



# **CMDBuild**

***Open Source Configuration and Management Database  
Technical Manual***

**Versione 0.5.4  
Marzo 2007**

No part of this document may be reproduced, in whole or in part, without the express written permission of Tecnoteca s.r.l.

CMDBuild leverages many great technologies from the open source community:  
PostgreSQL, Apache, Tomcat, Eclipse, JasperReports, IReport  
We are thankful for the great contributions that led to the creation of that products

CMDBuild è un progetto realizzato e gestito da:

Comune di Udine – Servizio Sistemi Informativi e Telematici



Tecnoteca S.r.l. ([www.tecnoteca.com](http://www.tecnoteca.com))



Cogitek S.r.l. ([www.cogitek.it](http://www.cogitek.it))



CMDBuild è rilasciato con licenza GPL ([www.gnu.org/copyleft/gpl.html](http://www.gnu.org/copyleft/gpl.html))  
Copyright ©2006 Tecnoteca srl

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

## Sommario

Introduzione.....	4
I moduli di CMDBuild.....	4
Installazione del sistema.....	5
Requisiti hardware.....	5
Requisiti software.....	5
Installazione del sistema.....	6
Installazione del database.....	6
Installazione dell'applicazione.....	7
Configurazione dell'applicazione.....	7
Test di funzionamento dell'applicazione.....	8
Struttura dell'applicazione.....	11
Generalità.....	11
Albero delle classi.....	11
Diagrammi UML.....	12
Progettazione del database .....	15
Criteri di base.....	15
Descrittori integrativi.....	20
Funzioni.....	21
Viste.....	24
API di CMDBuild.....	25
Generalità.....	25
Lista delle API.....	25
Attuali limitazioni note del sistema.....	26
Export / Import.....	26
Campi tipo "reference".....	26

# Introduzione

CMDBuild è una applicazione Open Source per la configurazione e gestione del database della configurazione (CMDB) degli oggetti in uso presso il Dipartimento IT di una organizzazione.

Gestire un Database della Configurazione 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.

CMDBuild si ispira alle "best practice" ITIL (Information Technology Infrastructure Library), ormai affermatesi come "standard de facto", non proprietario, per la gestione dei servizi informatici secondo criteri orientati ai processi.

Con CMDBuild l'amministratore del sistema può costruire autonomamente il proprio CMDB (da cui il nome del progetto), grazie ad un apposito programma di configurazione che consente di aggiungere progressivamente nel sistema nuove classi di oggetti, nuovi attributi e nuove tipologie di relazioni.

## I moduli di CMDBuild

Il sistema CMDBuild comprende due moduli principali:

- il Modulo Schema, dedicato alla definizione iniziale ed alle successive modifiche della struttura dati (classi e sottoclassi, attributi delle classi, tipologie di relazioni fra classi)
- il Modulo Gestione Dati, dedicato all'inserimento ed aggiornamento nel sistema dei dati descrittivi e delle relazioni funzionali fra le diverse entità, nonché alla produzione di report e tabulati.

Il presente manuale è dedicato ai tecnici informatici cui sono demandate le attività di installazione delle componenti software e di amministrazione del database.

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

- Overview concettuale del sistema
- Administrator Manual
- User Manual

## Installazione del sistema

L'installazione di CMDBuild richiede l'utilizzo di uno o più server su cui suddividere le componenti logiche costitutive del sistema:

- server web
- componenti di elaborazione
- database

Vengono descritti di seguito i requisiti software richiesti da CMDBuild, le modalità di installazione e configurazione, i criteri di verifica e test.

Nella progettazione dell'infrastruttura sistemistica va considerato che l'attivazione di applicazioni web come quella in oggetto richiede la disponibilità di componenti hardware e di rete dotate di adeguati livelli di sicurezza, sia rispetto accessi esterni indesiderati (firewall, DMZ) che rispetto le esigenze di disponibilità continuativa on line del sistema (backup della linea internet, mirroring hardware) e di adeguate prestazioni di accesso.

### Requisiti hardware

Per l'installazione di CMDBuild è richiesto un computer di classe server di recente generazione, avente le seguenti caratteristiche dimensionali minime:

- memoria RAM minimo 1 GB (consigliati 2 GB)
- spazio disco minimo 20 GB (consigliati 40 GB in previsione del prossimo rilascio della funzione di gestione allegati)

Sono altresì consigliati:

- la presenza di un sistema di dischi in configurazione RAID
- un sistema di backup giornaliero dei dati
- un sistema di continuità elettrica per preservare il server da interruzioni anomale dell'alimentazione

### Requisiti software

L'installazione di CMDBuild richiede la presenza dei componenti software di seguito elencati.

#### Sistema operativo

Qualunque sistema operativo supporti gli applicativi sotto elencati (consigliato il sistema operativo Linux sul quale CMDBuild è soggetto a test più estesi).

#### Database

PostgreSQL 8.0 o superiore (consigliato 8.1), l'installazione base è sufficiente, accertarsi che sia attivato il supporto al linguaggio "plpgsql".

Sito di riferimento: <http://www.postgresql.org/>

#### Web Server

CMDBuild richiede l'installazione di Apache 2.0 e di Jakarta Tomcat 4.5 o superiori (consigliato 5.0, Tomcat 6.0 non è al momento supportato).

Per la versione Tomcat 5.5 sono richieste librerie aggiuntive, posizionate nella cartella “tomcat55libs”.

Sito di riferimento per entrambi: <http://www.apache.org/>

### Librerie Java

Le librerie Java sono necessarie per il funzionamento di Jakarta Tomcat.

CMDBuild richiede JDK 1.5 o superiori.

Sito di riferimento: <http://www.sun.com/>

### Librerie incluse nel rilascio

CMDBuild contiene una serie di librerie già all'interno del pacchetto di installazione, ed in particolare:

- postgresql-8.0-313.jdbc3.jar per il collegamento a PostgreSQL 8.0
- commons-dbc-1.2.jar per la gestione del pool di connessione al database
- JasperReports-1.2.2.jar per la produzione di report
- shark\* per l'utilizzo del motore di workflow Enhydra Shark
- struts\* e commons\*, librerie utilizzate da Struts 1.2

Ulteriori informazioni su Struts si possono trovare sul sito di riferimento: <http://struts.apache.org/>

Per l'eventuale disegno di report custom è utilizzabile l'editor visuale IReport che produce il relativo descrittore in formato compatibile con il motore JasperReports.

Per l'eventuale disegno di workflow personalizzati è suggerito l'editor visuale JPED dal quale viene prodotto il relativo descrittore in formato XPDL compatibile con il motore Enhydra Shark.

Tutti i software sopra elencati sono rilasciati con licenza Open Source (ad eccezione eventualmente del Sistema Operativo se si optasse per una soluzione diversa da Linux).

## Installazione del sistema

Per l'installazione del sistema può essere predisposta una partizione unica, oppure due partizioni dedicate rispettivamente ai programmi e ai dati. Nella seconda ipotesi la partizione riservata ai dati comprenderà sia le tabelle del database PostgreSQL che una cartella sul filesystem ove archiviare i file allegati alle schede di database (immagini, documenti, ecc).

Come tipo di filesystem in ambiente Linux possono essere utilizzati indifferentemente “riscfs” o “ext3”, sono sconsigliate altre opzioni meno sperimentate.

