Universitat Oberta de Catalunya

Integració NMS – SIG

Memòria

Màster Programari Lliure

Títol: Integració NMS - SIG

Autor: Antoni Rosa Ruiz

Consultor: Gregorio Robles

Tutor extern: Carles Bock

Data: 15 de Juny de 2012

Aquest document és l'entrega final del projecte “Integració NMS – SIG” de final de màster de programari lliure.

L'objectiu principal del projecte és desenvolupar una solució de programari que permeti integrar un sistema de monitorització d'elements i serveis en xarxa amb un sistema SIG. La finalitat és d'una banda, poder disposar d'una geo-localització dels elements gestionats que puguin ser representats al damunt d'una base cartogràfica donada via un *mashup*[1], i de l'altra, implementar un sistema d'integració que permeti definir certa lògica de negoci usable per a què un motor SIG pugui decidir característiques estrictament de representació segons el seu estat.

[1] El terme anglès *mashups* normalment fa referència a una aplicació, normalment web, que usa i integra informacions de procedència variada per crear un resultat o servei nou

[2] http://creativecommons.org/licenses/by-sa/3.0/

Índex de continguts

Índex d'il·lustracions

Introducció

A continuació es pot trobar l'estudi realitzat per poder prendre decisions sobre el disseny i desenvolupar una solució que s'ajusti al resultat final desitjat.

Estudi NMS

NMS és l'acrònim de Network Management System i es defineix com una solució de programari per recollir informació de diferents elements de xarxa i que implementa certa funcionalitat d'administració dels elements i de la informació recollida. Esquemàticament, les principals característiques són:

- Adquisició de la informació:

  L'adquisició d'informació es pot realitzar mitjançant agents que usant diferents protocols recullen informació de les diferents interfícies *northbound* dels elements i components de la xarxa.

- Administració de la informació recollida:

  La informació recollida permet realitzar importants funcions de gestió: es poden definir i gestionar esdeveniments que a partir d'errors o valors llindar desencadenadors d'alarmes, o que en funció d'errors detectats en equips, s'executin canvis en la configuració , etc..

- Interfície d'administració:

  La interfície d'administració facilita a l'administrador de les xarxes les tasques d'operativa i gestió dels elements i dels informes d'estat, d'errors, etc..

Comparació Nagios i OpenNMS

Característiques OpenNMS

OpenNMS és un entorn NMS de programari lliure desenvolupat en Java, força popular i amb diferents reconeixements a nivell empresarial que compta amb una comunitat d'usuaris molt extensa, i que té suport comercial per part del mateix equip de desenvolupament, l'OpenNMS groupi. Tot el seu codi font està disponible sota la llicència General Public License v3.

De les seves característiques destaquenii:

- Descobriment automàtic d'equips, serveis i topologia de xarxa de capa 2 i capa 3
- Gestió d'esdeveniments externs (traps SNMP) o autogenerats segons la informació recollida
- Sistema detallat d'informes de les dades monitoritzades
- Diferents maneres i protocol de monitorització d'equips i serveis: pings ICMP, SNMP, *parsers* HTTP, integració amb JMX, etc..
- Té una interfície REST que permet una fàcil integració per part de terceres aplicacions

Característiques Nagios

Nagios és un entorn NMS de programari lliure també força popular desenvolupat majoritàriament en C, però també amb algunes parts en Perl, i és una de les eines NMS d'us empresarial més estàndard. Compta amb una comunitat d'usuaris molt extensa. Tot el codi font està disponible sota la llicència General Public License v2, a més d'una llicència pròpia.iii

De les seves característiques principals destaquen:

- Motorització de múltiples serveis i protocols
- Gestió d'esdeveniments externs (traps SNMP) o autogenerats segons la informació recollida
- Sistema detallat d'informes de les dades monitoritzades
- Suport per servidors de monitorització redundats
- Gran quantitat de plugins que n'estenen la funcionalitat

Solució escollida

Les característiques dels dos sistemes NMS són equivalents si es té en compte que ambdós sistemes permeten augmentar la funcionalitat base mitjançant complementsiv. Això és important sobretot per Nagios que de sèrie no ofereix tantes prestacions com l'OpenNMSv**.** No obstant, el fet que OpenNMS sigui una mica més fàcil de posar en marxa i, considerant que en el present projecte es volen monitoritzar equips i serveis en un entorn força heterogeni, fa que l'ús d'aquest sistema sigui més adequat pel present projecte.

Així mateix, el sistema d'*auto discovery* d'OpenNMS es considera una funcionalitat interessant per l'escenari de proves dissenyat per realitzar proves, com també ho és el fet de que disposa d'una interfície REST d'integració, apart de les llibreries de desenvolupament de complements (tal i com també té Nagios, que li sumen punts en quant a facilitat d'integració).

REST interface

OpenNMS disposa d'una interfície REST amb les següents característiques: Format de la informació: XML (per defecte) i JSON (si s'especifica en un paràmetre HTTP de la crida). A continuació es mostra una llista de la funcionalitat més rellevant pel present projecte que OpenNMS ofereix a través de la interfície REST:

Dominis d'informació :

NODES

Verbs suportats **GET POST PUT DELETE**

Aquest *end-point* es pot usar per llistar i administrar tots els elements monitoritzats a l'OpenNMS.

EVENTS

Verbs suportats **GET PUT**

Aquest *end-point* es pot usar per llistar i gestionar tots els successos ocorreguts al sistema.

ALARMS

Verbs suportats **GET PUT**

Aquest *end-point* es pot usar per llistar i gestionar tots els successos catalogats com alarmes ocorregudes al sistema.

ACKNOWLEDGEMENTS

Verbs suportats **GET POST**

Aquest *end-point* es pot usar per llistar i gestionar totes les desactivacions d'alarmes del sistema.

NOTIFICATIONS

Verbs suportats **GET**

Aquest *end-point* es pot usar per llistar totes les notificacions d'esdeveniments del sistema.

OUTAGES

Verbs suportats GET

Aquest *end-point* es pot usar per llistar la llista d'elements que han sigut considerats fora de servei pel sistema.

REQUISITIONS

Verbs suportats **GET POST PUT DELETE**

Aquest *end-point* es pot usar per llistar i gestionar la grups de nodes al sistema.

FOREIGN SOURCES

Verbs suportats **GET POST PUT DELETE**

Aquest *end-point* permet llistar i gestionar les fonts d'autodiscovery d'elements monitoritzats pel sistema.

SNMP Configuration

Verbs suportats **GET PUT**

Aquest *end-point* permet llistar i gestionar la informació SNMP dels nodes registrats al sistema.

MAPS

Verbs suportats **GET POST PUT DELETE**

Aquest *end-point* permet llistar i gestionar mapes lògics de la topologia d'elements al sistema.

USERS

Verbs suportats **GET POST PUT DELETE**

Aquest *end-point* permet llistar i gestionar grups d'usuaris del sistema.

GROUPS

Verbs suportats **GET POST PUT DELETE**

Aquest *end-point* permet llistar i gestionar grups d'usuaris del sistema.

ALARMS Statistics

Verbs suportats **GET**

Aquest *end-point* permet llistar estadístiques bàsiques referents a les alarmes registrades del sistemes.

PostgreSQL, PostGIS

PostgreSQL és un Sistema Gestor de Bases de Dades Relacional i Orientat a Objectes Robust, i PostGIS és una extensió d'aquest que aporta característiques pel manteniment i manipulació de dades geoespacials segons l'estàndard definit per la Open Geospatial Consortiumvi.

L'ús de PostgreSQL i PostGIS ve donat pel context en el que es desenvolupa el projecte, on ja es disposa d'algunes solucions basades en aquestes eines i, per tant, se'n coneixen les principals característiques i se sap que aquestes donen cobertura a les necessitats plantejades pel projecte. Totes dues eines són programari lliure.

Desenvolupament del projecte

El primer pas en el desenvolupament és aconseguir totes aquelles eines que es necessitaran per dur a terme el projecte, bé ja siguin eines amb un rol actiu en l'arquitectura final de la solució desenvolupada (p.e. Un sistema gestor de Base de dades) o coma part necessària d'un desenvolupament estructurat (p.e. Un gestor de versions de codi font.)

Instal·lació del programari

El sistema operatiu de base amb el que es du a terme el desenvolupament i s'instal·len els serveis/aplicacions pel desenvolupament, sempre que no es digui el contrari, és l'Ubuntu 10.04 LTS Server de 64 bits.

Instal·lació Subversion

De cara al desenvolupament es va decidir disposar d'un repositori de control de versions per poder gestionar els canvis, versions i futures ampliacions i ramificacions del codi. A continuació es descriu la feina duta a terme per instal·lar Subversion i crear aquest repositori de codi.

Instal·lació de subversion:

```
# sudo apt-get install subversion libapache2-svn
```

Configurar un repositori: crear-lo

```
# mkdir /home/usuari/repositori
```

```
# svnadmin create /home/usuari/repositori/projecteX
```

Configurar un repositori: servir-lo via apache

```
# sudo chown -R www-data:www-data /home/usuari/repositori/
```

```
# sudo nano /etc/apache2/sites-available/default
```

I afegir-hi la informació necessària per servir el repositori via HTTP i amb autenticació bàsica:

```
<Location /svn>
 DAV svn
 SVNParentPath  /home/usuari/repositori
 SVNListParentPath On
 AuthType Basic
 AuthName "Repositori de Control de Versions"
 AuthUserFile /etc/subversion/passwd
 Require valid-user
</Location>
```

Si volguéssim afegir dos usuaris podríem fer-ho així:

```
# sudo htpasswd -c /etc/subversion/passwd usuari_http
# sudo htpasswd /etc/subversion/passwd SegonUsuari_http
```

I caldria reiniciar el servidor web...

```
# sudo apache2ctl restart
```

Instal·lació PostgreSQL

```
$ sudo apt-get install postgresql postgresql-contrib
```

Instal·lació de l'*admin pack*

```
$ su - postgres
$ psql template1 <
/usr/share/postgresql/8.4/contrib/adminpack.sql
```

Establir contrasenya de l'usuari administrador (postgres) de la BBDD:

```
$ sudo -u postgres psql postgres
```

```
\password postgres
\q
```

Addicionalment, caldrà executar la següent comanda en una consola de PostgreSQL per habilitar la creació de STORED i TRIGGERS a la BBDD:

```
CREATE FUNCTION plpgsql_call_handler () RETURNS OPAQUE AS
'/usr/lib/postgresql/8.4/lib/plpgsql.so' LANGUAGE 'C';CREATE
LANGUAGE 'plpgsql' HANDLER plpgsql_call_handler LANCOMPILER
'PL/pgSQL';
```

Finalment, per poder fer consultes entre diferents BD cal instal·lar el complement DBLINK de PostgreSQLvii:

```
#psql  -Upostgres -W NOM_DE_LA_BBDD <
/usr/share/postgresql/8.4/contrib/dblink.sql
```

