Agenzia nazionale per le nuove tecnologie, l'energia e lo sviluppo economico sostenibile

Ministero della Transizione Ecologica

Consolidamento dell'uso della Piattaforma PELL per dati statici ed implementazione sezione per la raccolta dati dinamici

Cosma Damiano De Angelis, Pierfrancesco De Angelis

Report RdS/PTR2021/020

CONSOLIDAMENTO DELL'USO DELLA PIATTAFORMA PELL PER DATI STATICI ED IMPLEMENTAZIONE SEZIONE PER LA RACCOLTA DATI DINAMICI

Cosma Damiano De Angelis (arch4energy), Pierfrancesco De Angelis (arch4energy)

Novembre 2021

Report Ricerca di Sistema Elettrico

Accordo di Programma Ministero dello Sviluppo Economico - ENEA
Piano Triennale di Realizzazione 2019-2021 - II annualità
Obiettivo: Tecnologie
Progetto: Tecnologie per la penetrazione efficiente del vettore elettrico negli usi finali
Work package: Local Energy District
Linea di attività: Progettazione Software PELL Edifici
Responsabile del Progetto: Claudia Meloni, ENEA
Responsabile del Work package: Claudia Meloni, ENEA

Il presente documento descrive le attività di ricerca svolte all'interno del Contratto "TERIN/2020/226 Progettazione software PELL Edifici (CIG 8472167D5E– CUP I34I19005780001)"
Responsabile Unico del Procedimento ENEA: Claudia Meloni (ENEA)
Responsabile del Contratto per il Contraente: Cosma Damiano De Angelis (arch4energy)

Indice

Sommario

La piattaforma PELL è una piattaforma di tipo smart city as-a-service, la cui architettura generale definisce il recupero dei dati da diverse infrastrutture e gestori e la creazione di una serie di servizi per gli utenti finali. Costituisce il punto di accesso per tutti gli utenti finali legati al progetto PELL (Public Energy Living Lab), fornendo l'entry point di registrazione, accesso al caricamento dati ed ai servizi correlati.
Il ruolo centrale della piattaforma è consentire il caricamento dei dati statici, ovvero le schede censimento relative allo stato corrente degli edifici, e dei dati dinamici, ovvero i dati relativi alle misure elettriche e termiche acquisite dai meters.
Le attività che sono state svolte e qui riportate sono le restanti – considerato il primo rilascio relativo a due task già effettuato nel marzo 2021 - previste dal contratto TERIN/2020/226 ovvero:

- manutenzione della piattaforma UBD, sia sotto il profilo della implementazione di protocolli di accesso sia sotto il profilo del supporto alla gestione del sistema
- progettazione della struttura dati per il LENICALC WEB

Il primo task è consistito nella progettazione delle configurazioni necessarie alla implementazione del protocollo di accesso ai dati tramite protocollo WebHDFS nonché nella progettazione e realizzazione di una sistema di supporto alla gestione del sistema nel suo complesso. Il sistema di supporto alla gestione, basato sulla fondamentale funzione di monitoring, si rende necessario per aumentare quanto possibile l'uptime della infrastruttura UBD. Il secondo task è relativo alla progettazione della struttura dati che possa consentire, in futuro, la completa integrazione della funzionalità di calcolo del LENI nel portale edifici.

1 Introduzione

Fra i diversi ambiti in cui ENEA sta operando in vista della ottimizzazione dell'utilizzo dell'energia a livello nazionale, è previsto lo sviluppo di un sistema che ha come oggetto gli edifici di pubblica proprietà.
Facendo seguito al progetto PELL relativo alla pubblica illuminazione, grazie al quale è stata implementata una piattaforma Big Data denominata UBD, ed una serie di servizi applicativi verticali, è del tutto consistente pensare ad uno sviluppo di sistema, per gli edifici, che faccia tesoro di tale esperienza.
Arch4energy ha partecipato alla realizzazione del sistema attuale, nell'ambito di progetti precedenti, come riportato in dettaglio nel documento di gara richiesto per le esperienze precedenti, specifiche per il progetto PELL.
Il progetto PELL Edifici vedrà dunque, tra gli elementi costitutivi, quelli che si possono definire come i trasposti delle componenti PELL IP, utilizzando peraltro l'infrastruttura ICT, sia per gli aspetti di comunicazione che di storage ed elaborazione parallela, di cui è previsto il necessario potenziamento.
L'infrastruttura che attualmente accoglie i dati relativi al PELL IP è denominata Urban Big Data, costituita da un cluster Hadoop attualmente in esercizio.
Una prima area di intervento è costituita dalla realizzazione di un sistema di supporto all'esercizio della Piattaforma UBD, la cui complessità in termini di numero di server e numero di processi attivi richiede strumenti di gestione adeguati.
D'altro canto la specializzazione della piattaforma verso Pell Edifici richiede l'analisi della integrazione di strumenti specifici come Lenicalc, nel nuovo portale.
Il presente documento è relativo dunque al rilascio dei deliverables che seguono:

- Manutenzione piattaforma UBD, che comprende
  - Accesso utenti tramite protocollo WebHDFS
  - Sistema di monitoring UBD e documentazione per Coaching
- Progettazione integrazione Lenicalc

2 Manutenzione Piattaforma UBD

Nel corso degli ultimi anni ENEA ha sviluppato una piattaforma informatica estremamente articolata, oggetto di continua evoluzione, per dare servizi ai diversi progetti nell'area Smart City. Come di consueto nei grandi sistemi ICT, l'infrastruttura consta di un certo numero di server che ospitano processi e/o piattaforme, la cui complessità ed interazione crescono in misura esponenziale rispetto al numero dei server e delle applicazioni. L'infrastruttura deve inoltre dare servizi verso l'esterno, a tutti i partecipanti - a vario titolo – ai diversi progetti. Si pongono quindi – tra i tanti – due questioni che vengono affrontate nel progetto, ovvero la progettazione della comunicazione fra piattaforma UBD e i diversi utenti che possono accedervi, nonché il problema della stabilità e della continuità dei servizi erogati. Da qui la necessità di implementare un sistema di gestione che consenta di raggiungere i livelli di qualità – in termini di continuità del servizio - opportuni. In tal senso si pone la progettazione e realizzazione di un sistema di monitoraggio a supporto delle operazioni di gestione della infrastruttura, oggetto del presente documento. Le operazioni di manutenzione della piattaforma si sostanziano dunque in quanto ai capitoli 3 e 4 seguenti.

3 Protocollo WebHdfs

L'accesso alla piattaforma Hadoop, che è alla base del sistema Big data UBD, viene regolato dalle configurazioni che Hadoop stesso prevede. Nel caso di ENEA, si è considerato la possibilità di dare accesso agli utenti non amministratori tramite il protocollo WebHdfs. Tale protocollo permette di interagire agli utenti riconosciuti dal sistema operativo – siano essi tramite Kerberos oppure tramite configurazione sul sistema operativo in modalità nativa - con il file system HDFS. Il vantaggio di questa scelta è dovuto al fatto che un qualsiasi client web, ad esempio un browser, oppure il comando "curl", siano sufficienti ad interagire con il file system in modo completo. Come di consueto, in considerazione del fatto che il cluster UBD è in esercizio, si è proceduto a verificare la ipotizzata configurazione di Hadoop su un cluster Arch4energy dedicato allo sviluppo e test delle soluzioni ipotizzate.

3.1 Cluster di Test

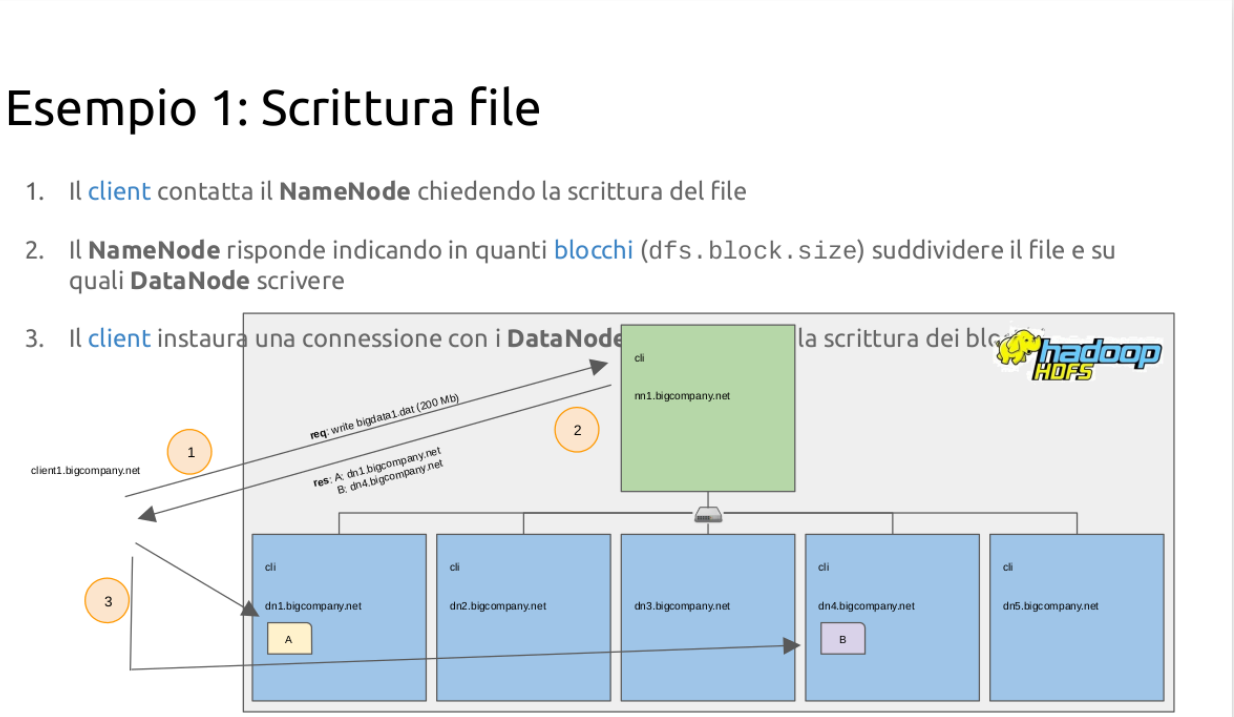
Il cluster di test è stato realizzato installando le stesse componenti, in termini di distribuzione e versione, esistenti sul cluster UBD. In particolare la versione di SO Centos 7.6 e la versione di hadoop 2.7.6. La aderenza alle versioni è assolutamente necessaria poiché le configurazioni di Hadoop sono specifiche per ogni versione. Per verificare le funzionalità è infatti sufficiente installare un cluster multiserver, anche se non è necessario utilizzare lo stesso numero di server di UBD. Nel cluster di test vengono utilizzate tre macchine, denominate "master", "slave01" e "slave02". Sono stati creati due utenti, uno principale "hadoop" ed uno ospite "hdfsguest". Il cluster è stato installato secondo la modalità "kerberos security", implementando sul master il servizio kerberos. Questa modalità coinvolge tutti i servizi software a supporto della piattaforma. Nel realizzare l'impianto di test si è potuto partire da una situazione "pulita", da scratch, e ciò ha evidenziato un potenziale problema laddove si dovesse operare su una piattaforma in esercizio, come UBD. Il problema è l'impatto notevolissimo che le operazioni di modifica delle configurazioni hanno sulla installazione corrente. Apportare tali modifiche in esercizio vorrebbe dire sostanzialmente reinstallare tutta la piattaforma passando per il backup e restore dei dati in essere. Inoltre si è considerato che nelle versioni successive di Hadoop l'approccio alla sicurezza si è molto evoluto. Tutto ciò ha comportato la necessità di progettare una modalità alternativa alla kerberizzazione per giungere alla disponibilità di una interfaccia web ai dati hadoop. Facendo riferimento al documento originale "WeHDFS REST API" è stato possibile implementare l'accesso al file system tramite i comandi elencati di seguito:

Operations

- HTTP GET
  - OPEN (see FileSystem.open)
  - GETFILESTATUS (see FileSystem.getFileStatus)
  - LISTSTATUS (see FileSystem.listStatus)
  - GETCONTENTSUMMARY (see FileSystem.getContentSummary)
  - GETFILECHECKSUM (see FileSystem.getFileChecksum)
  - GETHOMEDIRECTORY (see FileSystem.getHomeDirectory)
  - GETDELEGATIONTOKEN (see FileSystem.getDelegationToken)
  - GETDELEGATIONTOKENS (see FileSystem.getDelegationTokens)
  - GETXATTRS (see FileSystem.getXAttr)
  - GETXATTRS (see FileSystem.getXAttrs)
  - GETXATTRS (see FileSystem.getXAttrs)
  - LISTXATTRS (see FileSystem.listXAttrs)
  - CHECKACCESS (see FileSystem.access)
- HTTP PUT
  - CREATE (see FileSystem.create)
  - MKDIRS (see FileSystem.mkdirs)
  - CREATESYMLINK (see FileContext.createSymlink)
  - RENAME (see FileSystem.rename)
  - SETREPLICATION (see FileSystem.setReplication)
  - SETOWNER (see FileSystem.setOwner)
  - SETPERMISSION (see FileSystem.setPermission)
  - SETTIMES (see FileSystem.setTimes)
  - RENEWDELEGATIONTOKEN (see DelegationTokenAuthenticator.renewDelegationToken)
  - CANCELDELEGATIONTOKEN (see DelegationTokenAuthenticator.cancelDelegationToken)
  - CREATESNAPSHOT (see FileSystem.createSnapshot)
  - RENAMESNAPSHOT (see FileSystem.renameSnapshot)
  - SETXATTR (see FileSystem.setXAttr)
  - REMOVEXATTR (see FileSystem.removeXAttr)
- HTTP POST
  - APPEND (see FileSystem.append)
  - CONCAT (see FileSystem.concat)
  - TRUNCATE (see FileSystem.concat)
- HTTP DELETE
  - DELETE (see FileSystem.delete)
  - DELETESNAPSHOT (see FileSystem.deleteSnapshot)

