Versione

3.4

CMDBuild®

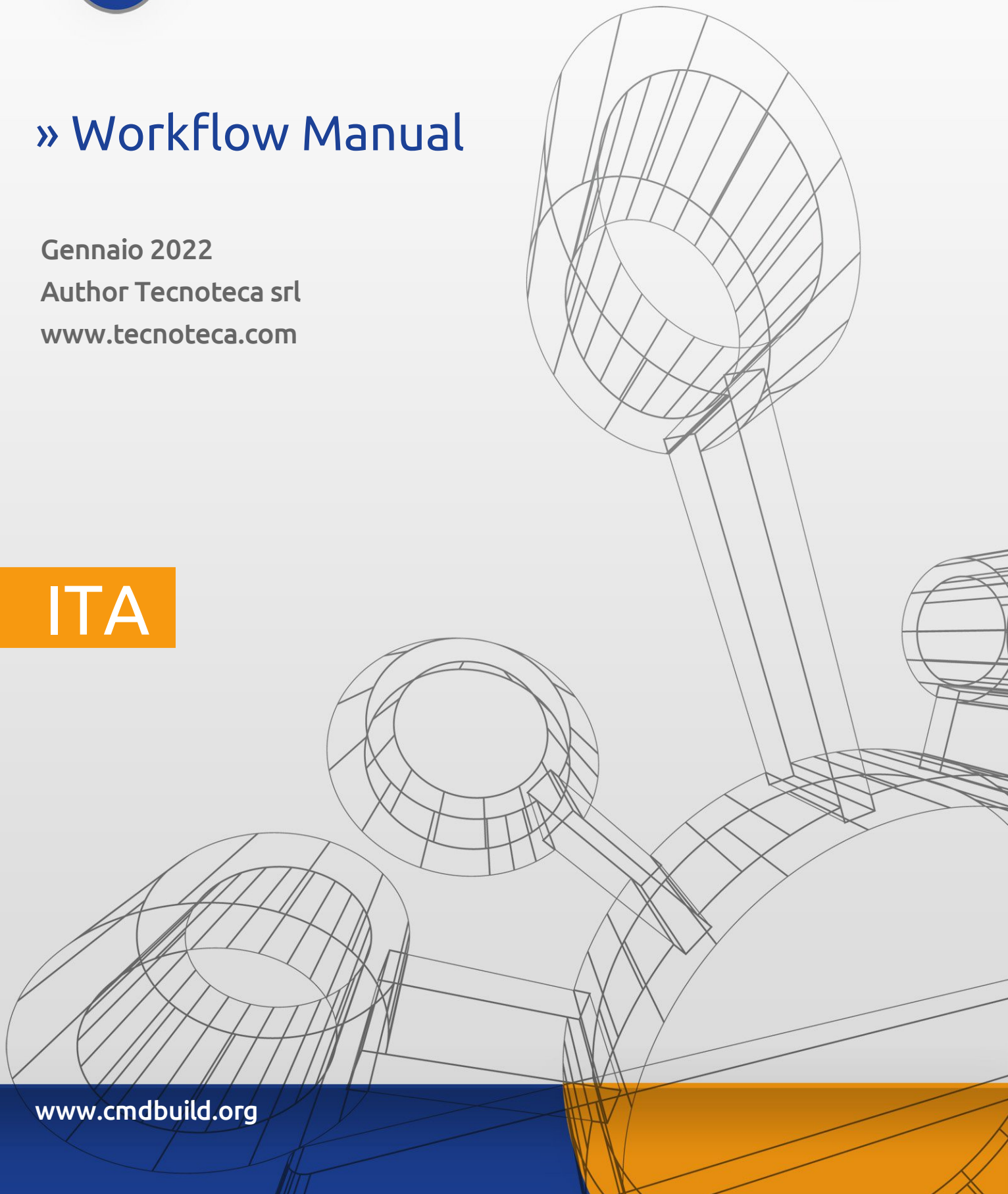
» Workflow Manual

Gennaio 2022

Author Tecnoteca srl

www.tecnoteca.com

ITA



CMDBuild ® leverages many great technologies from the open source community:
PostgreSQL, Apache, Tomcat, Eclipse, Ext JS, JasperStudio, Enhydra Shark, TWE, OCS Inventory, Liferay, Alfresco, GeoServer, OpenLayers, Quartz, BiMserver, Xeokit.
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.

