MÀSTER UNIVERSITARI EN PROGRAMARI LLIURE
TFM – *Desenvolupament d'aplicacions*

Disseny i desenvolupament d'una eina de monitorització remota per a Sistemes VoIP basats en FreePBX

Raül Martínez Díaz

Màster Universitari en Programari Lliure
rmartinezdi@uoc.edu

Equip Docent UOC:

Santi Caballé Llobet
Professor

Gregorio Robles Martínez
Professor col·laborador

Equip Col·laborador Infordisa S.A:

Robert Ferrando Bergadà
Gerent d' Infordisa SA

José Suárez Laporta
Tècnic VoIP / Elastix Certified Engineer

Abstract

El present document constitueix el resultat d'un treball de recerca, anàlisis i desenvolupament en el camp de la monitorització de sistemes remots de programari lliure. Per a la seva elaboració va ser necessari fer un estudi les problemes específics que comporta la monitorització de sistemes remots. Fruït d'aquest estudi es va dissenyar i implementar una aplicació client-servidor que permet la gestió i administració de les monitoritzacións en sistemes remots, el present document recull el detall del disseny elaborat i la seva implementació.

Índex de continguts

1. Introducció

1.1 Justificació i antecedents del projecte pag. 6
1.2 Context del projecte pag. 7
1.3 Descripció del projecte i objectius pag. 8
1.3.1 Component Servidor pag. 8
1.3.2 Component Client pag. 9
1.4 Calendari i planificació del projecte pag. 10
1.5 Producte obtingut pag. 12
1.6 Estructura del present document pag. 13

2. Àmbit tecnològic del projecte

2.1 Desenvolupament amb eines de programari lliure pag. 14
2.2 Eines de desenvolupament de projectes de programari lliure pag. 15
2.2.1 Eines de gestió de projectes pag. 15
2.2.2 Eines control de versions pag. 16
2.2.3 Editors de text pag. 17
2.2.4 Eines de compilació i depuració pag. 18
2.2.5 Generadors de documentació pag. 19

3. Requisits del sistema

3.1 Recollida de requisits inicials pag. 20
3.2 Observació i investigació contextual pag. 21
3.3 Entrevistes pag. 22
3.3.1 Anàlisi de les entrevistes pag. 23
3.4 Anàlisi de tasques pag. 24

4. Anàlisi del sistema

4.1 Anàlisis del sistema: Component Servidor pag. 25
4.1.1 Casos d'ús pag. 26
4.1.2 Anàlisi detallat dels casos d'ús del component servidor pag. 27
4.1.2.1 CU-A01 - Validar usuari pag. 28
4.1.2.2 CU-A02 – Afegir NODE pag. 29
4.1.2.3 CU-A03 – Editar NODE pag. 30
4.1.2.4 CU-A04 – Llistar NODE pag. 30
4.1.2.5 CU-A05 – Afegir SENSOR de servei pag. 31
4.1.2.6 CU-A06 – Editar SENSOR de servei pag. 32

4.1.2.7 CU-A07 – Llistar SENSOR de servei pag. 33
4.1.2.8 CU-A08 – Assignar SENSOR de servei pag. 34
4.1.2.9 CU-A09 – Afegir REGISTRE pag. 35
4.1.2.10 CU-A10 – Llistar REGISTRE pag. 36
4.1.2.11 CU-A11 – Autenticar NODE pag. 36
4.1.2.12 CU-A12 – Anàlisi de registres pag. 37

4.2 Anàlisis del sistema: Component Client pag. 38
4.2.1 Casos d'ús pag. 38
4.2.2 2 Anàlisi detallat dels casos d'ús del component client pag. 38
4.2.2.1 CU-B01 - Afegir NODE pag. 39
4.2.2.2 CU-B02 – Autenticar NODE pag. 40
4.2.2.3 CU-B03 – Llistar SENSORS de servei pag. 40
4.2.2.4 CU-B04 – Actualitzar SENSORS de servei pag. 41
4.2.2.5 CU-B05 – Llistar TASQUES de monitorització pag. 42
4.2.2.6 CU-B06 – Actualitzar llistat de TASQUES de monitorització pag. 43
4.2.2.7 CU-B07 – Sincronitzar NODE pag. 44
4.2.2.8 CU-B08 – Executar tasques pag. 45
4.2.2.9 CU-B09 – Afegir registre pag. 45
4.2.2.10 CU-B10 – Analitzar registre pag. 46
4.2.2.11 CU-B11 – Llistar registres pag. 46
4.2.2.12 CU-B12 – Enviar registres pag. 47

5. Disseny del sistema

5.1 Disseny arquitectònic del sistema pag. 48
5.1.1 Disseny arquitectònic del component servidor pag. 49
5.1.2 Disseny arquitectònic del component client pag. 50
5.2 Disseny del model de dades pag. 52
5.2.1 Disseny del model de dades del component servidor pag. 52
5.2.2 Disseny del model de dades del component client pag. 54

6. Implementació

6.1 Component Servidor pag. 55
6.1.1 Relació de components i llibreries de l'aplicació servidor pag. 55
6.1.2 Arquitectura general del component servidor pag. 56
6.1.3 API REST pag. 57
6.1.4 Autenticació d'usuaris i nodes client pag. 59
6.1.5 Consideracions addicionals de seguretat pag. 63
6.1.6 Consola d'Administració Web pag. 64
6.1.7 Captures de l'aplicació pag. 66

6.2 Component client pag. 67
6.2.1 Relació de components i llibreries de l'aplicació client pag. 68
6.2.2 Arquitectura general del component client pag. 69
6.2.3 Model de dades pag. 74
6.2.4 Comunicació amb el servidor pag. 75
6.2.4.1 Llibreria de comunicacions libcurl pag. 77
6.2.4.2 Serialització i deserialització de dades JSON pag. 78
6.2.5 Autentificació pag. 79
6.2.5.1 Descripció del procés d'autentificació pag. 79
6.2.5.2 Esquema de seqüència del procés d'autentificació pag. 81
6.2.6 Fils d'execució pag. 82
6.2.6.1 Descripció dels fils d'execució pag. 82
6.2.6.2 Espais de memòria compartida pag. 83
6.2.6.3 Esquema de seqüència dels fils d'execució pag. 84
6.2.7 Consideracions addicionals de seguretat pag. 85
6.2.7 Captures del component client pag. 86

7. Conclusions i treballs futurs

7.1 Treball realitzat pag. 87
7.2 Treballs futurs pag. 88
7.3 Conclusions pag. 89

8. Referències pag. 90
9. Llicència pag. 91

1. Introducció

1.1 Justificació i antecedents del projecte

Als darrers anys la implementació de sistemes de VoIP en l'àmbit de la petita i mitjana empresa ha experimentat un fort creixement. La millora continua de les línies de transmissió de dades, accentuada amb la implementació cada cop més estesa de les línies de fibra òptica, l'increment en el nombre d'operadors que ofereixen servies de VoIP així com la continua reducció de preus , han provocat que s'hagin dissipat les tradicionals reticències dels responsables dels departament de IT respecte a la qualitat i fiabilitat dels sistemes de VoIP, provocant alhora un considerable augment en la demanda d'aquestes solucions.

Entre les múltiples solucions de VoIP existents al mercat, destaquen per les seves prestacions, les solucions de programari lliure basades en FreePBX i sistemes derivats com ara les solucions Asterisk o Elastix. La primera versió de FreePBX va sorgir als voltants de l'any 2004 i des de llavors el seu desenvolupament ha estat constant, integrant cada cop més funcionalitats fins esdevenir un sistema madur que compta amb el suport d'una amplia comunitat.

Tot i l'increment de la qualitat i la fiabilitat d'aquests sistemes, la possibilitat d'una fallida ha de ser considerada. Un sistema de VoIP no deixa d'estar subjecte a les mateixes amenaces que qualsevol altre sistema informàtic. Als possibles problemes derivats de factors interns, com caigudes del rendiment del sistema o fallida de components, cal afegir-hi també d'altres causats per factors externs, com intrusions al sistema, caiguda de línies de comunicacions, etc. Tenint present que els sistemes de telefonia representen un servei crític per a moltes empreses i organitzacions, sorgeix la necessitat d'anticipar aquests problemes, o si més no, detectar-ne la causa tan aviat com sigui possible a fi de restablir el servei i evitar noves interrupcions. Es davant d'aquesta necessitat d'anticipació, on sorgeix la necessitat d'un sistema de monitorització amb capacitat per recopilar i analitzar les dades dels diferents serveis que integren el sistema.

És objectiu del present projecte el desenvolupament d'una eina de monitorització que doni una possible resposta a aquesta necessitat, i d'altres necessitats específiques del context del projecte.

1.2 Context del projecte

Aquest projecte es desenvolupa amb la col·laboració del Departament de Telefonia IP de l'empresa de serveis informàtics Infordisa S.A

A finals de 2013 Infordisa va impulsat un nou departament de Telefonia IP (VoIP) basat en centraletes FreePBX i sistemes derivats, especialment Elastix i Asterisk. Actualment el departament de Telefonia IP gestiona al voltant de 150 centraletes físiques distribuïdes en una radi d'acció que agrupa clients ubicats a Catalunya, Comunitat Valenciana i Balears.

Ben aviat va sorgir al Departament de Telefonia IP la necessitat de monitoritzar aquests equipaments per tal d'anticipar-se a possibles incidències que puguin afectar el serveis VoIP del client. Derivada d'aquesta necessitat, al llarg del temps s'han realitzat diverses proves d'integració de les centraletes en el sistemes de monitorització Nagios que l'empresa utilitza per a la monitorització de múltiples infraestructures de xarxes i comunicacions. Malgrat tot, diversos factors han fet que aquestes probes no hagin estat del tot satisfactòries

Ara fa uns anys, davant una problemàtica similar per monitoritzar una amplia infraestructura d'impressores en règim de Cost per Pàgina, l'empresa va decidir desenvolupar un eina de monitorització especialitzada que els ha donat molt bons resultats. Partint d'aquesta experiència prèvia, i davant la possibilitat sorgida arrel del present Treball de Final de Màster, es vol impulsar i col·laborar en l'estudi i desenvolupament d'un sistema de monitorització similar per la infraestructura de telefonia IP.

1.3 Descripció del projecte, objectius i funcionalitats.

L'objectiu principal d'aquest projecte és la anàlisis i desenvolupament d'una eina de monitorització remota per a sistemes de telefonia IP (VoIP) basats en FreePBX. Aquesta eina ha de permetre la monitorització centralitzada d'un conjunt infraestructures VoIP que es troben àmpliament distribuïdes en el territori. Cal que sigui adaptativa, de manera permeti definir de forma remota les especificacions per a cada sistema monitoritzat, així com permetre activar o desactivar les tasques de monitorització de

cada infraestructura. També extensible, permetent la incorporació a la monitorització de nous serveis amb relativa facilitat.

