef *reference*	
selectReferenceByCustomAttribute	Restituisce un oggetto reference corrispondente alla scheda indicata tramite un attributo generico	ClassName *string* AttributeName *string* AttributeValue *string*	OutRef *reference*	
selectReferenceByReference	Restituisce un oggetto reference corrispondente ad un attributo di tipo reference presente nella scheda indicata tramite reference	ObjReference *reference* *AttributeName string*	OutRef *reference*	
selectRelations	Restituisce un array di reference corrispondenti alle card in relazione con la card identificata dall'id e dalla classe indicati, su uno specifico dominio	ClassName *string* CardId *integer* *DomainName string*	RefArray *relations*	
selectRelationsByReference	Restituisce un array di reference corrispondenti alle card in relazione con la card identificata dal reference indicato, su uno specifico dominio	ClassName *string* CardId *integer* *DomainName string*	RefArray *relations*	
updateAttribute	Modifica una scheda	ClassName *string* AttributeName *string* ObjId *integer* AttributeValue *string*	Done *boolean*	Il metodo modifica solo l'attributo indicato Per modificare più attributi in blocco si deve definire un metatool di tipo updateCard
updateAttributeRef	Modifica una scheda indicata tramite reference	ObjRef *reference* AttributeName *string* AttributeValue *string*	Done *boolean*	

Metodi esterni

Tool	Descrizione	Parametri input	Parametri output	Note
getCurrentTimestamp	Restituisce la data e ora di sistema		TheDate *date*	
getCurrentGroupReference	Restituisce una variabile di tipo reference corrispondente al gruppo dell'utente corrente		GroupRef *reference*	L'oggetto restituito corrisponde alla scheda del gruppo corrente nella tabella del CMDB “Role”
getCurrentUserReference	Restituisce una variabile di tipo reference corrispondente all'utente corrente		UserRef *reference*	L'oggetto restituito corrisponde alla scheda dell'utente corrente nella tabella del CMDB “User”
getReportFullUrl	Restituisce il link al report creato con l'extended attribute createReport	ReportUrl *string*	ReportUrl *string*	
sendMail	Invia una mail	FromAddresses *string* ToAddresses *string* CCAddresses *string* BCCAddresses *string* Subject *string* Content *string* UrlAttachments *string* MimeType *string*		Il tool presuppone siano correttamente configurati i parametri di Shark relativi all'invio di mail I parametri From, To e Attach possono contenere più valori concatenati con “,” I parametri CCAddresses, BCCAddresses e UrlAttachments possono essere valorizzati con una stringa vuota MimeType può assumere i valori “text/html” o “text/plain”

Template metodi automatici utilizzabili nel workflow

Per l'utilizzo con Together Workflow Server 2.3 (superato dal sistema basato su Together Workflow Server 4.4) CMDBuild rende disponibili alcuni template di metodi automatici (meta-tool), utilizzabili per la definizione di tool effettivi.

Per la creazione dei nuovi “tool” custom CMDBuild prevede i seguenti passaggi:

- creazione in TWE Together Workflow Editor 4.4 di una nuova “Application” (la lista è accessibile dalle proprietà del processo) con l'apposito pulsante “Create new element”
- completamento della definizione dell' “Application” cliccando sulla nuova riga aggiunta alla lista e impostando i seguenti parametri:
  - Id = nome che si vuole attribuire al nuovo “tool”
  - Name = per semplicità può essere impostato lo stesso valore scelto per il campo precedente
  - Formal parameters = aggiungere tanti parametri di input e di output quanti ne prevede il tool da creare (come descritto alla tabella successiva)
  - Extended attribute “ToolAgentClass”
  - eventuali ulteriori extended attribute specifici del singolo meta-tool (come descritto alla tabella successiva)