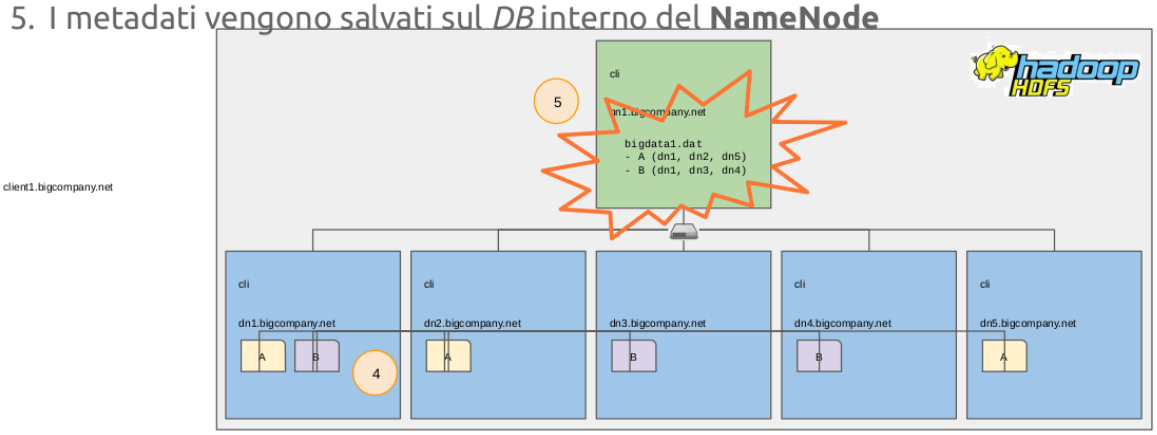
3.2 Architettura di rete

Una importante sottolineatura è che le funzionalità di accesso al file system tramite WebHdfs richiedono la raggiungibilità a livello IP da parte del client che emette il comando verso l'intero cluster UBD. Ciò è dovuto alla architettura propria di HDFS che delega al Namenode la gestione della allocazione dei blocchi dati, ma lascia ai Datanode stessi il colloquio con il client. Questa modalità evita l'effetto collo di bottiglia sul namenode stesso ma richiede, ovviamente, la raggiungibilità client-datandes, anche se la richiesta viene fatta al namenode che gestisce il dispatching. Nelle figure che seguono sono riportati gli schemi logici di questo meccanismo:



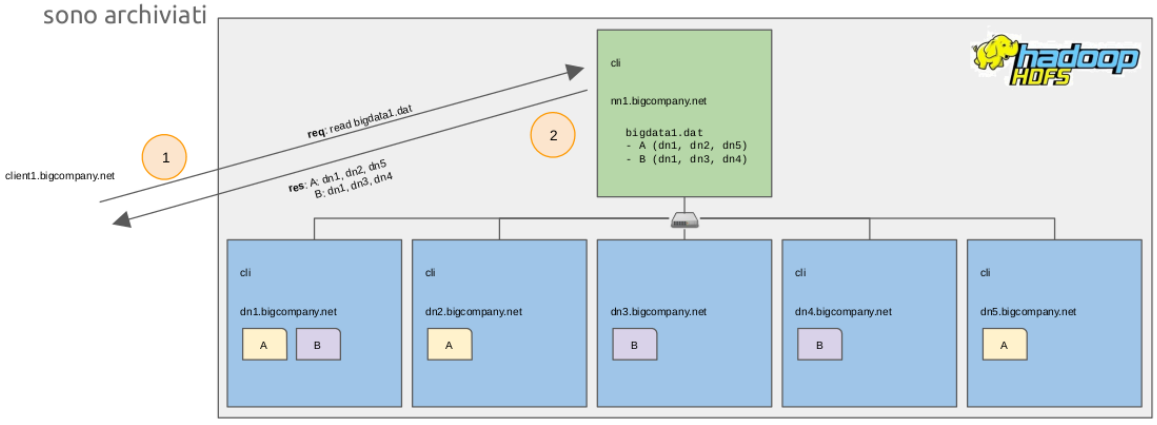
Esempio 1: Scrittura file

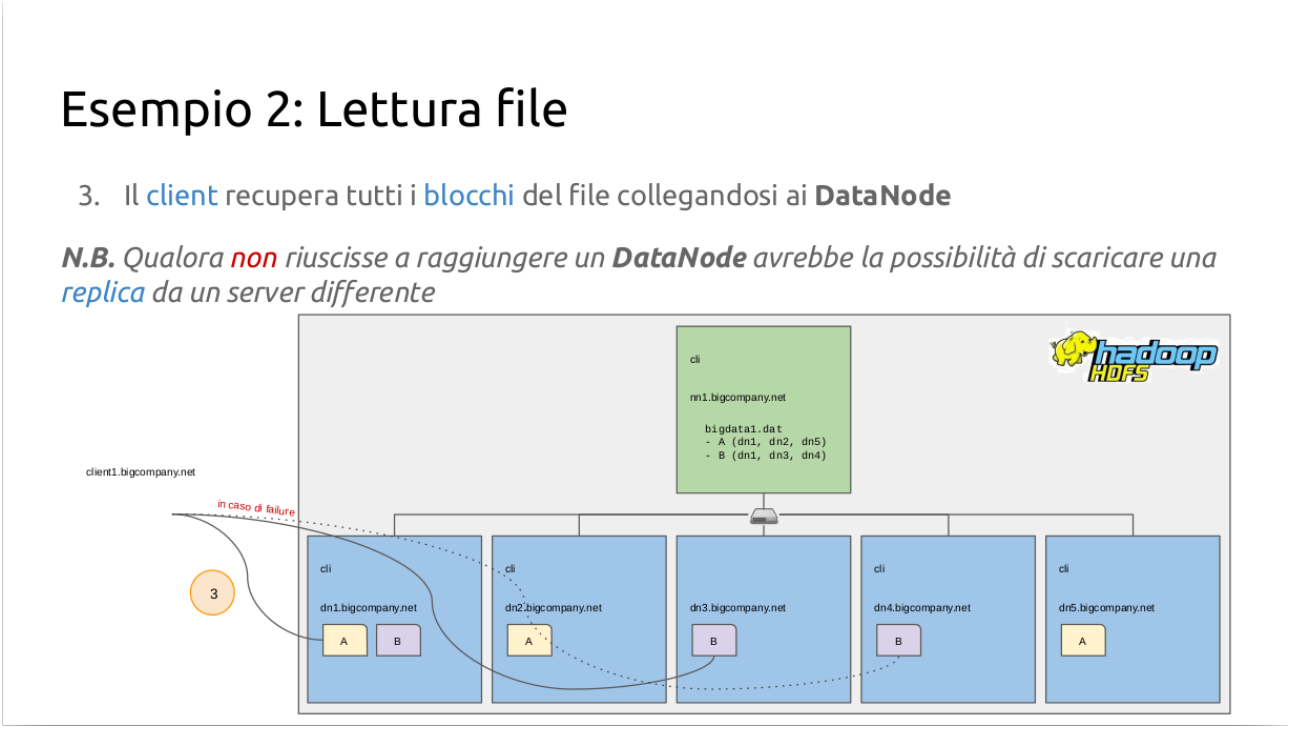
4. I blocchi vengono replicati *n* volte (*es:* 3) a seconda del valore impostato come *default* (`dfs.replication`) o come da richiesta del **client**
5. I metadati vengono salvati sul *DB* interno del **NameNode**



Esempio 2: Lettura file

1. Il client contatta il **NameNode** per ottenere i metadati del file bigdata1.dat al fine di poter accedere al contenuto dello stesso
2. Il **NameNode** risponde inviando la lista dei blocchi che compongono il file e dei **DataNode** dove sono archiviati





3.3 Esempi di comandi WebHdfs

Facendo riferimento al cluster di test di Arch4energy, vengono riportati una serie di esecuzione di comandi WebHdfs, che effettuano le operazioni tipiche di una interazione fra utenti e file system Hdfs. La situazione relativa ai permessi del file system HDFS è la seguente:

```
Found 5 items
drwxr-xr-t   - hadoop    supergroup          0 2017-08-16 13:10 /user/app
drwxr-xr-x   - hadoop    supergroup          0 2021-04-07 18:19 /user/hadoop
drwxr-x---   - hdfsguest supergroup          0 2021-11-02 09:39 /user/hdfsguest
drwxr-xr-x   - hadoop    supergroup          0 2017-09-11 09:50 /user/hive
drwxr-xr-x   - hadoop    supergroup          0 2018-07-03 16:46 /user/spark
```

Come evidente, esistono due utenti di cui hadoop è il superuser di HDFS mentre hdfsguest è un ospite. I permessi sono stati settati in modo opportuno, per cui da hdfsguest non si potrà accedere ai contenuti di hadoop, mentre ovviamente il superuser potrà vedere tutto. Nella directory di hdfsguest sono presenti alcuni files di esempio visibili da hdfs:

```
[hdfsguest@master ~]$ hdfs dfs -ls eneatest
Found 3 items
-rw-r--r--   2 hdfsguest supergroup        182 2021-11-02 11:32 eneatest/test.csv
-rw-r--r--   2 hdfsguest supergroup        182 2021-11-02 11:34 eneatest/test1.csv
-rw-r--r--   2 hdfsguest supergroup        182 2021-11-02 11:34 eneatest/test2.csv
```

Un esempio di lettura del file test.csv è dato dal comando seguente:

curl -i -L "http://master:50070/webhdfs/v1/user/hdfsguest/eneatest/test.csv?op=OPEN&user.name=hdfsguest"

Il cui risultato è l'equivalente di un cat del file test.csv a stdout.

3.4 Comandi custom

Per rendere il più possibile semplice l'accesso ai dati da WebHdfs, sono stati realizzati alcuni comandi custom che possono essere lanciati dalla shell dell'utente specifico, passando come parametri soltanto i dati minimi. Questi shell scripts sono da considerare anche come template per la creazione di altri comandi similari. Ad esempio volendo vedere a video il contenuto di un file su HDFS, di proprietà dell'utente "hdfsguest", si deve essere loggati come utente "hdfsguest" e lanciare il comando:

$ cat_file.webhdfs.sh eneatest/test.csv

Il risultato sarà il contenuto a video del file test.csv che si trova nella directory eneatest della home hdfs dell'utente hdfsguest. Il contenuto del comando custom è il seguente:

```
#!/bin/bash
# Il path deve esistere nel file system HDFS
FILE=$1
echo $USER $FILE | awk '{printf("curl -i -L
\"http://master:50070/webhdfs/v1/user/%s/%s?op=OPEN&user.name=%s\"\n",$1,$2,$1)}' | sh
```

Ovviamente nel caso del cluster UBD si dovrà modificare il nome del Namenode verso cui si lancia il comando. Gli altri comandi che sono stati implementati (ed i cui nomi sono parlanti) sono i seguenti:

get_dir_status_webhdfs.sh
get_file_status_webhdfs.sh
put_file_webhdfs.sh

3.5 Istruzioni per attivazione nuovo utente

Per mettere in grado un nuovo utente di accedere via WebHdfs alla piattaforma UBD sarà necessario effettuare le operazioni seguenti:

- creare il nome utente su UBD
- creare la home in hdfs e renderla leggibile soltanto a lui
- distribuire al nuovo utente i comandi custom
- assicurarsi la raggiungibilità IP dal suo client verso tutti i server UBD

A questo punto l'utente sarà in grado di inserire ed estrarre files dal file system HDFS.

4 Sistema di Monitoraggio

L'infrastruttura attuale si compone di server, prevalentemente linux, che ospitano i diversi servizi. Si tratta di circa 20 server linux con sistema operativo Centos, eccetto uno con sistema Windows Server 2012. I servizi principali sono http, https, mqtt, sql, load balancers, ed alcuni applicativi custom di ENEA.

Fra questi server ve ne sono cinque che ospitano la piattaforma Big Data, di Apache Software Foundation. A piattaforma assicura al sistema la capacità di gestire, sia in termini di quantità e di sicurezza del dato, sia in termini di capacità elaborativa, la quantità di dati – virtualmente senza limiti – che un sistema di tipo nazionale deve garantire.

La piattaforma Big Data è costituita – dal punto di vista hardware - da cinque server le cui configurazioni principali sono le seguenti.

- Hadoop: un Master e 4 Slave.
- Spark: un Master e 5 workers.
- Elasticsearch : cluster da 5 server paritari

A questi componenti si affiancano molti altri, sempre e solamente software, facenti parte del cosiddetto ecosistema Hadoop.

Le componenti sono molte, ovviamente non tutte quelle disponibili in Apache Software Foundation, ma sono state scelte in quanto offrono funzionalità che sono, o che possono essere, utili nelle varie fasi del progetto ENEA.