Per assolir l'objectiu, l'eina desenvolupada estarà integrada per dos components: un primer component servidor, la funcionalitat principal del qual és centralitzar la gestió de totes les tasques de monitorització dels nodes client. I un segon component client responsable d'executar les diferent tasques de monitorització a cada un dels nodes client i enviar-ne els resultats al servidor.

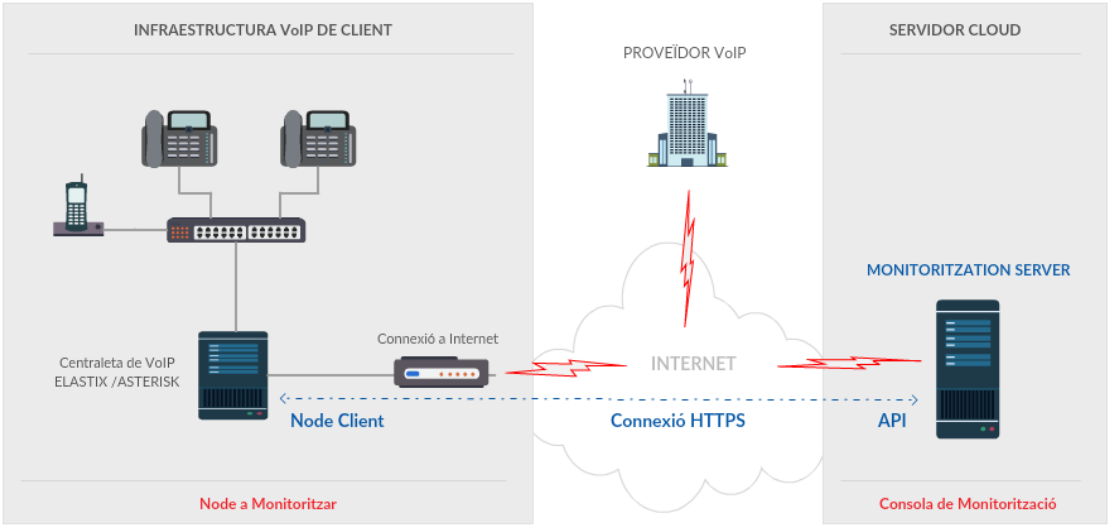


fig. 1 – Esquema general de la infraestructura de monitorització

A continuació es detallen els objectius i funcionalitats d'ambdós components:

1.3.1 Component Servidor

Estudi i desenvolupament d'un component servidor responsable de centralitzar la gestió de totes les tasques de monitorització dels nodes client, permetent assignar i configurar els serveis a monitoritzar per cada un dels nodes, així com recollir els resultats de les diferents tasques de monitorització i dipositar-los en una base de dades pel seu anàlisi. Aquest component es desenvoluparà en llenguatge Node.Js, i proporcionarà una API REST per a la comunicació i l'intercanvi de dades amb els node clients.

Principals funcionalitats :

- Identificar i autoritzar els nodes client

- Proporcionar a cada un dels nodes client la relació de serveis que ha de monitoritzar, així com la configuració específica de cada un d'ells.
- Recollir de forma passiva les dades reportades per cada node, processar-les i enregistrar-les a la base de dades.
- Analitzar les dades reportades per cada node, detectar possibles anomalies o errors i reportar-les generant les corresponents notificacions mitjançant correu electrònic.

Alhora, la API REST de la Consola de Monitorització també proporcionarà els recursos necessaris per permetre a un petit client web interactuar amb el servidor, de forma que es pugui visualitzar la informació obtinguda per cada un dels nodes, així com les diferents alertes o avisos existents al sistema.

1.3.2 Component Client

Desenvolupament en llenguatge C d'un client de monitorització que s'executarà a la infraestructura de centraletes de telefonia IP (VoIP) basades en FreePBX i derivats. Aquest component s'encarregarà de recopilar periòdicament informació sobre el sistema operatiu, així com el funcionament de diferents serveis instal·lats al sistema, principalment els relatius al sistema de VoIP. Aquestes dades és reportaran periòdicament per al seu processament al component servidor.

Principals funcionalitats:

- Identificar i autoritzar el node client al servidor de monitorització.
- Obtenir a partir del servidor de monitorització les especificacions de cada un dels serveis a monitoritzar.
- Executar periòdicament els diferents serveis de monitorització i reportar-ne els resultats als servidor de monitorització.

1.4 Calendari i planificació del projecte

El present projecte es desenvolupa entre el mes de Setembre de 2016 i el mes de Gener de 2017 i s'estructura en quatre fites a assolir en dates concretes. Cada una d'aquestes fites delimita una etapa necessària del projecte i en determina el calendari de treball.

Al següent quadre, es detallen cada una d'aquestes fites així com les tasques necessàries per assolir-les:

Tasques	Inici	Fi	
Pla de Treball	21/09/2016	1/10/2016	
Anàlisis previ i recollida d'objectius	21/09/2016	25/09/2016	3 dies
PAC1 - Redacció del Pla de Treball	26/09/2016	30/09/2016	9 dies
PAC1 - Redacció del Pla de Treball	1/10/2016	1/10/2016	**FITA**
Anàlisi, disseny i prototipatge de l'aplicació	26/09/2016	12/11/2016	
Anàlisi de requisits	26/09/2016	9/10/2016	14 dies
Investigació d'usuaris	26/09/2016	2/10/2016	7 dies
Especificació de requisits	3/10/2016	9/10/2016	7 dies
Disseny i prototipatge	10/10/2016	6/11/2016	28 dies
Elaborar escenaris d'us	10/10/2016	16/10/2016	7 dies
Definició fluxos interacció del sistema	17/10/2016	23/10/2016	7 dies
Disseny i construcció prototips del sistema	24/10/2016	2/11/2016	10 dies
Validació de prototips	3/11/2016	5/11/2016	3 dies
Identificació d'aspectes a millorar	10/10/2016	6/11/2016	28 dies
PAC2 - Redacció primera part de la memòria	1/11/2016	11/11/2016	12 dies
PAC2 - Lliurament primera part de la memòria	12/11/2016	12/11/2016	**FITA**
Desenvolupament de l'aplicació	7/11/2016	11/12/2016	
Desenvolupament API Servidor	7/11/2016	18/11/2016	12 dies
Desenvolupament Node Client	19/11/2016	4/12/2016	16 dies
Desenvolupament light Web Client	28/11/2016	10/12/2016	13 dies
PAC3 - Redacció segona part de la memòria	1/12/2016	10/12/2016	12 dies
PAC3 - Lliurament segona part de la memòria	11/12/2016	11/12/2016	**FITA**
Integració i jocs de prova	12/12/2016	25/12/2016	
Integració de l'aplicació en entorn de proves	12/12/2016	20/12/2016	9 dies
Jocs de proves	12/12/2016	20/12/2016	9 dies
Documentació de l'aplicació	21/12/2016	25/12/2016	5 dies
Lliurament final	19/12/2016	4/01/2017	
Redacció memòria del projecte	19/12/2016	4/01/2017	18 dies
Lliurament de la memòria del projecte	4/01/2017	4/01/2017	**FITA**

fig. 2 – Calendari del Projecte

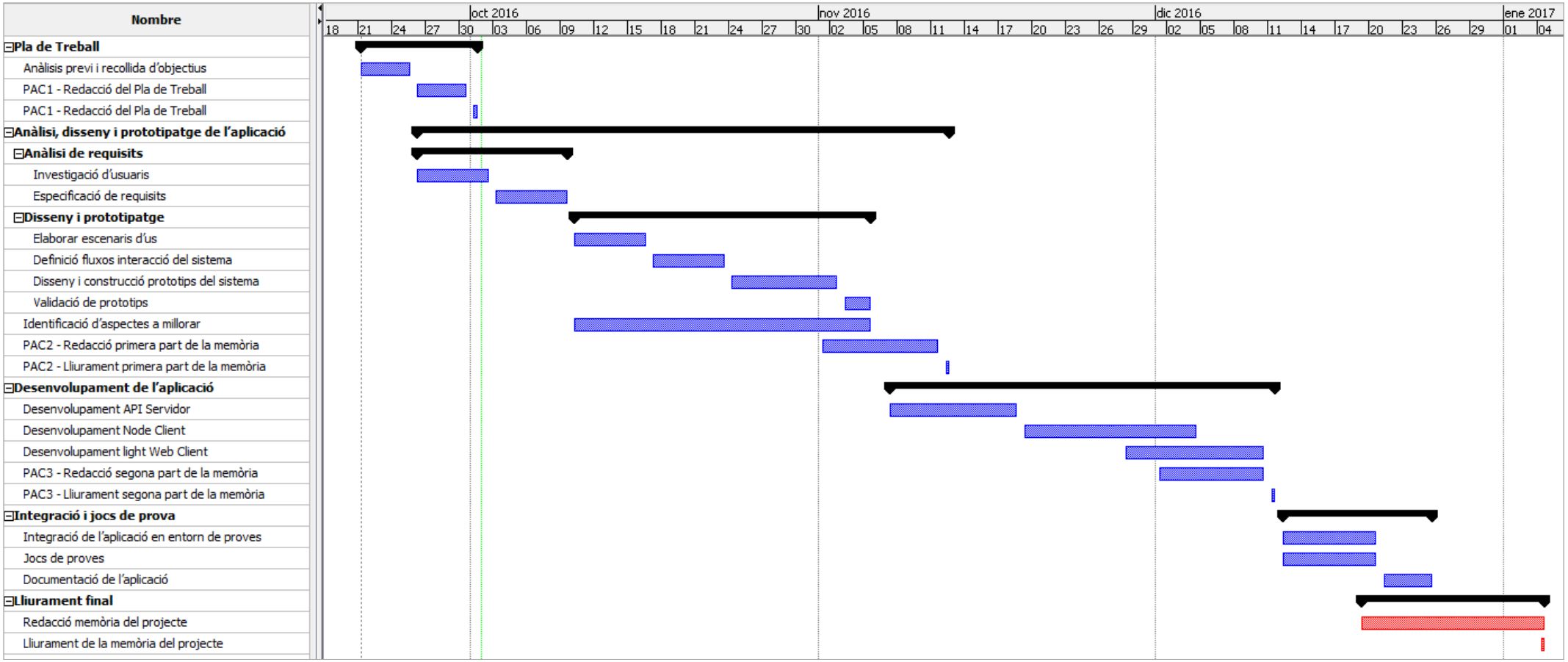


fig. 3 – Diagrama de Gantt del Projecte

1.5 Producte obtingut

Fruït de la elaboració del Treball Final de Màster i la realització de les Pràctiques Externes s'han generat una sèrie de documents que conformen la present memòria final, integrada pels següent documents:

- Pla de Treball
- Àmbit tecnològic del projecte
- Document d'anàlisi, disseny i prototipatge del sistema.
- Document de solucions d'implementació adoptades.
- Conclusions i aspectes a millorar

A banda, s'han obtingut un conjunt d'aplicacions agrupades dins el paquet *bnode-suite* on es desenvolupen les solucions d'implementació adoptades. Aquest paquet l'integren dos components:

- El component servidor *bnode-core*, desenvolupat amb una arquitectura NodeJS + MongoDB. La funcionalitat d'aquest component és centralitzar la gestió de totes les tasques de monitorització dels nodes client, permetent assignar i configurar els serveis a monitoritzar per cada node, així com recopilar els resultats de cada una de tasques de monitorització i dipositar-los en una base de dades pel seu anàlisi. El component servidor s'integra d'una API REST desenvolupada en NodeJS + MongoDB, així com una consola d'administració desenvolupada en AngularJS. Aquesta consola d'administració interacciona amb la API per configurar els serveis, assignar nodes i llistar els registres de monitorització.