Instal·lació PostGIS

Els passos a realitzar pel PostGIS són:

```
$ sudo apt-get install postgresql-8.4-postgis
$ sudo -u postgres psql postgres
$ psql -f /usr/share/postgresql/8.4/contrib/postgis.sql
$ psql -f /usr/share/postgresql/8.4/contrib/postgis_comments.sql
```

Un cop instal·lat, caldrà fer alguns pasos si es vol al crear una BD amb les característiques que aporta PostGIS habilitades:

```
$ createuser USUARI
$ createdb -E UTF8 -O USUARI NOM_DE_LA_BBDD
$ createlang plpgsql NOM_DE_LA_BBDD
$ exit
$ psql -f /usr/share/postgresql/8.4/contrib/postgis.sql -d
NOM_DE_LA_BBDD
```

Instal·lació UNMAPServer

Els passos a realitzar pel servidor de mapes són:

```
$ sudo apt-get install cgi-mapserver mapserver-bin mapserver-doc
php5-mapscript cgi-mapserver fontconfig-config libfcgi0ldbl
libfontconfig1 libgd2-xpm
```

Cal modificar la configuració de UNMapserver per configurar-hi els *datums* que s'usaran si aquests no hi són. Fins si tot si hi són, és interessant modificar l'arxiu per millorar-ne el rendiment:

Cal editar l'arxiu */usr/share/proj/epsg* de la següent manera:

```
# 1er els datums + usats, per + rendiment. MMN.

# ED50 / UTM zone 31N

<23031> +proj=utm +zone=31 +ellps=intl +units=m +no_defs  <>

# 900913 / Projeccio Google

<900913> +proj=merc +a=6378137 +b=6378137 +lat_ts=0.0
+lon_0=0.0 +x_0=0.0 +y_0=0

 +k=1.0 +units=m +nadgrids=@null +no_defs <>

# WGS 84
<4326> +proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs  <>

# ED50 / UTM zone 30N

<23030> +proj=utm +zone=30 +ellps=intl +units=m +no_defs  <>
```

(...)

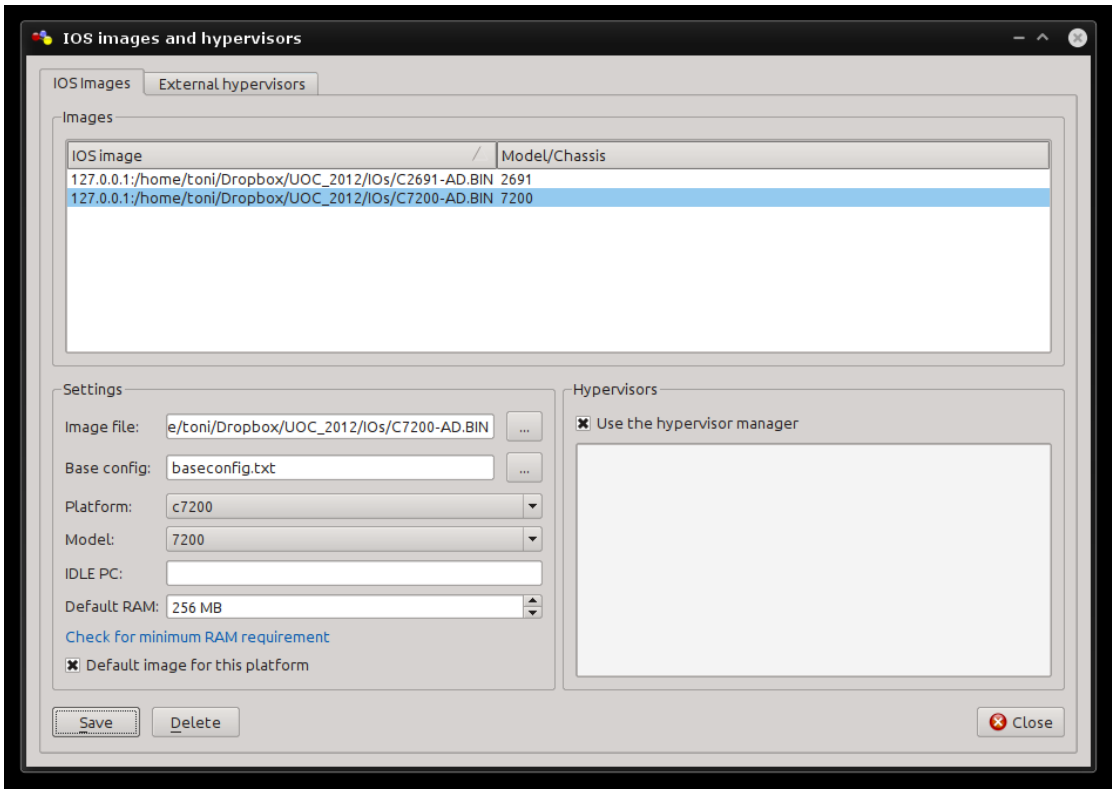
La resta de l'arxiu de configuració es pot deixar tal i com està.

Instal·lacióGNS3

Instal·lem el GNS3, el simulador de xarxes:

```
# sudo apt-get install gns3 dynamips
```

Posem en marxa l'aplicació i configurem les imatges del sistema IOs del routers. Concretament simularem els models CISCO C7200 i un C2691:



Il·lustració 1: Càrrega de les imatges binàries dels routers CISCO C7200 i C2691

Per poder usar les imatges dins de l'emulador es pot establir la configuració per defecte dels *routers* dins de l'arxiu *baseconfig.txt.* Pel present projecte s'estableix com es mostra a continuació:

```
!
hostname %h
no ip domain-lookup
line con 0
exec-timeout 0 0
logging synchronous
privilege level 15
```

```
service timestamps debug datetime msec
alias configure show do show
alias interface show do show
line vty 0 15
no login
exec-timeout 0 0
logging synchronous
privilege level 15
```

Mini introducció a CiscoIOs

La interfícies *ethernet* de Xarxa s'anomenen FasthEthernetN/N (i en les comandes es pot usar l'acrònim FN/N) i els enllaços amb cable *serial* SerialN/N (i en les comandes es pot usar l'acrònim SN/N) on N és un numeral que indica la posició de la targeta (normalment els equips en tenen més d'una).

Algunes comandes bàsiques:

Comanda: *show running-configuration*

Mostra la configuració actualment en ús pel dispositiu

Comanda:*show interface*

Mostra la informació de la interfície de xarxa

Comanda: *show ip interface*

Mostra informació IP del interfície de xarxa

Comanda: *show ip route*

Mostra la taula d'enrutament del dispositiu

Text 1: Mini manual de supervivència de CISCO iOS

Instal·lació OpenNMS

Pel present projecte se segueixen les instruccions recomanades pel OpenNMS en la seva instal·lació en sistemes Debian/Ubuntu.viii

Els passos, esquemàticament, són:

1. Afegir el repositori del projecte:

```
deb http://debian.opennms.org stable main
deb-src http://debian.opennms.org stable main
```

2. Importar la clau del repostori

```
wget -O - http://debian.opennms.org/OPENNMS-GPG-KEY | sudo apt-key add -
```

3. Instal·lació de dependències si no existeixen prèviament al sistema: PostgreSQL i JDK

```
sudo apt-get install postgresql
```

Afegim a l'arxiu "/etc/apt/sources.list.d" el repositori del JDK:

```
deb http://archive.canonical.com/ubuntu lucid partner
```

I executem la instal·lació del paquet JDK

```
sudo apt-get update
sudo apt-get install sun-java6-jdk
```

4. Instal·lació del paquet

```
sudo apt-get update

sudo apt-cache install opennms
```

5. Inicialització de la Base de dades d'OpenNMS

```
sudo /usr/share/opennms/bin/install -dis
```

Arribats aquest punt ja es pot accedir a la interfície web d'OpenNMS. Apuntant un navegador a http://localhost:8980/openms des de la mateixa màquina on s'ha instal·lat l'NMS s'hi hauria de poder accedir. El directori amb les opcions de configuració és "/etc/openms".

Les opcions bàsiques de funcionament inicial caldrà ajustar-les a l'arxiu l'anomenat "oenmns.properties" ix.

Definició Escenari

L'objectiu en definir un escenari de proves és intentar reproduir un entorn semblant al que la solució desenvolupada tindrà un cop es posi en producció. A tal efecte, es procedeix a definir i implementar una simulació mínima però que contingui molts dels elements que trobaríem en una implementació de veritat.

Per la definició de l'escenari de proves s'usarà GNS3, un simulador de xarxes de programari lliure que permet la simulació de xarxes complexes. A l'emular el maquinari permet definir laboratoris i escenaris de proves emprant les mateixes imatges del *Firmware* dels elements de xarxa simulats. Gràcies a això, el comportament final dels equips de xarxa simulats és molt fidedigne al que tindrien en la realitat.

També s'usarà el programa de virtualització de maquinari VirtualBox. VirtualBox és en la seva majoria programari lliure, sent actualment un projecte d'Oracle Corporation.

De cara a la definició de l'escenari s'usaran algunes instàncies de màquines virtuals (MV) GNU/Linux. L'escenari que s'emprarà per proveir dades a l'NMS consistirà en simular uns routers CISCO controlant l'accés entre tres xarxes i aquestes MV amb alguns serveis en marxa (HTTP, SMTP). A els routers se'ls hi habilitarà SNMP per proporcionar informació i estadístiques a l'NMS.

Descripció del maquinari

Màquines Virtuals:

- VM_NMS. Debian GNU/Linux (LAN 1)
- VM_L2. Debian GNU/Linux (LAN 2)
- VM_L4. Debian GNU/Linux (LAN 4)

Routers:

- R1 IOs 2691 (LAN 1 – LAN 2)
- R2 IOs 2691 (LAN 2 – LAN 3)
- R3 IOs 2691 (LAN 3 – LAN 4)

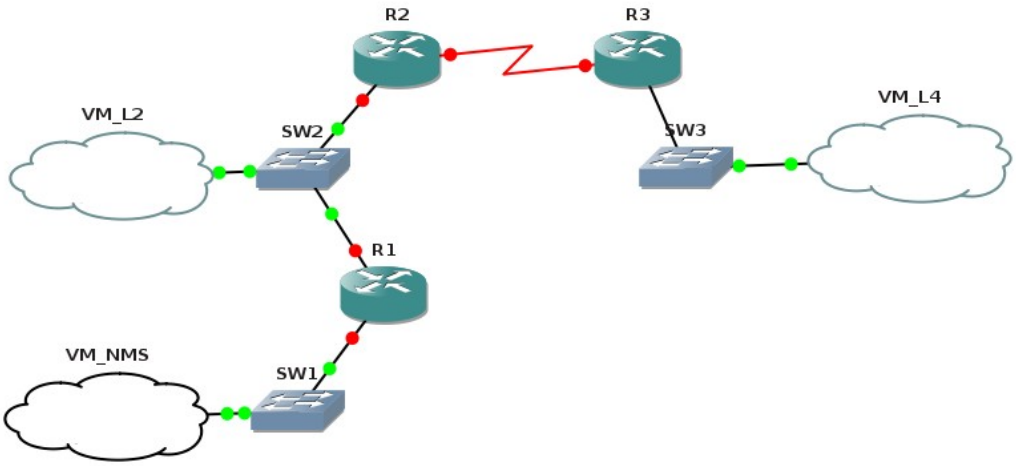
Switches:

- SW1 (LAN1)
- SW2 (LAN2)
- SW3 (LAN4)

Topologia

L'escenari consistirà en quatre segments de xarxa (LAN 1 -> LAN 4) interconectats per 3 *routers* mitjançant *FastEthernet,* exceptuant l'enllaç entre R2 i R3 que s'efectuarà per un port Serial.