Sono da sottolineare due aspetti:

- il software ASF è completamente open source e disponibile senza limitazioni;
- in ogni momento si può aggiungere un componente la cui compatibilità e armonizzazione con gli altri è assicurata da una corretta configurazione secondo gli standard ASF.

In questa logica si intende implementare anche il sistema di monitoraggio.

Esso dovrà infatti essere disponibile in open source in modo da rendere ENEA del tutto autonoma nella sua implementazione e manutenzione.

4.1 Basi Progettuali

Data la complessità e l'importanza dei servizi erogati, si prevede un sistema di monitoraggio a supporto dell'esercizio, sistema che dovrà però costituire anche strumento per le diverse finalità delineate. Gli scopi del sistema di monitoraggio sono infatti molteplici, sia come tenore tecnico che comunicativo. Se infatti il sistema di monitoraggio deve assicurare un supporto al gruppo tecnico che gestisce le operation, d'altro canto dovrà essere capace di comunicare all'esterno, ovvero a livello non tecnico, le complessità e la potenzialità della infrastruttura stessa.

I criteri tecnici devono assicurare non solo che si abbia cognizione del reale stato di salute dei vari sottosistemi, ma anche che a fronte di un malfunzionamento il quadro sinottico dia informazioni sulle correlazioni, a volte inaspettate, fra i servizi che costituiscono la costellazione di complessiva di sottoservizi che è responsabile del servizio nel suo complesso.

Dal punto di vista della comunicazione poi il sistema di monitoraggio deve rendere "visibile" a chiunque – con una vista di tipo qualitativo – sia la potenza che la complessità del sistema nel suo complesso.

Questa seconda prospettiva assume tanto peso quanto più complesso e tecnologicamente evoluto è il sistema di cui si vuole comunicare la presenza.

Si consideri che il sistema di monitoraggio dovrà operare ad un ordine di grandezza del minuto, nel senso che dovrà essere possibile configurare la verifica con il taglio del minuto, da uno ad N.

4.1.1 Criteri Tecnici

I criteri con cui dovrà essere realizzato il sistema di monitoraggio sono tali da minimizzare il pericolo di avere malfunzionamenti applicativi di cui il sistema non da visione, così come il pericolo di avere allarmi che non corrispondono a reali malfunzionamenti. In sostanza si tratta di evitare il pericolo di falsi negativi, così come di falsi positivi.

Per approssimarsi a questo risultato è necessario operare a diversi livelli, secondo le schema che segue:

- monitorare le risorse del server : le risorse da monitorare saranno la CPU, la RAM, il disco e la network;
- monitorare la presenza dei processi per ciascun servizio : per ciascun servizio si dovrà verificare la presenza del/i relativo/i processo/i.
- monitorare la disponibilità dei socket di erogazione del servizio : ogni servizio espone un socket, ovvero una porta collegata ad un indirizzo IP. Quindi si dovrà verificare la presenza effettiva della porta per ciascun servizio di ciascun server.
- monitorare la funzionalità del servizio overall : la verifica della presenza del processo, così come del socket, devono essere completate con la verifica della funzionalità finale, tramite un client di utilizzo del servizio stesso. Ciò completerà il quadro dei componenti che danno così una ragionevole certezza che la funzione venga effettivamente svolta.

4.1.2 Criteri Comunicativi

Il sistema di monitoraggio dovrà essere in grado di offrire, ad ogni istante, l'intero insieme i dati di stato per ogni componente, in modo da presentare – ovvero da consentire lo sviluppo – una vista sinottica omnicomprensiva.

La vista sinottica sarà così utilizzabile sia a livello tecnico come vista di correlazione di eventi, sia a livello comunicativo per far "vedere" di quanti elementi il sistema sia costituito e come questi elementi possano interagire fra loro.

Nell'ottica dello UBD questa funzione diventa straordinariamente importante in quanto una piattaforma Big Data, come quella di ENEA, può erroneamente apparire come una sorta di grande disco per la registrazione di dati digitali, nascondendo la sua potenza come piattaforma di storage sicuro e di elaborazione parallela, di potenzialità sostanzialmente senza limiti.

4.2 Realizzazione della Piattaforma

Nel mondo open source la problematica del monitoraggio è ormai ampiamente frequentata e negli anni sono state realizzate, e rese disponibili alla comunità, moltissime soluzioni software. Alcuni analisti testimoniano diverse decine di piattaforme disponibili per il monitoring. Tutti o quasi presentano una struttura composta da un motore, da plugins – lanciati dal motore - per le azioni sui sistemi da monitorare ed una interfaccia Web per la consultazione e/o amministrazione. Fra i vari sistemi disponibili alcuni si sono

distinti per la loro semplicità di configurazione, di uso e di integrazione e sono stati quindi prevalentemente scelti da diverse organizzazioni ICT. Un criterio di scelta per UBD è certamente la presenza di plugins open source dedicati proprio alla piattaforma Hadoop, che sono direttamente utilizzabili o fonte di ispirazione per la creazione di plugins custom.
La piattaforma che si propone nel presente progetto è la soluzione open di Nagios, o di sue derivate (a puro titolo di esempio OMD, check_mk, etc) ed i motivi che sottendono – oltre a quanto accennato il precedenza - questa opzione sono molteplici:

- Estrema semplicità di integrazione di plugins custom, anche shell script, come elementi di connessione fra il motore di polling ed il sistema target da monitorare.
- Configurazioni su files ascii editabili, che consentono un interfacciamento molto semplice con il motore;
- Ottenimento di un file ascii di snapshot dello stato dell'intera infrastruttura, consentendo la realizzazione di sinottici grafici;
- Estrazione dei dati storici generati dal polling;
- Capacità di polling con intervalli del minuto;
- Semplicità di integrazione con sistemi di reporting come Kibana, soluzione utilizzata già da ENEA nell'ambito del progetto PELL;
- Utilizzo di files e non di RDBMS per i dati statici o dinamici;
- Possibilità di rotazione dei files storici, per la analisi ex post di problemi che si sono evidenziati in momenti diversi da quelli che hanno generato il problema.
- Disponibilità di interfaccia web di presentazione e navigazione già completa, senza necessità di ulteriori sviluppi web.

4.3 Consistenza Infrastruttura PELL-UBD

La piattaforma ENEA può essere suddivisa logicamente in due componenti. I server dedicati alla parte Big Data e i restanti server dedicati si diversi servizi di rete ed applicativi.

Alias macchina	Descrizione breve	IP
broker	Pell Broker	192.168.34.95; 192.107.61.113 (pub)
pellbroker1	broker load balancer	192.168.34.239
pellbroker2	broker load balancer	192.168.34.240
pellbroker3	broker load balancer	192.168.34.241
pellfe	Pell Frontend (doppio ip)	192.107.61.62; 192.168.34.96
pellbe	Pell Backend	192.168.34.122
pellwin	Pell windows	192.168.34.123
pelldb	Pell db	192.168.34.124
pelldev	Pell Dev	192.168.34.125
pelltest	Pell Test	192.168.34.145
datalake.enea.it	Datalake	IP 192.107.61.50/24; Gw: 192.107.61.16
pell-namenode	Pell namenode	192.168.34.109
pell-datanode1	pell-datanode1	192.168.34.235
pell-datanode2	pell-datanode2	192.168.34.236
pell-datanode3	pell-datanode3	192.168.34.237

Alias macchina	Descrizione breve	IP
pell-datanode4	pell-datanode4	192.168.34.238
bd-test1	Big data test	192.168.34.106
bd-test2	Big data test	192.168.34.107

4.4 Monitoraggio Hadoop UBD

La piattaforma Hadoop di ENEA è costituita dai server – presenti nella lista di cui sopra - che ospitano i diversi componenti ASF. Oggetto di questa prima parte di monitoraggio sono i due principali servizi, che sono propedeutici a tutti gli altri ovvero Hadoop e Spark.

4.4.1 Controlli Server

Sui server si prevede il monitoraggio di :

- Uso della CPU
- Occupazione RAM
- Occupazione Disco/File Systems
- Traffico di Network

4.4.2 Componenti Software

Principali componenti cluster:

- Hadoop/HFDS
- Yarn
- Zookeeper
- Spark

4.4.3 Controllo Hadoop/HDFS

- hdsf dfsadmin -report
- hdfs fsck /

4.4.4 Controllo Processi

- ps -ef
- jps

E' possibile utilizzare gli script, opportunamente modificati per renderli integrabili con il sistema di monitoraggio, presenti in /home/hdp/Utilities del sever pellnn.

4.4.5 Controllo Socket

- Il servizio HDFS espone la porta 54310 del namenode, quindi pellnn:54310
- Il servizio YARN espone la porta 8032 del namenode, quindi pellnn:8032
- Il servizio SPARK espone la porta 7077 del namenode, quindi pellnn:7077

4.4.6 Controllo Web Gui

Ciascun servizio espone una o più interfacce di verifica in http, che dovranno essere verificate così come fatto con i socket:

- Web GUI Hadoop/HDFS espone la porta 50070 del namenode, quindi http://pellnn:50070
- Web GUI Cluster Hadoop YARN espone la porta 8088 del namenode, quindi http://pellnn:8088
- Web GUI Storico Hadoop YARN espone la porta 18888 del namenode, quindi http://pellnn:18888
- Web GUI Spark Master espone la porta 8080 del namenode, quindi http://pellnn:8080
- Web GUI Spark Storico espone la porta 18080 del namenode, quindi http://pellnn:18080

4.5 Controlli Funzionali

I controlli funzionali, opportunamente incapsulati in uno script linux, verificano l'effettiva disponibilità di ciascun servizio, semplicemente usandolo.

4.5.1 Hadoop/HDFS

Verifica della effettiva disponibilità del fiel system HDFS. Lancio di uno script hdfs che copia un file

$ hdfs -put nomefile /tmp

4.5.2 Hadoop/YARN

Verifica della effettiva disponibilità della piattaforma di elaborazione parallela Hadoop/YARN. Lancio remote shell con Yarn:
$ yarn org.apache.hadoop.yarn.applications.distributedshell.Client -jar /opt/hadoop/share/hadoop/yarn/hadoop-yarn-applications-distributedshell-2.7.6.jar -shell_command df -shell_args -h
O anche il lancio un job che usa il cluster applicativo hadoop
$ ./mahout_test.sh
oppure
$ hadoop jar /opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.6.jar pi 4 4

4.5.3 Spark

Verifica della effettiva disponibilità della piattaforma di elaborazione parallela Spark. Lancio di un programma di test scala – sviluppato ad hoc - in modalità scripting:
$ spark-shell -i test_batch.scala

4.5.4 Zookeeper

Verifica della effettiva disponibilità della funzionalità di distribuzione di file di configurazione Zookeeper. Si può utilizzare lo script VerifyZoo.sh presente nel direttorio Utilities.

4.6 La piattaforma di Monitoraggio OMD

La piattaforma di monitoraggio adottata è la OMD, Open Monitoring Distribution, nella sua versione Checkmk, basata a sua volta su Nagios. La versione è quella – libero utilizzo - denominata Raw, nella versione 1.6. Questa scelta rispetta la linea di aderenza alla policy Open Source di ENEA. Come orami spesso accade nel mondo Open, le organizzazioni sviluppano soluzioni che vengono sempre rilasciate rispettando questa impostazione, lasciandosi però la possibilità di implementare soluzioni verticali che possono essere offerte con un prezzo di uso legato alla specificità della soluzione stessa. In ogni caso non viene mai meno la disponibilità del pacchetto “base” o “Raw”, come nel caso di OMD, il cui acronimo contiene il termine Open.

4.7 OMD-Manuale di Installazione

La richiesta di ENEA prevede la realizzazione della piattaforma di Monitoraggio insieme al coaching necessario per consentire ad ENEA stessa la gestione in autonomia della piattaforma nel tempo. A questo scopo la parte di documento che segue costituisce manuale di installazione, di configurazione e d’uso della piattaforma così come realizzata.

4.7.1 Pacchetti di installazione

In relazione al sistema operativo CentOs 7 adottato sui server ENEA, sono necessari due packages, uno relativo al server ed uno relativo all’agente che va installato su tutte le macchine da monitorare, server compreso:

- check-mk-raw-1.6.0p13-el7-38.x86_64.rpm
- check-mk-agent-1.6.0p13-1.noarch.rpm

4.7.2 Prima Configurazione

Una volta installato il server e l’agente sul server stesso, i passi fondamentali sono:

- Creazione di una istanza OMD
- Attivazione dell’agente
- Configurazione con WATO del monitoraggio del server stesso
- Attivazione del log dei dati storici
- Attivazione della rotazione dei log dei dati storici
- Scrittura di plugins custom come estensione dell’agente per controlli specifici

4.7.3 Attivazione Log Storici

Il sistema in modo nativo offre gli andamenti storici di tutte le grandezze monitorare in forma di grafici immediatamente disponibili, con l’orizzonte di un anno. Volendo però avere la libertà di accedere ai dati originali, senza limitazioni di tempo, è necessario implementare una modalità che consente la registrazione su file ascii dell’esito di tutti i controlli eseguiti. Poiché OMD crea per ogni istanza un file system dedicato alla istanza stessa, il nome della istanza è ricorrente nei PATH del sistema.Quindi di seguito si indicherà con ISTANZA il nome della istanza generica. I passi da compiere per la configurazione sono i seguenti:

- *Editare il file /omd/sites/ISTANZA/etc/nagios/nagios.d/obsess.cfg e modificare le righe:*
  - *obsess_over_service=1*
  - *ocsp_command=write-service-data*
- *Creare il file /omd/sites/ISTANZA/etc/nagios/conf.d/write-service-data.cfg contentente:*
  - *define command{*
  - *command_name write-service-data*
  - *command_line $USER2$/write-service-data.sh "$LASTSERVICECHECK$" "$HOSTNAME$" "$SERVICEDESC$" "$SERVICESTATE$" "$SERVICESTATETYPE$" "$SERVICEEXECUTIONTIME$" "$SERVICELATENCY$" "$SERVICEPERFDATA$" "$SERVICEOUTPUT$"*
  - *}*
- *Creare il file /omd/sites/ISTANZA/local/lib/nagios/plugins/write-service-data.sh contenente:*
  - *#!/bin/sh*
  - */bin/echo "$1|$2|$3|$4|$5|$6|$7|$8|$9" >> /omd/sites/ISTANZA/var/nagios/service-data.log*

Riavviando il sistema con OMD restart si attiverà la configurazione.