- Components client bnode i bnode-conf desenvolupats en Ansi C. Són els components que s'executen al nodes. La utilitat bnode-conf s'encarrega de la configuració i registre de cada node al servidor, mentre que la aplicació bnode, dissenyada per executar-se com a servei, s'encarrega de descarregar del servidor la configuració dels serveis a monitoritzar, així com crear i executar les tasques de monitorització i enviar-ne els resultats al servidor.

El codi font d'aquest paquet, amb llicència GNU General Public License , es troba adjunt al present document.

1.6 Estructura del present document

El present document constitueix la memòria final del projecte i s'estructura en una sèrie d'apartats, cada un d'ells vinculat a una etapa diferent del projecte.

Al primer apartat de la memòria s'exposa breument el Pla de Treball, on s'hi descriuen els objectius del projecte, la seva motivació, així com les estratègies i accions a emprendre. Al segon apartat es realitza un breu anàlisi de l'àmbit tecnològic sobre el qual s'elabora el projecte i la seva relació amb els estudis del Máster Universitari de Programari Lliure cursat. Als següents apartats es descriu breument el procés de recollida de requisits, l'anàlisi de sistema i dels possibles escenaris, així com les solucions de disseny adoptades. Al sisè apartat, es fa una descripció detallada de les solucions d'implementació que constitueixen el producte final del projecte. La darrera part de la memòria esta constituïda per les conclusions finals del projecte així com una breu exposició d'alguns dels aspectes a millorar.

2. Àmbit tecnològic del projecte

2.1 Ús de Programari lliure

El present projecte és realitza dins el context del Treball Final de Màster en l'especialitat de Desenvolupament de Programari lliure. Per aquest motiu a banda dels objectius merament funcionals del projecte, també s'ha establert l'objectiu de desenvolupar el projecte utilitzant únicament eines de programari lliure.

2.2 Eines de desenvolupament de programari lliure

En el desenvolupament d'un projecte de programari lliure, com el present, sovint intervenen un o més grups de treballs que requereixen d'una sèrie de tecnologies que els permetin compartir un mateix entorn de treball, unificant criteris i metodologies. La utilització d'una eina de gestió de projectes permet distribuir i gestionar les diferents tasques que integren el projecte, així com establir un calendari i un espai de comunicació comú. Un altre eina força important són els sistemes de control de versions, que permeten a diferents individus o grups de treball col·laborar en una mateixa tasca de desenvolupament, i alhora que mantenir un històric de versions.

Al present apartat es fa una breu relació d'algunes d'aquestes eines i tecnologies, així com l'aplicació que se'n ha fet dins el desenvolupament del projecte.

fig. 8 – Relació de les eines de desenvolupament de programari lliure implementades

2.2.1 Eines de gestió de projectes

Un projecte de desenvolupament és descompon en un conjunt de tasques assignades a diferents membres d'un grup de treball, o fins i tot, entre múltiples grups de treball. Els sistemes de gestió de projectes permeten gestionar un o més projectes des d'una plataforma que permet crear i assignar tasques, monitoritzar la progressió del projecte, recollir els problemes i incidències sorgits durant el desenvolupament, etc.. A banda proporciona als grups de treball un espai de comunicació comú.

Per al desenvolupament del present projecte s'utilitza la següent eina de gestió de projectes, principalment per gestionar el calendari i les fites del projecte així com la definició i seguiment de tasques.

OpenProject

OpenProject és una solució de programari lliure per a la gestió de projectes. Aquesta eina permet la gestió de projectes basats en equips de treball, permetent l'assignació i control de les diferents tasques, la gestió de problemes i incidències o la gestió documental del projecte. A banda disposa d'una wiki per projecte i permet implementar metodologies de gestió de projectes Agile i Scrum. També permet establir fites, calendaris, càlcul de costos, etc ...

Com en d'altres projectes de programari lliure actual, OpenProject disposa d'una versió Community que podem descarregar i desplegar a la nostra pròpia infraestructura de servidors GNU/Linux , així com versions cloud en subscripció

2.2.2 Eines de control de versions

Per al desenvolupament del present projecte s'utilitza la següent eina de control de versions:

GitLab

L'eina GitLab és un projecte de codi lliure que implementa totes les funcionalitats de Git, el popular sistema de control de versions que va crear Linus Torvals per a la gestió del Kernel de Linux. A aquestes funcionalitats hi afegeix un acurat entorn web, similar al d'altres plataformes com GitHub o BitBucker, però a diferència d'aquestes, GitLab

permet la implementació d'un servidor en la nostra pròpia infraestructura sense necessitat de llicència.

GitLab implementa les següents funcionalitats:

- Permet la implementació d'un servidor GitLab a la nostra pròpia infraestructura o bé allotjar els nostres repositoris als servidors que GitLab té núvol.
- Permet implementar múltiples projectes en un mateix servidor, cada un amb els seus usuaris i permisos.
- Implementa sistema de control de versions basat en Git
- Seguiment errors
- Seguiment detallat de l'activitat d'un projecte
- Wiki de projecte

El present projecte es compon de tres components diferenciats: una API desenvolupada en Node.js, un client desenvolupat en C, i per últim, una web de gestió. Per a cada un d'aquets components s'ha creat un repositori en un servidor GitLab propi desplegat en una infraestructura virtualitzada. Aquest repositoris permetran implementar el control de versions i el seguiment d'errors per cada un dels components del projecte.

2.2.3 Editors de text

El codi font d'una aplicació es compon d'una sèrie de fitxers de text escrits en un llenguatge de programació determinat mitjançant un editor de text. Els desenvolupadors disposen d'una amplia varietat d'editors de text, des de editors bàsics que només permet l'edició de text en pla, com editors molt més complexes que incorporen funcionalitat com formateig automàtic de codi, detecció d'erros en la sintaxis, correcció ortogràfica, etc ..

A continuació és realitza una breu descripció de les diferents solucions de programari lliure utilitzades en el desenvolupament del present projecte:

Eclipse IDE for C/C++

Eclipse és una plataforma de desenvolupament integrat, originàriament desenvolupada per Java, està disponible també per al desenvolupament d'aplicacions

en altres llenguatges de programació, com ara C/C++, PHP, Cobol, entorns web HTML,CSS,Javascript, JSP, SQL, etc ...

Eclipse és una plataforma de desenvolupament integrat (IDE), això significa que integra en una mateixa plataforma tots els recursos necessaris per a desenvolupar en un llenguatge de programació determinat. A banda, Eclipse disposa d'un gran nombre d'extensions que permeten la seva integració amb altres funcionalitats com la integració amb sistemes de control de versions com Git i derivats, la generació de documentació, etc ...

L'entorn de desenvolupament Eclipse IDE for C/C++ utilitzat al projecte integra un conjunt d'eines conegudes com a Eclipse C/C++ Development Tools (CDT) que permeten gestionar el procés de compilació, anàlisis del codi, enllaçat de llibreries, gestió d'errors, etc.. directament des de l'entorn Eclipse. Aquest entorn també proporciona una amplia col·lació d'eines per depurar el codi executable, facilitant la detecció d'errors o comportaments anòmals.

Brackets

Brackets és un projecte de programari lliure, impulsat per Adobe, específicament dissenyat per a la creació d'aplicacions web que integrin tecnologies HTML, CSS i Javascript. A banda de les múltiples eines visuals que integra i que li proporcionen un entorn de desenvolupament web molt àgil, Brackets també permet integració amb Git, validació de HTML i CSS, o integració amb les eines per desenvolupadors dels navegador Chrome o Firefox, etc ...

Brackets s'ha utilitzat al present projecte per desenvolupar en Javascript de la API REST del servidor, així com en el desenvolupament HTML/CSS/Javascript del client web de gestió.

Vim

L'eina Vim ('Vi IMproved') és una evolució de l'editor de text Vi, present als sistemes Unix. Aquest editor, disponible a gairebé tots els derivats GNU/Linux, manté totes les característiques d'edició que tan popular van fer el seu antecessor e incorpora un gran nombre de noves funcionalitats, especialment adreçades als programadors.

L'eina Vim s'utilitza en el desenvolupament del present projecte per crear i editar diferents tipus de fitxers de text, com ara fitxers bash script o fitxers de configuració, des de sessions de terminal en mode text.

2.2.4 Eines de compilació i depuració

Un compilador és una eina que transforma el codi font d'una aplicació en un programa binari executable per un computador. Al present projecte s'utilitza un compilador GCC (GNU Compiler Collection) per transformar un conjunt de fitxers de codi font C, en una aplicació executable per sistemes GNU/Linux.

L'entorn de desenvolupament Eclipse IDE for C/C++ utilitzat al projecte integra un conjunt d'eines conegudes com a Eclipse C/C++ Development Tools (CDT) que permeten gestionar el procés de compilació, anàlisis del codi, enllaçat de llibreries, gestió d'errors, etc.. directament des de l'entorn Eclipse. Aquest entorn també proporciona una amplia col·lació d'eines per depurar el codi executable, facilitant la detecció d'errors o comportaments anòmals.

No obstant això, durant el desenvolupament del present projecte s'utilitzen diferents eines addicionals per depurar el codi executable:

Valgrind

L'eina Valgrind és una aplicació basada de programari lliure que permet diferents tipus d'anàlisi sobre el procés d'execució d'un programa binari en un sistema GNU/Linux. Al present projecte Valgrind s'utilitza, principalment, per a la detecció i correcció de problemes en la gestió de memòria.

2.2.5 Generadors de documentació

Documentar el codi d'una ampliació consisteix a afegir dins d'aquest codi tota la informació necessària per a poder identificar que fa, com i per què . En qualsevol projecte de desenvolupament la correcta documentació del codi resulta essencial, tant per permetre a altres membres de l'equip identificar i comprendre el codi, com per reutilitzar el codi o agilitzar futures revisions i correccions.

Les eines de generació de documentació analitzen el codi font d'una aplicació recollint tota la documentació que conté i la compacten en un format determinat, com ara un conjunt de pàgines web, un fitxer Latex, etc. ..