Topologia de xarxa de l'escenari de proves dissenyat per la validació del desenvolupament del projecte:



Il·lustració 2: Captura amb els diferents nodes que conformen l'escenari

Configuració Xarxa Routers i OpenNMS

Primer cal establir la configuració de xarxa dels dispositius: la màquina virtual del VrtualBox on s'executa OpenNMS i els *routers* virtuals del GNS3 on s'executen els nodes.

Configuració Router R1

Les dades que s'ha establert al *router* segons l'escenari definit són aquestes:

Interfície f0/0

```
Use the 100 Base-TX (RJ-45) connector
```

IP:

```
192.168.10.150
```

Interfície f0/1

```
Use the 100 Base-TX (RJ-45) connector
```

IP:

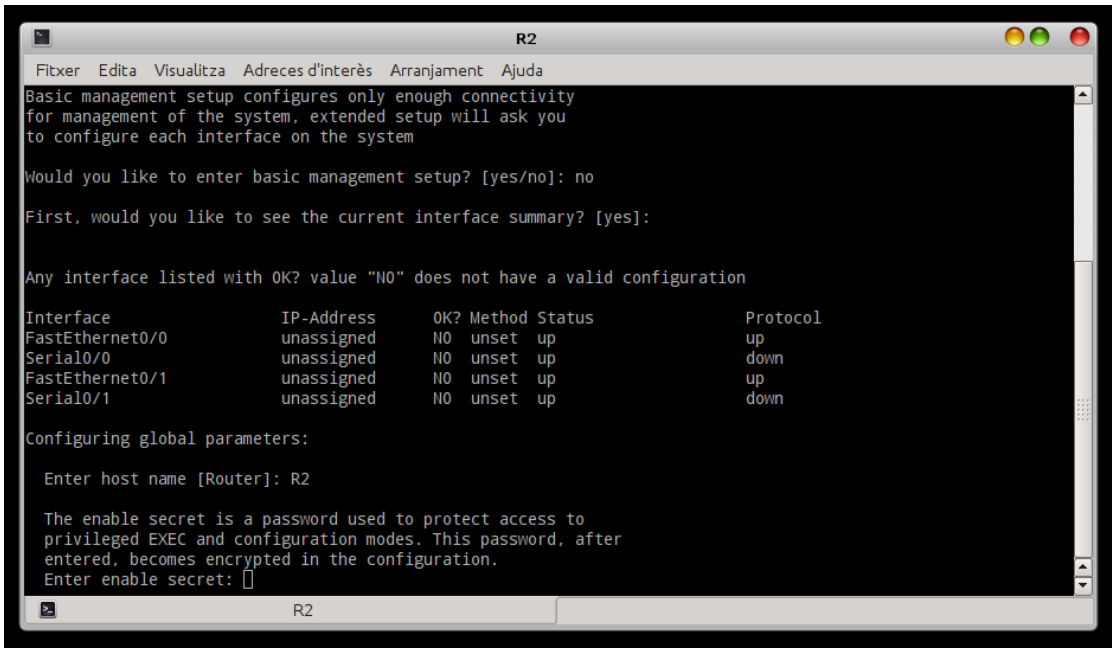
```
192.168.11.150
```

Rutes estàtiques afegides

```
ip route 192.168.12.0 255.255.255.0 192.168.11.200
```

Comunitat SNMP

```
uoc
```



Il·lustració 3: Captura del procés de configuració inicial del router R2

Configuració Router R2

Les dades que s'ha establert al *router* per l'escenari definit per desenvolupar el projecte són aquestes:

Interfície f0/0

```
Use the 100 Base-TX (RJ-45) connector
```

IP:

```
192.168.11.200
```

Serial f0/0

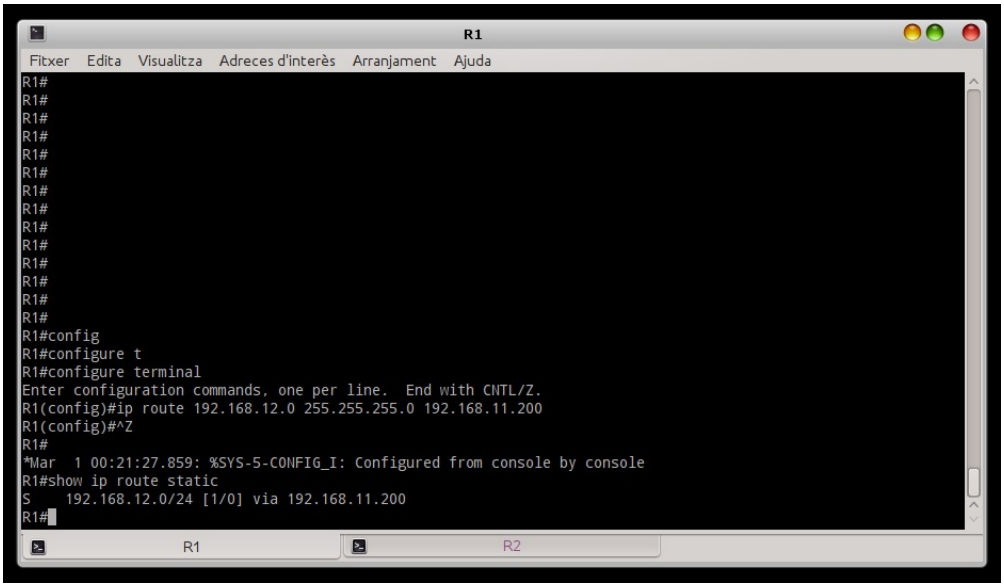
```
Use async serail
```

IP:

```
192.168.12.200
```

Rutes estàtiques afegides:

```
ip route 192.168.10.0 255.255.255.0 192.168.11.150
```



Il·lustració 4: Captura de la configuració de la ruta estàtica al router R2

Comunitat SNMP:

```
uoc
```

Configuració OpenNMS

L'equip OpenNMS és un sistema Linux estàndard i establirem la configuració de xarxa modificant-ne l'arxiu "/etc//network/interfaces". Les dades de xarxa són:

IP:

```
192.168.10.15
```

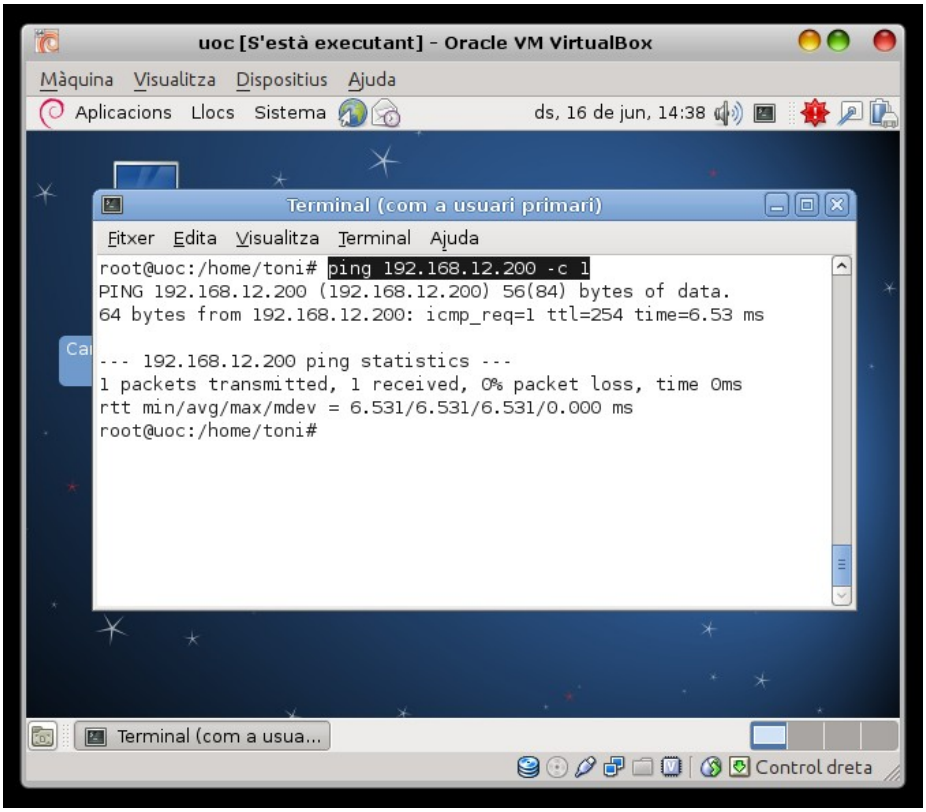
Rutes estàtiques afegides a mà:

```
route add -net 192.168.11.0 netmask 255.255.255.0 gw
192.168.10.150
```

```
route add -net 192.168.12.0 netmask 255.255.255.0 gw
192.168.10.150
```

Validació de la connectivitat

Des de l'equip OpenNMS es pot que les comunicacions amb l'escenari funcionen realitzant *pings* a les diferents interfícies dels routers R1 i R2.