4.7.4 Concetti di Plugin Custom

La piattaforma OMD contiene una notevolissima raccolta di plugin per monitorare i tanti diversi aspetti di una infrastruttura complessa. Ma come avviene sempre nell'IT, ogni sistema ha le sua peculiarità e l'infrastruttura PELL non fa eccezione. La sua complessità e varietà richiede lo sviluppo di plugins custom ad hoc. La piattaforma OMD consente in modo molto semplice di creare lugins che vengono poi inclusi automaticamente dall'agente di sistema. I plugins possono essere scritti in qualsiasi linguaggio, cosa che rende estremamente potente la realizzazione di plugin in shell scripting. La convenzione da rispettare è relativa all'output del plugin stesso. Il sistema si aspetta infatti un output così organizzato(SPAZIO significa un blank) :

Codice_Ritorno SPAZIO Nome_Plugin SPAZIO val1=val;warning;critical;min;max|val2=val SPAZIO Stringa di Output che può contenere anche spazi

Il Codice_Ritorno è un intero che vale 0,1,2,-1 avendo come significato Ok, Warning, Critical, Unknown
Quindi un esempio di output reale potrebbe essere:

0 mioplugin val1=42|val2=23;30;50;0;100|val3=44 OK – Il valore è corretto

L'eseguibile va inserito nel direttorio
/usr/lib/check_mk_agent/local
L'azione va compiuta da root poiché la /usr/lib è una directory di sistema. Importante rendere eseguibile a tutti il plugin. Quando si torna alla configurazione WATO dell'agente check-mk, compare il plugin con il nome "mioplugin" fra quelli disponibili.

4.7.5 Esempi di Plugin Custom

Un plugin molto utile è quello che effettua la verifica della disponibilità di una porta tcp su un server esterno visto dal server OMD. Il plugin che segue è la verifica della porta 6556 su un server denominato OMD_Client con indirizzo IP 192.168.1.152.
Il plugin è denominato /usr/lib/check_mk_agent/local/check_6556_OMD_Client.sh

```
==========================================================================
#!/bin/sh
FILE=/tmp/tmp00$$
# contenuto del file del check_tcp
# TCP OK - 0.001 second response time on 192.168.1.152 port 6556|time=0.001403s;;;0.000000;10.000000
/opt/omd/versions/1.6.0p13.cre/lib/nagios/plugins/check_tcp -H 192.168.1.152 -p 6556 &>/dev/null
ret=$?
if [ $ret == "0" ]
then
    /opt/omd/versions/1.6.0p13.cre/lib/nagios/plugins/check_tcp -H 192.168.1.152 -p 6556 > $FILE
    awk -F"|" '{printf("0 port_6556_192.168.1.152 %s %s\n",$2,$1)}' $FILE
    rm $FILE
```

```
    exit 0
else
    echo "2 port_6556_192.168.1.152 time=0.00s;;;0.000000;10.000000 ERROR - Port not available"
    exit 2
fi
```

==========

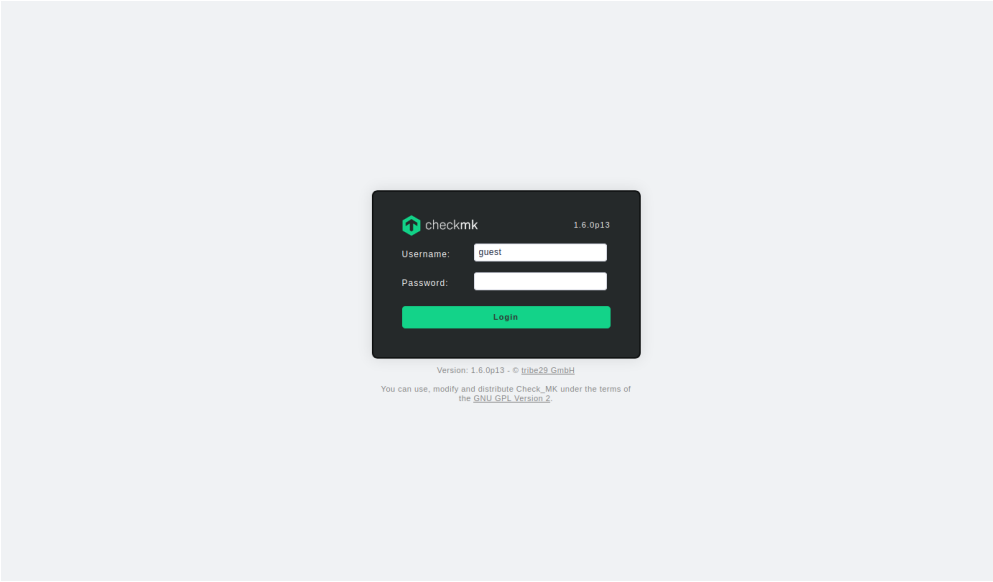
Il plugin non è relativo ad una istanza, perché è una estensione del sistema complessivo ed è disponibile ad ogni agente che lo trova della directory specifica.

4.8 OMD-Manuale d'uso e configurazione

In relazione al progetto di monitoraggio della infrastruttura UBD di ENEA, di seguito viene descritto il sistema nella sua prima implementazione. In una prima si fa riferimento ad un manuale utente sintetico, che consenta di utilizzare il sistema già pronto all'uso. In una seconda parte si riportano note sulla configurazione che è stata realizzata, per consentire ad ENEA l'ulteriore sviluppo della piattaforma per coprire il perimetro, sempre in evoluzione, della infrastruttura PELL-UBD. Il sistema check_mk fornisce una interfaccia web molto ricca, di seguito vengono indicate soltanto le viste più significative in funzione della configurazione effettuata. Rimane la massima libertà nel navigare, tramite il menù sinistra, per ottenere le più diverse viste.

4.8.1 Primo Utilizzo

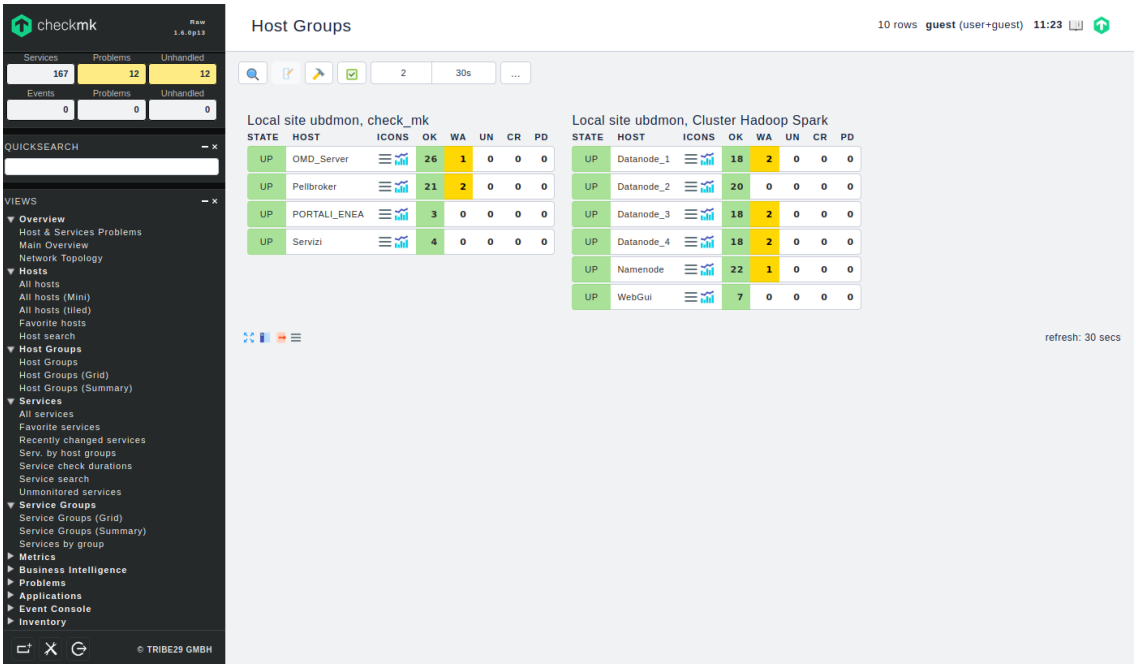
Il sistema grafico utilizzato, on top ad OMD, è il check_mk, derivato da Nagios e l'interfaccia è raggiungibile all'indirizzo http://192.168.34.108/ubmdon



E' stato creato un utente non amministratore, che non può quindi modificare la configurazione, che è "guest" con password "ubdmon".

4.8.2 Viste Principali

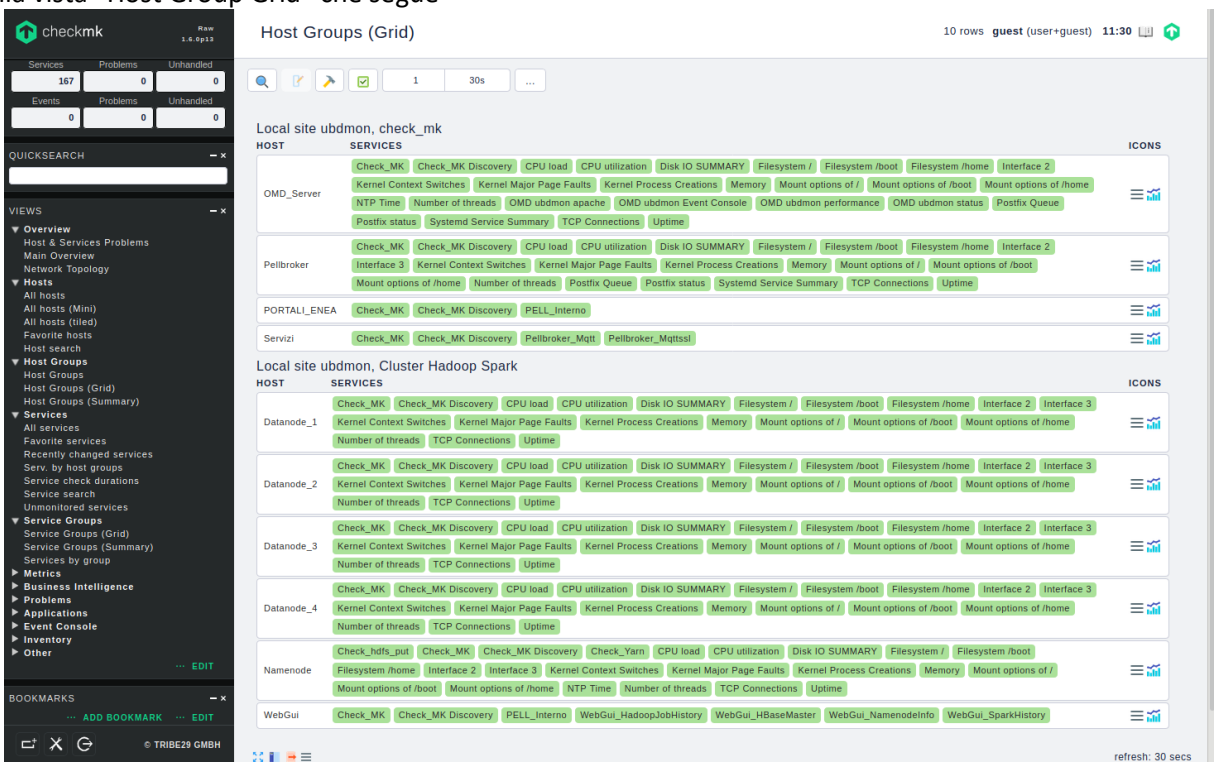
Dal menù si passa alla vista "Host groups"