## Contenuti del rilascio

Il file “tar.gz” rilasciato contiene i seguenti componenti:

INSTALL.txt	file guida, i cui contenuti sono riportati ai punti successivi del presente documento
GPL.txt	termini della licenza
CMDBuild.Sql	comandi SQL per la creazione di un database vuoto
CMDBuild-demo.Sql	comandi SQL per la creazione di un database di test
\tomcat55libs	Librerie aggiuntive richieste da Tomcat 5.5 e 6.0
\cmdb	file di CMDBuild per Tomcat
\cmdb\WEB-INF\src	codice sorgente
\cmdb\WEB-INF\lib	librerie Java
\cmdb\WEB-INF\classes	classi Java

## Installazione del database

Per l'installazione del database vanno effettuate le seguenti operazioni:

- tramite uno strumento con interfaccia grafica (ad esempio pgAdmin3 di PostgreSQL) o da linea di comando creare il database utilizzando il nome "cmdb" o altro equivalente:

```
CREATE DATABASE cmdb
WITH OWNER = postgres
ENCODING = 'UNICODE'
TABLESPACE = pg_default;
```

- accertarsi che l'encoding sia UNICODE
- eseguire i comandi SQL contenuti nei file CMDBuild.sql o CMDBuild-demo.sql (quest'ultimo contiene dei dati di esempio). Se si è creato il database tramite interfaccia grafica (ad es. PGAdmin) accertarsi che non esista nessun language associato al database, in caso contrario commentare (aggiungendo all'inizio i caratteri "--") le seguenti righe:

```
-- CREATE TRUSTED PROCEDURAL LANGUAGE 'plpgsql'
-- HANDLER plpgsql_call_handler;
```

## Installazione dell'applicazione

Copiare la directory "cmdb" nella sottocartella "webapps" della directory ove è installato Tomcat (eseguire i passi successivi prima di avviarlo).

## Configurazione dell'applicazione

La configurazione di CMDBuild è strettamente dipendente dalla specifica versione di Tomcat utilizzata, essendo presenti significative differenze fra le recenti versioni rilasciate.

Il funzionamento di CMDBuild è stato verificato sulle seguenti versioni di Tomcat: 4.1.34, 5.0.28, 5.5.20 and 6.0.10 .

Le configurazioni da effettuare riguardano i seguenti passaggi:

- configurazione del file di contesto per i parametri relativi all'applicazione CMDBuild
- configurazione del file di contesto per i parametri relativi all'utilizzo del database
- posizionamento del file di contesto
- installazione libreria interfaccia PostgreSQL
- eventuale installazione librerie aggiuntive
- avvio di Tomcat

### Passaggio a)

Il presente punto va eseguito solamente se CMDBuild è stato posizionato in una cartella diversa da quella di default:

```
TomcatDIR/webapps/cmdb
```

Supponendo di averlo posizionato nella cartella:

```
TomcatDIR/webapps/cmdbuild
```

editare il file:

TomcatDIR/webapps/cmdbuild/META-INF/context.xml

Modificare il nodo <Context>, posizionato all'inizio del file XML, come mostrato nei due casi sotto riportati:

```
<Context displayName="CMDBuild"
  path="/cmdb"
  workDir="work/Catalina/localhost/cmdb">

<Context displayName="CMDBuild"
  path="/cmdbuild"
  workDir="work/Catalina/localhost/cmdbuild">
```

### Passaggio b)

Il presente passaggio richiede di modificare con un normale editor il file di configurazione:

TomcatDIR/webapps/cmdb/META-INF/context.xml

per modificare:

- l'indirizzo del server che ospita PostgreSQL
- il nome del database scelto in fase di installazione
- username e password necessari a CMDBuild per accedere al database

Per comodità dell'utilizzatore abbiamo ripetuto la descrizione delle operazioni necessarie in funzione della versione di Tomcat utilizzata.

Il file è predisposto con attiva la sezione per Tomcat 4.1 e 5.0 (la versione al momento consigliata e impostata come default) e commentata la sezione per Tomcat 5.5 e 6.0 .

#### Tomcat 4.1 e 5.0

Spostarsi nella sezione:

```
<!-- BEGIN REGION TOMCAT 4.1 AND 5.0 -->
<Resource name="jdbc/cmdbSrc" type="javax.sql.DataSource"
  auth="Container"></Resource>
<ResourceParams name="jdbc/cmdbSrc">
  <parameter>
    <name>driverClassName</name>
    <value>org.postgresql.Driver</value>
  </parameter>
  <parameter>
    <name>url</name>
    <value>jdbc:postgresql://localhost/{DATABASE NAME}</value>
  </parameter>
  <parameter>
    <name>username</name>
    <value>{POSTGRES USER}</value>
  </parameter>
  <parameter>
    <name>password</name>
    <value>{POSTGRES PASSWORD}</value>
  </parameter>
  <parameter>
    <name>maxActive</name>
    <value>300</value>
  </parameter>
</parameter>
```



```
        <name>maxIdle</name>
        <value>2</value>
    </parameter>
    <parameter>
        <name>maxWait</name>
        <value>5000</value>
    </parameter>
    <parameter>
        <name>factory</name>
        <value>org.apache.commons.dbcp.BasicDataSourceFactory</value>
    </parameter>
</ResourceParams>
<!-- END REGION TOMCAT 4.1 AND 5.0 -->
```

Modificare i parametri sopra indicati (URL, username e password), salvare il file ed uscire.

### Tomcat 5.5 e 6.0

Spostarsi nella sezione:

```
<!-- BEGIN REGION TOMCAT 5.5 AND 6.0
<Resource name="jdbc/cmdbSrc" auth="Container" type="javax.sql.DataSource"
    factory="org.apache.commons.dbcp.BasicDataSourceFactory"
    maxActive="100" maxIdle="30" maxWait="10000"
    driverClassName="org.postgresql.Driver"
    username="{POSTGRES USER}"
    password="{POSTGRES PASSWORD}"
    url="jdbc:postgresql://localhost/{DATABASE NAME}"
    defaultAutoCommit="true" removeAbandoned="true"
    removeAbandonedTimeout="60" logAbandoned="true"/>
END REGION TOMCAT 5.5 AND 6.0 -->
```

Modificare i parametri sopra indicati (URL, username e password) e rendere attiva la sezione per Tomcat 5.5 e 6.0 modificando la prima riga:

```
<!-- BEGIN REGION TOMCAT 5.5 AND 6.0
```

in:

```
<!-- BEGIN REGION TOMCAT 5.5 AND 6.0 -->
```

e l'ultima riga:

```
END REGION TOMCAT 5.5 AND 6.0 -->
```

in:

```
<!-- END REGION TOMCAT 5.5 AND 6.0 -->
```

Spostarsi sulla sezione per Tomcat 4.1 e 5.0 e commentarla modificando la prima riga:

```
<!-- BEGIN REGION TOMCAT 4.1 AND 5.0 -->
```

in:

```
<!-- BEGIN REGION TOMCAT 4.1 AND 5.0
```

e l'ultima:

```
<!-- END REGION TOMCAT 4.1 AND 5.0 -->
```

in:

```
END REGION TOMCAT 4.1 AND 5.0 -->
```

Salvare il file ed uscire.