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

Sommario

1. Introduzione....4
1.1. Descrizione dell'applicazione....4
1.2. Sito web del progetto....5
1.3. I moduli di CMDBuild....5
1.4. Manualistica disponibile....5
1.5. Applicazioni basate su CMDBuild....6
2. Descrizione del sistema di workflow....7
2.1. Generalità....7
2.2. Obiettivi....7
2.3. Strumenti utilizzati....7
2.4. Terminologia....8
3. Modalità di implementazione....10
3.1. Workflow come classi particolari....10
3.2. Costruzione del flusso del processo....10
3.3. Definizione di un nuovo processo....11
3.4. Avvio ed avanzamento di un processo....12
4. Widget utilizzabili nelle attività utente del workflow....14
4.1. Lista widget....14
4.1.1. Informazioni ulteriori per l'utilizzo degli "string template" nel tool manageEmail....17
4.1.2. Esempio 1....18
4.1.3. Esempio 2....18
5. API utilizzabili nelle attività automatiche del workflow....20
5.1. Parole chiave....20
5.2. Gestione degli oggetti di CMDBuild....20
5.2.1. ReferenceType....20
5.3. Metodi di accesso al CMDBuild....23
5.3.1. NewCard....23
5.3.2. ExistingCard....23
5.3.3. NewProcessInstance....26
5.3.4. ExistingProcessInstance....27
5.3.7. NewRelation....28
5.3.8. ExistingRelation....29
5.3.9. QueryClass....29
5.3.10. QueryLookup....30
5.3.11. CallFunction....30
5.3.12. QueryRelations....31
5.3.13. CreateReport....32
5.3.14. NewMail....32
5.4. Metodi per la conversione di tipi....34
5.4.1. ReferenceType....34
5.4.2. LookupType....35
5.4.3. CardDescriptor....35
5.4.4. Card....35
6. Appendice: Glossario....37

1. Introduzione

1.1. Descrizione dell'applicazione

CMDBuild è un ambiente web open source tramite cui è possibile configurare applicazioni personalizzate per l'Asset Management.

Da un lato dispone di meccanismi nativi per l'amministratore, implementati in un codice "core" mantenuto separato dalla logica di business, per configurare il sistema in tutte le sue funzionalità.

Dall'altro genera dinamicamente una interfaccia web per gli operatori, consentendo loro di mantenere sotto controllo la situazione degli asset, di conoscerne in ogni momento la composizione, la dislocazione, le relazioni funzionali e le modalità di aggiornamento nel tempo, per gestirne il ciclo di vita in modo completo.

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 una apposita interfaccia che consente di aggiungere progressivamente nuove classi di oggetti, nuovi attributi e nuove tipologie di relazioni. Può definire filtri, "viste" e permessi di accesso ristretti a righe e colonne di ciascuna classe.

L'amministratore può disegnare in modo visuale, con un editor esterno, workflow operanti sulle classi modellate nel database, importarli in CMDBuild e metterli a disposizione degli operatori che li eseguiranno secondo i flussi previsti e con gli automatismi configurati.

In modo analogo può disegnare in modo visuale, con un editor esterno, report di diverso genere (tabulati, stampe con grafici, etichette barcode, ecc) sui dati del CMDB, importarli nel sistema e metterli a disposizione degli operatori.

Può poi configurare delle dashboard, costituite da grafici che mostrino in modo immediato la situazione di alcuni indicatori dello stato corrente del sistema (KPI).

Un task manager incluso nell'interfaccia utente del Modulo di Amministrazione consente di schedulare in background diverse tipologie di operazioni (avvio di processi, ricezione e invio di mail, esecuzione di connettori) e diverse tipologie di controlli sui dati del CMDB (eventi sincroni e asincroni) a fronte dei quali inviare notifiche, avviare workflow ed eseguire script.

L'interoperabilità con altri sistemi è gestita tramite il CMDBuild Service BUS, denominato WaterWAY.

Grazie all'integrazione con sistemi documentali che supportano lo standard CMIS (Content Management Interoperability Services), fra cui la diffusissima soluzione open source Alfresco, gli operatori potranno allegare alle schede archiviate in CMDBuild documenti, immagini, video ed altre tipologie di file.

E' poi disponibile uno scadenziario, alimentabile sia automaticamente alla compilazione di una scheda dati che manualmente, per gestire scadenza singole o ricorrenti, relative ad esempio a certificazioni, garanzie, contratti con clienti e fornitori, adempimenti amministrativi, ecc.

E' anche possibile utilizzare funzionalità GIS per il georiferimento degli asset e la loro visualizzazione su una mappa geografica (servizi mappe esterni) e/o su planimetrie vettoriali (server locale GeoServer e database spaziale PostGIS) e funzionalità BIM per la visualizzazione di modelli 3D basati su file in formato IFC.

E' poi incluso nel sistema un webservice REST, tramite cui gli utilizzatori di CMDBuild possono implementare soluzioni personalizzate di interoperabilità con sistemi esterni.