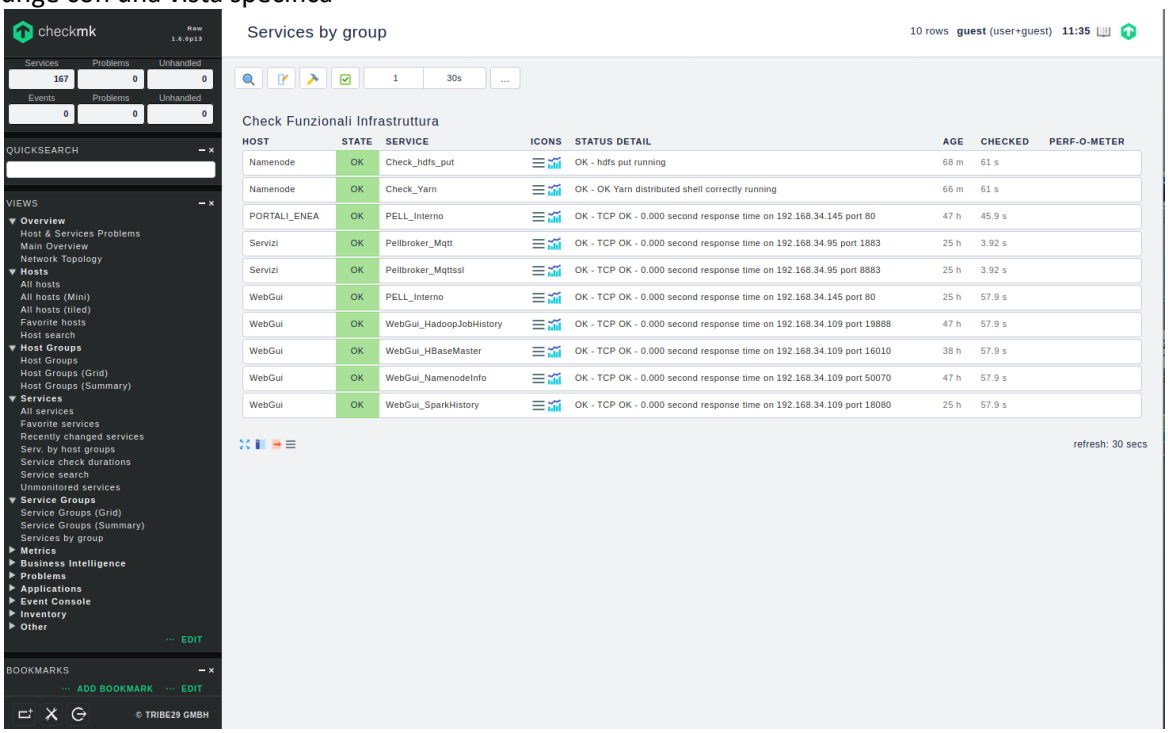
si ottiene una vista dei server e dello stato dei servizi configurati:

- OMD_Server è il server su cui è installato il motore e l'interfaccia Web di monitoraggio.
- Pellbroker è il server Mqtt
- PORTALI_ENEA è un server fittizio su cui si possono configurare le disponibilità dei diversi portali enea, e fra questi è stato configurato il portale PELL
- Servizi è un server fittizio su cui si possono configurare servizi di rete come Mqtt
- Namenode ed i vari Datanode sono i costituenti la piattaforma Big Data Hadoop
- WebGui è un server fittizio sul quale cono verificate le disponibilità delle Web Gui dei servizi Hadoop

Su tutti i server sono configurati sia verifiche tecniche che funzionali.
Dalla vista "Host Group Grid" che segue



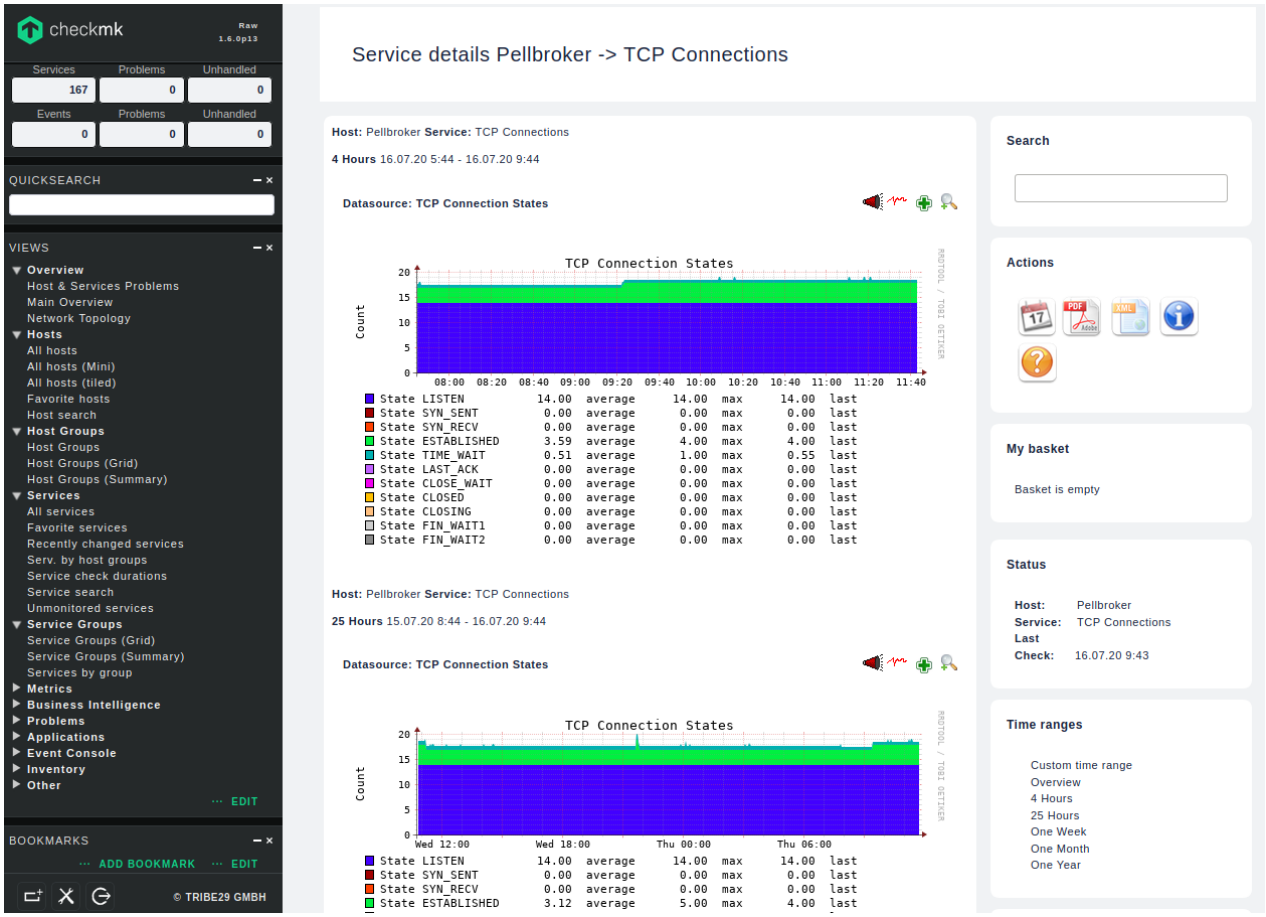
e’ possibile vedere tutti i server e tutti i servizi monitorati, nel loro stato istantaneo. La pagina web si rinnova ogni 30 secondi oppure ogni volta che si naviga fra le diverse pagine. Quindi una vista simile consente un colpo d’occhio molto significativo dello stato complessivo della infrastruttura. Considerando che vi sono controlli di tipo tecnico su ogni server, mentre i controlli funzionali sono quelli che ci assicurano della disponibilità di applicazioni o servizi specifici, è stato creato un gruppo di servizi (Service Group) che si raggiunge con una vista specifica



Qui si vedono evidenziati i servizi e/o le applicazioni core che l’infrastruttura deve erogare affinchè tutte le funzioni vengano esplicate correttamente.

4.8.3 Dati di Performance

I servizi che sono configurati per restituire un dato di performance, ad esempio un tempo di risposta o un valore numerico di percentuale disco occupato, e simili, offrono una interfaccia grafica del dato storico graficizzato. Andando un mouse-over sulla icona che rappresenta i grafici, i ottiene il grafico della prima grandezza disponibile, mentre cliccando sulla icona si apre una interfaccia con tutti i grafici delle grandezze, se disponibili, relative al controllo effettuato. Il tempo massimo di storicizzazione e presentazione è di un anno solare. Questo limite viene risolto da una personalizzazione di cui si farà cenno nella parte configurativa. A puro titolo di esempio si riporta il grafico relativo alle connessioni TCP del server Pellbroker:

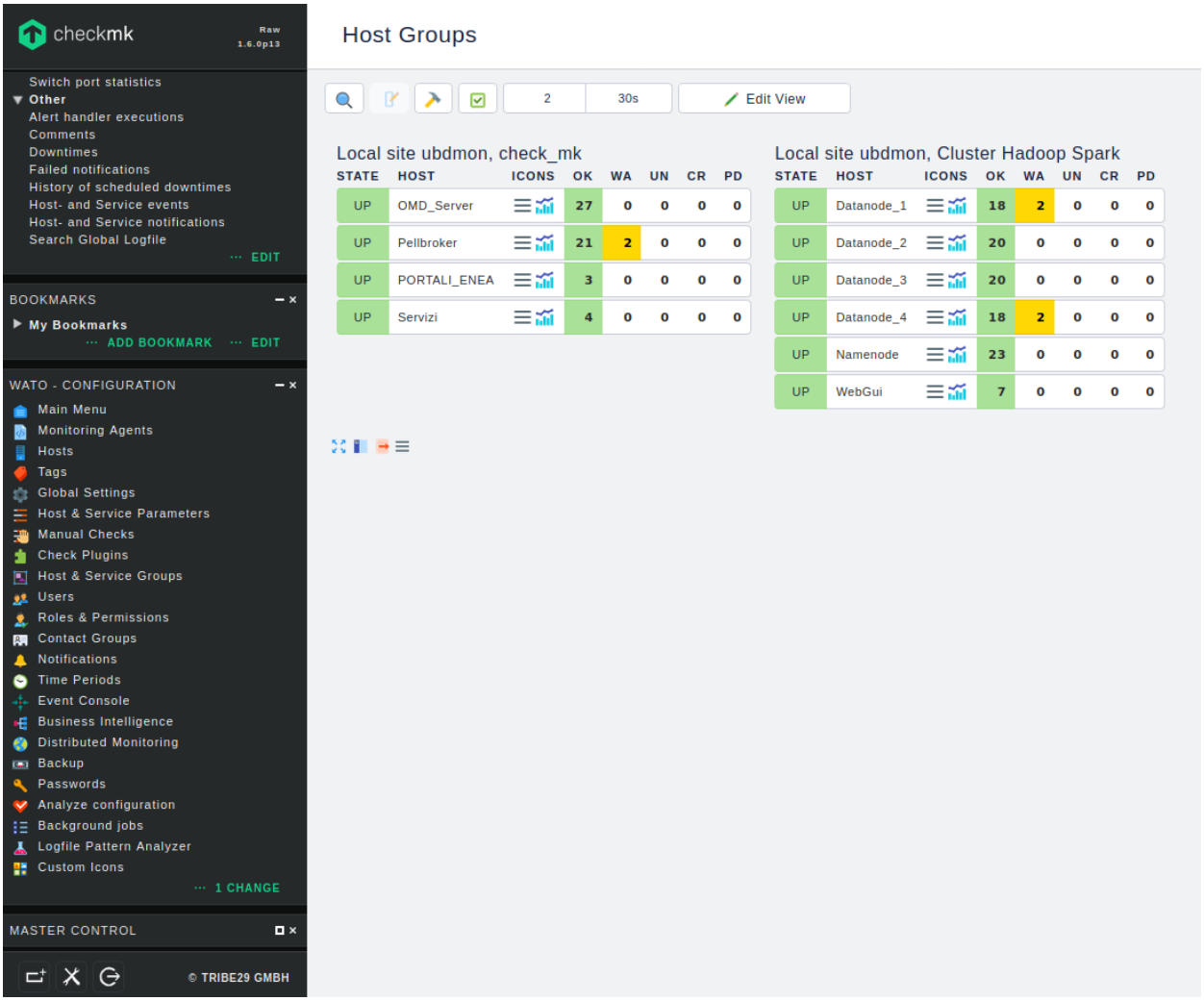


4.8.4 Logiche di controllo

Le logiche di controllo che sono state implementate sono di tipo diverso e vanno esaminate per interpretare correttamente i diversi stati che possono presentarsi. Esistono controlli che vengono eseguiti sul server stesso per verificare lo stato di una funzione o servizio presenti sul server stesso, e controlli che vengono eseguiti su un server ma che testano la disponibilità di una funzione o servizio su un altro server. Ad esempio, facendo riferimento alla vista precedente dei Service By Group, i servizi Check_hdf_put e Check_Yarn vengono eseguiti sul namenode per determinare lo stato della funzionalità put di hdfs e la corretta esecuzione di una shell distribuita su Yarn. Il servizio PELL_Interno invece viene eseguito sul server PORTALI_ENEA, che è lo stesso OMD_Server ridefinito, ma va a testare la disponibilità della porta 80 sul server 192.168.34.145 che è quello che espone il portale PELL. Questa situazione è peraltro immediatamente visibile nell'output del plugin. Lo stesso vale per il server WebGui e Servizi, ovvero si tratta sempre del server OMD_Server su cui vengono eseguiti controlli che testano la disponibilità delle porte TCP dei vari servizi sia su Pellbroker che sul Namenode che espone le WebGui di Hadoop.