A continuació és realitza una breu descripció dels generadors de documentació utilitzats al present projecte:

Doxygen

Doxygen és una eina de programari lliure que escaneja el codi font d'una aplicació cercant certes etiquetes que utilitza per generar la documentació corresponent. Doxygen permet generar documentació en format HTML, LATEX, MAN Pages o XML.

Per agilitzar la generació i manteniment de la documentació durant el desenvolupament amb la plataforma Eclipse, s'utilitza el plugin Eclox. Entre d'altres característiques, aquest plugin ressalta dins l'entorn d'edició d'Eclipse els comentaris que utilitzen etiquetes de Doxygen, facilitant-ne la seva edició, gestió i identificació.

3. Requisits del sistema

3.1 Recollida de requisits

L'objectiu del present projecte és l'anàlisis i desenvolupament d'un sistema de monitorització per a sistemes VoIP basats en FreePBX que proporcioni al responsables tècnics d'aquestes infraestructures un eina capaç de recopilar informació exhaustiva dels sistemes que implementen, a fi d'anticipar possibles incidències i alhora proporcionar-los informació acurada que els permeti implementar la política de actuació més adient quan aquestes incidències es produeixin.

Per desenvolupament d'aquesta eina és necessari aprofundir en les característiques dels sistemes VoIP basats en FreePBX, definint el conjunt de serveis que implementen i establint aquells que són crítics per al funcionament del sistema, i que per tant, han de ser objecte de l'eina de monitorització. A banda, també cal conèixer quines són les necessitats del responsables tècnics d'aquests equipaments, per establir, en base a la seva experiència, el requisits que el sistema ha de complir per adequar-se a aquestes necessitats.

S'estableixen dos mètodes d'indagació. D'una banda s'utilitza un mètode d'observació i investigació contextual que ens permetrà definir el funcionament específic de sistemes basats en FreePBX, determinar el seu àmbit d'implementació i establint els seus serveis crítics.

D'altra banda, s'utilitzarà un mètode basat en entrevistes que ens permetrà conèixer les necessitats específiques dels responsables tècnics. Aquestes entrevistes tenen per objectiu establir , en base a l'experiència prèvia dels tècnics en sistemes FreePBX i de les eines per a la seva monitorització, els requisits funcionals que ha de satisfer el sistema de monitorització.

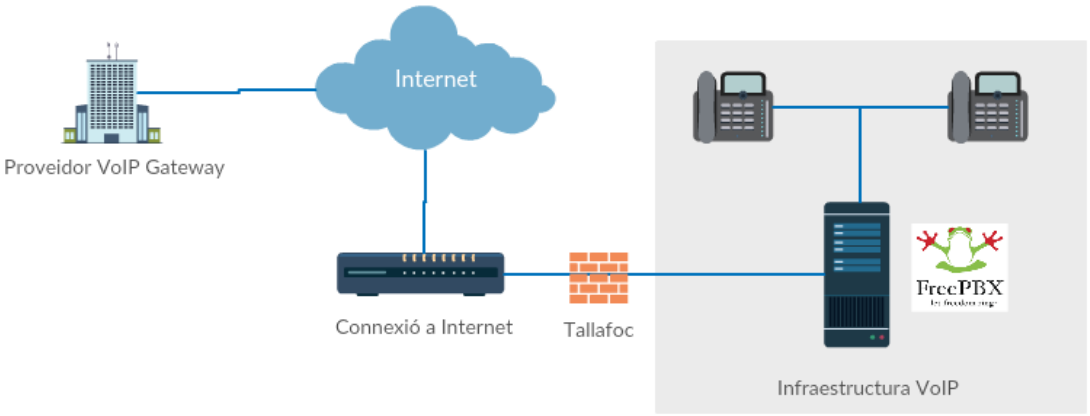
De l'anàlisi dels resultats obtinguts en ambdós mètodes d'indagació establirem les característiques i funcionalitat que l'eina de monitorització ha d'implementar per assolir els objectius del projecte.

3.2 Observació i investigació contextual

El present mètode d'indagació consisteix a observar el funcionament en producció d'un sistema de telefonia IP basat en FreePBX. D'una banda s'analitza el tipus d'infraestructura en la qual s'implementa aquest sistema, d'altra banda s'analitza com interacciona amb ell l'equip tècnic responsable de la seva implementació i manteniment.

D'aquest mètode d'estudi se'n desprenen les següent conclusions:

- Els sistemes de telefonia IP són elements crítics dins la infraestructura de comunicacions d'una empresa o organització. Qualsevol caiguda de servei té un impacte immediat en l'activitat de l'empresa i ha de ser corregida de forma immediata. D'aquesta observació se'n desprèn la necessitat de detectar qualsevol incidència tan aviat com es produeixi.
- El sistemes de telefonia IP basats en FreePBX utilitzen troncals de VoIP o *Gateways VoIP* remots proporcionats per operadors externs. La connexió amb aquests troncals es realitza a través d'una connexió a Internet, que pot o no dedicada a aquest servei. Qualsevol caiguda de la connexió a internet produeix la interrupció immediata del servei amb l'exterior.



- Els sistemes FreePBX són, com qualsevol altre sistema informàtic, vulnerables a atacs i intents de frau, això motiva que molts d'ells es trobin protegits per sistemes tallafoc. Cal considerar sensors de monitorització que permetin detectar i alertar de possibles intrusions al sistema.

- Un dels problemes més freqüents del sistemes de telefonia IP és la baixa qualitat d'àudio, que es manifesta en forma de talls de veu durant la comunicació. Aquests problema pot estar produït per problemes derivats de la connexió a internet que utilitza el sistema, per problemes interns al propi sistema com saturació de recursos, o problemes interns de l'operador que proporciona el servei de VoIP. El sistema de monitorització ha de proporcionar prou informació per poder determinar l'origen d'aquesta degradació del servei.
- Els sistemes FreePBX poden utilitzar alhora interfases FXS o FXO que els permeten interactuar amb línies analògiques convencionals. Cal considerar que en aquestes línies també es poden produir falles, provocant la interrupció dels serveis que suporten.
- Els sistemes FreePBX disposen d'una consola web que proporciona un entorn gràfic per a la gestió del sistema i la monitorització del seus recursos. No obstant això, per motius de seguretat aquesta consola no sempre es accessible remotament, especialment qual la centraleta es troba en sistemes gestionats per tercers on es proporciona un accés limitat al sistema. Un altre inconvenient és el gran nombre de centraletes implementades (FreePBX i derivats) que fan inviable accedir, una a una, a la consola web i monitoritzar els seus estats.

3.3 Entrevistes

El present mètode d'indagació consisteix realitzar una sèrie d'entrevistes amb els diferents components del departament de telefonia IP de l'empresa col·laboradora, *Infordisa S.A.*, més concretament amb els responsables tècnics dels sistemes de VoIP als quals va adreçada l'eina que es desenvolupa al present projecte.

L'objectiu d'aquestes entrevistes és, d'una banda, definir quins són els mecanismes de monitorització que utilitzen actualment i determinar quin avantatges i inconvenients presenten, així com determinar quines funcionalitat hi troben a faltar. D'altre banda, les entrevistes ha de servir per, en base a l'experiència dels tècnics, obtenir informació sobre el tipus d'incidents més habituals dels sistemes que gestionen.

Tot plegat ens permetrà determinar quin és el perfil d'usuari al que destinem l'eina de monitorització i quins requisits ha de tenir aquest eina per satisfer les seves necessitats.

3.3.1 Anàlisi de les entrevistes

Com a resultat de les entrevistes realitzades amb els responsables tècnics del departament de telefonia IP de l'empresa col·laboradora, Infordisa S.A, s'ha obtingut la següent relació de factors a considerar:

- De les entrevistes es desprèn que un sistema de monitorització actiu, on el sistema de monitorització executa directament els sensors de monitorització, és inviable. Les centraletes de telefonia IP es troben distribuïdes en multitud de clients amb infraestructures molt diferenciades. Els intent d'implementar aquest sistema de monitorització amb Nagios no han estat satisfactoris especialment en infraestructures de client gestionades per tercers. Les principals dificultats, l'accés remot al sistema, problemes amb canvis constant de les IP públiques, bloqueig de ports,etc... Tot plegat fa que s'arribi a la conclusió que el sistema a desenvolupar ha d'implementar monitoritzacions passives, on el client executa a intervals regulars de temps les tasques de monitorització i les reporta a la consola de monitorització pel seu anàlisi.

- Un altre fet destacable és que no tots els sistemes tenen les mateixes necessitats de monitorització, hi ha sistemes que integren múltiples troncals o Gateways Ip, sistemes que incorporen targetes especials amb ports FXS o FXO, sistemes més susceptibles a problemes amb les comunicacions, etc... Tot plegat fa que el departament tècnic requereixi d'un eina adaptable al sistema que ha de monitoritzar.

- Respecte al conjunt de serveis o sensors genèrics que cal monitoritzar a cada node destaquen aquells relatius al rendiments del sistema, com ara espai de disc, consum de memòria o temps actiu del sistema, així com els relatius a les comunicacions i l'accés a internet, i en especialment l'estat dels troncals Ip – connexions amb els Gateway VoIP del proveïdor de VoIP

- Una de les principals necessitats del departament tècnic és la detecció de trucades a numeracions especials amb cost addicionals. Una eina capaç de detectar i notificar en temps reals aquestes trucades seria de gran utilitat tan per al departament tècnic com per als clients.

- Els principals problemes detectats en els sistemes de telefonia IP gestionats per departament tècnic són el relatius a problemes amb la qualitat del servei. Gran part d'aquest tipus de problemes estan originats per incidències amb la connexió a internet, latència molt altes o pèrdues de paquets. Fins ara, el departament tècnic ha d'accedir remotament als equipament afectats per tal de realitzar els diagnòstics oportuns a les línies de comunicació.

- Un altre problema greu són els atacs i els accessos fraudulents al sistema. Si bé garantir la completa seguretat del sistema resulta una tasca difícil, especialment en infraestructures gestionades per tercers, un eina de monitorització capaç de detectar i informar d'accessos al sistema podria ser de gran ajuda per reduir els intents de frau.

3.4 Anàlisis de tasques

De la recollida de requisits desenvolupada als punts anteriors es desprèn la següent relació de funcionalitats i característiques que el sistema de monitorització ha d'oferir als usuaris per tal d'assolir els objectius del projecte:

1. Sistema **Client-Servidor**: Cal que el sistema a implementar permeti centralitzar en una mateixa eina de monitorització tota la infraestructura de centraletes de VoIP basades en FreePBX que es troba distribuïda en centres de client. Per tant, caldrà que el sistema de monitorització disposi d'una aplicació client responsable d'executar les tasques de monitorització a cada una de les centraletes, i d'una aplicació servidor o Consola de Monitorització que recollirà els registres dels diferents nodes client.