CMDBuild comprende inoltre due framework esterni:

- il CMDBuild Advanced Connector, scritto in linguaggio Java e configurabile in Groovy, che tramite tramite logiche native per la sincronizzazione di dati agevola la implementazione di connettori con fonti dati esterne, ad esempio con sistemi di automatic inventory o di virtualizzazione o di monitoraggio (fornito con licenza non open source a chi sottoscrive la Subscription annuale con Tecnoteca)
- il CMDBuild GUI Framework, che agevola la implementazione di interfacce grafiche aggiuntive, ad esempio pagine web semplificate per utenti non tecnici, da pubblicare su portali esterni (suggerita la soluzione open source Liferay) ed in grado di interagire con il CMDB tramite il webservice REST

CMDBuild dispone infine di una interfaccia “mobile” (per smartphone e tablet), implementata come “APP” multipiattaforma (iOS, Android) ed anch'essa in grado di interagire con il CMDB tramite il webservice REST (fornita con licenza non open source a chi sottoscrive la Subscription annuale con Tecnoteca).

CMDBuild è un sistema web enterprise: Java lato server, GUI web Ajax, architettura SOA (Service Oriented Architecture) basata su webservice, implementato riutilizzando le migliori tecnologie open source e seguendo gli standard di settore.

CMDBuild è un sistema in continua evoluzione, rilasciato per la prima volta nel 2006 ed aggiornato con più rilasci annuali per offrire sempre nuove funzionalità ed il supporto delle nuove tecnologie.

1.2. Sito web del progetto

CMDBuild dispone di un sito web dedicato al progetto: http://www.cmdbuild.org

Il sito raccoglie una estesa documentazione per chi desidera approfondire la caratteristiche tecniche e funzionali del progetto: brochure, slide, manuali (vedi paragrafo successivo), testimonianze, case history, newsletter, forum.

1.3. 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.

1.4. Manualistica disponibile

Il presente manuale è dedicato alla descrizione del Modulo di Amministrazione, tramite cui il gestore del sistema può eseguire la configurazione del modello dati, definire utenti e permessi ed eseguire altre attività di servizio.

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

- overview concettuale del sistema (“Overview Manual”)
- utilizzo del sistema da parte degli operatori (“User Manual”)

- amministrazione del sistema (“Administrator Manual”)
- installazione e gestione tecnica del sistema (“Technical Manual”)
- utilizzo del webservice per l'interoperabilità con sistemi esterni (“Webservice Manual”)

1.5. Applicazioni basate su CMDBuild

Tecnoteca ha utilizzato il proprio ambiente CMDBuild per implementare due diverse soluzioni preconfigurate:

- CMDBuild READY2USE, per la gestione degli asset e dei servizi IT, orientato ad infrastrutture IT interne o a servizi erogati a clienti esterni (www.cmdbuildready2use.org) secondo le best practice ITIL (Information Technology Infrastructure Library)
- openMAINT, per la gestione dell’inventario di asset ed impianti di patrimoni immobiliari e delle relative attività di manutenzione preventiva e a guasto (www.openmaint.org)

Entrambe le applicazioni sono rilasciate con licenza open source, con esclusione di alcune componenti esterne (connettori di sincronizzazione dati, portale Self-Service, APP mobile, ecc), riservate a chi sottoscrive la Subscription annuale con Tecnoteca.

2. Descrizione del sistema di workflow

2.1. Generalità

Un importante valore aggiunto di CMDBuild è la possibilità di definire processi tramite cui gli operatori eseguono in modo collaborativo e controllato le attività di gestione previste.

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 non implementare una serie di processi rigidi e predefiniti, ma un sistema generico tramite il quale utenti esperti 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.

Nella seconda parte del documento vengono invece documentati 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.

2.2. Obiettivi

L'utilizzo di un sistema di workflow garantisce:

- 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

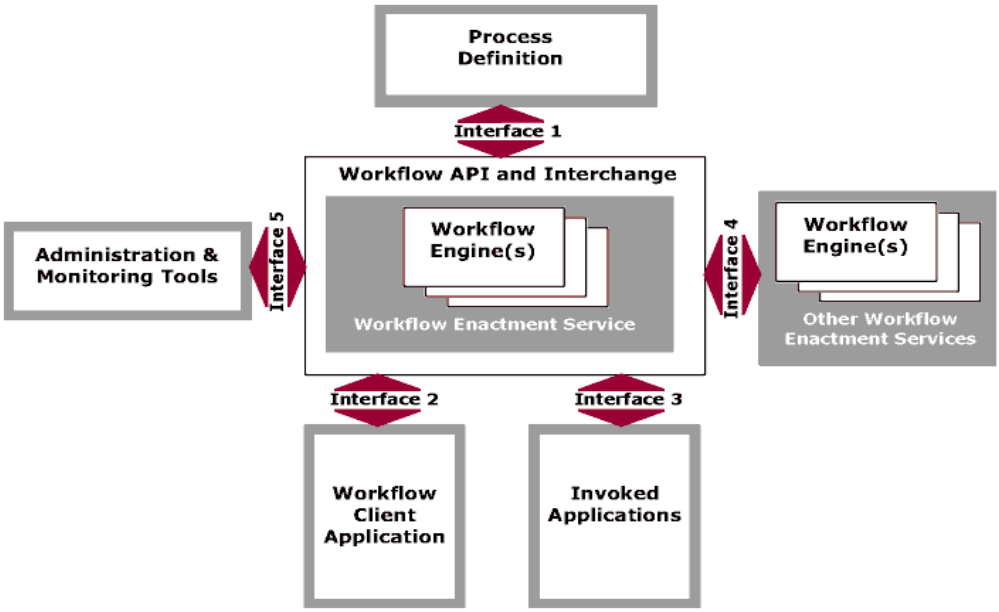
Nell'ambito IT con i meccanismi disponibili possono essere ad esempio configurati tutti i processi previsti dalle "best practice" ITIL, inclusi quelli di Incident Management, Change Management, Request Fulfillment, Service Catalog, ecc, nell'ambito del Facility Management possono essere configurati tutti i processi della manutenzione.