Il·lustració 5: Captura de pantalla del moment on l'equip OpenNMS valida la connectivitat amb el router R2

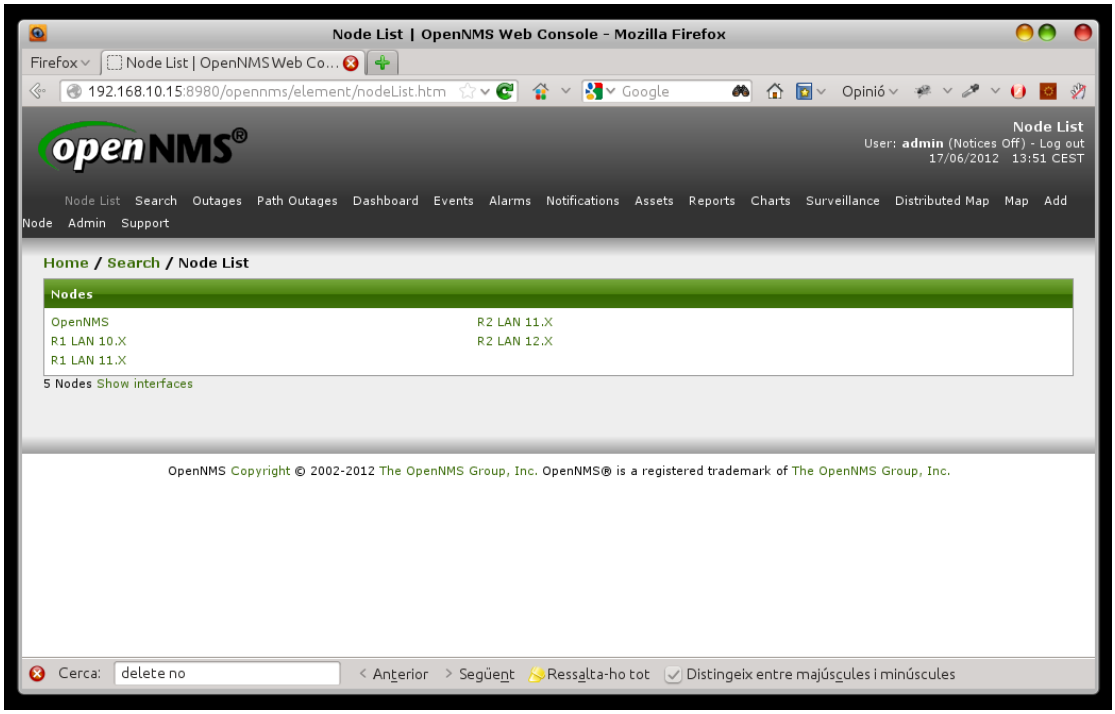
Configuració OpenNMS

Un cop feta la instal·lació cal establir la configuració de la topologia de l'escenari. OpenNMS porta eines per executar aquesta passos també des de la línia de comandes, que són les s'executen a continuació.

Escaneig dels nodes

Per escanejar el propi OpenNMS i les diferents interfícies de xarxa dels *routers* R1 i R2 es poden executar les següents comandes:

```
   # perl /usr/share/opennms/bin/send-event.pl --interface
192.168.10.15 uei.opennms.org/internal/discovery/newSuspect
   # perl /usr/share/opennms/bin/send-event.pl --interface
192.168.10.150 uei.opennms.org/internal/discovery/newSuspect
   # perl /usr/share/opennms/bin/send-event.pl --interface
192.168.11.150 uei.opennms.org/internal/discovery/newSuspect
   # perl /usr/share/opennms/bin/send-event.pl --interface
192.168.11.200 uei.opennms.org/internal/discovery/newSuspect
   # perl /usr/share/opennms/bin/send-event.pl --interface
192.168.12.200 uei.opennms.org/internal/discovery/newSuspect
```



Il·lustració 6: Captura del llistat de nodes a OpenNMS després d'executar la comanda per descobrir-los

Configuració SNMP

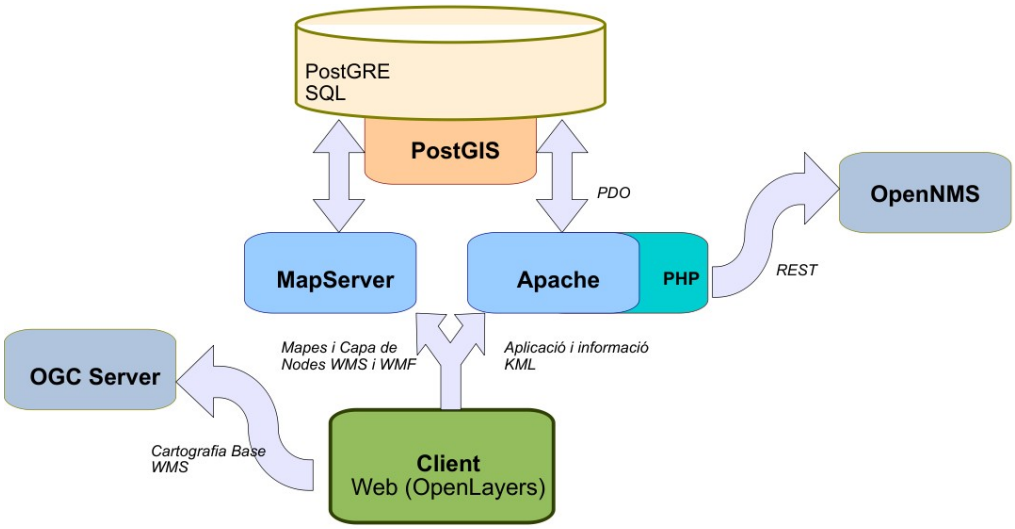
Per afegir a l'OpenNMS a la comunitat SNMP “uoc”, tal i com s'ha fet amb els *routers,* es pot executar la següent comanda:

```
# snmpwalk -v2c -c uoc 192.168.10.15 system
```

Gràcies a l'SNMP l'OpenNMS podrà monitoritzar diferents mètriques de consum de Xarxa dels *routers* i del propi sistema on s'executa.

Arquitectura

A continuació es mostra l'arquitectura dels components que conformen la solució



Il·lustració 7: Diagrama d'arquitectura dels elements de la solució

La solució desenvolupada inclou el desenvolupament dels mòduls d'inventari que al diagrama s'executen a la part "Apache PHP", així com la captació d'informació dels elements a representar des de l'OpenNMS mitjançant l'accés REST a aquest.

El *front-end* geoespacial, és a dir el mapa interactiu en la seva major part s'executa al "Client" amb la llibreria OpenLayersx de Javascript, i obté la informació des del propi MapServer i des de servidors WMS externs per la cartografia de base.

Es pot veure que tant l'aplicació desenvolupada com MapServer se serveixen de PostGIS per poder usar dades geoespacials.

Desenvolupament

Convencions i decisions de disseny programàtic

S'ha decidit usar l'estàndard “lowerCamelCase” per desenvolupar el codi del projecte. Consisteix en els noms de les variables compostes eliminant els espais i posant en majúscula la primera lletra de cada paraula, excepte la primera lletra de totes, que està en minúscula.

Ex.: nomVariableExemple

Malgrat el projecte és una prova de concepte s'ha optat per aplicar el patró MVC (de l'acrònim anglès Model–View–Controller) per separar el model de dades, la interfície usuari i la lògica de control i poder donar una fàcil re utilització del codi desenvolupat i major rellevància a les parts més innovadores desenvolupades durant el projecte.

S'ha decidit afegir el sufix “tfm” al nom de les classes específiques del projecte, però no al de les classes que s'han programat amb la intenció de ser re aprofitades en altres projectes.

S'ha decidit que l'aplicació pogués acabar sent multi idioma i per tant, exceptuant algun possible oblit, no s'usa cap text directament sinó que s'usen cadenes codificades en un arxiu d'idioma.

Malgrat s'ha previst que l'aplicació sigui multi usuari, de moment només s'ha implementat la validació de credencials però no gestió d'usuaris i només existeix un usuari “test”.

La majoria de variables de configuració de l'aplicació s'establiran dins d'una classe “config” amb una variable estàtica[3] anomenada “var”, de forma que des de qualsevol lloc es pugui accedir a config::$var['NOM_DE_LA_VARIABLE'].

També s'ha decidit usar l'anglès com a idioma principal tant pel nom de les variables al codi, com als camps de la bases de dades i, exceptuant algun possible oblit, també als comentaris del codi, malgrat s'ha intentat mantenir al mínim la quantitat de comentaris.

[3] http://en.wikipedia.org/wiki/Static_variable

Estructura del codi font

Malgrat s'ha decidit usar uns patrons MVC pel codi font de la solució, no s'ha fet us de cap dels *framework* existents per implementar-lo. L'estructura del codi, arxius i carpetes reflecteix aquesta tria:

Carpeta arrel "/":

Conté **index.php que és el punt d'entrada central de tota la funcionalitat** desenvolupada. En terminologia MVC és el Controlador frontal.

També conté un "tests.php", que és un arxiu on s'hi haurien d'anar afegint les proves unitàries de les diferents parts del codi.

Carpeta "/setup":

Conté l'arxiu "configuration.php" de configuració de l'aplicació

Carpeta "/classes":

Conté les diferents classes del projecte, el "Model" del patró MVC

Carpeta "/lib":

Conté les diferents classes genèriques del projecte i aquelles que s'han desenvolupat pensant en ser reutilitzables des d'altres projectes.

Carpeta "/lang":

Conté l'arxiu amb les cadenes d'idioma, actualment només el Català, però que es podria ampliar fàcilment.

Carpetes "template":

Conté els arxius amb les pantalles de l'aplicació, vindrien a ser el "View" del patró MVC.

Carpetes "/css i /js":

Contenen els arxius d'estil i funcionalitat javascript respectivament de l'aplicació web.

Accés als WebServices OpenNMS

Com s'ha comentat anteriorment s'ha previst l'accés programàtic a les funcionalitats de l'OpenNMS realitza mitjançant crides REST. A tal efecte s'ha procedit a programar un client REST genèric que ha permès crear, al seu torn, una classe *proxy* de la funcionalitat d'OpenNMS.

El Client Rest desenvolupat

- Ubicació: "/lib/restClient.php"

El Client Rest desenvolupat usa la biblioteca de comunicacions cURL de php i ofereix una funcionalitat de més alt nivell per facilitar l'ús dels *end-points* REST.

La principal interfície d'ús és:

call($requestURI, $method = 'GET',$postData = null)

Descripció dels paràmetres:

$requestURI. És la URL del REST on ens volem connectar

$method. És el verb REST, si no s'especifica realitzarà una crida GET

$postData. És una variable del tipus vector associatiu, on es poden informar a mode "clau" => "valor" totes aquelles variables que es vulguin transmetre amb la crida REST i, malgrat el nom, funciona tant pel verb "POST" com per "PUT" i "DELETE".

Classe Proxy funcionalitat OpenNMS

- Ubicació: "/classes/tfmOpenNMSRestClient.php"

S'ha preparat una classe anomenada "tfmOpenNMSRestClient " que, mitjançant l'ús de la classe "restClient" descrita anteriorment, conté els mètodes per dur a terme crides de la funcionalitat d'OpenNMS. La intenció és mapejar la funcionalitat REST 1 a 1 dins de la classe.

Actualment només hi ha 3 mètodes GET, ja que la durada i l'abast del projecte no ha permès mapejar tota la funcionalitat, però no hauria de resultar gens complicat anar ampliant la classe quan es requereixi alguna del noves funcionalitats.

Les crides a OpenNMS es fan en format JSON i les dades es retornen *parsejades* de manera que poden ser utilitzades des del codi PHP comuna classe estàndard genèrica (això en PHP és una stdClass)

Integració NMS

La integració amb OpenNMS consisteix a desar l'identificador i l'etiqueta que OpenNMS dóna als elements que es vulguin referenciar. Malgrat això suposarà certa càrrega cada cop es vulgui consultar la informació sobre l'estat dels elements, redueix enormement la complexitat de la integració i facilita l'estructura de la informació a mantenir. L'obtenció d'informació es durà a terme mitjançant crides contra la interfície REST amb l'identificador que OpenNMS hagi assignat als elements.