### **Passaggio c)**

Per comodità dell'utilizzatore abbiamo ripetuto la descrizione delle operazioni necessarie in funzione della versione di Tomcat utilizzata.

#### Tomcat 4.1

Spostare il file:

```
TomcatDIR/webapps/cmdb/META-INF/context.xml
```

in:

```
TomcatDIR/webapps/cmdb.xml
```

Nel caso l'applicazione CMDBuild fosse stato copiato in una directory diversa il file di contesto deve essere rinominato di conseguenza (ad esempio cmdbuild.xml se si fosse utilizzata la directory cmdbuild).

#### Tomcat 5.0

Spostare il file:

```
TomcatDIR/webapps/cmdb/META-INF/context.xml
```

in:

```
TomcatDIR/conf/Catalina/localhost/cmdb.xml
```

Nel caso l'applicazione CMDBuild fosse stato copiato in una directory diversa il file di contesto deve essere rinominato di conseguenza (ad esempio cmdbuild.xml se si fosse utilizzata la directory cmdbuild).

#### Tomcat 5.5 e 6.0

Il file:

```
TomcatDIR/webapps/cmdb/META-INF/context.xml
```

va lasciato nella posizione iniziale e con il nome iniziale.

### **Passaggio d)**

#### Tomcat 4.1, 5.0 e 5.5

Copiare la libreria (nel caso si utilizzino diverse versioni di PostgreSQL è consigliabile scaricare ed utilizzare la corrispondente versione della libreria di interfaccia):

```
postgresql-8.0-313.jdbc3.jar
```

nella cartella:

```
TomcatDIR/Common/lib
```

#### Tomcat 6.0

Copiare la libreria (nel caso si utilizzino diverse versioni di PostgreSQL è consigliabile scaricare ed utilizzare la corrispondente versione della libreria di interfaccia):

```
postgresql-8.0-313.jdbc3.jar
```

nella cartella:

```
TomcatDIR/lib
```

## Passaggio e)

### Tomcat 5.5 e 6.0

Nel caso si utilizzasse Tomcat 5.5 o 6.0 è richiesta la presenza di librerie aggiuntive:

- commons-collections
- commons-dbcp
- commons-pool

Le librerie sopra elencate vanno copiate dalla directory:

```
tomcat55libs/ *
```

nel caso di Tomcat 5.5 alla directory:

```
TomcatDIR/common/lib
```

nel caso di Tomcat 6.0 alla directory:

```
TomcatDIR/lib
```

## Passaggio f)

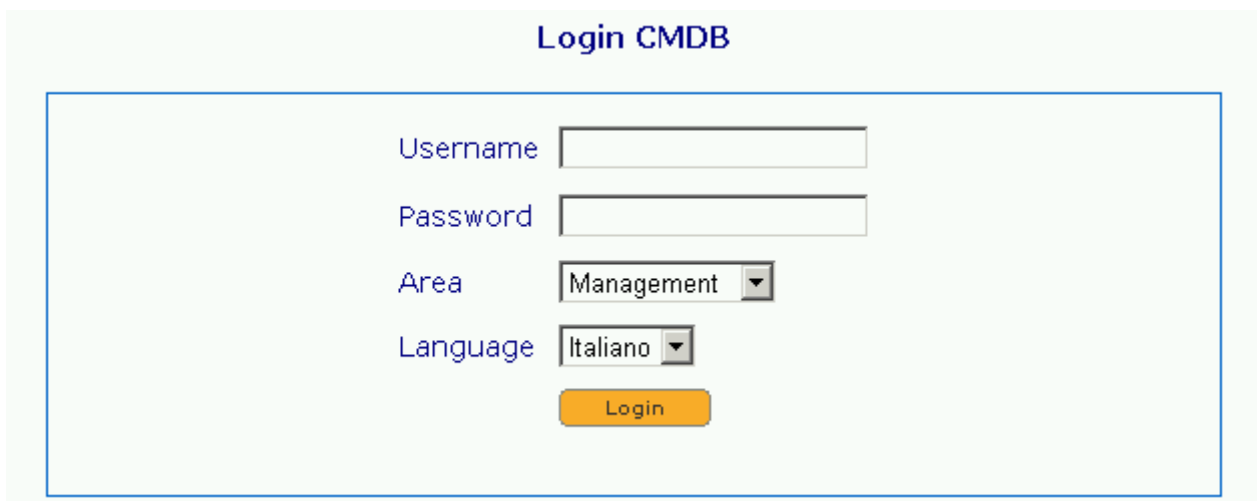
Al termine delle operazioni sopra descritte riavviare Tomcat.

## Test di funzionamento dell'applicazione

Alla conclusione delle operazioni descritte, se tutto è stato fatto correttamente, si potrà aprire il browser e accedere all'indirizzo <http://localhost:8080/cmdb/>

Nel caso sia stata configurata durante l'installazione di Tomcat una porta diversa da quella standard, tale porta dovrà essere sostituita alla 8080 all'interno della URL sopra specificata.

Si presenterà la seguente schermata:



Username

Password

Area

Language

Autenticandosi con l'account:

Username: admin

Password: admin

si potrà a questo punto accedere all'applicazione CMDBuild.

I manuali disponibili sul sito (Overview Document, Administrator Manual, User Manual) forniranno il supporto necessario per un efficace utilizzo del sistema.

Nel caso si presentasse invece la seguente schermata:

## HTTP Status 500 -

**type** Exception report

**message**

**description** The server encountered an internal error () that prevented it from fulfilling this request.

**exception**

```
org.apache.jasper.JasperException
  org.apache.jasper.servlet.JspServletWrapper.service (JspServletWrapper.java:372)
  org.apache.jasper.servlet.JspServlet.serviceJspFile (JspServlet.java:292)
  org.apache.jasper.servlet.JspServlet.service (JspServlet.java:236)
  javax.servlet.http.HttpServlet.service (HttpServlet.java:802)
```

**root cause**

```
java.lang.NullPointerException
  cmdbuild.action_forms.LoginForm.reset (LoginForm.java:53)
  org.apache.struts.taglib.html.FormTag.initFormBean (FormTag.java:460)
  org.apache.struts.taglib.html.FormTag.doStartTag (FormTag.java:433)
  org.apache.jsp.index_jsp._jspService (index_jsp.java:134)
  org.apache.jasper.runtime.HttpJspBase.service (HttpJspBase.java:94)
  javax.servlet.http.HttpServlet.service (HttpServlet.java:802)
  org.apache.jasper.servlet.JspServletWrapper.service (JspServletWrapper.java:324)
  org.apache.jasper.servlet.JspServlet.serviceJspFile (JspServlet.java:292)
  org.apache.jasper.servlet.JspServlet.service (JspServlet.java:236)
  javax.servlet.http.HttpServlet.service (HttpServlet.java:802)
```

**note** The full stack trace of the root cause is available in the Apache Tomcat/5.0.28 logs.

può essere ipotizzato un errore di connessione con il database, CMDBuild non trova cioè il database oppure sono sbagliati username e password per connettersi a PostgreSQL.

Si consiglia in tal caso di ritornare al paragrafo relativo alla configurazione di CMDBuild e di modificare il file struts-config.xml.