2.3. 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 Tecnoteca River che fornisce una implementazione standard delle specifiche WfMC (http://www.wfmc.org/) per la parte di interesse della gestione dei workflow in CMDBuild, utilizzando al suo interno XPDL come linguaggio nativo
- l'editor visuale TWE Together Workflow Editor 5.5 o precedente (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.



2.4. Terminologia

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

- processo: una sequenza di passaggi (“attività”) descritti nel sistema per svolgere una determinata azione in forma guidata e secondo regole prestabilite
- attività: uno dei passaggi che costituiscono il flusso del processo
- istanza di processo una specifica attivazione di un “processo”, effettuata tramite avvio e conferma del primo passaggio previsto
- istanza di attività: 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”, cioè fermo in una attività intermedia fino a ripresa dell’esecuzione sucessiva

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

3. Modalità di implementazione

3.1. Workflow come classi particolari

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

La gestione del workflow comprende:

- 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

3.2. 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, 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

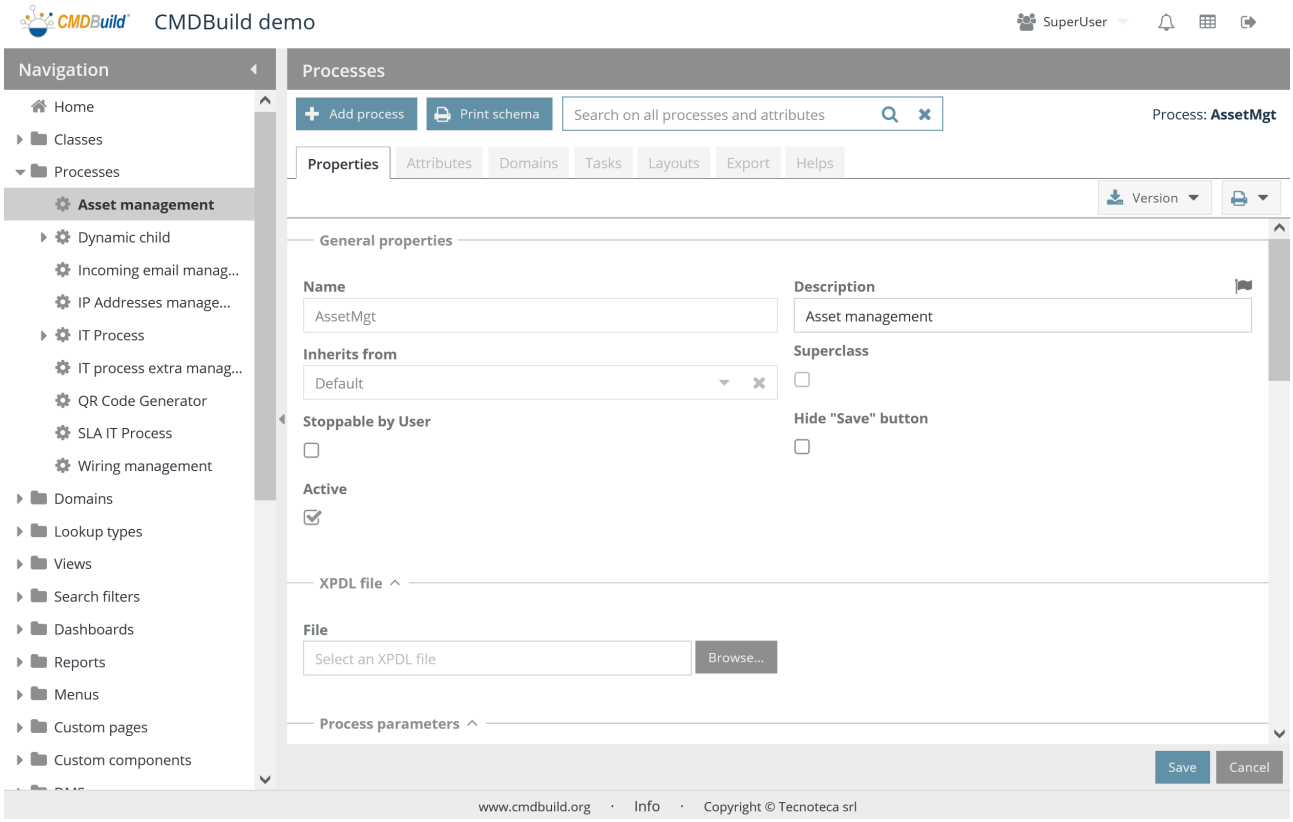
3.3. Definizione di un nuovo processo

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 Tecnoteca River.