2. Sistema basat en **monitorització passives**: Donat que no es té accés remot a molts dels sistemes de VoIP, el sistema estarà basat en monitoritzacions passives, es a dir, cada cert interval de temps els nodes client executaran les tasques de monitorització que tinguin assignades i reportaran els registres resultants a la Consola de Monitorització.

3. Sistema de **monitorització adaptable**: Els serveis a monitoritzar poden variar d'un client a un altre, per tant a la Consola de Monitorització s'han de poder definir i assignar serveis per a cada client, així com modificar o deshabilitar els existents. La

aplicació client o node client, descarregarà aquest llistat de serveis i executarà les tasques segons s'hagin definit per a cada servei.

4. Seguretat del sistema: Les dades reportades pels nodes client a la Consola de Monitorització poden contenir dades sensibles, per tant el sistema ha d'implementar un sistema de comunicació segura.

5. Entorn de gestió: S'ha de dotar el sistema d'un entorn de gestió gràfic que permeti, d'una banda, definir i assignar serveis o sensors als nodes client, i d'altra banda visualitzar l'estat dels diferents serveis monitoritzats en cada client, així com els registres reportats.

6. Compatibilitat amb serveis de monitorització existents: El sistema ha de ser compatible, o parcialment compatible, amb els plugins de monitorització existents per a sistemes Nagios, de manera els client pugui executar aquesta mena de serveis.

7. Notificacions i alertes: El sistema ha de notificar per correu electrònic les incidències detectades en cada un serveis monitoritzats.

8. Retenció e històric de registres: El sistema ha de permetre definir diferent tipus de retenció de registres per servei, de manera que per uns serveis es pugui mantenir un històric mentre per d'altres només caldrà mantenir el darrer registre obtingut.

Hi han també una sèrie de funcionalitats que per la seva complexitat queden fora de l'àmbit del projecte, però que alhora, han de ser considerades per permetre que puguin ser implementades al sistema en un futur:

9. Anàlisi de registres telefònics: Com qualsevol sistema de telefonia els sistemes basats en FreePBX incorporen un registre de les trucades realitzades pel sistema. En un futur el sistema ha de permetre recopilar informació detallada d'aquests registres mantenint un històric del mateixos. Alhora el sistema ha de ser capaç d'analitzar aquest registres per detectar trucades a numeracions especials o susceptibles de ser fraudulentes.

4. Anàlisis del sistema

De la recollida de requisits es desprèn que el sistema de monitorització l'integraran dos components amb un desenvolupament independent però fortament vinculats funcionalment. D'una banda, un primer component servidor o back-end responsable de la definició i gestió del conjunt d'elements del sistema, així com de la gestió de les dades i la seva persistència, i d'altra banda, un segon component client o node client, responsable d'executar les tasques de monitorització a cada un dels sistemes client i reportar-ne els resultat al component servidor.

Al diagrama següent es mostren de manera esquemàtica els diferents components que integren l'eina de monitorització a desenvolupar:

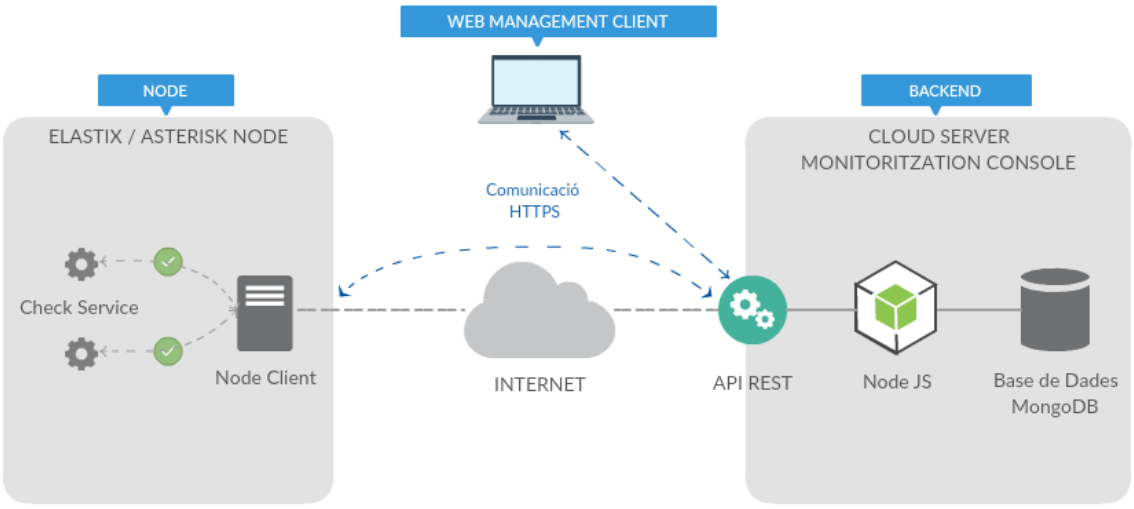


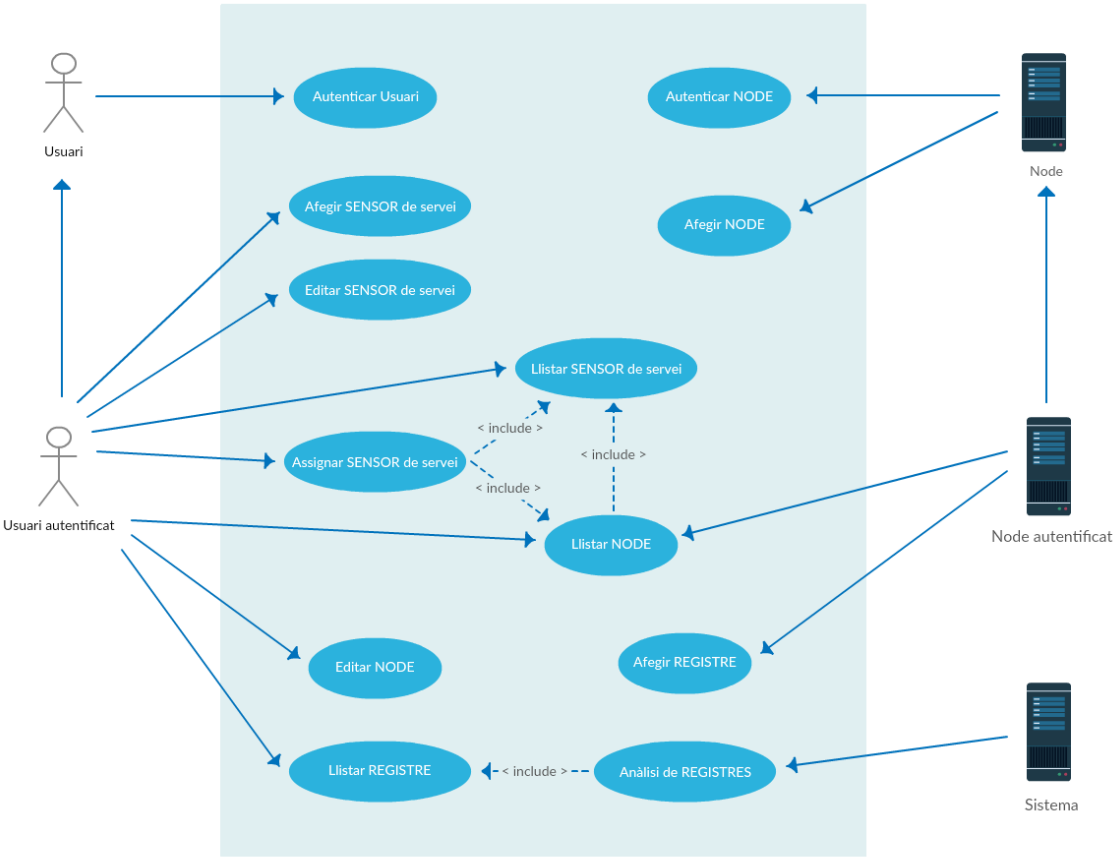
fig.9 – Arquitectura del sistema

Al present apartat es fa una relació, de forma independent, dels casos d'ús relatius al conjunt d'activitats a implementar per cada un del components.

4.1 Anàlisis del sistema: Component Servidor

4.1.1 Casos d'ús

El present apartat recull el llistat de casos d'ús relatius al component servidor obtinguts a patir dels requisits i funcionalitats adquirits en la fase d'anàlisi.



Al següent taula es llisten els casos d'ús resultants a partir dels requisits i funcionalitats obtinguts en la fase d'anàlisi i que es detallen a l'apartat següent.

Codi	Descripció		Actor
CU-A01	Autenticar usuari		Usuari
CU-A02	Afegir NODE		Node autentificat
CU-A03	Editar NODE		Usuari autentificat
CU-A04	Llistar NODE	Inclou CU-A06	Usuari autentificat Node autentificat

CU-A05	Afegir SENSOR de servei		Usuari autentificat
CU-A06	Llistar SENSOR de servei		Usuari autentificat Node autentificat
CU-A07	Editar SENSOR de servei		Usuari autentificat
CU-A08	Assignar SENSOR de servei	Inclou CU-A06 Inclou CU-A04	Usuari autentificat
CU-A09	Afegir REGISTRE		Node
CU-A10	Llistar REGISTRE		Usuari autentificat
CU-A11	Autenticar NODE		Node
CU-A12	Anàlisi de registres	Inclou CU-A10	Sistema

4.1.2 Estudi detallat dels casos d'ús del component servidor

4.1.2.1 CU-A01 – Autenticar usuari

L'usuari acaba d'accedir a l'aplicació i se li requereix autentificació per a poder accedir al seu espai de treball.

Escenari d'ús:	**Autenticar usuari**
Versió:	1.0
Actors:	Usuari
Identificació	CU-A01
Propòsit:	Identifica un usuari del sistema i obté cookie d'identificació
Precondicions:	L'usuari no està validat.
Seqüència normal:	P1) El sistema sol·licita identificació a l'usuari P2) L'usuari introdueix identificador i contrasenya P3) El sistema. P4) Campus Virtual valida usuari i retorna token. P5) El sistema recupera token, el salvaguarda i passa a escenari d'ús CU02 *Taulell*.
Excepcions:	E1) El Campus Virtual no respon – Mostra missatge error connexió. E2) El Campus Virtual no valida usuari – Mostra missatge d'error i torna a P1
Post condicions:	L'usuari està validat al sistema i disposem d'un cookei d'autenticació.

4.1.2.2 CU-A02 - Afegir NODE

L'usuari sol·licita al servidor la creació d'un nou NODE al que se li assigna un identificador únic i un token d'identificació.