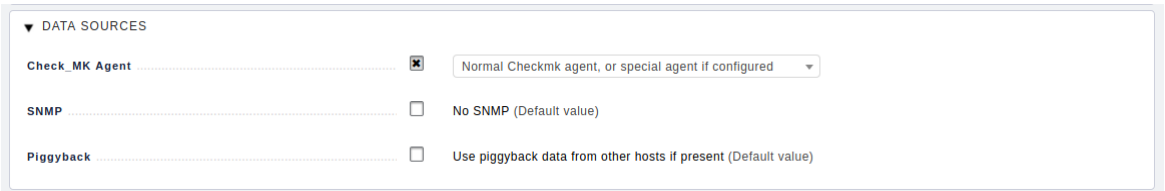
4.9 OMD-Note di configurazione

Il sistema è stato configurato completamente con WATO (Web Administration TOol) disponibile all'amministratore "cmkadmin" nella parte inferiore del menu, il cui pannello di controllo è visibile a sinistra nella figura.



4.9.1 Plugin Standard

Come accennato in precedenza, è importante capire quali tipologie di controllo sono state implementate, per interpretare correttamente i diversi stati che possono presentarsi. La configurazione automatica per ogni server è assicurata dalla presenza del servizio check_mk_agent, che una volta installato si mette in ascolto sulla porta 6556 e fornisce tutti i controlli tecnici di default.
Il nuovo Host deve essere configurato come segue:



In questo modo il servizio di agente offre la discovery di tutti i possibili servizi configurabili e rimane nella possibilità dell'amministratore di accettarli o meno.

4.9.2 Plugin Personalizzati

E' inevitabile che ciascun sistema di monitoraggio debba rappresentare al meglio la complessità della infrastruttura monitorata, per cui è fondamentale poter sviluppare plugin ad hoc, che devono poi essere inseriti nel framework. Esistono diverse modalità per estendere il sistema con plugin custom e nel progetto UBD sono stati utilizzati due metodi, ovvero:

- **Custom Plugins**

Si tratta di plugin, nel nostro caso si tratta di shell script, che vengono posizionati in modo da essere rilevati dall'agente come disponibili. I criteri da rispettare sono due:
Il plugin deve avere come output su stdout una stringa che si composta come segue:
NUMERO SPAZIO NOME_SERVIZIO SPAZIO VALORI_DI_PERFORMANCE SPAZIO STRINGA DI OUTPUT
Ad esempio, un plugin che controlla la funzionalità di upload – installato sul Namenode - di un file in hdfs risponde come segue:
0 Check_hdfs_put Time=2 OK - hdfs put running
L'eseguibile deve essere posizionato nel direttorio
/usr/lib/check_mk_agent/local
per essere visibile all'agente.
I plugin di questo tipo presenti nel server OMD_Server sono:

- check_HadoopJobHistory.sh
- check_NamenodeInfo.sh
- check_Pellbroker_Mqttssl.sh
- check_SparkHistory.sh
- check_HBaseMaster.sh
- check_Pellbroker_Mqtt.sh check_PELL_Interno.sh

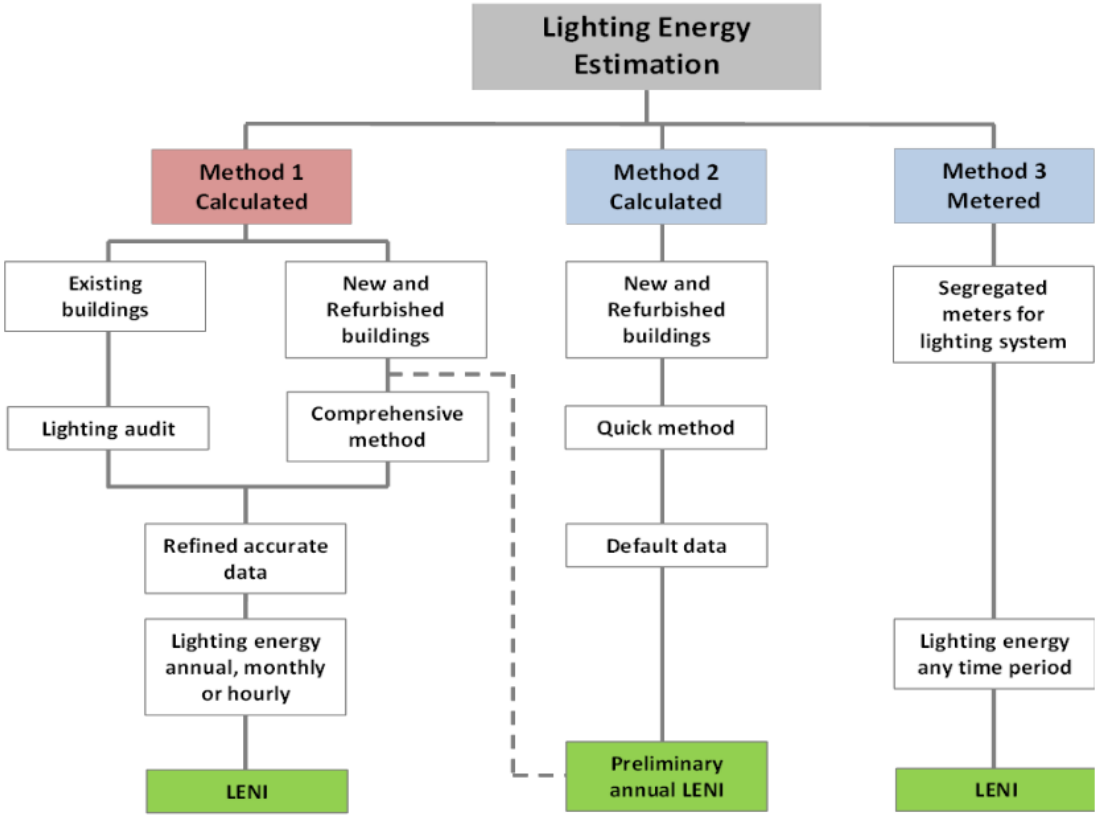
- **MRPE Plugins**

  Nel caso in cui si vogliano utilizzare plugin che possono avere un tempo di esecuzione maggiore del minuto, che è il default di sistema, si dovrà realizzare l'eseguibile in modo che il suo output sia quello standard di Nagios. E' il caso del plugin che esegue, sul namenode, un comando di esecuzione distribuita tramite Yarn, di un comando shell. In questo modo lo Yarn esegue su tutte le macchine del cluster un comando, cosa che prova la funzionalità di distribute execution. Il plugin è lo /usr/local/bin/Check_Yarn presente sul namenode, e viene configurato in /etc/check_mk/mrpe.cfg
  Il contenuto di mrpe.cfg è il seguente:
  Check_Yarn (interval=300) /usr/local/bin/Check_Yarn
  L'opzione interval indica 300 secondi per il rilancio dello script, fermo restando che se l'esecuzione ci mettesse un tempo superiore, il sistema terrebbe buono il valore precedente senza andare in errore. E' una sorta di differimento preventivo. L'output del plugin deve avere la forma che segue:
  OK Yarn distributed shell correctly running |Time=7
  E deve restituire una valore 0,1,2 o -1, ad indicare Ok, Warning, Critical, Unknown. Una volta realizzato e posizionato correttamente, il plugin viene rilevato dalla discovery dell'agente check_mk e reso disponibile immediatamente alla configurazione.

5 Lenicalc Web

5.1 Premesse

Il LENI, ovvero il Light Energy Numeric Indicator Il Leni (Lighting Energy Numeric Indicator) è un indicatore numerico dell'energia annuale richiesta per l'illuminazione di un edificio. Il valore ottenuto viene sfruttato per confrontare i consumi all'interno di edifici caratterizzati da utenze simili, ma con differenti dimensioni.
Il Leni è infatti dato dal rapporto W/A [kWh/m2 anno] ,dove W indica l'energia totale annua impiegata per l'illuminazione nel corso di un anno, mentre A è l'area occupata dall'edificio. La norma UNI EN 15193-1:2017 mette a disposizione tre metodologie di valutazione delle prestazioni energetiche per i sistemi di illuminazione artificiale, attraverso la stima dei consumi di energia elettrica imputabili all'illuminazione artificiale anche in presenza dei sistemi di controllo. Le metodologie possono essere applicate per edifici nuovi, esistenti o ristrutturati e sono riportate nella figura seguente:



In questo panorama, ENEA, nell'ambito delle attività previste dalla "Ricerca di Sistema Elettrico" ha sviluppato il tool LENICALC. Si tratta di uno strumento software per la determinazione del LENI secondo il metodo completo (metodo 1) della UNI EN 15193-1:2017. Scopo del progetto è quello di progettare la struttura dati che consenta di portare su Web le funzionalità che oggi sono del LENICALC. L'obiettivo è di consentire, in futuro, l'integrazione della funzionalità di calcolo nei portali ENEA relativi ai progetti in cui in calcolo del LENI sia parte integrante.

5.2 Condizioni al contorno

Il calcolo del LENI è sostanzialmente funzione di due insiemi di dati: i dati fisici dell'edificio ed i dati relativi alla illuminazione. Dal punto di vista pratico spesso i dati seguono ambiti separati. Quelli fisici sono relativi al momento della progettazione architettonica ed ingegneristico strutturale, quelli sulla illuminazione attengono alla progettazione degli impianti di illuminazione. In entrambi i casi i progettisti utilizzano sistemi software specializzati, e quindi diversi fra loro. Si aprono dunque diversi scenari, entro cui un sistema universale di calcolo del LENI deve posizionarsi.

5.3 Considerazioni sui formati dati

Si è partiti dalla considerazione che nel mondo architettonico esiste oggi uno standard di fatto che è il software AudoCad di Autodesk, nelle sue diverse espressioni bidimensionale e tridimensionale. Al di là del caso specifico, quello che è diventato un vero standard è la struttura di files ASCII che è capace di contenere tutte le informazioni geometriche dei progetti, ovvero il formato DXF. Il fatto che Autodesk abbia reso aperto il formato, fa si che esso sia amplissimamente utilizzato anche da software non autodesk. Questa apertura ha fatto si che il DXF sia diventato un formato di scambio, nel senso che diversi software che siano capaci di leggere e scrivere DXF sono in gradi di scambiarsi l'intero progetto. Seguendo questo ragionamento si può traguardare un futuro simile ruolo per altri tipi di files, ad esempio gli shapefile. Questi ultimi sono notoriamente files che rappresentano un universo vettoriale, dove troviamo enti geometrici e le informazioni a loro associate.

5.4 Ipotesi progettuale

Partendo dalla ipotesi che il formato DXF è di fatto il formato in cui tutti i progetti, nella loro componente bidimensionale, possono essere esportati, si è proceduto ad uno studio specifico della capacità del DXF di gestire dei metadati. Si è cercato dunque di esaminare se e come si potessero inserire nel DXF dati che non sono direttamente relativi alla geometria. In effetti si è visto che il DXF prevede questa possibilità e quindi si è lavorato concretamente su questa ipotesi. Lo scenario che si è ipotizzato è stato il seguente: se il progettista ha sviluppato il suo progetto geometrico, si potrebbe inserire nel progetto stesso i dati ulteriori necessari al calcolo del LENI. In questo modo si potrebbe ipotizzare un workflow di questo tipo:

- Il progettista completa il progetto architettonico
- Segue le istruzioni per inserire i dati ulteriori per il calcolo del LENI
- Esporta tutto in DXF
- Il LENICALC WEB legge il dxf, calcola il LENI, esporta il report

Lo studio attuale ha dunque come obiettivo di definire le metodologie possibili per consentire quanto al secondo item precedente, ovvero dare la possibilità di inserire metadati per il LEN utilizzando gli strumenti di CAD standard, ed eventualmente altri strumenti di uso comune come i componenti di Office Automation universalmente diffusi.

5.5 Inserimento metadati per LENI

Partendo dalla ipotesi che il formato DXF è di fatto il formato in cui tutti i progetti, nella loro componente bidimensionale, possono essere esportati, si è proceduto ad uno studio specifico della capacità del DXF di gestire dei metadati ed alle modalità per inserire tali dati nel DXF stesso. Grazie alla disponibilità di progetti reali che Arch4energy realizza nella sua componente di progettazione Architettonica, si è potuto realizzare una simulazione "reale" basata sugli elaborati grafici di un progetto di edificio di villeggiatura già realizzato. Si sono ipotizzate due modalità di inserimento dati per il LENI: la prima utilizzando direttamente l'interfaccia utente di Autocad, la seconda utilizzando una struttura dati esterna.

5.5.1 Inserimento metadati da Autocad

Sfruttando la struttura dati di Autocad, si è ipotizzato di inserire i metadati per LENI direttamente da interfaccia utente. La struttura dati prevede infatti, tra gli altri, i blocchi. Il blocco costituisce una serie di entità raggruppate e ad esso possono essere associati attributi non grafici ma testuali. Lo scopo è proprio quello di sfruttare l'integrazione con altre componenti che vadano ad estrarre i dati testuali per trattarli poi opportunamente. Ovviamente un valore numerico seppur trattato come testuale dal cad, è immediatamente gestibile a livello programmatico, effettuando la conversione da testo a numero. Il vantaggio di questa metodologia è che l'utente utilizza l'interfaccia a cui è abituato, ma con lo svantaggio di dover inserire i dati seguendo una sequenza di operazioni da imparare e che è richiesta solo per questo scopo.