Tipo template	**Descrizione**	**Parametri input**	**Parametri output**	**Note**
createCard	Creazione scheda nel CMDB	Lista attributi da valorizzare sulla nuova scheda oppure ClassName *string* Lista parametri di input previsti dalla funzione	CardReference *reference*	Restituisce l'id della scheda creata La seconda forma di specifica dei parametri di input è utilizzabile se si esclude ClassName dalla lista degli attributi esterni (vedi tabella successiva)
createReport	Esecuzione report	Lista parametri di input previsti dal report	ReportURL *string*	L'URL restituita può essere utilizzata per allegare il report ad una mail con il tool sendMail
executeFunction	Esecuzione funzioni PostgreSQL	Lista parametri di input previsti dalla funzione	Lista parametri di output previsti dalla funzione	Deve sempre essere presente almeno un parametro di input e uno di output, anche fittizio
startProcess	Avvio istanza altro processo	Lista attributi da valorizzare all'avvio del processo	ProcessInstanceId *string*	Restituisce il nome dell'istanza del processo (di tipo stringa)
updateCard	Aggiornamento scheda nel CMDB	ClassName *string* ObjId *integer* Lista attributi da aggiornare sulla	Done *boolean*	

		scheda oppure ObjRef *reference* Lista parametri di input previsti dalla funzione		
updateProcess	Salvataggio o avanzamento istanza altro processo	ProcessInstanceId *string* Lista attributi da valorizzare	Done *boolean*	

Per motivi di migliore leggibilità viene riportata separatamente nella tabella successiva l'indicazione del ToolAgent e di altri eventuali attributi da specificare in TWE nella definizione del metatool.
Tutti i valori degli attributi sono di tipo stringa.

Tipo template	**Attributo metatool**	**Valore metatool**
createCard	ToolAgentClass ClassName	org.cmdbuild.shark.toolagent.CreateCardToolAgent [Nome classe]
createReport	ToolAgentClass Type Code Format	org.cmdbuild.shark.toolagent.CreateReportToolAgent custom [Codice report] pdf oppure csv
executeFunction	ToolAgentClass Procedure oppure CursorProcedure	org.cmdbuild.shark.toolagent.ExecuteStoredProcedureToolAgent [nome funzione PostgreSQL con ritorno valore singolo] [nome funzione PostgreSQL con ritorno valori multipli]
startProcess	ToolAgentClass ProcessClass Complete	org.cmdbuild.shark.toolagent.ProcessStartToolAgent [Nome classe] 1 (per avanzare il processo all'attività successiva) oppure 0 (per fermare il processo sulla prima attività)
updateCard	ToolAgentClass	org.cmdbuild.shark.toolagent.UpdateAttributeToolAgent
updateProcess	ToolAgentClass ProcessClass Complete	org.cmdbuild.shark.toolagent.ProcessUpdateToolAgent [Nome classe] 1 (per avanzare il processo all'attività successiva) oppure 0 (per fermare il processo sulla prima attività) Se il processo è in stato “Sospeso” deve prima essere eseguito il metodo resumeProcess.

APPENDICE: Glossario

ALLEGATO

Per “allegato” si intende un qualunque file associabile ad una scheda dati inserita nel sistema.

Per la gestione degli allegati CMDBuild utilizza in modalità embedded un qualunque sistema documentale compatibile con il protocollo standard CMIS (oppure il DMS Alfresco fino alla versione 3 tramite il proprio webservice nativo).

La gestione degli allegati supporta il versioning di file caricati più volte, con numerazione automatica.

ATTIVITA'

Per “attività” si intende uno dei passaggi che costituiscono il processo.

Una attività è caratterizzata da un nome, un esecutore, un tipo, eventuali attributi, eventuali metodi associati ad API di CMDBuild per poter essere eseguiti.

Per “istanza di attività” si intende una specifica attivazione di una attività, effettuata automaticamente dal sistema o manualmente da un operatore.

Vedi anche: Processo

ATTRIBUTO

Il termine indica nel sistema CMDBuild la generica tipologia di informazione descrittiva di una determinata classe.