Escenari d'ús:	**Afegir NODE**
Versió:	1.0
Actors:	Usuari
Identificació	CU-A02
Propòsit:	Afegeix un nou NODE al sistema i retorna un identificador únic i el seu corresponent *token* d'autenticació
Precondicions:	El NODE no existeix
Seqüència normal:	P1) Un NODE no registrat sol·licita al BACKEND la creació d'un nou NODE P2) El BACKEND sol·licita identificació a l'usuari P3) L'usuari introdueix el seu identificador i contrasenya P4) El BACKEND autentifica l'usuari i afegeix un nou NODE a la base de dades. Com a resultat de crear el nou NODE s'obté un identificador únic i un token d'identificació. P5) El BACKEND retorna al NODE el seu identificador únic i el token d'identificació corresponent. El node utilitzarà aquestes dades en un futur per identificar-se al sistema.
Excepcions:	E1) El BACKEND no valida usuari – Mostra missatge d'error i torna a P1
Post condicions:	S'ha afegit un nou NODE a la base de dades i s'ha retornat un identificador únic i un token de identificació.

4.1.2.3 CU-A03 - Editar NODE

Donat un identificador de NODE únic, permet a l'usuari modificar els atributs i la configuració del NODE.

Escenari d'ús:	**Editar NODE**
Versió:	1.0
Actors:	Usuari
Identificació	CU-A03

Propòsit:	Permet modificar els atributs d'un NODE existent al sistema
Precondicions:	El NODE amb l'identificador donat existeix a la base de dades L'usuari està autentificat i té permisos per modificar el node
Seqüència normal:	P1) L'usuari sol·licita al BACKEND editar un node i li proporciona un conjunt d'atributs que s'han d'actualitzar. P2) El BACKEND verifica que l'usuari està autentificat i té els permisos corresponent per editar nodes. P3) El BACKEND actualitza a la base de dades els atributs del NODE amb l'identificador donat.
Excepcions:	E1) L'usuari no està autentificat al BACKEND. Retorna error: *Usuari no autentificat.* E2) El NODE amb l'identificador donar no existeix a la base de dades. Retorna error: *Element no existeix a la base de dades.*
Post condicions:	S'han modificat els atributs del NODE a la base de dades.

4.1.2.4 CU-A04 - Llistar NODE o NODES

Donat un identificador de NODE únic, el servidor retorna el node especificat i el seu conjunt d'atributs, així com el llistat de serveis que té assignats. Si no s'especifica un node concret, el BACKEND retornar una relació de tots els node del sistema i els seus corresponents atributs.

Escenari d'ús:	**Llistar NODE**
Versió:	1.0
Actors:	Usuari
Identificació	CU-A04
Propòsit:	Retorna un node o conjunt de nodes i els seus atributs, així com el llistat de sensors de servei que tenen assignats cada un d'ells.
Precondicions:	El NODE amb l'identificador donat existeix a la base de dades L'usuari o el node està autentificat i té permisos per llistar el node
Seqüència normal:	P1) Un usuari o node sol·liciten al BACKEND la configuració d'un node especificant el seu identificador únic P2) El BACKEND verifica que l'usuari o el node està autentificat i té els permisos corresponent per llistar nodes.

	P3) El BACKEND recupera de la base de dades el node amb l'identificador especificat. P4) Crida cas d'ús *CUA06-Llistar SENSORS de servei* recuperant el llistat de sensors assignats al node. P5) Retorna el conjunt d'atributs de node , així com el llistat de sensors serveis que té assignats.
Seqüència alternativa:	P1) L'usuari sol·licita al BACKEND informació sobre el nodes de sistema P2) El BACKEND verifica que l'usuari o el node està autentificat i té els permisos corresponent per llistar nodes. P3) El BACKEND recupera de la base de dades el llistat de nodes del sistema P4) Per a cada un dels nodes recuperats, crida el cas d'ús *CUA06-Llistar SENSORS de servei* recuperant el llistat de sensors assignats al node. P5) Retorna un llistat amb el conjunt de nodes del sistema , així com el llistat de sensors serveis que té assignats cada un d'ells.
Excepcions:	E1) L'usuari no està autentificat al BACKEND. Retorna error: *Usuari no autentificat.* E2) El NODE amb l'identificador donar no existeix a la base de dades. Retorna error: *Element no existeix a la base de dades*
Post condicions:	S'ha retornat a l'usuari el NODE especificat i el seu conjunt d'atributs, així com el llistat de serveis que té assignats.

4.1.2.5 CU-A05 – Afegir SENSOR de servei

L'usuari introdueix al sistema un nou sensor de servei amb un conjunt d'atributs donat.

Escenari d'ús:	**Afegir SENSOR de servei**
Versió:	1.0
Actors:	Usuari
Identificació	CU-A05
Propòsit:	Afegeix un nou sensor de SERVEI al sistema.
Precondicions:	L'usuari està autentificat i té permisos per afegir sensors al sistema
Seqüència normal:	P1) L'usuari sol·licita al BACKEND la creació d'un nou SENSOR al sistema i li proporciona un conjunt d'atributs.

	P2) El BACKEND verifica que l'usuari està autentificat i té els permisos corresponent per afegir serveis. P3) El BACKEND afegeix un nou sensor de servei a la base de dades amb els atributs donats. Si un atribut no existeix, s'utilitza la configuració per defecte per l'atribut. P4) El BACKEND retorna l'identificador únic del SENSOR de servei creat.
Excepcions:	E1) L'usuari no està autentificat al BACKEND. Retorna error: *Usuari no autentificat.* E2) El NODE amb l'identificador donar no existeix a la base de dades. Retorna error: *Element no existeix a la base de dades*
Post condicions:	S'ha afegit un nou SENSOR de SERVEI a la base de dades i s'ha retornat el seu identificador únic.

4.1.2.6 CU-A06 - Llistar SENSOR o SENSORS de servei

Donat un identificador de sensor de servei únic, el sistema retorna el sensor especificat i el seu conjunt d'atributs. Si no s'especifica un sensor concret, el sistema retorna una relació de tots els sensors del sistema i els seus atributs corresponents.

Escenari d'ús:	**Llistar SENSOR de servei**
Versió:	1.0
Actors:	Usuari / Node
Identificació	CU-A06
Propòsit:	Retorna a un usuari o node. un sensor de servei o llistat de sensors i els seus atributs.
Precondicions:	El SENSOR de servei amb l'identificador donat existeix a la base de dades L'usuari està autentificat i té permisos per modificar el node
Seqüència normal:	P1) Un usuari o node sol·liciten al BACKEND la configuració d'un sensor de servei especificant el seu identificador únic P2) El BACKEND verifica que l'usuari està autentificat i té els permisos corresponent per llistar serveis. P3) El BACKEND recupera de la base de dades el sensor especificat i retorna el seu conjunt d'atributs

Seqüència alternativa:	P1) L'usuari sol·licita al BACKEND informació sobre e conjunt des sensors del sistema P2) El BACKEND verifica que l'usuari o el node està autentificat i té els permisos corresponent per llistar sensors. P3) El BACKEND recupera de la base de dades el llistat de sensors del sistema i el retorna a l'usuari/node
Excepcions:	E1) L'usuari no està autentificat al BACKEND. Retorna error: *Usuari no autentificat.* E2) El SENSOR amb l'identificador donar no existeix a la base de dades. Retorna error: *Element no existeix a la base de dades*
Post condicions:	S'ha retornat a l'usuari el sensor especificat i el seu conjunt d'atributs.

4.1.2.7 CU-A07 - Editar SENSOR de servei

Donat un identificador únic de sensor i un conjunt d'atributs, el BACKEND cerca a la base de dades el sensor de servei i actualitza els seus atributs.

Escenari d'ús:	**Editar SENSOR de servei**
Versió:	1.0
Actors:	Usuari
Identificació	CU-A07
Propòsit:	Permet modificar els atributs d'un SENSOR de servei existent al sistema
Precondicions:	El sensor amb l'identificador donat existeix a la base de dades L'usuari està autentificat i té permisos per modificar el node
Seqüència normal:	P1) L'usuari sol·licita al BACKEND editar un sensor i li proporciona un conjunt d'atributs que s'han d'actualitzar. P2) El BACKEND verifica que l'usuari està autentificat i té els permisos corresponent per editar nodes. P3) El BACKEND actualitza a la base de dades els atributs del sensor amb l'identificador donat
Excepcions:	E1) L'usuari no està autentificat al BACKEND. Retorna error: *Usuari no autentificat.* E2) El sensor amb l'identificador donar no existeix a la base de dades. Retorna error: *Element no existeix a la base de dades*

Post condicions:	S'han modificat els atributs d'un sensor a la base de dades i s'ha retornat el seu identificador únic.

4.1.2.8 CU-A08 - Assignar SENSOR de servei

Donats un identificador de node i un identificador de sensor, el sistema afegeix el servei al llistar de sensors a monitoritzar per un NODE especificat.

Escenari d'ús:	**Assignar SENSOR de servei**
Versió:	1.0
Actors:	Usuari
Identificació	CU-A08
Propòsit:	Permet afegir un sensor de servei al llistat de sensors a monitoritzar per un NODE
Precondicions:	El NODE amb l'identificador donat existeix a la base de dades El sensor amb l'identificador donat existeix a la base de dades L'usuari està autentificat i té permisos per modificar el node
Seqüència normal:	P1) L'usuari sol·licita al BACKEND afegir un SENSOR al llistat de sensors d'un NODE determinat. P2) El BACKEND verifica que l'usuari està autentificat i té els permisos corresponent per editar nodes. P3) El BACKEND crida el cas d'ús *CU-A04-Llistar Node* recuperat les dades d'un node així com el llistat de sensors que té assignats. P4) Si el llistat de sensors del node no conté el sensor especificat, el BACKEND afegeix el sensor al llistat i actualitza la base de dades
Excepcions:	E1) L'usuari no està autentificat al BACKEND. Retorna error: *Usuari no autentificat.* E2) El NODE amb l'identificador donat no existeix a la base de dades. Retorna error: *Element no existeix a la base de dades* E3) El SENSOR amb l'identificador donat no existeix a la base de dades. Retorna error: *Element no existeix a la base de dades*
Post condicions:	S'ha afegit el SENSOR al llistat de SENSORS a monitoritzar pel NODE

4.1.2.9 CU-A09 – Afegir REGISTRE o REGISTRES

Un NODE introdueix al sistema un nou registre, o un conjunt de registres, així com els seus d'atributs.

Escenari d'ús:	**Afegir REGISTRE**
Versió:	1.0
Actors:	Node
Identificació	CU-A09
Propòsit:	Afegeix un nou REGISTRE al sistema.
Precondicions:	EL NODE disposa d'un identificador únic i el seu corresponent token d'identificació. EL NODE disposa d'un o més registres per reportar al BACKEND
Seqüència normal:	P1) El NODE envia al BACKEND un nou registre o conjunt de registres, així com el seu identificador únic i el token d'identificació. P2) El BACKEND autentica el NODE a partir de l'identificador i el token donats. P3) El BACKEND afegeix el registre o conjunt registres especificats a la base de dades del sistema.
Excepcions:	E1) El NODE no disposa d'un identificador o *token* vàlids. Retorna error: *Node no autentificat.*
Post condicions:	S'ha afegit a la base de dades el registre o conjunt de registres donats.