Nel log di Tomcat si possono trovare ulteriori informazioni sull'accaduto, quali ad esempio:

**GRAVE: Initializing application data source cmdbSource**

*org.apache.commons.dbcp.SQLNestedException: Cannot create PoolableConnectionFactory (Attivazione del backend fallita: FATAL: database "cmdbSource" does not exist.)*

Nel caso l'errore persista verificare che l'utente di amministrazione del database PostgreSQL (di default 'Postgres') disponga dei necessari permessi di accesso, in genere specificati nel file "pg\_hba.conf" escludendo ogni restrizione per l'accesso da localhost:

```
host all all 127.0.0.1/32 md5
```

# Struttura dell'applicazione

## Generalità

Gli strumenti individuati per la scrittura del software comprendono l'ambiente di sviluppo Eclipse e l'editor UML Omondo EclipseUML.

L'applicazione CMDBuild è stata realizzata nel linguaggio JAVA con utilizzo di pagine JSP per l'interfaccia web. L'applicazione è basata sul framework Struts che implementa il paradigma MVC (Model View Controller).

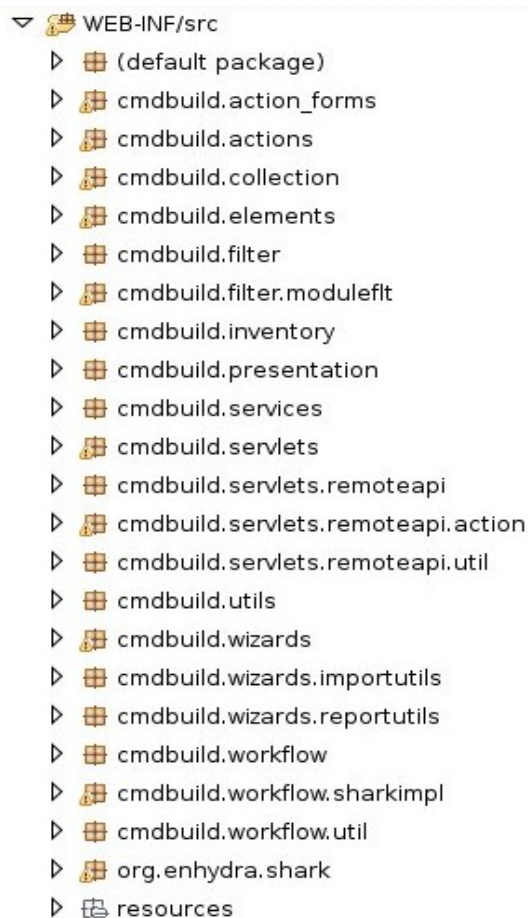
Il "modello" (responsabile tra l'altro della raccolta informazioni) è gestito da classi Java, così come il "controllore" responsabile del comportamento del sistema a seguito di un'azione dell'utente.

La "vista" è rappresentata dalle pagine JSP, che nei casi di generazione complessa del layout sfruttano alcune classi Java per la generazione di codice HTML.

Attraverso un semplice file XML di configurazione, Struts controlla CMDBuild e si occupa dell'interazione dei tre componenti sopracitati.

## Albero delle classi

Le classi sono organizzate secondo la seguente struttura:



I diversi insiemi di classi hanno il seguente significato:

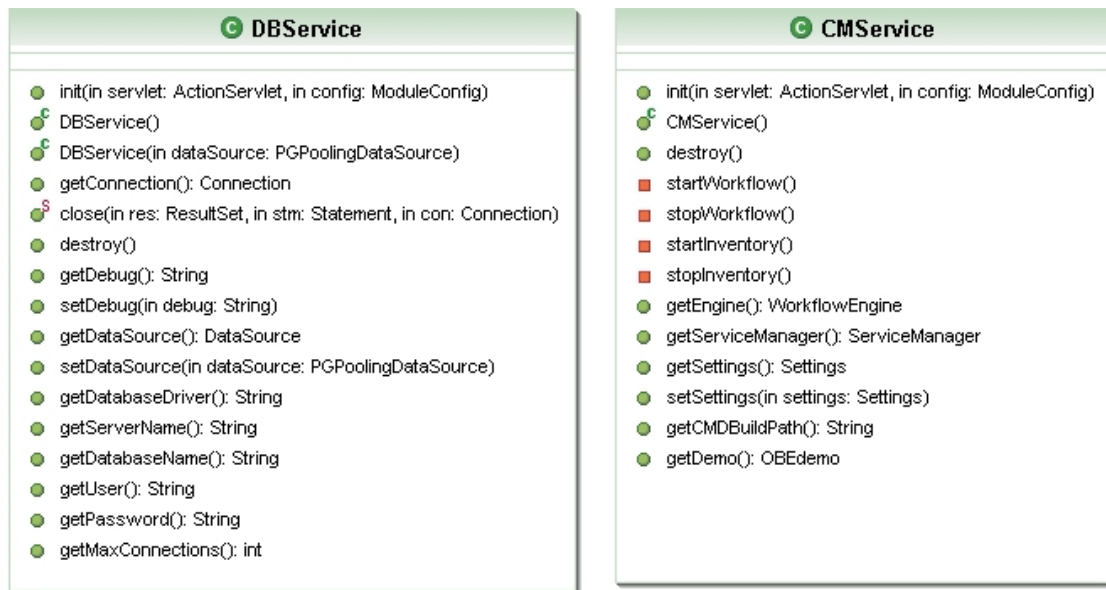
- `action_forms`: rappresentano il modello di Struts
- `actions`: rappresentano il controllore di “Struts”
- `collection`: insieme di classi utilizzate in CMDBuild per il trasporto delle informazioni
- `elements`: classi delegate all’interrogazione del database e gestori di CMDBuild.
- `Filter`: filtro delle richieste HTTP, configurabile
- `filter.moduleleft`: filtri per gestione workflow e remote api
- `inventory`: classi per l’integrazione con OCSInventory
- `presentation`: classi per la generazioni di codice HTML usate dai gestori
- `services`: classi “plugin” per la gestione globale di CMDBuild (impostazioni, database, workflow, etc)
- `servlets`: classe utilizzata per il flusso dei dati in pdf
- `servlets.remoteapi`: contiene le chiamate remote predisposte in CMDBuild
- `servlets.remoteapi.action`: azioni corrispondenti alle chiamate remote
- `servlets.remoteapi.util`: funzioni di utilità (ad esempio traduzione oggetti in stringhe e viceversa)
- `utils`: generiche classi di utilità
- `wizards`: classi wizard
- `wizard.importutils`: classi utili per il wizard di importazione
- `wizard.reportutils`: classi utili per il wizard dei report
- `workflow`: classi di definizione della struttura per la gestione del workflow
- `workflow.sharkimpl`: implementazione del supporto al workflow tramite Enhydra Shark
- `workflow.util`: classi di utilità per il workflow