CMDBuild consente tramite il Modulo Schema di creare nuovi attributi in una classe o in un dominio e di modificarne alcune caratteristiche.

Nella classe “Fornitore” gli attributi sono ad esempio il nome, l'indirizzo, il numero di telefono, ecc.

Ogni attributo corrisponde nel Modulo di Gestione a campi di inserimento dati sulla apposita scheda di gestione della classe e a colonne della corrispondente tabella nel database.

Vedi anche: Classe, Dominio, Relazione, Superclasse, Tipo di attributo

BIM

Metodologia che si pone l'obiettivo di supportare l’intero ciclo di vita di un edificio, dall'idea iniziale alla fase di costruzione, di utilizzo e manutenzione, fino alla eventuale demolizione finale.

La metodologia BIM (Building Information Modeling) è supportata da numerosi programmi informatici che possono interagire tramite un formato aperto di scambio dati denominato IFC (Industry Foundation Classes).

Vedi anche: GIS

CI

Si definisce Configuration Item (Elemento della Configurazione) ogni elemento che concorre a fornire il servizio IT all’Utente, considerato ad un livello di dettaglio sufficiente per la sua gestione tecnica e patrimoniale.

Esempi di CI sono: server, workstation, programma applicativo, sistema operativo, stampante, ecc

Vedi anche: Configurazione

CLASSE

Il termine rappresenta un tipo di dati complesso caratterizzato da un insieme di attributi che nel loro insieme descrivono quel tipo di dato.

Una classe modella una tipologia di oggetto da gestire nel CMDB, quale ad esempio un computer, una applicazione software, un servizio, un fornitore, ecc

CMDBuild consente all'Amministratore del Sistema, attraverso il Modulo Schema, di definire nuove classi e di cancellare o modificare la struttura di classi già definite.

Una classe è rappresentata a video da una apposita scheda di gestione dati e nel database da una tavola generata automaticamente al momento della definizione della classe.

Vedi anche: Scheda, Attributo

CONFIGURAZIONE

Il processo di Gestione della Configurazione ha lo scopo di mantenere aggiornata e disponibile per gli altri processi la base di informazioni relativa agli oggetti informatici gestiti (CI), alle loro relazioni ed alla loro storia.

E' uno dei principali processi gestiti dal sistema ITIL.

Vedi anche: CI, ITIL

DASHBOARD

Una dashboard corrisponde in CMDBuild ad una raccolta di grafici di diversa tipologia, tramite cui avere immediata evidenza di alcuni parametri chiave (KPI) relativi ad un particolare aspetto di gestione del servizio IT.

Vedi anche: Report

DATABASE

Il termine indica un insieme di informazioni strutturato ed organizzato in archivi residenti sull'elaboratore server, nonché l'insieme dei programmi di utilità dedicati alla gestione dei tali informazioni per attività quali inizializzazione, allocazione degli spazi, ottimizzazione, backup, ecc.

CMDBuild si appoggia sul database PostgreSQL, il più potente, affidabile e completo database Open Source, di cui utilizza in particolare le sofisticate funzionalità e caratteristiche object oriented.

DOMINIO

Un dominio rappresenta una tipologia di relazione fra una coppia di classi.

E' caratterizzato da un nome, dalle descrizioni della funzione diretta ed inversa, dai codici delle due classi e dalla cardinalità (numerosità degli elementi relazionabili) ammessa, nonché dagli eventuali attributi configurati.

CMDBuild consente all'Amministratore del Sistema, attraverso il Modulo Schema, di definire nuovi domini e di cancellare o modificare la struttura di domini già definiti.

E' possibile caratterizzare ciascun dominio tramite definizione di attributi custom.

Vedi anche: Classe, Relazione

FILTRO DATI

Un filtro dati è una restrizione della lista degli elementi contenuti in una classe, ottenuta specificando condizioni booleane (uguale, diverso, contiene, inizia, ecc) sui possibili valori assumibili da ciascun attributo della classe.