4.1.2.10 CU-A10 - Llistar REGISTRE o REGISTRES

Un usuari sol·licita al BACKEND un llistat de registres que compleixin amb unes condicions determinades, com ara node origen, data de creació, etc...

Escenari d'ús:	**Llistar REGISTRE**
Versió:	1.0
Actors:	Usuari
Identificació	CU-A10

Propòsit:	Llista un conjunt de registres
Precondicions:	L'usuari està autentificat i té permisos per llistar registres
Seqüència normal:	P1) L'usuari sol·licita al BACKEND un llistat de registres amb unes condicions especifiques, com ara el node origen, la data de creació , etc .. P2) El BACKEND verifica que l'usuari està autentificat i té els permisos corresponents per llistar serveis. P3) El BACKEND recupera de la base de dades el conjunt de registres especificats i els retorna a l'usuari.
Excepcions:	E1) L'usuari no està autentificat al BACKEND. Retorna error: *Usuari no autentificat.*
Post condicions:	S'ha retornat a l'usuari un o més registres que compleixen les condicions especificades.

4.1.2.11 CU-A11 – Autenticar NODE

Un NODE s'identifica al sistema per accedir a recursos que requereixen identificació.

Escenari d'ús:	**Autenticar NODE**
Versió:	1.0
Actors:	Node
Identificació	CU-A11
Propòsit:	Autentifica un NODE a partir d'un identificador i el seu token corresponent.
Precondicions:	EL NODE disposa d'un identificador únic i el seu corresponent *token* d'identificació.
Seqüència normal:	P1) El BACKEND recupera de la base de dades el NODE en base a l'identificador especificat. P2) Comprova la correspondència entre el token donat i el que figura a la base de dades. Si hi ha correspondència autentica el NODE.
Excepcions:	E1) El NODE no disposa d'un identificador o *token* vàlids. Retorna error: *Node no autentificat.*
Post condicions:	S'ha autenticat un NODE al sistema

4.1.2.12 CU-A12 – Anàlisi de REGISTRE

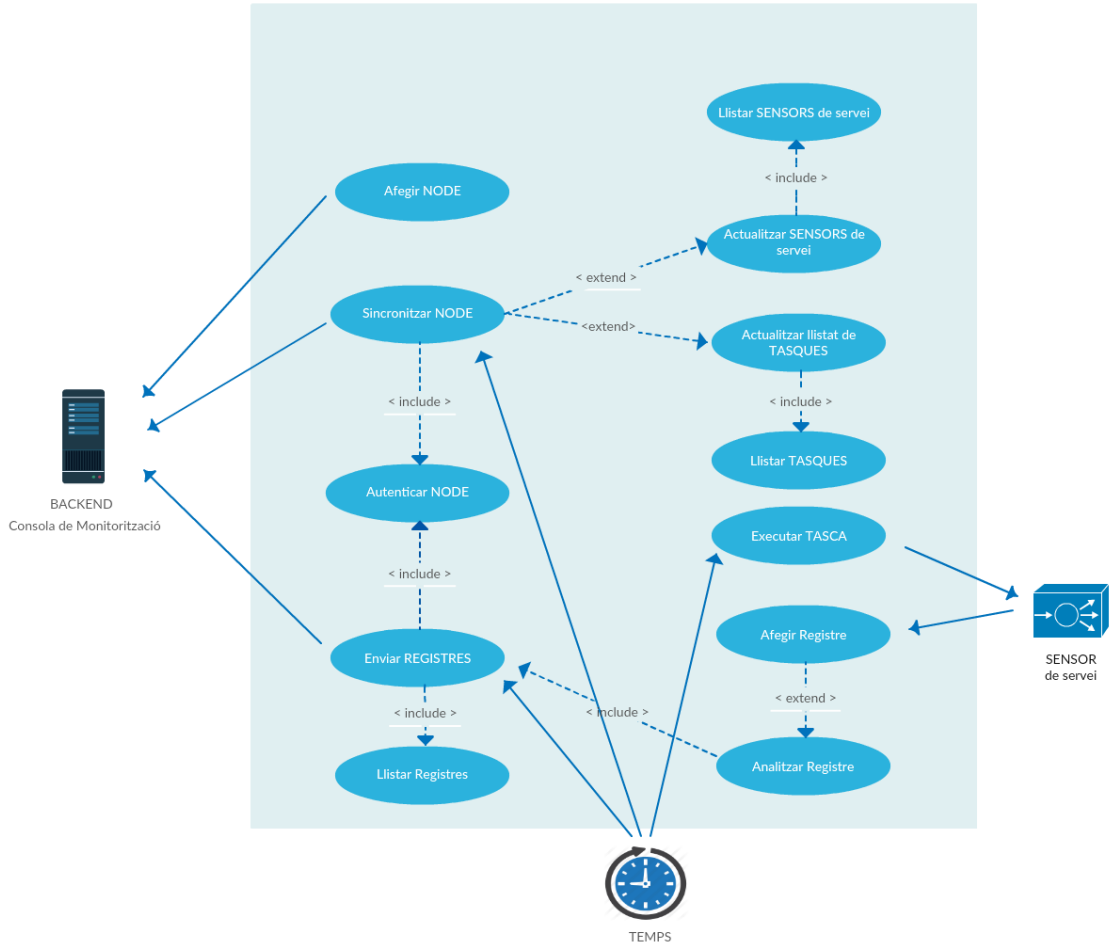
El sistema accedeix a un registre o conjunt de registres per analitzar-ne els seus atributs. Per a cada registre recupera la configuració del sensor de servei que l'ha generat, en base a aquesta configuració, determina si cal generar una alerta o notificació.

Escenari d'ús:	**Anàlisi de REGISTRE**
Versió:	1.0
Actors:	Sistema
Identificació	CU-A12
Propòsit:	Analitza un registre un conjunt de registres
Precondicions:	Existeixen al sistema un registre o conjunt de registres pendents d'analitzar.
Seqüència normal:	P1) El sistema recupera de la base de dades un registre o llistat de registres per analitzar-ne els atributs. P2) De cada registre recupera de la base de dades el sensor de servei que l'ha generat. P3) El sistema analitza els atributs del registre en relació a la configuració del sensor de servei i determina si es necessari generar una nova alerta/notificació.
Excepcions:	E1) L'usuari no localitza un sensor de servei. Retorna error: *Element no existeix a la base de dades*
Post condicions:	S'han analitzat un o més registres i s'ha generat les notificacions corresponents.

4.2 Anàlisis del sistema: Component Client

4.2.1 Casos d'ús

El present apartat recull el llistat de casos d'ús relatius a l'aplicació client responsable d'executar els sensors de servei en cada un dels nodes client, així com reportar periòdicament els resultats d'aquestes execucions al servidor BACKEND.



Al següent taula es llisten els casos d'ús resultants a partir dels requisits i funcionalitats obtinguts en la fase d'anàlisi i que es detallen a l'apartat següent.

Codi	Descripció	Actor
CU-B01	Afegir NODE	Temps
CU-B02	Autenticar NODE	

CU-B03	Llistar SENSORS de servei		
CU-B04	Actualitzar SENSORS de servei	Inclou CU-B03	
CU-B05	Llistar TASQUES de monitorització		
CU-B06	Actualitzar llistat de TASQUES de monitorització	Inclou CU-B05	
CU-B07	Sincronitzar NODE	Extens CU-B04 Extens CU-B06 Inclou CU-B02	Temps
CU-B08	Executar tasques		Temps
CU-B09	Afegir registre	Extens CU-B08	Sensor de servei
CU-B10	Analitzar registre	Inclou CU-B10	
CU-B11	Llistar registres		
CU-B12	Enviar registres	Inclou CU-B02	Temps

4.2.2 Estudi detallat dels casos d'ús del component client

4.2.2.1 CU-B01 – Afegir NODE

Un node que no disposa de identificador envia una petició de registre a una Consola de Monitorització i n'obté un identificador i token únics.

Escenari d'ús:	**Afegir NODE**
Versió:	1.0
Actors:	Sistema
Identificació	CU-B01
Propòsit:	Envia una petició de registre a la Consola de Monitorització i n'obté un identificador i token únics.
Precondicions:	El node no disposa de identificador.
Seqüència normal:	P1) El sistema envia a una Consola de Monitorització una petició de registre. P2) El sistema espera la resposta de la Consola de Monitorització i en recupera l'identificador i token únics. P3) El sistema desa l'identificador i token únics al fitxer de configuració del node.

Excepcions:	E1) La Consola de Monitorització no respon – Mostra missatge error connexió. E2) La Consola de Monitorització no accepta la petició de registre – Mostra missatge d'error i torna a P1
Post condicions:	El node s'ha registrat a la Consola de Monitorització i disposa d'un identificador i token únics.

4.2.2.2 CU-B02 – Autenticar NODE

Un node s'identifica en una Consola de Monitorització per accedir a un conjunt de recursos protegits.

Escenari d'ús:	**Autenticar NODE**
Versió:	1.0
Actors:	Sistema
Identificació	CU-B02
Propòsit:	Identifica el node en una Consola de Monitorització
Precondicions:	El node disposa d'un identificador i *token* únics.
Seqüència normal:	P1) Un node sol·licita identificació a la Consola de Monitorització, enviant-hi el seu *token* i identificador únics.
Excepcions:	E1) La Consola de Monitorització no respon – Mostra missatge error connexió. E2) La Consola de Monitorització no accepta la identificació del node – Mostra missatge d'error i torna a cas d'ús CU-BO1
Post condicions:	El node s'ha identificat correctament a la Consola de Monitorització

4.2.2.3 CU-B03 – Llistar SENSORS de servei

Llista els sensors de servei d'un node.

Escenari d'ús:	**Llistar SENSORS de servei**
Versió:	1.0
Actors:	Sistema
Identificació	CU-B03

Propòsit:	Llista els sensors de servei d'un node, això com els atributs de cada un d'ells.
Precondicions:	El disposa d'un o més sensors de servei
Seqüència normal:	P1) Recorre el llistat de sensors de servei i retorna el seus atributs.
Excepcions:	E1) El node no disposa de cap sensor de servei, retorna un llista buida.
Post condicions:	S'ha retornat el llistat de sensors de servei que conté el node, així com una relació dels atributs de cada un dels sensors.

4.2.2.4 CU-B04 – Actualitzar SENSORS de servei

Donat un conjunt de sensors de servei, el node els compara amb el sensors de la seva llista. Si la llista ja conté un del sensors, compara els atributs i si es necessari els actualitza. Si la llista no conté el sensor l'afegeix.