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.



3.4. Avvio ed avanzamento di un processo

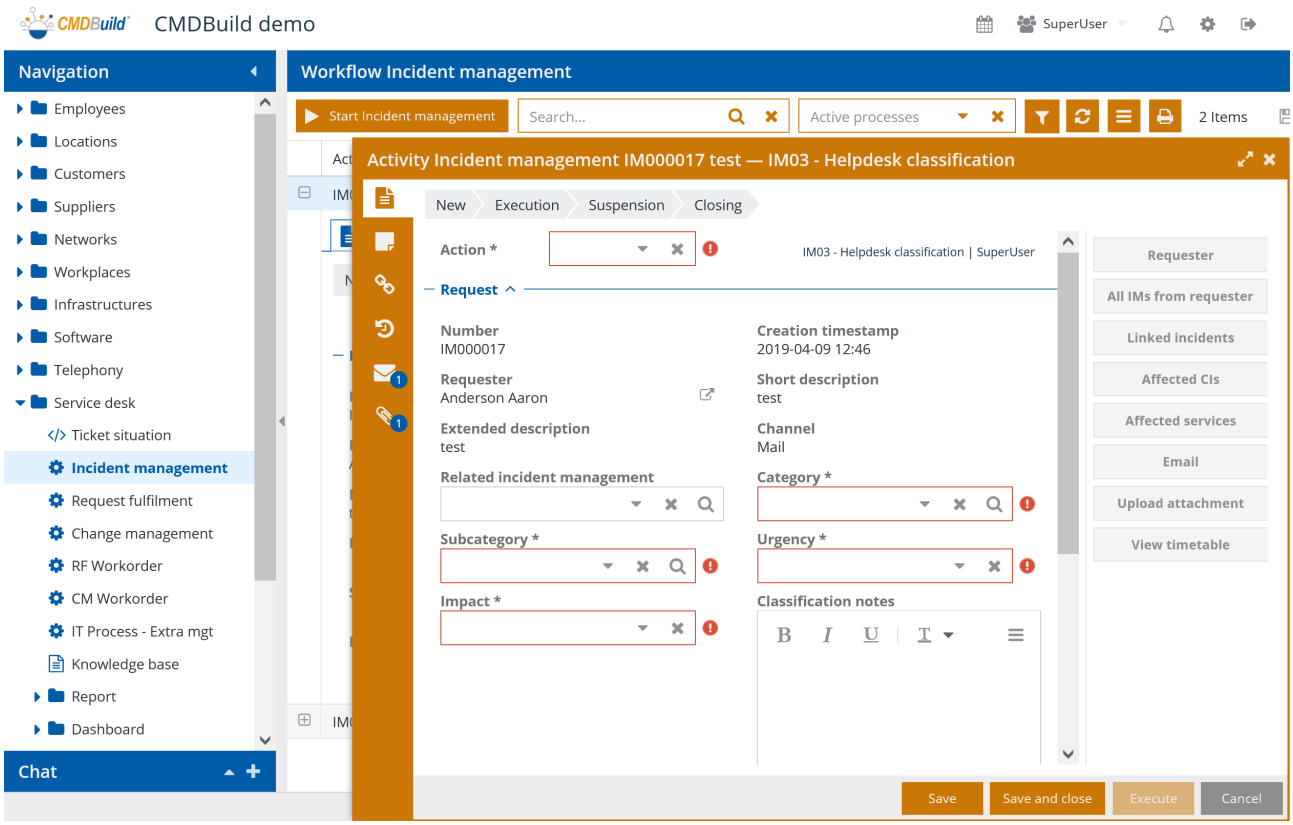
L'applicazione CMDBuild comprende nel Modulo Gestione la possibilità di interpretare, tramite il supporto del motore di workflow Tecnoteca River, i processi disegnati esternamente con TWE Together Workflow Editor e poi importati tramite il Modulo di Amministrazione.

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, chiamata "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)
- nel TAB “Email” è possibile visualizzare tutte le email collegate all’istanza di processo in questione
- nel TAB “Allegati” (se il servizio DMS è attivo), è possibile visualizzare tutti gli allegati collegati all’istanza di processo in questione

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.