I filtri dati possono essere definiti ed utilizzati “una tantum”, oppure possono essere memorizzati dall'operatore e richiamati successivamente (dallo stesso operatore o da operatori di altri gruppi di utenti ai quali l'Amministratore del sistema abbia concesso l'utilizzo).

Vedi anche: Classe, Vista

GIS

Un sistema GIS è un sistema informatico in grado di produrre, gestire e analizzare dati spaziali associando a ciascun elemento geografico una o più descrizioni alfanumeriche.

Le funzionalità GIS implementate in CMDBuild consentono di creare attributi geometrici, in aggiunta a quelli testuali, tramite cui rappresentare su scala locale (planimetrie) o su scala più estesa (mappe esterne) elementi puntuali (ad esempio gli asset IT), poligonali (ad esempio linee dati) o aree (piani, stanze, ecc).

Vedi anche: BIM

GUI FRAMEWORK

E' una interfaccia utente completamente personalizzabile e orientata a fornire un accesso semplificato all'applicazione, pubblicabile su portali web di qualsiasi tecnologia ed interoperabile con CMDBuild tramite il webservice REST standard.

Vedi anche: Mobile, Webservice

ITIL

Sistema di "best practice" ormai affermatosi come "standard de facto", non proprietario, per la gestione dei servizi informatici secondo criteri orientati ai processi (Information Technology Infrastructure Library).

Fra i processi fondamentali coperti da ITIL ci sono quelli del Service Support, comprendenti l'Incident Management, il Problem Management, il Change Management, il Configuration Management ed il Release Management.

Per ogni processo considera la descrizione, i componenti di base, i criteri e gli strumenti consigliati per la misura della qualità del servizio, i ruoli e le responsabilità delle risorse coinvolte, i punti di integrazione con gli altri processi (per eliminare duplicazioni e inefficienze).

Vedi anche: Configurazione

LOOKUP

Con il termine “LookUp” si indica una coppia di valori del tipo (Codice, Descrizione) impostabili dall'Amministratore del Sistema tramite il Modulo Schema.

Tali valori vengono utilizzati dall'applicazione per vincolare la scelta dell'utente, al momento della compilazione del relativo campo sulla scheda dati, ad uno dei valori preimpostati.

Il Modulo Schema consente la definizione di nuove tabelle di “LookUp” secondo le necessità dell'organizzazione.

MOBILE

E' una interfaccia utente ottimizzata per strumenti “mobile” (smartphone e tablet), implementata come “app” multipiattaforma (iOS, Android) ed interoperabile con CMDBuild tramite il webservice REST standard.

Vedi anche: GUI Framework, Webservice

PROCESSO

Per “processo” (o workflow) si intende una sequenza di passaggi (“attività”) descritti nel sistema per svolgere in forma guidata e secondo regole prestabilite una determinata azione.

Per ogni processo saranno avviate in CMDBuild una serie di “istanze di processo”, una per ogni necessità di effettiva esecuzione dell'azione corrispondente, che avrà luogo su “asset” specifici e sarà svolta da utenti specifici.

Una ”istanza di processo” viene attivata tramite avvio e conferma del primo passaggio previsto e termina alla esecuzione dell'attività finale prevista nella definizione.

Vedi anche: Attività

RELAZIONE

Per “Relazione” si intende in CMDBuild un collegamento effettivo di due schede appartenenti a due classi, o in altri termini una istanza di un dato dominio.

Una relazione è quindi definita da una coppia di identificativi univoci delle due schede collegate e dall'identificativo del dominio utilizzato per il collegamento, nonché dalla valorizzazione degli eventuali attributi previsti nel dominio.

CMDBuild consente agli operatori del Sistema, attraverso il Modulo Gestione Dati, di definire nuove relazioni fra le schede archiviate nel database.