5.5.2 Inserimento metadati da fonti esterne

Una delle capacità di Autocad è quella di prevedere la connessione del dissegno a fonti di dati esterne, come ad esempio un database relazionale. Il punto di forza di questa metodologia è che si opera esternamente al disegno verso e proprio. Se si dovesse scegliere un database relazionale, ad esempio, si dovrebbe anche prevedere lo sviluppo di una applicazione di interfaccia specifica. Il problema allora diventa la scelta della base dati esterna, in modo da rendere questo passaggio il più semplice possibile.

5.5.3 Soluzione proposta

Nell'ottica di orientarsi su una procedura semplice e che non richieda altro che l'utilizzo di strumenti di uso comune, Arch4energy ha ipotizzato di utilizzare un semplice foglio Excel come base dati. Il tema diventa quindi quello di definire una struttura logica dei dati per il LENI che sia facilmente inscrivibile nella creazione di uno o più fogli di calcolol excel e che sia altrettanto facilmente agganciabile al relativo file grafico. La soluzione è stata quella di organizzare i dati per il LENI in tabelle, il cui nome viene richiamato come blocco del file Autocad. Questa semplicissima modalità offre evidenti vantaggi:

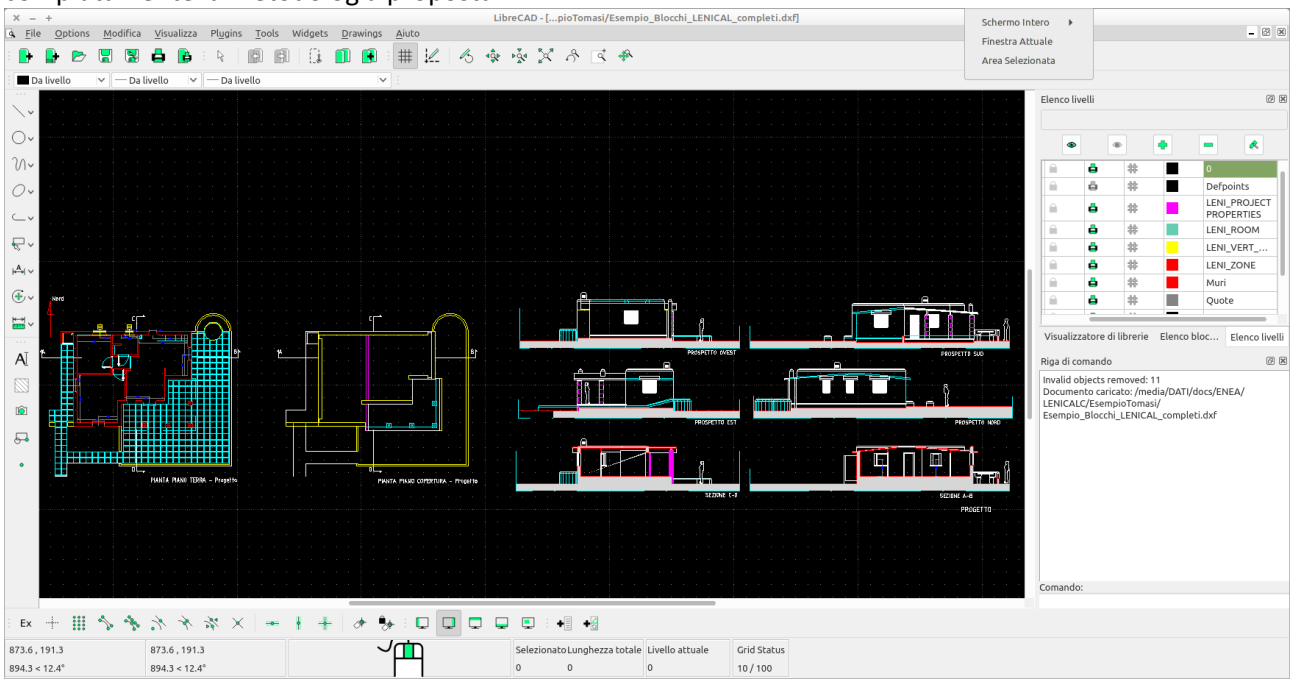
- l'operatore organizza i dati in tabelle relative ai vari ambiti richiesti (PROJECT, ROOMS, ZONES, WINDOWS, etc)
- inserisce i dati in excel
- assegna ai blocchi lo stesso nome del foglio
- esporta tutto in DXF

Il file dxf a quel punto contiene tutto quanto necessario, sia a livello geometrico, sia a livello di dati per il calcolo del LENI, ed è pronto per essere elaborato da un processo di tipo "filtro". Con filtro si intende – come di consueto – un processo che legge il file di input e crea un file di output. Ovviamente sarà demandato al livello applicativo il controllo formale e sostanziale dei dati che sono stati integrati nel DXF, ma a questo punto il workflow sarà ben delineato e può essere descritto come segue:

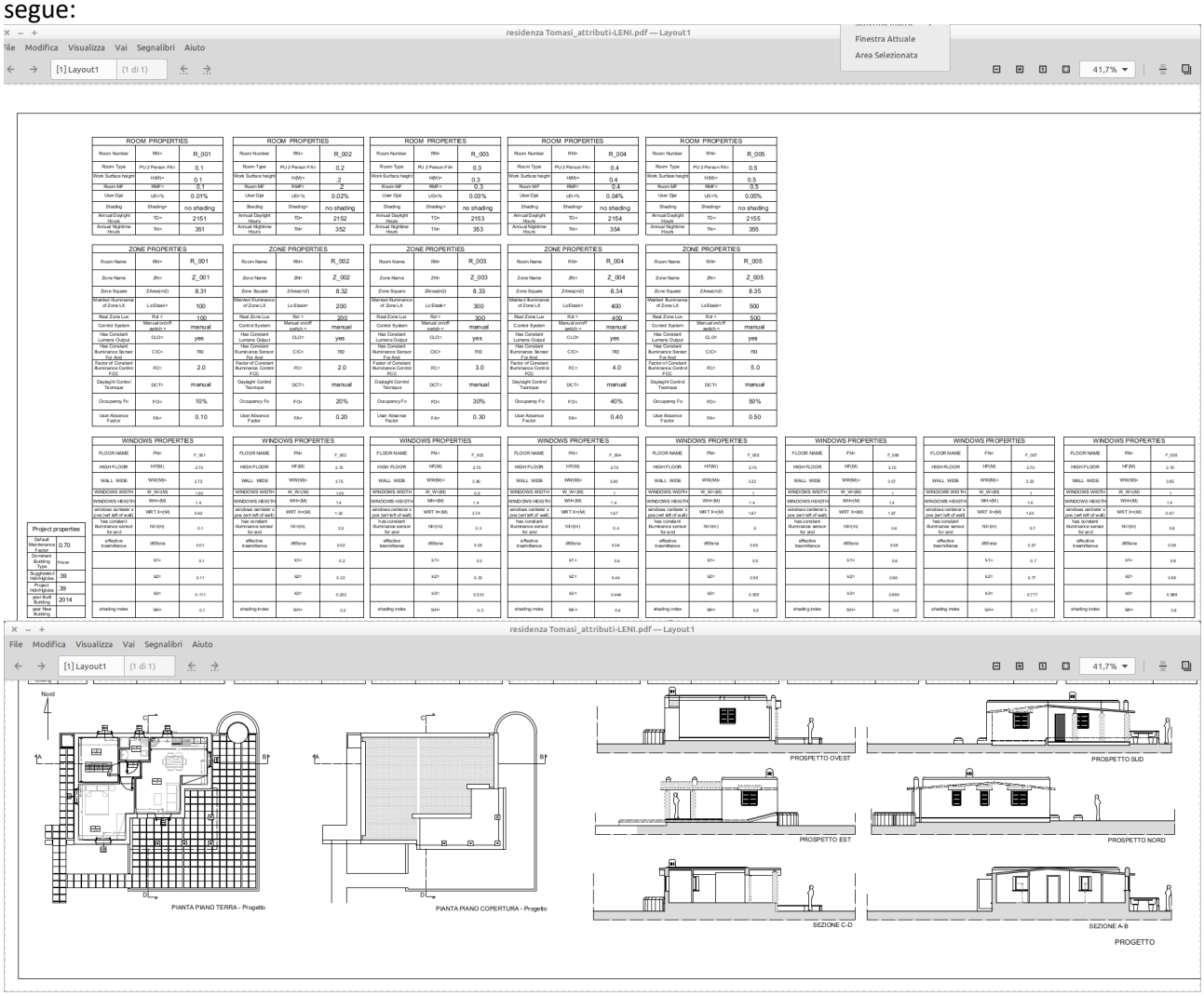
- l'operatore organizza i dati in tabelle relative ai vari ambiti richiesti (PROJECT, ROOMS, ZONES, WINDOWS, etc)
- inserisce i dati in excel
- assegna ai blocchi lo stesso nome del foglio
- esporta tutto in DXF
- il filtro esamina il contenuto
- se il contenuto è validato
    - i dati passano al calcolo del LENI
- se il contenuto NON è validato
    - si torna all'inizio

5.5.4 Esempio reale

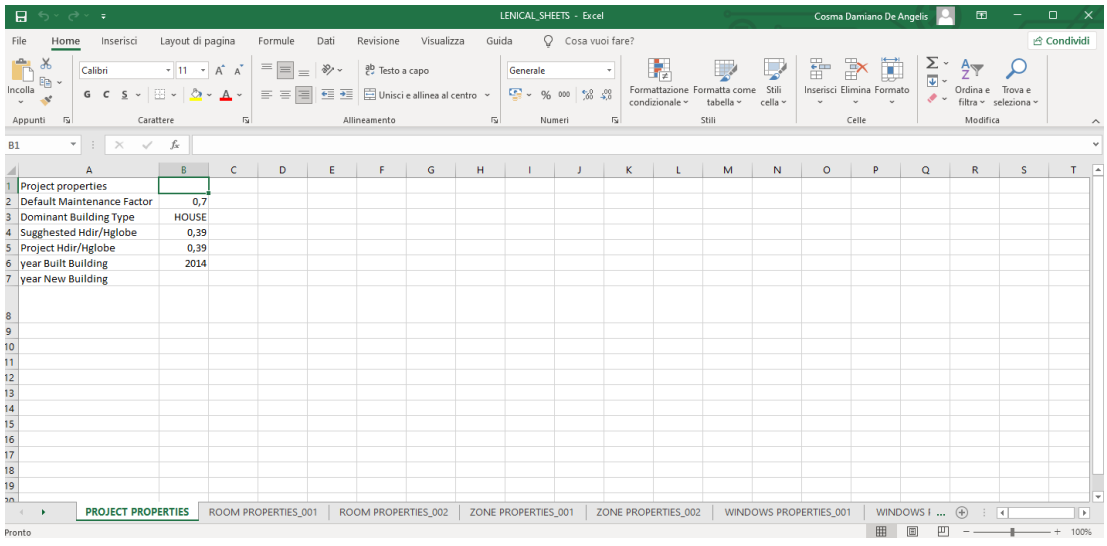
Come accennato in precedenza Arch4energy comprende un settore di Progettazione Architettonica, per cui è stato possibile verificare le varie ipotesi in relazioni ad un edificio reale. Nelle figure che seguono sono riportati tutti gli elementi che costituiscono l'intera base di dati, sia grafici che meta, tali da rappresentare compiutamente la metodologia proposta.