Bases de Dades

Ús de PDO

Pel desenvolupament de l'accés a dades s'ha optat per usar els objectes d'accés a dades PDO ja que ofereixen una interfície lleugera i consistent d'accés a bases de dades en PHPxi. Una altre avantatge és que si en algun moment calgués canviar de sistema de gestor de base de dades, les implicacions programàtiques són molt menors. Malgrat això, PDO simplement exposa les funcions de les bases de dades i per tant, si les sentències SQL de l'aplicació utilitzen funcions específiques d'un sistema, aquestes no funcionarien si es canviés de sistema gestor de base de dades.

Creació de Bases de Dades amb les característiques PostGIS

Per crear una base de dades nova amb la integració PostGIS funcional es pot usar com a plantilla alguna base de dades que ja tingui PostGIS habilitat. La comanda per usar una BD existent com a plantilla seria la següent:

```
   # createdb -T [nom_Base_Dades_Plantilla]
[nom_nova_BD]
```

Nota PostrgeSQL:

En la seva instal·lació per defecte, per executar les utilitats de PostgreSQL des de la línia de comandes, caldrà fer-ho amb l'usuari postgres. Gràcies a la utilitat “su” i mitjançant la comanda “su postgres” podrem fer-ho sense conèixer-ne la contrasenya.

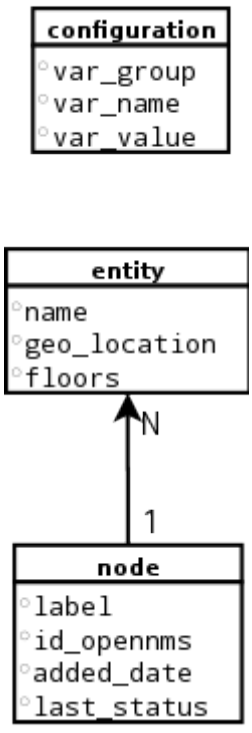
El paràmetre -T és el que permet especificar quina Base de Dades ha de ser la plantilla.

Si no s'especifica aquesta plantilla, la nova Base de Dades no tindrà les extensions de PostGIS habilitades (ni de tipus de dades ni les extensions al llenguatge plpgsql)

Alternativament, si un cop tenim la base de dades creada sense suport de PostGIS volem habilitar-lo, caldrà executar els *scripts* "spatial_ref_sys.sql" i *"postgis.sql"* contra la Base de Dades. En els sistemes Ubuntu/Debian, i per la versió 8.4 de PostgreSQL, la comanda seria la següent:

```
   # psql -d [nom_nova_BD] -f
/usr/share/postgresql/8.4/contrib/spatial_ref_sys.sql
   # psql -d [nom_nova_BD] -f
/usr/share/postgresql/8.4/contrib/postgis.sql
```

Diagrama ER



Il·lustració 8: Diagrama Entitat Relació

Taula **configuration**: Aquesta taula està pensada per poder ordenar tot tipus d'informació agrupada per àrees funcionals. A més a més, s'ha desenvolupat una classe per poder-hi accedir i establir les variables de manera ràpida, cosa que permet treure-li el màxim rendiment amb el mínim cost de desenvolupament.

Taula **entity**: Aquesta taula està pensada com a agrupador de nodes. De manera fàcil podem pensar en cada una de les entrades a aquesta taula com si es tractés d'un edifici, i d'aquí l'atribut “floors” en referència al nombres de plantes. Una “entity” pot tenir N equips de xarxes, i el camp “geo_location” en guarda la ubicació al mapa.

La taula **nodes**: Cada registre d'aquesta taula representa un equip de xarxa que es monitoritza a OpenNMS. Té una clau forana ja que la seva ubicació serà la de l'”entity” (l'edifici) on es trobi.

A l'annex II d'aquest document s'hi poden trobar els *scripts* de creació usats per crear les taules descrites.

Codi d'integració Geoespacial

Malgrat la intenció del projecte és mostrar els elements de la xarxa servits mitjançant capes WMS de MapServer, el poc temps disponible per al desenvolupament del projecte fa que s'hagi buscat una solució més ràpida a la representació de les dades i s'hagi optat per mostrar la informació mitjançant una capa KML a OpenLayers.

Per poder generar documents KMLxii de manera eficient s'ha generat una funció en PL/pgSQL directament al PostgrSQL que retorna la informació en format KML. L'avantatge d'executar-ho directament al sistema Gestor de Base de dades és que el rendiment és superior i que és fàcilment re utilitzable. El codi de la funció és el següent:

```
   CREATE OR REPLACE FUNCTION kmlNodes(name text, description text)
RETURNS text AS $$
    DECLARE
      result text;
      curs1 CURSOR FOR SELECT e.name, ST_X(the_geom) as longitude,
ST_Y(the_geom) as latitude ,ST_AsKML(the_geom) as kml,
```

```
array_to_string(array_agg(n.label), ', ') as nodes FROM entity e left
join node n on n.id_entity = e.id GROUP BY the_geom,e.name;
    BEGIN
      result := '<?xml version="1.0" encoding="UTF-8"?>' || E'\n' ||
                        '<kml xmlns="http://www.opengis.net/kml/2.2">'
|| E'\n' ||
                        '<Document>' || E'\n' ||
                        '<name>' || name || '</name>' || E'\n' ||
                        '<description>' || description ||
'</description>' || E'\n\n' ||
                        '<Style id="defaultStyle">' || E'\n' ||
                        '  <LineStyle>' || E'\n' ||
                        '    <color>ff00ff00</color>' || E'\n' ||
                        '    <width>1</width>' || E'\n' ||
                        '  </LineStyle>' || E'\n' ||
                        '  <PolyStyle>' || E'\n' ||
                        '    <color>5f00ff00</color>' || E'\n' ||
                        '  </PolyStyle>' || E'\n' ||
                        '  <IconStyle>' || E'\n' ||
          '        <Icon>' || E'\n' ||
          '          <href>images/kml.png</href>' || E'\n' ||
          '    </Icon>' || E'\n' ||
          '  </IconStyle>' || E'\n' ||
                        '</Style>' || E'\n\n';
      FOR recordvar IN curs1 LOOP
                    result := result || '<Placemark>' || E'\n';
                    result := result ||
'<styleUrl>#defaultStyle</styleUrl>' || E'\n';
                    result := result || '<name>' || recordvar.name ||
'</name>' || E'\n';
                    result := result || '<description>Ubicació
(Lon/Lat): ' || recordvar.longitude || '/' || recordvar.latitude  ||
E'\n. Nodes que conté: ' || recordvar.nodes || '</description>' ||
E'\n';
      result := result || recordvar.kml || E'\n';
      result := result || '</Placemark>' || E'\n\n';
           END LOOP;
           result :=  result ||'</Document>' || E'\n' ||
                        '</kml>';
    RETURN result;
    END;
```

```
$$ LANGUAGE plpgsql;
```

Se'n pot destacar l'ús de les funcions de PostGIS "ST_X", "ST_Y" i "ST_AsKML":

ST_X: Se li passa una geometria espacial de PostGIS i retorna longitud en la projecció en que es trobin les dades.[4]

ST_Y: Se li passa una geometria espacial de PostGIS i retorna latitud en la projecció en que es trobin les dades.

ST_AsKML: Se li passa una geometria espacial de PostGIS i retorna el text en format KML.

Procés d'importació de dades

La manera d'importar dades al sistema serà mitjançant dues pantalles:

1. Un manteniment CRUD[5] d'entities (o sigui d'edificis) que també permetrà referenciar-los damunt del mapa amb un sol *click*.

2. Un llistat dels elements de l'OpenNMS que permetrà integrar-los a l'aplicació, que a efectes pràctics, tal i com ja s'ha comentat, voldrà dir desar-ne l'identificador d'OpenNMS.

1. Manteniment CRUD d'entities

Per poder introduir el punt exacte on es troba un edifici damunt del mapa, s'ha desenvolupat un sistema d'obtenció de les coordenades mitjançant la llibreria OpenLayers. A continuació es mostra un extracte del codi per la captura de les coordenades mitjançant un *click* damunt del mapa:

```
OpenLayers.Control.Click = OpenLayers.Class(OpenLayers.Control, {
                defaultHandlerOptions: {
                    'single': true,
```

[4] Per la projecció, veure el punt de georefereciació d'aquest mateix document més endavant.

[5] CRUD ve de l'acrònim anglès "CReate Update Delete", és a dir "Alta Baixa i Modificació " de registres.

```
                'double': false,
                'pixelTolerance': 0,
                'stopSingle': false,
                'stopDouble': false
            },

            initialize: function(options) {
                this.handlerOptions = OpenLayers.Util.extend(
                    {}, this.defaultHandlerOptions
                );
                OpenLayers.Control.prototype.initialize.apply(
                    this, arguments
                );
                this.handler = new OpenLayers.Handler.Click(
                    this, {
                        'click': this.trigger
                    }, this.handlerOptions
                );
            },

            trigger: function(e) {
                var lonlat = map.getLonLatFromPixel(e.xy);
                $("#lat").val(lonlat.lat);
                $("#lon").val(lonlat.lon);
                $("#save").attr("disabled",null);
            }

        });
```

Es pot veure com mitjançant l'objecte “OpenLayers.Handler.Click” es determina que el *click* serà tractat dins de la funció “Trigger”, i com allà se n'obtenen les coordenades (la longitud i la latitud) amb el mètode “getLonLatFromPixel” de l'objecte “map” d'OpenLayers.