## Diagrammi UML

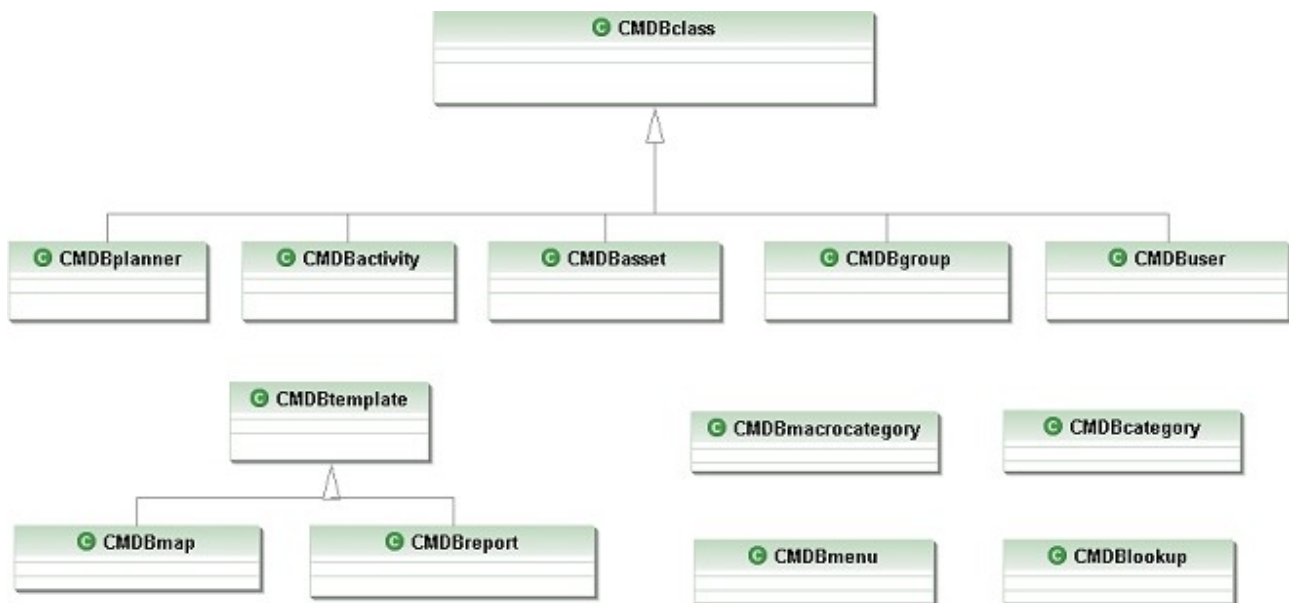
Si riportano di seguito alcuni diagrammi UML esemplificativi dell’architettura delle classi di CMDBuild.

Il seguente diagramma descrive la classe che gestisce la connessione al database e la classe per la gestione della configurazione.

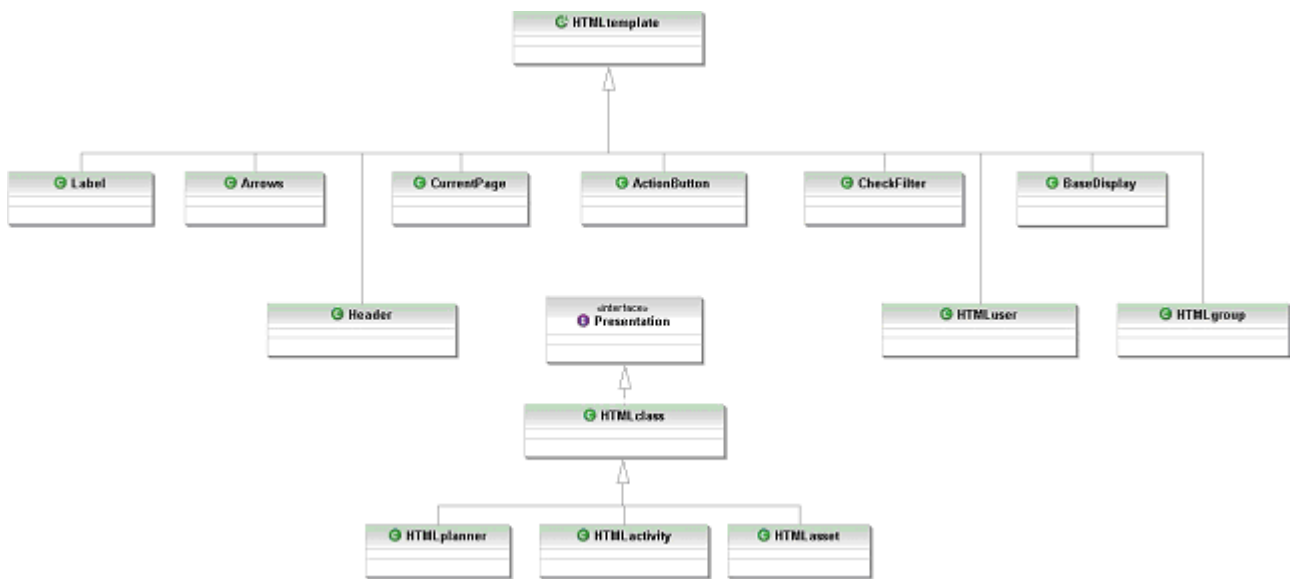
Entrambe le classi vengono caricate all’avvio dell’applicazione (avvio di Tomcat).



Segue il diagramma UML della classi e delle interfacce relativa alla gestione delle tabelle e dei domini nel database:



Il diagramma UML delle classi java delegate alla generazione delle schede di CMDBuild per l'editing dei dati è il seguente:



Gli oggetti sotto elencati sono invece utilizzati per il trasferimento delle informazioni all'interno di CMDBuild:





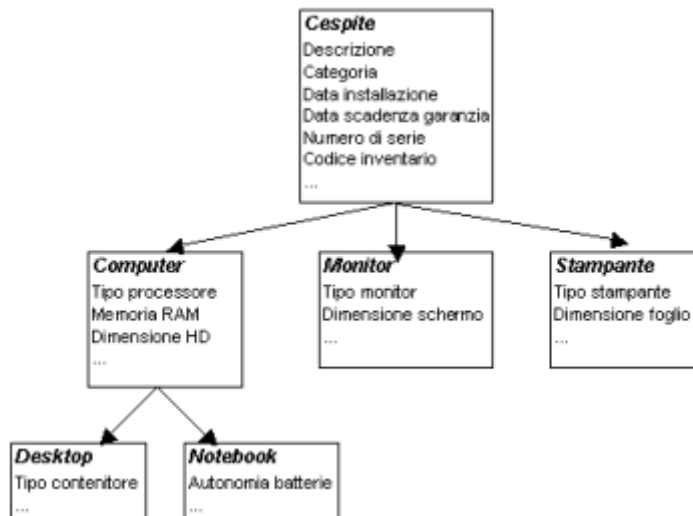
# Progettazione del database

## Criteri di base

### PostgreSQL

La progettazione del database ha dovuto rispondere ad un primo insieme di requisiti di base:

- gestire la strutturazione gerarchica a più livelli delle classi (superclassi / sottoclassi), al fine di poterle specializzare mantenendo nelle superclassi gli attributi generali



- gestire relazioni “molti a molti” fra le classi
- tracciare la storia completa delle modifiche dei dati e delle relazioni nel tempo

Per tutte e tre le esigenze è stato individuato come particolarmente interessante il meccanismo di “derivazione” fra classi reso disponibile da alcuni database object-relational ed in particolare, in ambito open source, da PostgreSQL.

Si è quindi deciso di utilizzare PostgreSQL come database di supporto a CMDBuild, ottenendo in effetti di disegnare in modo particolarmente naturale le strutture sopra descritte.

### Strutturazione a più livelli gerarchici

Tramite la parola chiave “INHERITS” è possibile infatti in PostgreSQL creare una tabella che ne “specializza” un’altra, aggiungendo alcuni attributi specifici e ritrovandosi tutti gli attributi definiti nella superclasse.