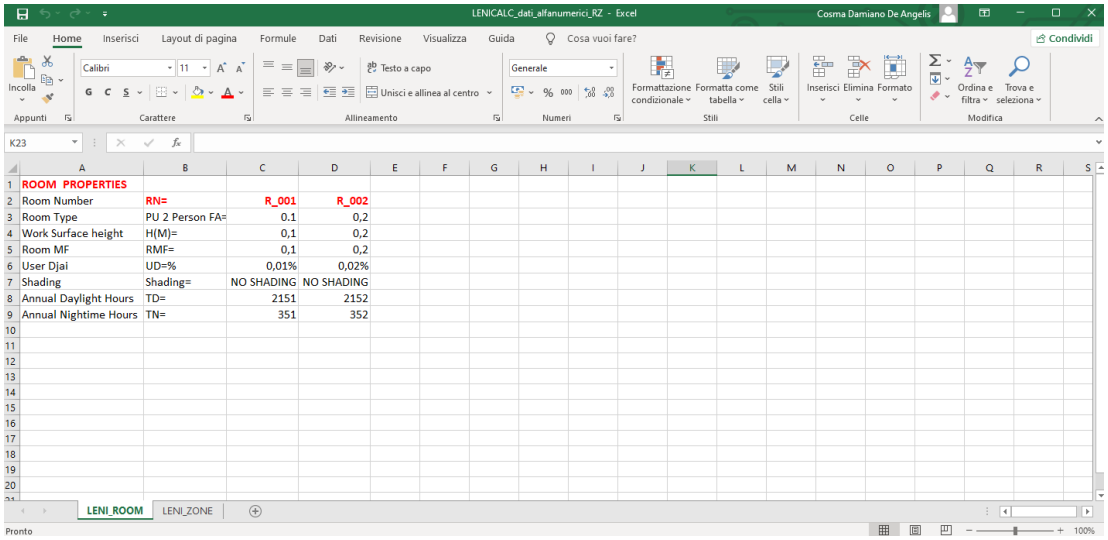


Nella precedente figura si può vedere l'architettonico e, a destra, la vista dei livelli riferiti ai metadati per il LENI. I dati tabellari, contenenti le variabili ed i valori sono riportati – in forma stampata - nella figura che segue:

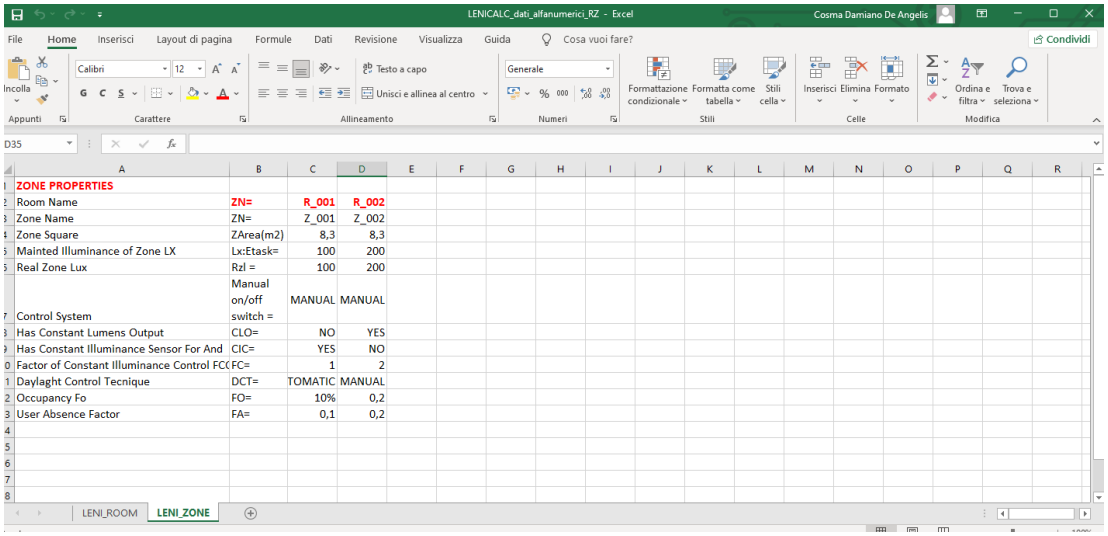


Di fatto il progettista dovrà prepararsi tutta questa serie di dati tabellari, che avrà poi cura di inserire materialmente in un foglio excel. Anche le formattazione di fogli excel è stata ipotizzata in modi diversi, sempre allo scopo di individuare la procedura più semplice. Di seguito sono riportati esempi delle strutture excel sperimentate.





	A	B	C	D
1	ROOM PROPERTIES			
2	Room Number	RN=	R_001	R_002
3	Room Type	PU 2 Person FA=	0.1	0,2
4	Work Surface height	H(M)=	0,1	0,2
5	Room MF	RMF=	0,1	0,2
6	User Djai	UD=%	0,01%	0,02%
7	Shading	Shading=	NO SHADING	NO SHADING
8	Annual Daylight Hours	TD=	2151	2152
9	Annual Nightime Hours	TN=	351	352



	A	B	C	D
1	ZONE PROPERTIES			
2	Room Name	ZN=	R_001	R_002
3	Zone Name	ZN=	Z_001	Z_002
4	Zone Square	ZArea(m2)	8,3	8,3
5	Mainted Illuminance of Zone LX	Lx:Etask=	100	200
6	Real Zone Lux	Rzl =	100	200
7	Control System	Manual on/off switch =	MANUAL	MANUAL
8	Has Constant Lumens Output	CLO=	NO	YES
9	Has Constant Illuminance Sensor For And	CIC=	YES	NO
10	Factor of Constant Illuminance Control FC(	FC=	1	2
11	Daylaght Control Tecnique	DCT=	TOMATIC	MANUAL
12	Occupancy Fo	FO=	10%	0,2
13	User Absence Factor	FA=	0,1	0,2

5.5.5 Utilities di supporto

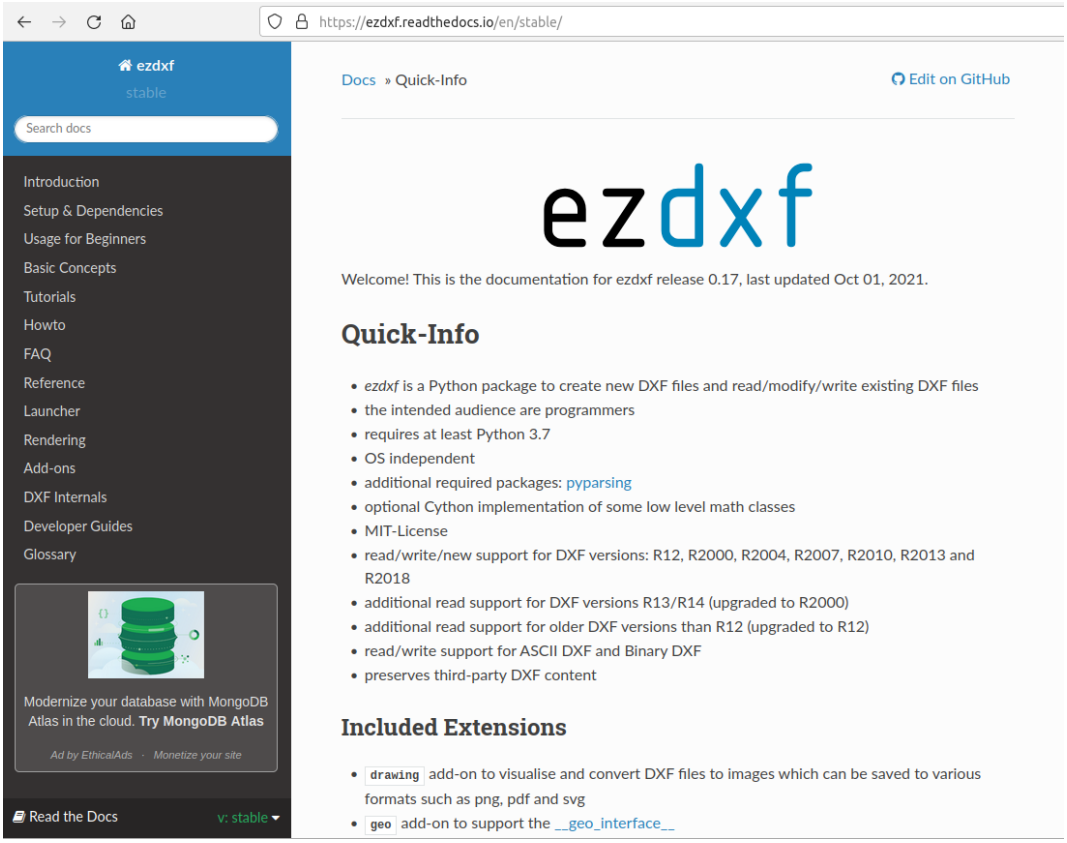
A dimostrazione della estrema gestibilità dei formato DXF, si riportano di seguito alcuni tools open source che permettono di esaminare i files DXF. Sono tools che possono essere usati direttamente a command line, o anche integrati in un filtro di validazione della metodologia proposta. Il primo tool "ogrinfo" che è in grado di filtrare e trasformare in diversi modi il file DXF. Il filtro fa parte della suite "GDAL" (https://gdal.org) che permette di analizzare e trasformare molti formati dati di file raster e vettoriali geospaziali. A puro titolo di esempio è immediato verificare che in un file DXF siano presenti i layer che ci aspettiamo. Il file che esaminiamo è il LENICALC_blocchi_completi.dxf:

$ ogrinfo -al LENICALC_blocchi_completi.dxf | grep LENI_ | sort -u

Layer (String) = LENI_ROOM

Layer (String) = LENI_ZONE

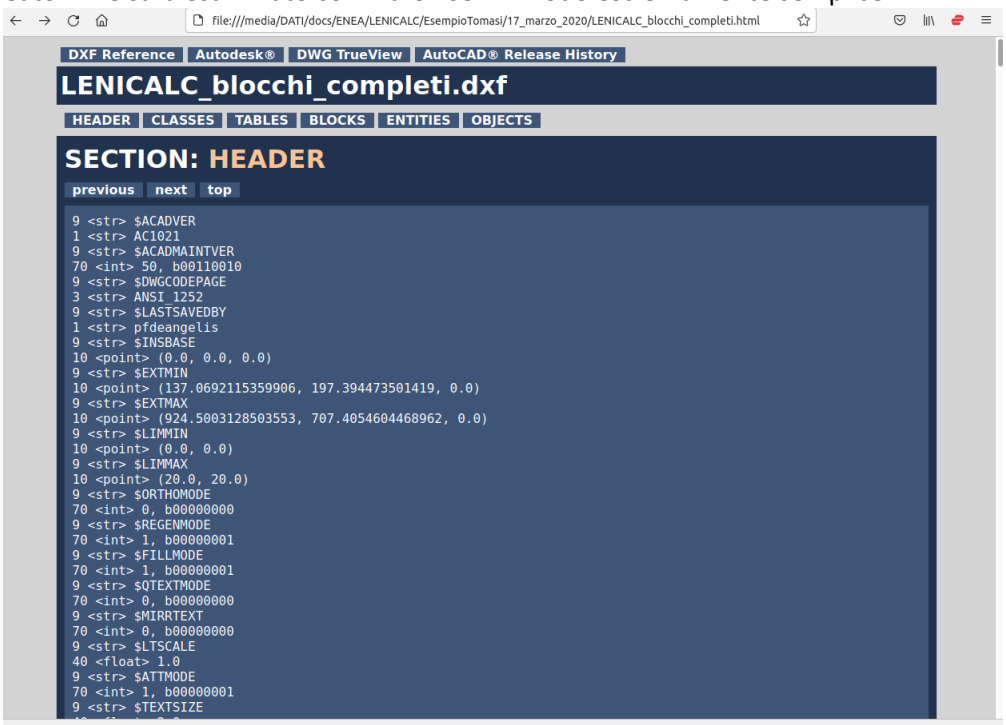
Il secondo tool è una libreria python open source denominata "ezdxf" di cui di seguito il sito:



Un comando molto utile è il ezdxf.pp dove pp sta per “Pretty Print”. Eseguendo il comando si trasforma il DXF in un html facilmente navigabile con un normale browser web. Eseguendo il comando sul file precedente si ottiene un file html con lo stesso nome.

$ python -m ezdxf.pp LENICALC_blocchi_completi.dxf
dxfpp created 'LENICALC_blocchi_completi.html'

Una volta creato il file sarà esaminato con ll browser in modo estremamente semplice



5.6 Esperienza su edificio reale

Nell'ottica della ipotesi di progetto di effettuare una verifica del calcolo del LENI rispetto ai dati reali di consumo, è stato installato un sistema di monitoraggio dei parametri elettrici nella scuola sita in Via Sorelle Agazzi in Sonnino Scalo (LT). Si tratta di un edificio moderno, ad un solo piano, con aule luminose, grande atrio e cortile esterno utilizzato come parco giochi. Dato però il particolare periodo dovuto alla pandemia Covid, i dati di consumo sono risultati essere lontani da quelli prevedibili in una condizione di normalità. Ciò ha reso impossibile effettuare ad oggi la verifica prevista. Al fine di poter effettuare la verifica prevista al ristabilimento delle condizioni di normale attività, il sistema di data-logging e monitoring viene lasciato attivo per un tempo ulteriore, ad oggi previsto fino alla fine del 2022.

Di seguito uno screenshot del sistema allo stato attuale:

Local site scuole, Sonnino_Scalo

STATE	SERVICE	ICONS	STATUS DETAIL	AGE	CHECKED	PERF-O-METER
OK	Luce_Corrente_3F		CORRENTE TRIFASE[A] [0.300]	2020-09-12 10:53:27	2021-11-17 09:05:05	
OK	Luce_Corrente_L1		CORRENTE LINEA_L1[A] [0.678]	2020-09-12 10:54:08	2021-11-17 09:05:46	
OK	Luce_Corrente_L2		CORRENTE LINEA_L2[A] [0.000]	2020-09-12 10:54:48	2021-11-17 09:06:26	
OK	Luce_Corrente_L3		CORRENTE LINEA_L3[A] [0.000]	2020-09-12 10:55:28	2021-11-17 09:02:06	
OK	Luce_Energia_3F		ENERGIA TRIFASE[kWh] [25165.0]	2020-09-12 10:51:46	2021-11-17 09:03:24	
OK	Luce_Potenza_3F		POTENZA TRIFASE[kW] [0.126]	2020-09-12 10:52:27	2021-11-17 09:04:05	
OK	Luce_Potenza_L1		POTENZA L1[kW] [0.126]	2020-09-12 10:53:07	2021-11-17 09:04:45	
OK	Luce_Potenza_L2		POTENZA L2[kW] [0.000]	2020-09-12 10:53:47	2021-11-17 09:05:25	
OK	Luce_Potenza_L3		POTENZA L3[kW] [0.000]	2020-09-12 10:54:28	2021-11-17 09:06:06	
OK	Luce_Volt_3F		VOLTAGGIO TRIFASE[V] [396]	2020-09-12 10:55:08	2021-11-17 09:01:46	
OK	Temperatura_Quadro		TEMPERATURA[C] [18.0]	2020-09-12 10:55:49	2021-11-17 09:02:27	