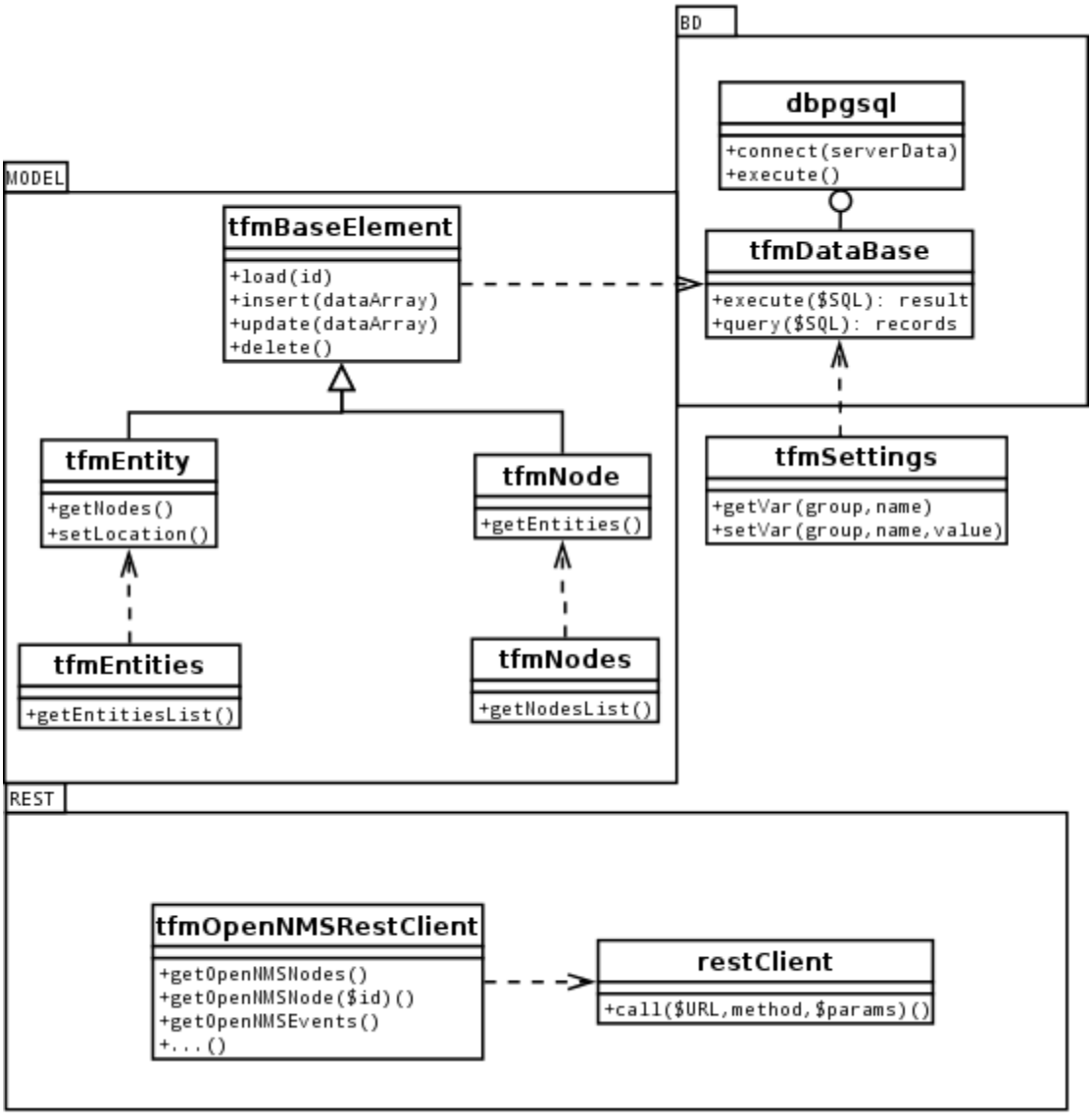
2. Elements d'OpenNMS

Pel llistat dels elements de l'OpenNMS, es programa un mètode GET a la classe d'accés REST a l'OpenNMS:

```
function getNodeList() {
```

```
        $result = $this->rest->callJSON(\config::
$var['restURL'].'nodes/');
        $status = $this->rest->status;
        if ($status == 200) {
              return $result;
        }
        return false;
}
```

Diagrama de Classes



Il·lustració 9: Diagrama de Classes UML

Es pot veure que s'ha dissenyat una classe Base "tfmElementBase" de la qual hereten les classes de representació de les entitats (els edificis) i els nodes (els equips). D'aquesta manera tota la funcionalitat CRUD d'aquestes classes s'ha generalitzat i no ha calgut implementar-la per partida doble. Això també deixa oberta la porta a futures ampliacions de funcionalitat de manera fàcil.

Així mateix, es pot veure com s'han dissenyat les classes per accedir a la Base de Dades independents de la funcionalitat de l'aplicació i les classes per fer crides REST i solucionar la integració amb OpenNMS.

Desenvolupament part Geoespacial

S'ha establert que pel mapa base s'usarà la cartografia d'OpenStreetMaps i la de l'Institut Cartogràfic de Catalunya (ICC) mitjançant l'ús de WMS. Es recuperaran únicament les porcions dels mapes que interessen.

Georeferenciació

La informació geoespacial que desarem es reduirà a un conjunt de coordenades per cada edifici.

Es desarà amb el tipus de dades "geometry" que és la implementació de PostGIS per l'especificació de dades geoepacials estàndard OGC SPEC s2.1.1.

Bàsicament és un tipus de dades on la seva representació com a cadena de caràcters vindria a ser: 'LINESTRING', 'POLYGON', 'MULTIPOINT' i 'POINT', seguit dels valors de les coordenades específiques en cada cas.

La projecció o *Datum* que s'utilitzarà és l'EPSG 23021, ja que és la que millor s'ajusta a Catalunya.[6]

Representació en pantalla

Com ja s'ha avançat, la representació en pantalla es farà mitjançant la llibreria OpenLayers. La projecció de les dades serà EPSG 23031 pels mapes de l'ICC i la WGS84 pels mapes de l'OpenStreets Map. A continuació hi ha una mostra del codi per carregar dues capes amb el topogràfic i el mapa de l'ortofotomapa per Barcelona des de l'ICC:

```
var topo_tilecache = new OpenLayers.Layer.WMS("Topo ICC",
   ["http://sagitari.icc.cat/tilecache/tilecache.py?" ,
   "http://84.88.72.13/tilecache/tilecache.py?"] ,
   {layers: 'topo', format:"image/jpeg",
   exceptions:"application/vnd.ogc.se_xml"},
   {buffer:4, transitionEffect:'resize'} );
```

[6] A http://spatialreference.org/ref/epsg/23031/ se'n poden consultar els detalls.

```
var orto_tilecache = new OpenLayers.Layer.WMS("Orto ICC",
  ["http://sagitari.icc.cat/tilecache/tilecache.py?" ,
  "http://84.88.72.13/tilecache/tilecache.py?"],
  {layers: 'orto', format:"image/jpeg",
  exceptions:"application/vnd.ogc.se_xml"},
  {buffer:4, transitionEffect:'resize'});

map = new OpenLayers.Map('map', mapOptions);
map.addLayers(layers);
```

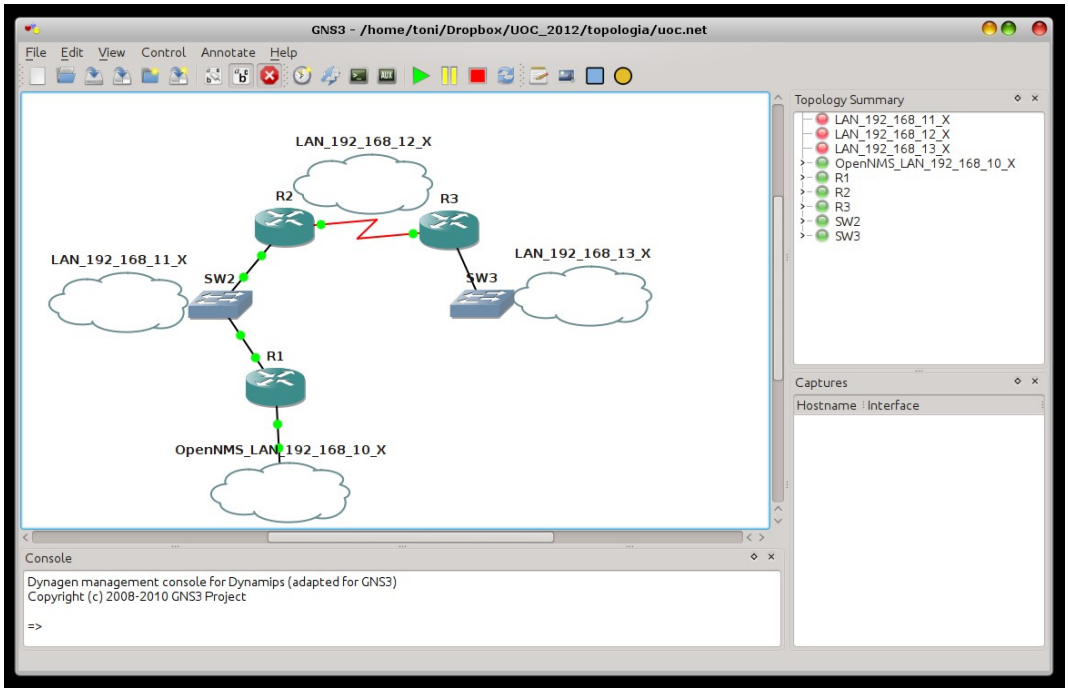
Apart de l'adreça dels servidors de mapa i les opcions obligatòries com un nom identificatiu, s'ha especificat l'opció “buffer” a 4, que fa que OpenLayers recuperi fins a 4 rajoles (o “tiles”) addicionals per cada cantó del mapa a mode de memòria cau per si l'usuari decideix desplaçar el centre del mapa.

Resultat final

A continuació és fa un recorregut per la solució implementada. Aquest recorregut comença pels elements arquitectònics de la solució i acaba per l'aplicació desenvolupada com a prova de concepte.

1. GNS3. L'escenari amb els Routers en Marxa.

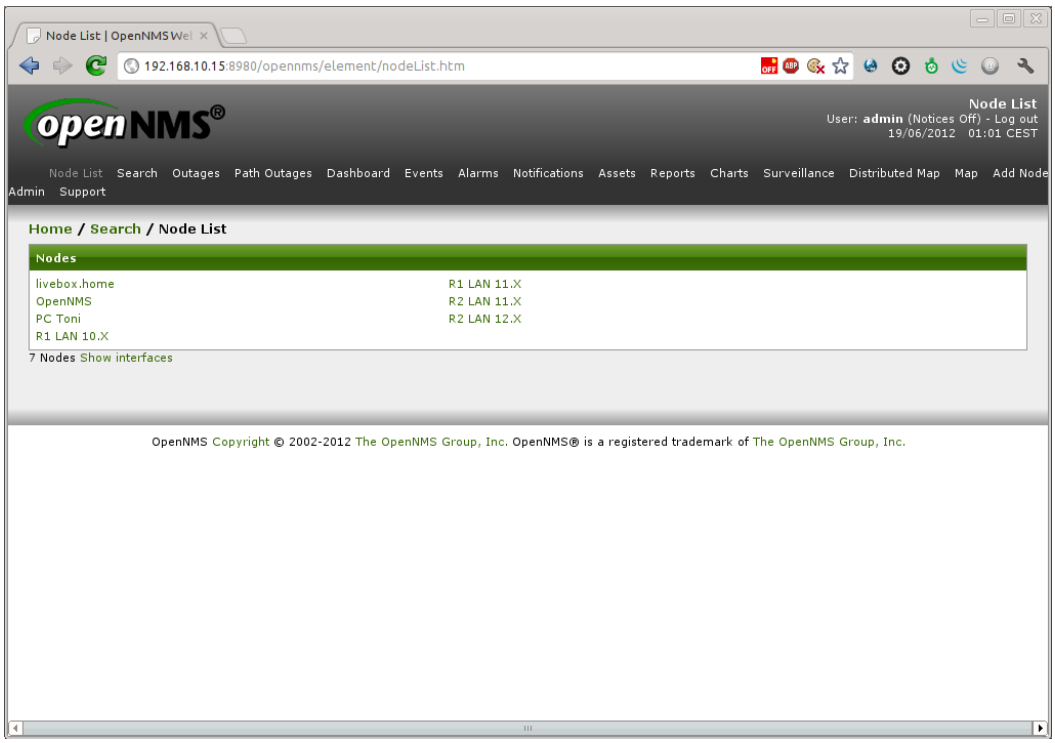
El verd dels enllaços entre *routers* i *switches* ens diu que la connexió està aixecada i funcionant.