Esempio:

```

CREATE TABLE "Asset"
(
    "Id" numeric NOT NULL DEFAULT nextval('Asset_SEQ'::text),
    "Description" varchar(250),
    "SerialNo" varchar(40),
    "VersionNo" varchar(32),
    "InstallationDate" timestamp,
    "WarrantyExpireDate" timestamp,

```

```
"State" varchar(16),
"StateDate" timestamp,
CONSTRAINT asset_pkey PRIMARY KEY ("Id")
)
CREATE TABLE "Monitor"
(
"MonitorType" varchar,
"ScreenSize" varchar(16),
"MaxScreenRes" varchar(16)
) INHERITS ("Asset")
```

### **Relazioni “molti a molti”**

Le diverse tipologie di relazioni fra classi vengono implementate ciascuna con una apposita tabella di relazioni “molti a molti”, creata tramite derivazione da una superclasse predefinita allo scopo di semplificare la creazione delle sottoclassi e di garantirne l’omogeneità strutturale.

Esempio:

```
CREATE TABLE "Map"
(
"Id" numeric NOT NULL DEFAULT nextval("Map_SEQ)::text),
"Id1" numeric NOT NULL,
"Id2" numeric NOT NULL,
"Value" varchar(40),
CONSTRAINT map_pkey PRIMARY KEY ("Id")
)
CREATE TABLE " Map_aggregazione"
(
) INHERITS ("Map")
```

### **Storia delle modifiche**

Anche per la gestione della storia delle modifiche viene sfruttato il meccanismo di derivazione delle classi in PostgreSQL, creando una classe derivata per ogni tipologia di oggetto, nella quale tramite appositi trigger del database verrà archiviato il record corrente, prima di modificarne gli attributi, associandogli l’”Id” del record base, la data di modifica ed il login dell’operatore che ha effettuato la modifica.

Esempio:

```
CREATE TABLE "Map"
(
"Id" numeric NOT NULL DEFAULT nextval("Map_SEQ)::text),
"Id1" numeric NOT NULL,
"Id2" numeric NOT NULL,
"Value" varchar(40),
CONSTRAINT map_pkey PRIMARY KEY ("Id")
)

CREATE TABLE "Monitor_history"
(
"HistoryId" numeric NOT NULL,
"HistoryDate" timestamp NOT NULL DEFAULT now()
) INHERITS ("Monitor")
```

Tramite lo stesso meccanismo viene gestita la storia delle modifiche nelle relazioni, creando cioè una classe derivata per ogni tipologia di relazione, nella quale tramite appositi trigger di database verranno archiviati i record di relazione correnti, prima di eliminarli, associandogli l' "Id" del record base, la data di modifica ed il login dell'operatore che ha effettuato la modifica.

## Flessibilità

Il secondo insieme di requisiti cui la progettazione del database ha dovuto rispondere riguarda il supporto alla strutturazione autonoma del modello dati, ed in particolare:

- definizione di nuove classi
- definizione degli attributi delle classi
  - di tipologia base: stringa, data e ora, interi, numerici, booleani, memo
  - di tipologia complessa: lookup (basati sulla omonima tabella di decodifica), reference (analoghi alle foreign key dei database relazionali, ma legati ai domini)
- definizione di nuovi domini (tipologie di relazione) fra le classi

Per migliori garanzie di coerenza intrinseca del sistema si è scelto di archiviare le informazioni necessarie alla gestione dei meccanismi sopra descritti non all'interno di un "dizionario" esterno, ma nel "dizionario" stesso di PostgreSQL, utilizzando il campo "Commento" degli oggetti interessati tramite un apposito formalismo di codifica ed interpretazione.

Una descrizione dettagliata di tale formalismo è descritta nel seguito del presente capitolo (paragrafo "Descrittori integrativi").

Sono altresì elencate e descritte nel seguito le "function" definite nel database PostgreSQL per "incapsulare" al suo interno anche la logica applicativa di gestione delle classi e dei domini creati dinamicamente ed autonomamente tramite il Modulo Schema (paragrafi "Funzioni" e "Viste").

## Schema risultante

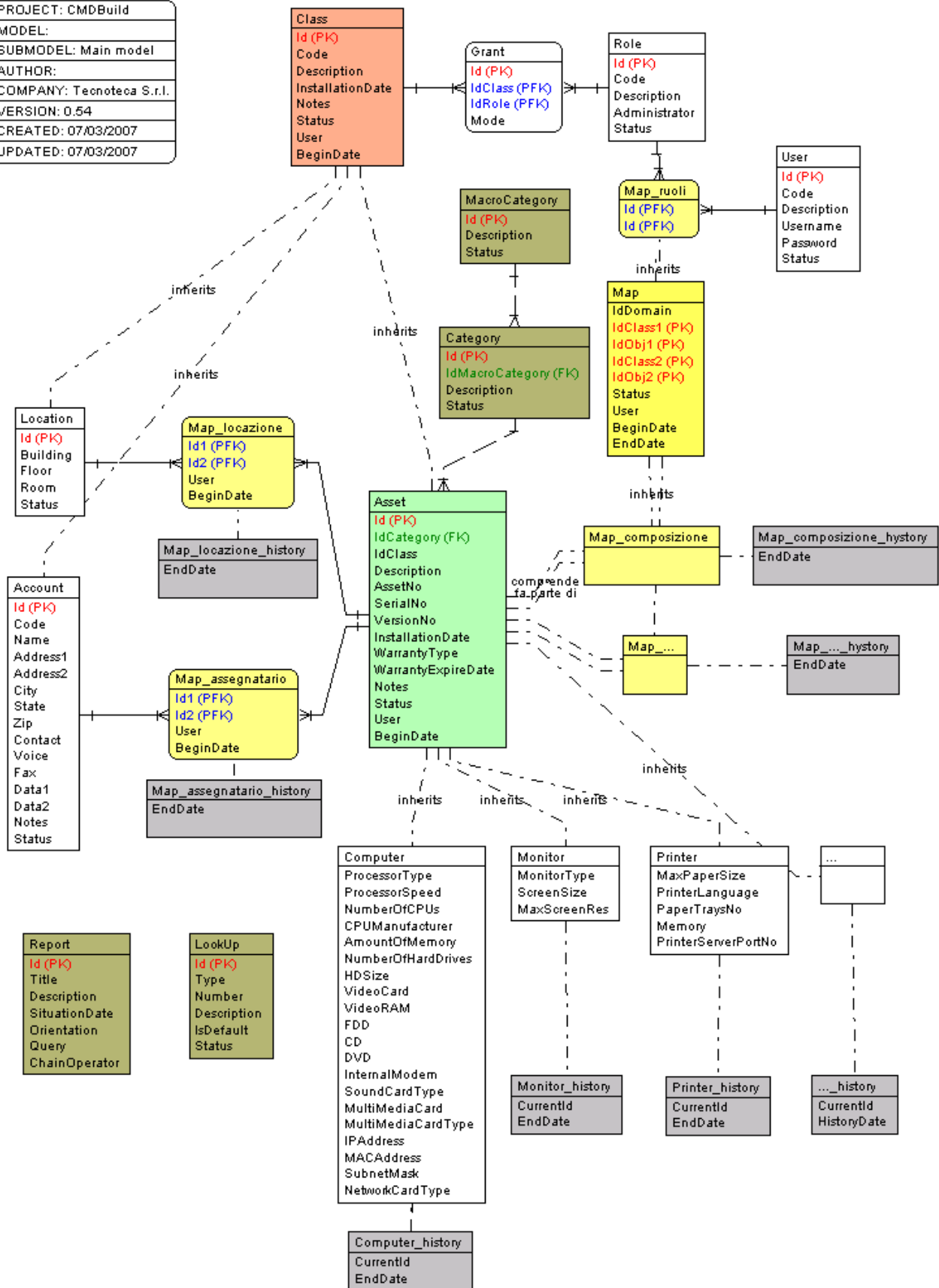
Dai requisiti sopra ricordati e da ulteriori necessità di gestione individuate è risultato il seguente schema di base, contenente le classi e le strutture generali sopra descritte, riferite a titolo di esempio ad alcune classi applicative più diffuse.