Vedi anche: Classe, Dominio

REPORT

Il termine indica in CMDBuild una stampa (in formato PDF o CSV) riportante in forma analitica le informazioni estratte da una o più classi fra le quali sia definita una catena di domini.

I report possono essere generati e modificati dagli operatori di CMDBuild tramite una apposita funzione del Modulo di Gestione Dati e la relativa definizione viene memorizzata nel database per poter essere riutilizzata successivamente.

Vedi anche: Classe, Dominio, Database

SCHEDA

Con il termine “Scheda” in CMDBuild si riferisce un elemento archiviato in una determinata classe.

Una scheda è caratterizzata da un insieme di valori assunti da ciascuno degli attributi definiti per la sua classe di appartenenza.

CMDBuild consente agli operatori del Sistema, attraverso il Modulo Gestione Dati, di archiviare nuove schede nel database e di aggiornare schede già archiviate.

Le informazioni di ogni scheda saranno memorizzate nel database alle opportune colonne di una riga della tavola generata per la classe di appartenenza della scheda.

Vedi anche: Classe, Attributo

SUPERCLASSE

Una superclasse è una classe astratta utilizzabile per definire una sola volta attributi condivisi fra più classi. Da tale classe astratta è poi possibile derivare classi reali che conterranno i dati effettivi e che comprenderanno sia gli attributi condivisi (specificati nella superclasse) che quelli specifici della sottoclasse.

Ad esempio è possibile definire la superclasse “Computer” con alcuni attributi base (RAM, HD, ecc) e le sottoclassi derivate “Desktop”, “Notebook”, “Server”, ciascuna delle quali con i soli attributi specifici.

Vedi anche: Classe, Attributo

TIPO DI ATTRIBUTO

Ogni attributo definito per una determinata classe è caratterizzato da un “Tipo” che determina le caratteristiche delle informazioni contenute e la loro modalità di gestione.

Il tipo di attributo viene definito con il Modulo Schema e può essere poi modificato entro alcuni limiti dipendenti dalla tipologia dei dati già archiviati.

CMDBuild gestisce i seguenti tipi di attributo: “Boolean” (booleano, Si / No), “Date” (data), “Decimal” (decimale), “Double” (virgola mobile in doppia precisione), “Inet” (indirizzo IP), “Integer” (numero intero), “LookUp” (tabellato da lista configurabile in “Impostazioni” / “LookUp”), “Reference” (riferimento o foreign key), “String” (stringa), “Text” (testo lungo), “TimeStamp” (data e ora).

Vedi anche: Attributo

VISTA

Una vista è un insieme di schede definito in modo “logico” anziché dal fatto di costituire l'intero contenuto di una classe nel CMDB.

In particolare una vista può essere definita in CMDBuild applicando un filtro ad una classe (quindi conterrà un insieme ridotto delle stesse righe) oppure specificando una funzione SQL che estragga attributi da una o più classi correlate.

La prima tipologia di vista mantiene tutte le funzionalità disponibili per una classe, la seconda consente la sola visualizzazione e ricerca con filtro veloce.

Vedi anche: Classe, Filtro

WEBSERVICE

Un webservice è un'interfaccia che descrive una collezione di operazioni, accessibili attraverso una rete mediante messaggistica XML.

Tramite un webservice una applicazione può rendere accessibili le proprie funzionalità ad altre applicazioni operanti attraverso il web.

CMDBuild dispone di un webservice SOAP e di un webservice REST.

WIDGET

Un widget è un componente grafico di una interfaccia utente di una applicazione software, che ha lo scopo di facilitare all'utente l'interazione con l'applicazione stessa.

CMDBuild prevede l'utilizzo di widget sotto forma di “pulsanti” posizionabili su schede dati o su schede di avanzamento di processi. I pulsanti aprono finestre di tipo “popup” tramite cui inserire se richiesto informazioni aggiuntive e visualizzare poi l'output della funzione richiamata.