Il·lustració 10: Resultat final 1. Xarxa simulada a GNS3

2. OpenNMS. Llista de nodes detectats.

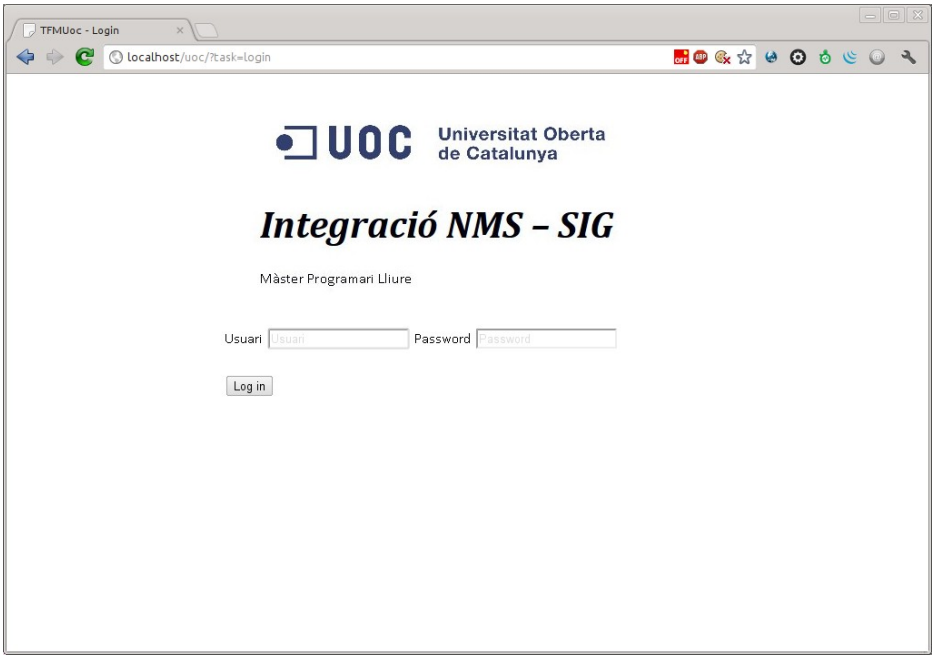
Apart dels nodes definits a l'escenari, OpenNMS ha detectat 2 equips més, un *router* d'Internet anomenat “livebox.home” i un PC anomenat “PC Toni”



Il·lustració 11: Resultat final 2. Nodes a OpenNMS

3. La pantalla d'inici de sessió de L'aplicació i missatge mostrat el primer cop s'hi accedeix .

Aspecte de la pàgina d'inici de sessió. De moment, l'únic usuari vàlid és test/test.

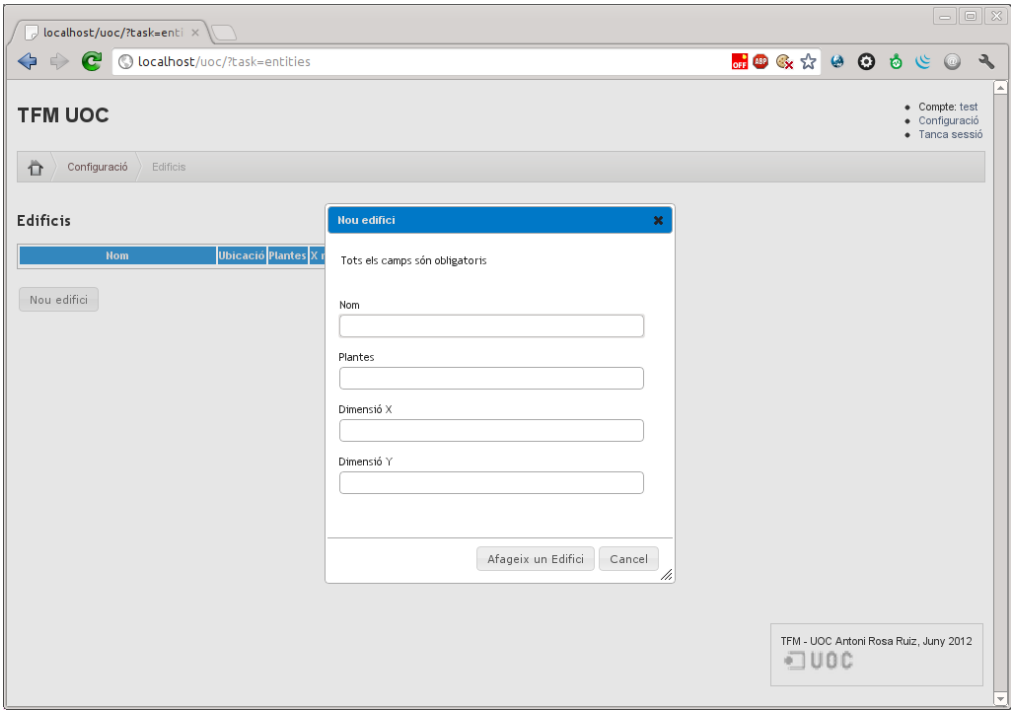


Il·lustració 12: Resultat final 3.1. Pantalla d'inici de sessió

Il·lustració 13: Resultat final 3.2. Missatge de benvinguda el 1er login

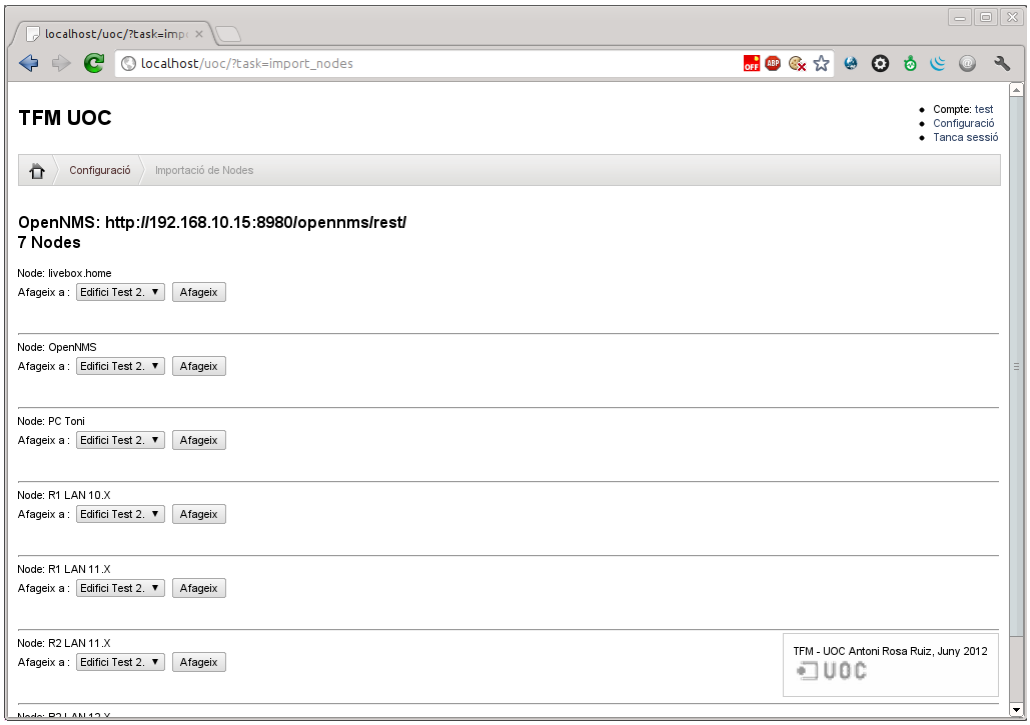
4. Pantalles de configuració d'Edificis i integració amb OpenNMS

Afegint una entity (un edifici) a l'aplicació.



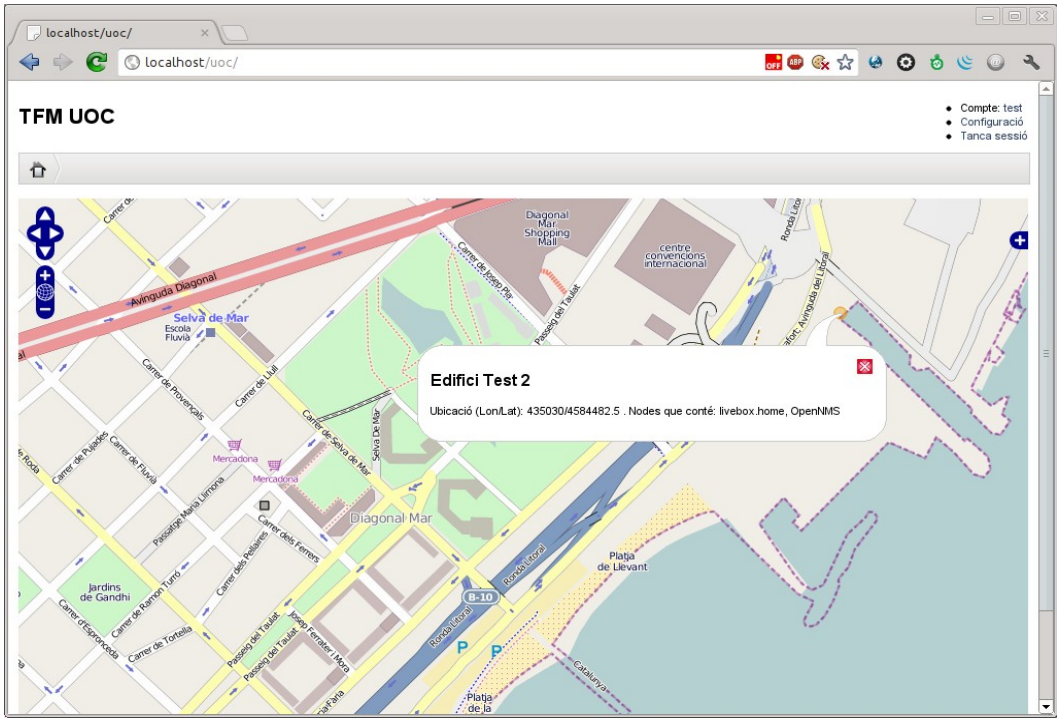
Il·lustració 14: Resultat fial 4.1. Alta d'un edifici

Lligam de les dades d'OpenNMS amb els edificis.



Il·lustració 15: Resultat final 4.2. Associar element OpenNMS a edificis

5. Visualització de la informació Georefenciada



Il·lustració 16: Resultat final 5. La informació georeferenciada damunt del mapa

Conclusions

Aquest capítol presenta les conclusions del projecte d'integració entre l'NMS escollit i PostGIS.