4. Widget utilizzabili nelle attività utente del workflow

4.1. 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
Widget_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* DisableGridFilterToggler *boolean* Output: CheckArray *references*	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 = {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”
Widget_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*	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 integer oppure Input: ClassName *string* ObjId *integer* ButtonLabel *string* *ReadOnly integer* Output: Reference *reference*	ReadOnly=1 presenta la card in sola lettura
Widget_createReport		Input: ReportCode *string* ButtonLabel *string* ForcePDF *integer* ForceCSV *integer* Parametro-1 Parametro-2 ... Parametro-n Output: ReportURL *string*	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
Widget_manageEmail	Permette di produrre tramite template o scrivere email libere che verranno inviate all'avanzamento del processo.	Input: ButtonLabel *string* Template*1* *string* Condition*1* *string* NotifyWith*1* *string* *...*	Alla richiesta di visualizzazione delle email viene controllata la casella di posta per nuove email Template*1* … Template*n* corrisponde all’attributo “Nome” del template email Condition*1* … Condition*n* è una espressione *cql* che abilita o meno la creazione della email con il Template1 NotifyWith*1* … NotifyWith*n* è il nome del template da utilizzare per l’invio di una email di notifica
Widget_openNote	Visualizza la pagina comprendente l'editor HTML per l'inserimento di note	Input: ButtonLabel *string*	Non utilizzabile nella prima attività di un processo
Widget_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
Widget_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). EventStartDate indica l’attributo da utilizzare come data di inizio dell’evento. EventEndDate indica l’attributo da utilizzare

		EventType *string* EventTypeLookup *string* DefaultDate *string*	come data di fine dell'evento. È opzionale. EventTitle indica l'attributo da cui prelevare il testo da riportare sul calendario per ogni evento. EventType indica l'attributo da utilizzare per identificare la tipologia dell'evento. EventTypeLookup è il LookupType da cui prelevare i valori da usare come tipologie di eventi. Solitamente è il LookupType dell'attributo "EventType". DefaultDate indica l'attributo da utilizzare da cui leggere la data sul quale aprire il calendario. È opzionale.
Widget_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.
Widget_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.
Widget_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]"* ImportDisabled *"[true|false]"* ModifyDisabled *"[true|*	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 function – i valori di output di una funzione I dati possono essere serializzati come tipo

		false]" SerializationType *"[json|text]"* KeyValueSeparator *string* AttributesSeparator *string* RowsSeparator *string* Output: Variabile di output *string*	testo (si veda il widget grid) o come tipo json.

4.1.1. 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="..."
...

4.1.2. 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

4.1.3. Esempio 2

...
Content="Il richiedente, **{js:JoinJS}**, appartenente all'ufficio **{cql:QueryRichiedente.Ufficio_value}** richiede:

{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

5. 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.

5.1. Parole chiave

Processo
ProcessId: Long 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

5.2. 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.

5.2.1. ReferenceType

Metodi
getId(): Long restituisce l'id del Reference
getDescription(): String restituisce la descrizione del Reference

LookupType

Metodi
getId(): Long 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(): Long 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

5.3. Metodi di accesso al CMDBuild

5.3.1. 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();
```

5.3.2. ExistingCard

Costruttori

existingCard(String className, Long 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
copyToAndMerge() Se è stato usato il modificatore attachments, il metodo copia un allegato selezionato della card in una destinazione specificata saltandolo se già esistente per la card target
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();
```

5.3.3. 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();
```

5.3.4. ExistingProcessInstance

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

5.3.5.

Modificatori
withProcessInstanceId(String value): ExistingProcessInstance imposta il process instance id
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

5.3.6.

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();
```

5.3.7. NewRelation

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

Modificatori
withCard1(String className, long cardId): NewRelation imposta la card al lato sorgente della relazione
withCard2(String className, long cardId): NewRelation imposta la card al lato destinazione della relazione
withAttribute(String attributeName, Object attributeValue): NewRelation imposta il valore di un attributo 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();
```

5.3.8. ExistingRelation

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

Modificatori
withCard1(String className, long cardId): ExistingRelation imposta l'IdClass e l'ObjId della Card dal lato sorgente della relazione
withCard2(String className, long 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();
```

5.3.9. 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();
```

5.3.10. QueryLookup

Builders

queryLookup(String type): QueryAllLookup
istanzia una query per richiedere a CMDBuild le lookup del tipo specificato

Actions

fetch(): Iterable<Lookup>
esegue la query su CMDBuild utilizzando il tipo fornito precedentemente

5.3.11. 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)".

5.3.12. QueryRelations

Costruttori
queryRelations(CardDescriptor cardDescriptor): ActiveQueryRelations istanzia una query per richiedere a CMDBuild le Card in relazione con quella specificata
queryRelations(String className, long 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();
```