I colori scelti per lo sfondo dei diversi gruppi di classi sono:

- rosso per la classe "class", da cui sono derivate tutte le classi contenenti oggetti utente
- giallo per la classe "map" e per le classi di relazione derivate
- verde per la classe "asset" (o cespite), la classe più importante per gli utenti di CMDBuild
- bianco per le altre classi utente
- marrone per le classi di sistema
- grigio per le classi dedicate alla storicizzazione dei dati e delle relazioni

Lo schema riportato è ovviamente del tutto indicativo per quanto riguarda le "classi applicative", potendo queste essere create e strutturate autonomamente da ogni utente di CMDBuild in funzione delle proprie necessità.

PROJECT: CMDBuild
MODEL:
SUBMODEL: Main model
AUTHOR:
COMPANY: Tecnoteca S.r.l.
VERSION: 0.54
CREATED: 07/03/2007
UPDATED: 07/03/2007



## Descrittori integrativi

CMDBuild utilizza per la gestione dei propri meccanismi di gestione informazioni relative a classi ed attributi aggiuntive rispetto a quelle trattate dal database PostgreSQL.

Tali informazioni vengono archiviate nel campo "Comment" rispettivamente di classi ed attributi, con la seguente sintassi:

TIPO OGGETTO	TIPO ELEMENTO	FORMATO TAG
Classe (tabella)	Modo di utilizzo, per distinguere classi di servizio, classi non modificabili e classi utente	MODE: reserved   read   write <i>reserved = non visualizzato</i> <i>read = non modificabile</i> <i>write = modificabile</i>
	Tipo, per distinguere classi e domini (nella fattispecie classe)	TYPE: class   domain <i>class = classe</i> <i>domain = dominio</i>
	Descrizione della classe, normalmente una stringa più esplicativa del nome	DESCR: <i>valore (stringa)</i>
	Flag superclasse	SUPERCLASS: true   false <i>true = superclasse</i> <i>false = non superclasse</i>
	Gestore della classe: gestore base ("class") o nome del metodo custom java implementato	MANAGER: <i>gestore (stringa)</i>
	Stato della classe, utilizzato per registrare la cancellazione logica	STATUS: active   noactive <i>active = attivo</i> <i>noactive = cancellato</i>
Attributo (colonna)	Modo di utilizzo, per distinguere attributi di servizio (ad esempio Id, data validità, ecc), attributi non modificabili (derivati da superclassi) e attributi utente	MODE: reserved   read   write <i>reserved = non visualizzato</i> <i>read = non modificabile</i> <i>write = modificabile</i>
	Descrizione dell'attributo, normalmente una stringa più esplicativa del nome	DESCR: <i>valore (stringa)</i>
	Indice dell'attributo (posizione nella lista degli attributi)	INDEX: <i>valore (numero intero)</i>
	Riferimento alla lista delle voci di decodifica per attributi di tipo LookUp	LOOKUP: <i>valore (LookUp.Type)</i>
	Riferimenti alle modalità di gestione dei campi "Reference", con indicazioni del dominio associato, della modalità di cancellazione (non ammessa, a cascata), del fatto che la classe corrente sia la prima o la seconda della coppia in relazione	REFERENCEDOM: <i>valore (stringa)</i> REFERENCETYPE: <i>restrict = non ammessa</i> <i>cascade = a cascata (non ancora implementata)</i> REFERENCEDIRECT:true   false <i>true = prima classe nella coppia</i> <i>false = seconda classe nella coppia</i>
	Flag campo data da trattare come "scadenza"	DATEEXPIRE: true   false

		<i>true = campo scadenza false = non campo scadenza</i>
	Flag campo base per visualizzazione (ad esempio sarà visualizzato anche nelle pagine "a tabulato" contenenti un numero ridotto di colonne)	BASEDSP: true   false <i>true = base per visualizzazione false = non base per visualizzazione</i>
	Stato della classe, utilizzato per registrare la cancellazione logica	STATUS: active   noactive <i>active = attivo noactive = cancellato</i>
Dominio (tabella)	Modo di utilizzo, per distinguere domini di servizio e domini utente	MODE: reserved   read   write <i>reserved = non visualizzato read = non modificabile write = modificabile</i>
	Tipo, per distinguere classi e domini (nella fattispecie dominio)	TYPE: class   domain <i>class = classe domain = dominio</i>
	Prima classe della coppia in relazione	CLASS1: <i>valore (stringa)</i>
	Seconda classe della coppia in relazione	CLASS2: <i>valore (stringa)</i>
	Descrizione diretta	DESCRDIR: <i>valore (stringa)</i>
	Descrizione inversa	DESCRINV: <i>valore (stringa)</i>
	Flag dominio da gestire rappresentando le due classi in relazione in forma master-detail	MASTERDETAIL: true   false <i>true = gestione master-detail false = non gestione master-detail</i>
	Cardinalità	CARDIN: 1:N   N:1   N:M
	Stato del dominio, utilizzato per registrare la cancellazione logica	STATUS: active   noactive <i>active = attivo noactive = cancellato</i>

## Funzioni

Si è scelto di utilizzare in modo esteso le "Function" di PostgreSQL per "incapsulare" al suo interno anche la logica applicativa di gestione delle classi e dei domini.