L'objectiu inicial de trobar una forma d'integrar informació provinent d'un NMS amb informació geospacial s'ha acomplert i s'ha demostrat factible amb el desenvolupament d'una aplicació de mostra.

S'ha comprovat que gràcies a la API Rest d'OpenNMS i una aproximació amb el plantejament tant senzill de guardar poc més que la referència dels elements NMS i un camp d'informació geoespacial, no presenta gaires més complicacions que la pròpia d'una arquitectura d'aplicacions i serveis heterogènia.

L'escenari de proves plantejat, compost d'un simulador de xarxes, GSN3, amb imatges de routers Cisco executant-s'hi, i una màquina virtual amb un sistema Linux i la solució OpenNMS és, per si mateix, un sub projecte molt interessant que mostra la possibilitat de simular xarxes i entorns reals i complexos mitjançant l'emulació del maquinari. L'avantatge de l'emulació per davant d'altres possibilitats, com la simulació, resideix en el fet que a l'executar-se imatges reals dels sistemes, la fiabilitat dels resultats obtinguts en aquest tipus d'escenaris simulats és molt alta.

L'aplicació obtinguda és una bona prova de concepte on es veu que el plantejament inicial del projecte anava ben encaminat i compleix amb la seva funció: mostrar informació d'OpenNMS damunt d'un mapa. Malgrat l'abast i limitació temporal del projecte no ha permès dotar aquesta aplicació de gaire funcionalitat més enllà dels punts descrits anteriorment, s'ha focalitzat en obtenir una solució genèrica i ampliable. Les funcions i llibreries desenvolupades posen una bona base per a posteriors productes derivats d'aquesta aplicació o ampliacions de la mateixa.

Glossari

REST (Representational State Transfer): Arquitectura de programari per a sistemes distribuïts basats en hipermèdia, com ara el web. Aquest terme va ser introduït l'any 2000 a la tesi doctoral de Roy Thomas Fielding, un dels autors principals de les especificacions del protocol HTTP.

Tot i que en un principi REST es referia tan sols a un conjunt de principis d'arquitectura de xarxa i la definició i adreçament dels recursos, actualment aquest concepte s'utilitza per referir-se a una interfície web que utilitza XML o JSON i HTTP sense cap conjunt de capçaleres com podria ser en el cas de SOAP o XML-RPC.

Ping: Eina de les xarxes TCP/IP, que permet, determinar si un equip està funcionant i si s'hi pot arribar des de la xarxa concreta amb l'ordinador des d'on es fa la prova. El programa envia paquets ICMP i n'espera les respostes.

NMS (Network Monitoring System): Aquest terme descriu l'ús d'un sistema que monitoritza constantment una xarxa Informàtica en cerca de components lents o components que fallen i que notifica a l'administrador de xarxa (via mail, SMS o altres tipus d'alarmes) si hi ha hagut talls. És un subconjunt de les funcions implicades en la gestió de xarxes.

KML (Keyhole Markup Language): Llenguatge de marcatge basat en XML per a representar dades geogràfiques en tres dimensions. Va ser desenvolupat per ser gestionat amb Keyhole LT, precursor de Google Earth (Google va adquirir l'empresa al 2004). Actualment KML és un estàndard internacional de l'Open Geospatial Consortium.

OpenNMS: Plataforma de gestió de xarxa desenvolupat per l'empresa del mateix nom sota el model de programari lliure. L'OpenNMS Group, que ofereix serveis comercials, formació i suport. Els objetius d'OpenNMS són que la plataforma de gestió pugui funcionar de forma distribuïda i escalable per a tots els aspectes del model de gestió de la xarxa FCAPS, i malgrat aquestes característiques de màxim nivell empresarial, que no deixi de ser lliure i de codi obert. L'eina és força madura i actualment, l'atenció la centren en millorar la gestió d'errors i rendiment de la solució.

Router: El *router* (en català, encaminador o enrutador) és un dispositiu de xarxa de nivell 3 del model OSI. Pren la informació del nivell de xarxa (adreça IP) per a prendre les decisions d'encaminament: escollir el camí o ruta més adequada per on reenviar les dades rebudes.

La successió de decisions de diferents *routers* possibilita l'arribada de la informació des del seu origen fins al sistema destí.

Tot enrutador disposa d'una taula d'encaminament que informa dels camins que han de prendre els paquets en funció de certs paràmetres (origen, destí, congestió, seguretat...).

Aquestes taules poden ser estàtiques (introduïdes manualment) o dinàmiques (RIP, BGP...).

Mashup (remescla en català): Aplicació web híbrida que utilitza contingut d'altres aplicacions web per a crear un nou contingut complet, consumint serveis directament.

El contingut d'una remescla normalment prové de llocs web de tercers consumits a través d'una interfície pública o usant una API. Alguns mètodes que constitueixen l'origen de dades inclouen: sindicació web (RSS o Atom), *screen scraping*, etc.

Les remescles estan revolucionant el desenvolupament web ja que permeten que qualsevol combini, de forma innovadora, dades que existeixen en diferents pàgines web.

GeoRefereciació: Posicionament amb el qual es defineix la localització d'un objecte espacial (representat mitjançant punt, vector, àrea, volum) en un sistema de coordenades i *datum*

(projecció espacial) determinat. Aquest procés és utilitzat freqüentment en els Sistemes d'Informació Geogràfica (SIGs).

La georeferenciació, en primer lloc, té una definició tecnocientífica, aplicada a l'existència de les coses en un espai físic, mitjançant l'establiment de relacions entre les imatges vectorials o de ràster sobre una projecció geogràfica o sistema de coordenades. No obstant això, l'acte de georeferenciar ha anat més enllà de les especialitats de Ciències de la Terra i dels SIGs.

L'ús massiu d'eines com Google Earth ha implicat un salt qualitatiu pel que fa a georeferenciació. Ja no es tracta només de geo-dades limitades als especialistes de les geo-ciències i Sistemes d'Informació Geogràfica. Ara la georeferenciació té un impacte sociològic ja que es realitza sobre tots els continguts socials presents en el món.

Cisco IOS (Internetwork Operating System original): És el sistema operatiu multitasca utilitzat en la majoria dels routers (enrutadors) i switches (commutadors) de Cisco Systems (alguns commutadors antics utilitzaven CatOS). IOS és, doncs, una solució per gestionar l'enrutament (routing), la commutació (switching), i la interconnexió de xarxes i telecomunicacions integrades.

La interfície de línia de comandes IOS (IOS CLI) proporciona un conjunt fix de comandes de diverses comandes. El conjunt de comandes disponible es determina mitjançant el "mode" i el nivell de privilegis de l'usuari actual. El mode de “configuració global" ofereix ordres per a canviar la configuració del sistema, i el mode “configuració de la interfície" ofereix ordres per a canviar la configuració d'una interfície concreta. Típicament a totes les comandes se'ls assigna un nivell de privilegi, de 0 a 15, i només poden accedir-hi els usuaris amb els privilegis necessaris.

PL/pgSQL (Procedural Language/PostgreSQL Structured Query Language): Llenguatge imperatiu que proveeix el gestor de bases de dades PostgreSQL. Permet executar comandes SQL mitjançant un llenguatge de sentències imperatives i ús de funcions, atorgant molt més control automàtic que les sentències SQL bàsiques.

Des de PL/pgSQL es poden realitzar càlculs complexes i crear nous tipus de dades d'usuari. De manera similar a un llenguatge de programació, disposa d'estructures de control repetitives i condicionals, a més de la possibilitat de creació de funcions que poden ser cridades en sentències SQL normals o executades mitjançant *triggers*.

Una dels principals avantatges d'executar programació en el servidor de bases de dades és que les consultes i el resultat no han de ser transportades entre el client i el servidor, ja que les dades resideixen en el propi servidor. A més, el gestor de base de dades pot planificar optimitzacions en l'execució de la recerca i actualització de dades.

Les funcions escrites en PL / pgSQL accepten arguments i poden prendre valors de tipus bàsic o de tipus complex (per exemple, registres, vectors, conjunts o fins i tot taules), permetent tipificació polimòrfica per a funcions abstractes o genèriques (referència a variables de tipus objecte).

Annexos

Annex I. Script de creació de les taules

```
CREATE TABLE configuration
(
  id serial NOT NULL,
  var_group character varying(50),
  var_name character varying(128),
  var_value character varying(512),
  created_on timestamp with time zone DEFAULT now(),
  created_by character varying(50),
  modified_on date,
  modified_by character varying(50),
  var_desc text,
  CONSTRAINT pk_configuration PRIMARY KEY (id ),
  CONSTRAINT "unique group_name" UNIQUE (var_group , var_name )
)
WITH (
  OIDS=FALSE
);


CREATE TABLE entity
(
  id serial NOT NULL,
  the_geom geometry,
  floors integer,
  name character varying(255),
  max_x integer,
  max_y integer,
  CONSTRAINT pk_entity PRIMARY KEY (id )
)
WITH (
  OIDS=FALSE
);
```

```
CREATE TABLE node
(
  id serial NOT NULL,
  label character varying(255),
  id_opennms integer,
  id_entity integer,
  x integer,
  y integer,
  floor integer,
  CONSTRAINT pk_node PRIMARY KEY (id )
)
WITH (
  OIDS=FALSE
);
```

Referències

[i] **Informació OpenNMS Group.** http://www.opennms.com/about-opennms-group/ (última visita Juny 2012)

[ii] **Característiques d'OpenNMS.** http://www.opennms.org/wiki/Features_List (última visita Juny 2012)

[iii] **Informació llicències Nagios.** http://www.nagios.com/legal/licenses (última visita Maig 2012)

[iv] **Informació tècnica d'integració.** http://nagiosplug.sourceforge.net/developer-guidelines.html#DEVREQUIREMENTS (última visita Maig 2012)

[v] **Comparació entre NMS.** http://en.wikipedia.org/wiki/Comparison_of_network_monitoring_systems (última visita Abril 2012)

[vi] **Especificació característiques OGC implementades a PostGIS.** http://www.opengeospatial.org/resource/products/details/?pid=509 (última visita Maig 2012)

[vii] **Resum de com usar dbLink a PostGreSQL.** http://www.postgresonline.com/journal/archives/44-Using-DbLink-to-access-other-PostgreSQL-Databases-and-Servers.html (última visita abril 2012)

[viii] **Guia d'instal·lació d'OpenNMS.** http://www.opennms.org/wiki/Installation:Debian (última visita Juny 2012)

[ix] **Preguntes més Frequents d'instal·lació d'OpenNMS.** http://www.opennms.org/wiki/FAQ-Configuration (última visita Juny2012)

[x] **Manual de referència d'OpenLayers.** http://dev.openlayers.org/docs/index/General.html (última visita Juny 2012)

[xi] **Documentació classe PDO en PHP.** http://php.net/manual/en/intro.pdo.php (última visita Juny 2012)

[xii] **Especificació Completa del format KML.** http://schemas.opengis.net/kml/ (última visita Juny 2012)