5.3.13. 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();
```

5.3.14. 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 il destinataro della mail da inviare
withTo(String... tos): NewMail imposta i destinatari della mail da inviare
withTo(Iterable<String> tos): NewMail imposta i destinatari della mail da inviare
withCc(String cc): NewMail imposta il destinatario in copia carbone della mail da inviare
withCc(String... ccs): NewMail imposta i destinatari in copia carbone della mail da inviare
withCc(Iterable<String> ccs): NewMail imposta i destinatari in copia carbone della mail da inviare
withBcc(String bcc): NewMail imposta il destinatario in copia carbone nascosta della mail da inviare
withBcc(String... bccs): NewMail imposta i destinatari in copia carbone nascosta della mail da inviare
withBcc(Iterable<String> bccs): NewMail imposta i destinatari in copia carbone 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
withAttachment(String url): NewMail imposta l'url (come stringa) di un documento da allegare alla mail
withAttachment(URL url, String name): NewMail imposta l'url di un documento con il nome specificato da allegare alla mail
withAttachment(String url, String name): NewMail imposta l'url di un documento (come stringa) con il nome specificato da allegare alla mail
withAttachment(DataHandler dataHandler): NewMail imposta il Data Handler come allegato della email
withAttachment(DataHandler dataHandler, String name): NewMail imposta il Data Handler con il nome specificato come allegato della email
withCard(@Nullable String className, @Nullable Long cardId): NewMail

imposta la card collegata alla email da inviare
withCard(@Nullable CardDescriptor card): NewMail imposta la card collegata alla email da inviare
withCard(@Nullable ReferenceType card): NewMail imposta la card collegata alla email da inviare
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();
```

5.4. Metodi per la conversione di tipi

5.4.1. 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(long 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);
```

5.4.2. LookupType

Metodi
selectLookupById(long 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");
```

5.4.3. 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);
```

5.4.4. 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);
```

6. 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.

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

Vedi anche: Scheda dati

ATTIVITÀ

Per “attività” si intende uno dei passaggi che costituiscono il flusso di un processo (workflow).

Una attività può essere costituita da una interazione con l’operatore (interattiva) o può essere costituita da uno script che esegue operazioni via API (automatica).

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 (ad esempio nella classe “Fornitore” gli attributi possono essere il nome, l'indirizzo, il numero di telefono, ecc).

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

Ogni attributo corrisponde nel database ad una colonna nella tabella che implementa la classe di appartenenza e corrisponde nel Modulo di Gestione Dati ad un campo di data entry sulla apposita scheda di gestione della classe.

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).

CMDBuild include un connettore per sincronizzare alcune informazioni (anagrafiche, tecniche o manutentive) di alcuni oggetti (CI o Configuration Item) e un visualizzatore interattivo del modello 3D dell’edificio rappresentato dal file IFC.

Vedi anche: CI, GIS

CI

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

Il termine viene applicato in CMDBuild ad un generico contesto di Asset Management estendendo il concetto normalmente utilizzato nella gestione delle infrastrutture informatiche.

Esempi di CI sono: server, workstation, programma applicativo, impianto, quadro elettrico, estintore, arredo, ecc

Vedi anche: Configurazione, ITIL

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'azienda, un edificio, un asset, un servizio, ecc

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

Una classe è rappresentata nel database da una tabella generata automaticamente al momento della definizione della classe e corrisponde nel Modulo di Gestione Dati ad una scheda di consultazione e aggiornamento delle informazioni previste nel modello.

Vedi anche: Scheda, Attributo

CMDB

Le "best practice" ITIL (Information Technology Infrastructure Library), ormai affermatosi come "standard de facto" non proprietario per la gestione dei servizi informatici, hanno introdotto il termine CMDB per indicare il database dei Configuration Item.

CMDBuild estende il concetto di CMDB per applicarlo ad un generico contesto di Asset Management.

Vedi anche: Database, ITIL

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 (Configuration Item) gestiti, alle loro relazioni ed alla loro storia.

Pur essendo conosciuto come uno dei principali processi delle Best Practice ITIL utilizzate nella gestione IT, in CMDBuild lo stesso concetto viene applicato a contesti generici di Asset Management.

Vedi anche: CI, ITIL

DASHBOARD

Una dashboard corrisponde in CMDBuild ad una pagina dell'applicazione contenente una o più rappresentazioni grafiche di diversa tipologia, tramite cui avere immediata evidenza di alcuni parametri chiave (KPI) relativi ad aspetti di gestione del servizio di Asset Management erogato.

Vedi anche: Report

DATABASE

Il termine indica un insieme di informazioni strutturate ed organizzate 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.

Il database di Asset Management implementato sulla base delle logiche e della filosofia di CMDBuild è anche indicato come CMDB (Configuration Management Data Base).

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 di Amministrazione, di definire nuovi domini e di cancellare o modificare la struttura di domini già definiti.

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

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, oppure possono essere configurati dall’Amministratore e resi disponibili agli operatori.

Vedi anche: Classe, Vista

GIS

Un sistema GIS è un sistema informatico in grado di produrre, gestire e analizzare dati spaziali 2D 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 territoriali) elementi puntuali (ad esempio gli asset), poligonali (ad esempio linee di trasmissione) o aree (piani, stanze, ecc).