Segue la lista delle principali funzioni utilizzate:

NOME	UTILIZZO	PARAMETRI
createattribute	Creazione dinamica nuovo attributo	<i>ClassName = nome classe AttributeName = nome attributo AttributeIndex = indice attributo (posizione)</i>

		<p> <i>AttributeDesc = descrizione attributo</i>  <i>AttributeType = tipo attributo (boolean, date, decimal, double, integer, reference, lookup, string, text, timestamp)</i>  <i>AttributeLookUp = lista LookUp collegata</i>  <i>AttributeReference =</i>  <i>AttributeReferenceDomain = dominio per tipo reference</i>  <i>AttributeReferenceType = tipo gestione cancellazione per tipo reference</i>  <i>AttributeReferenceDirect = prima o seconda classe nella coppia</i>  <i>AttributeDateExpire = flag data tipo scadenza</i>  <i>AttributeMandatory = obbligatorio</i>  <i>AttributeDefault = valore default</i>  <i>AttributeBaseDSP = base per visualizzazione</i>  <i>AttributeStatus = attributo attivo o eliminato</i> </p>
createclass	Creazione dinamica nuova classe	<p> <i>ClassName = nome classe</i>  <i>ClassDesc = descrizione classe</i>  <i>ClassStatus = classe attiva o eliminata</i>  <i>ParentClass = classe da cui eredita</i>  <i>SuperClass = flag superclasse</i>  <i>ClassManager = gestore classe</i> </p>
createdomain	Creazione dinamica nuovo dominio	<p> <i>DomainName = nome dominio</i>  <i>DomainClass1 = nome classe1</i>  <i>DomainClass2 = nome classe2</i>  <i>DomainDescrDir = descrizione diretta</i>  <i>DomainDescrInv = descrizione inversa</i>  <i>DomainMasterDetail = flag master-detail</i>  <i>DomainCardin = cardinalità</i>  <i>DomainStatus = dominio attivo o eliminato</i> </p>
createhistory	Creazione tabella derivata per storicizzazione classe (trigger esclusi)	<p> <i>ClassName = nome classe</i> </p>
createrelationhistory	Creazione tabella derivata per storicizzazione dominio (trigger esclusi)	<p> <i>DomainName = nome dominio</i> </p>
createrelationtriggers	Creazione triggers per la storicizzazione del dominio	<p> <i>DomainName = nome dominio</i> </p>
createtriggers	Creazione triggers per la storicizzazione della classe	<p> <i>ClassName = nome classe</i> </p>
deleteattribute	Cancellazione attributo (da utilizzarsi solamente a tabella vuota)	<p> <i>ClassName = nome classe</i>  <i>AttributeName = nome attributo</i> </p>
deleteclass	Cancellazione classe (da utilizzarsi solamente a tabella vuota)	<p> <i>ClassName = nome classe</i> </p>

deletedomain	Cancellazione dominio (da utilizzarsi solamente a tabella vuota)	<i>DomainName = nome dominio</i>
deleterelationtriggers	Cancellazione triggers per la storicizzazione del dominio	<i>DomainName = nome dominio</i>
deletetriggers	Cancellazione triggers per la storicizzazione della classe	<i>ClassName = nome classe</i>
deleteview	Cancellazione vista	<i>ViewName = nome vista</i>
modifyattribute	Modifica attributo	<i>ClassName = nome classe</i> <i>AttributeName = vecchio nome attributo</i> <i>AttributeNewName = nuovo nome attributo</i> <i>AttributeIndex = indice attributo (posizione)</i> <i>AttributeDesc = descrizione attributo</i> <i>AttributeType = tipo attributo (boolean, date, decimal, double, integer, reference, lookup, string, text, timestamp)</i> <i>AttributeLookUp = lista LookUp collegata</i> <i>AttributeReference =</i> <i>AttributeReferenceDomain = dominio per tipo reference</i> <i>AttributeReferenceType = tipo gestione cancellazione per tipo reference</i> <i>AttributeReferenceDirect = prima o seconda classe nella coppia</i> <i>AttributeDateExpire = flag data tipo scadenza</i> <i>AttributeMandatory = obbligatorio</i> <i>AttributeDefault = valore default</i> <i>AttributeBaseDSP = base per visualizzazione</i> <i>AttributeStatus = attributo attivo o eliminato</i>
modifyclass	Modifica classe	<i>ClassName = nome classe</i> <i>ClassNewName = nuovo nome classe</i> <i>ClassDesc = descrizione classe</i> <i>ClassStatus = classe attiva o eliminata</i> <i>SuperClass = flag superclasse</i> <i>ClassManager = gestore classe</i>
modifydomain	Modifica dominio	<i>DomainName = nome dominio</i> <i>DomainNewName = nuovo nome dominio</i> <i>DomainClass1 = nome classe1</i> <i>DomainClass2 = nome classe2</i> <i>DomainDescrDir = descrizione diretta</i> <i>DomainDescrInv = descrizione inversa</i> <i>DomainMasterDetail = flag master-detail</i> <i>DomainCardin = cardinalità</i> <i>DomainStatus = dominio attivo o eliminato</i>



## Viste

Per un più immediato accesso al “dizionario dati allargato”, costituito dal Data Dictionary di PostgreSQL e dai descrittori integrativi archiviati nei campi “Comment”, sono state definite ed utilizzate le viste descritte alla tabella successiva.

<b>NOME</b>	<b>UTILIZZO</b>
cmdbclasscatalog	Catalogo completo delle classi con gli attributi gestiti in CMDBuild
cmdbdomaincatalog	Catalogo completo dei domini con gli attributi gestiti in CMDBuild
cmdbgroupcatalog	Lista gruppi utenti definiti in CMDBuild
cmdbpolicycatalog	Lista permessi definiti in CMDBuild per i singoli utenti
navigationtree	Lista gerarchica delle classi utente definite in CMDBuild (utilizzato come albero di navigazione nel programma)
relationlist	Lista relazioni archiviate in CMDBuild
relationhistorylist	Lista relazioni archiviate in CMDBuild inclusi record cancellati e storicizzati

# API di CMDBuild

## Generalità

*[da completare]*

## Lista delle API

*[da completare]*

# Attuali limitazioni note del sistema

## Export / Import

Si è riscontrato che in PostgreSQL il comando "pg\_dump", anche se lanciato con opzione "-o" salva (e consente a "pg\_restore" di ripristinare) gli "oid" delle righe delle tabelle, ma non salva (e quindi non consente il ripristino) gli "oid" delle classi all'interno del catalogo di sistema.

Essendo tali "oid" utilizzati da CMDBuild come propri riferimenti interni (classe "Class" e derivate, classe "Map" e derivate), a seguito del restore di un database risultano non corrette le relazioni stabilite fra gli oggetti di CMDBuild.

E' stato realizzato e reso disponibile un comando di ripristino che va eseguito successivamente ad ogni "restore" e che aggiorna i riferimenti in questione a partire da informazioni che sono comunque disponibili nel sistema (sottolineiamo a questo proposito che non esiste alcun rischio di perdita di dati).

## Campi tipo "reference"

La struttura di CMDBuild consente la definizione di campi di tipo "reference" all'interno di una superclasse ed il loro utilizzo nell'ambito delle classi derivate.

Non è al momento gestito il caso di campi "reference" ereditati lungo catene "superclassi – sottoclassi" di lunghezza superiore a due (non è quindi ad esempio possibile definire un campo "reference" all'interno della classe "asset" ed utilizzarlo nella sottoclasse "notebook" se esiste una ulteriore superclasse "computer" fra "asset" e "notebook").

Sempre relativamente ai campi di tipo "reference" non è al momento gestita l'opzione di cancellazione in modalità "cascade", cioè con eliminazione automatica di tutti gli elementi collegati anche su più livelli al record da cancellare (ad esempio non si può cancellare la scheda di un "fornitore" se esiste una scheda "contatti" con un campo di tipo "reference" sulla classe "fornitori" ed esistono dei contatti ancora attivi per quel fornitore).

E' quindi al momento necessario provvedere manualmente alla cancellazione degli elementi collegati prima di poter cancellare la scheda principale.

Tale limitazione è legata al fatto che le uniche cancellazioni ammesse in CMDBuild sono cancellazioni di tipo "logico", il che non consente di usufruire delle opzioni di cancellazioni previste in PostgreSQL per le "foreign key".