Vedi anche: BIM

GUI FRAMEWORK

E' un framework messo a disposizione da CMDBuild per semplificare l’implementazione di interfacce utente esterne personalizzate tramite cui fornire un accesso semplificato al personale non tecnico, pubblicabili su portali web di qualsiasi tecnologia ed interoperabili con CMDBuild tramite il webservice REST standard.

Il CMDBuild GUI Framework è basato sulle librerie javascript JQuery.

Vedi anche: Mobile, Webservice

ITIL

E’ un 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, il Change Management ed il Configuration Management.

Per ogni processo vengono definite informazioni descrittive, 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).

CMDBuild riprende alcuni concetti di ITIL e li applica più in generale ad un generico contesto di Asset Management.

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 di Amministrazione.

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 (detti anche valori a scelta chiusa o picklist).

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

Vedi anche: Tipo di attributo

MOBILE

E' una interfaccia utente ottimizzata per strumenti utilizzabili in mobilità come smartphone e tablet.

E’ implementata come “APP” multipiattaforma (iOS, Android) ed è interoperabile con CMDBuild tramite il webservice REST standard.

Vedi anche: Webservice

PROCESSO

Per “processo” (o workflow) si intende una sequenza di passaggi (“attività”) definiti in CMDBuild per svolgere una determinata azione in forma guidata, collaborativa e secondo regole prestabilite.

Per ogni processo (tipologia di processo) sarà avviata in CMDBuild una nuova “istanza di processo” ogni qual volta si debba svolgere una determinata azione su oggetti (asset) da parte di utenti (appartenenti a ruoli) seguendo una procedura implementata sotto forma di workflow.

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

Il flusso dei processi gestiti in CMDBuild è descritto nel linguaggio di markup standard XPDL (XML Process Definition Language), normato dalla WFMC (WorkFlow Management Coalition).

Vedi anche: Attività

RELAZIONE

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

Una relazione è 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, attraverso il Modulo di Gestione Dati, di definire nuove relazioni fra le schede presenti nel CMDB.

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 configurati nel Modulo di Amministrazione importando in formato XML la descrizione del layout disegnato tramite l'editor visuale JasperReports e messi a disposizione degli operatori nel menu dell'applicazione.

I report possono essere poi stampati dagli operatori di CMDBuild dal Modulo di Gestione Dati, sotto forma di tabulati, stampe con grafici, documenti, etichette, ecc.

Vedi anche: Classe, Dominio, Database

SCHEDA DATI

Con il termine "Scheda dati" in CMDBuild si riferisce un elemento archiviato in una determinata classe (corrispondente al record di una tabella nel database).

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

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

Le informazioni di ciascuna scheda dati (attributi) sono memorizzate nel database nelle opportune colonne di una riga della tabella corrisponde alla classe su cui si sta operando.

Vedi anche: Classe, Attributo

SUPERCLASSE

Una superclasse è una classe astratta utilizzabile come "template" per definire una sola volta attributi e domini condivisi fra più sottoclassi. Da tale classe astratta, o da gerarchie di classi astratte, 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, oltre che le relazioni sui domini della superclasse e sui propri domini specifici.

Ad esempio è possibile definire la superclasse "Azienda" con alcuni attributi base (Partita IVA, Ragione sociale, Telefono, ecc) e le sottoclassi derivate "Cliente" e "Fornitore", ciascuna delle quali comprenderà sia gli attributi generici della superclasse che i propri attributi e le proprie relazioni.

Vedi anche: Classe, Attributo

TENANT

In CMDBuild un "tenant" è una partizione del CMDB riservata agli utenti appartenenti ad una sottoorganizzazione dell'istanza complessiva di CMDBuild (una Società del Gruppo, una Sede, una Divisione, ecc).

Lavorando in modalità "multitenant" ogni utente opera esclusivamente sui dati della propria sottoorganizzazione, ed eventualmente su dati dichiarati comuni a tutti i "tenant".

La lista dei "tenant" utilizzabili può essere definita da una classe applicativa di CMDBuild oppure da una funzione custom di database (in cui è possibile implementare regole di visibilità più complesse).

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 e le sue modalità di gestione vengono definite con il Modulo di Amministrazione.

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 tramite criteri “logici” di filtro applicati ad una o più classi del 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 informazioni e le proprie funzionalità ad altre applicazioni operanti attraverso il web.

CMDBuild dispone di un webservice SOAP e di un webservice REST, che vengono resi disponibili ad applicazioni esterne per leggere o scrivere dati nel CMDB o per eseguire operazioni.

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 consultare o inserire dati o eseguire altre operazioni.

CMDBuild include un insieme standard di widget per lo svolgimento delle operazioni più frequenti, ma fornisce anche le specifiche per implementare altri widget personalizzati.