Escenari d'ús:	**Actualitzar SENSOR de servei**
Versió:	1.0
Actors:	Sistema
Identificació	CU-B02
Propòsit:	Actualitza els sensors de servei d'un node
Precondicions:	S'ha proporcionar un conjunt de sensors de servei a actualitzar.
Seqüència normal:	P1) El node executa el cas d'ús *CU-B03-Llistar sensors* i recupera el llistat de sensors del node. P2) El node compara el conjunt de sensors proporcionat amb el conjunt de sensors llistat. P2a) Si el llistat conté el sensor, actualitza els seus atributs. P2b) Si el llistat no conté el sensor l'afegeix
Excepcions:	
Post condicions:	El node ha actualitzar correctament el seus llistat de sensors de servei.

4.2.2.5 CU-B05 – Llistar TASQUES de monitorització

Llista les tasques de monitorització d'un node

Escenari d'ús:	**Llistar TASQUES de monitorització**
Versió:	1.0
Actors:	Sistema
Identificació	CU-B05
Propòsit:	Llista les tasques de monitorització d'un node
Precondicions:	El no disposa d'una o més tasques de monitorització
Seqüència normal:	P1) Recorre el llistat de tasques de monitorització i retorna el seus atributs.
Excepcions:	E1) El node no disposa de cap tasca, retorna un llista buida.
Post condicions:	S'ha retornat el llistat tasques de monitorització que conté el node

4.2.2.6 CU-B06 – Actualitzar llistat de TASQUES de monitorització

Donat un conjunt de sensors de servei, el node els compara amb el sensors de la seva llista. Si la llista ja conté un del sensors, compara els atributs i si es necessari els actualitza. Si la llista no conté el sensor l'afegeix.

Escenari d'ús:	**Actualitzar llistat de TASQUES de monitorització**
Versió:	1.0
Actors:	Sistema
Identificació	CU-B02
Propòsit:	Actualitza el llistat de tasques de monitorització d'un node
Precondicions:	S'ha proporcionar un conjunt de sensors de servei.
Seqüència normal:	P1) El node executa el cas d'ús *CU-B03-Llistar tasques* i recupera el llistat de tasques del node. P2) El node compara el conjunt de sensors proporcionat amb el conjunt de tasques de monitorització del llistat. P2a) Si el llistat conté la tasca de monitorització corresponent al sensor, actualitza els seus atributs. P2b) Si el llistat no conté cap tasca de monitorització corresponent al sensor, crea una nova tasca i l'afegeix a llistat de tasques.

Excepcions:	
Post condicions:	El node ha actualitzar correctament el seus llistat de tasques de monitorització.

4.2.2.7 CU-B07 – Sincronitzar NODE

Transcorregut un interval de temps especificat, un node sincronitza amb la Consola de Monitorització la seva configuració. Per fer-ho s'identifica a la Consola i obté la seva configuració, així com el llistat de sensors de serveis que ha de gestionar. El node compara les dades obtingudes de la Consola, actualitzant el seu llistat de serveis i generant una tasca de monitorització per cada nou sensor de servei

Escenari d'ús:	**Sincronitzar NODE**
Versió:	1.0
Actors:	Temps
Identificació	CU-B07
Propòsit:	Recupera d'una Consola de Monitorització la configuració i llistat de serveis d'un node determinat. Actualitza el seu llistat de sensors de servei i genera una tasca de monitorització nova per cada nou servei.
Precondicions:	S'ha arribat a *l'interval de sincronització* especificat a la configuració del node. El node disposa d'un identificador i *token* únics.
Seqüència normal:	P1) Un node sol·licita una tasca de sincronització a la Consola de Monitorització, enviant-hi el seu *token* i identificador únics. P2) El node espera la resposta de la Consola de Monitorització i en recupera la seva configuració i el llistat de serveis que ha de monitoritzar. P3) El node crida el cas d'ús *CU-B04-Actualitzar SENSORS* actualitzant el seu conjunt de sensors de servei a partir de la llista recuperada de la Consola de Monitorització. P4) El node crida el cas d'ús *CU-B06-Actualitzar TASQUES de monitorització* actualitzant el seu conjunt de tasques a partir del llistat de sensors recuperat de la Consola de Monitorització.

Excepcions:	E1) La Consola de Monitorització no respon – Mostra missatge error connexió. E2) La Consola de Monitorització no accepta la identificació del node – Mostra missatge d'error i torna a cas d'ús CU-BO1
Post condicions:	El node s'ha actualitzat correctament

4.2.2.8 CU-B08 – Executar TASCA de Monitorització

Arribat l'interval d'execució d'una tasca de monitorització, el node demana al sistema operatiu base que executi la tasca corresponent i en recupera el resultat.

Escenari d'ús:	**Executar TASCA**
Versió:	1.0
Actors:	Temps
Identificació	CU-B08
Propòsit:	Executa una tasca de monitorització i en recupera el resultat.
Precondicions:	S'ha arribat a *l'interval d'execució* d'una tasca de monitorització. El sistema disposa d'un binari, script o comanda relacionat amb la tasca de monitorització.
Seqüència normal:	P1) Arribat l'interval d'execució d'una tasca de monitorització, el node demana al sistema operatiu base que executi la tasca corresponent. P2) El node espera el resultat d'executat la tasca P2a) Si es produeix un resultat, crea i afegeix el registre corresponent P2b) Si transcorregut un temps no hi ha resposta, cancel·la la execució de la tasca i genera un nou registre d'error. P3) El node defineix un nou interval d'execució per la tasca de monitorització.
Excepcions:	E1) El sistema base no respon – Mostra error executant la tasca
Post condicions:	La tasca de monitorització s'ha executat correctament i ha retornat un registre.

4.2.2.9 CU-B09 – Afegir REGISTRE

S'afegeix al llistat de registres un nou registre generat com a resultat d'una tasca de monitorització

Escenari d'ús:	**Afegir registre**
Versió:	1.0
Actors:	Sensor de Servei
Identificació	CU-B09
Propòsit:	Afegeix un nou registre al node
Precondicions:	S'ha executat una nova tasca de monitorització i s'ha obtingut un registre resultant.
Seqüència normal:	P1) El node recupera un nou registre resultant de l'execució d'una tasca de monitorització. P2) El node executa el cas d'ús *CU-B10-Analitzar Registre.* P3) Si l'execució de CU-B10 no genera un enviament, el node afegeix el registre al llistat de registres del node.
Excepcions:	E1) L'execució del cas d'ús *CU-B10-Analitza Registre* ha produït una nou enviament de registre i per tant no el registre no s'ha afegit al llistat de registres del node.
Post condicions:	S'ha afegit un nou registre al llistat de registres.

4.2.2.10 CU-B10 – Analitzar REGISTRE

S'analitza un registre per tal de determinar si cal reportat el registre de forma immediat a la Consola de Monitorització.

Escenari d'ús:	**Analitzar registre**
Versió:	1.0
Actors:	Sistema
Identificació	CU-B10
Propòsit:	Analitza un registre
Precondicions:	S'ha introduït un nou registre al sistema
Seqüència normal:	P1) El node analitza el registre introduït al sistema, comparant els seus atributs amb els atributs de notificació del sensors de servei vinculat al registre.

	P2) Si l'atribut de notificació indica enviament immeditat, el node crida el cas d'us *CU-B12-Enviar Registre* i reporta de forma immediata a la Consola de Monitorització el registre.
Excepcions:	
Post condicions:	S'ha analitzat un registre per determinar si es necessari reportar-lo de forma immediata a la Consola de Monitorització.

4.2.2.11 CU-B11 – Llistar REGISTRES

Llista els registres emmagatzemats per un node

Escenari d'ús:	**Llistar REGISTRES**
Versió:	1.0
Actors:	Sistema
Identificació	CU-B11
Propòsit:	Llista els registres emmagatzemats per un node
Precondicions:	El node disposa d'un o més registres de monitorització
Seqüència normal:	P1) Recorre el llistat de registres de monitorització i retorna el seus atributs.
Excepcions:	E1) El node no disposa de cap registre, retorna un llista buida.
Post condicions:	S'ha retornat el llistat de registres de monitorització que conté el node

4.2.2.12 CU-B12 – Enviar REGISTRES

El node envia a la Consola de Monitorització un o més registres

Escenari d'ús:	**Llistar REGISTRES**
Versió:	1.0
Actors:	Temps
Identificació	CU-B12

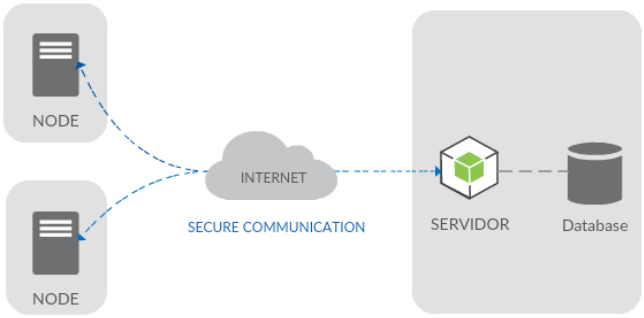
Propòsit:	Enviar registres de monitorització a la Consola de Monitorització
Precondicions:	El node disposa d'un o més registres de monitorització S'ha generat un enviament de notificació S'ha arribat a l'interval *d'enviament de registres*
Seqüència normal:	P1) S'ha arribat a l'interval *d'enviament de registres*, el node executa el cas d'ús *CU-B11-Llistar Registres* i recupera el llistat de registres del node. P2) El node enviar a la Consola de Monitorització el llistat de registres recuperat. P3) El node espera la resposta de la Consola de Monitorització. P3a) Si la Consola notifica la correcta recepció de les dades, el node buida el llistat de registres. P3b) Si la Consola no pot rebre les dades o no respon, es torna al pas P2
Seqüència alternativa:	P1) S'ha sol·licitat l'enviament immediat d'un registre a la Consola de Monitorització P2) El node enviar a la Consola de Monitorització de registre que ha generat l'enviament immediat. P3) El node espera la resposta de la Consola de Monitorització. P3a) Si la Consola notifica la correcta recepció de les dades, el node elimina el registre i continua l'execució. P3b) Si la Consola no pot rebre les dades o no respon, es torna al pas P2
Excepcions:	E1) La Consola de Monitorització no respon – Mostra missatge error connexió. E2) La Consola de Monitorització no accepta la identificació del node – Mostra missatge d'error i torna a cas d'ús CU-BO1
Post condicions:	S'ha produït l'enviament dels registres i s'ha buidat la llista de registres del node.

5. Disseny del sistema

L'objectiu d'aquest apartat és definir el funcionament general del sistema de monitorització en el seu conjunt, i de cada un dels components que l'integren en particular. Sense entrar en detall sobre els aspectes d'implementació tecnològica, es descriuen les entitats que integren el sistema, així com les relacions que s'hi estableixen.

5.1 Disseny arquitectònic del sistema

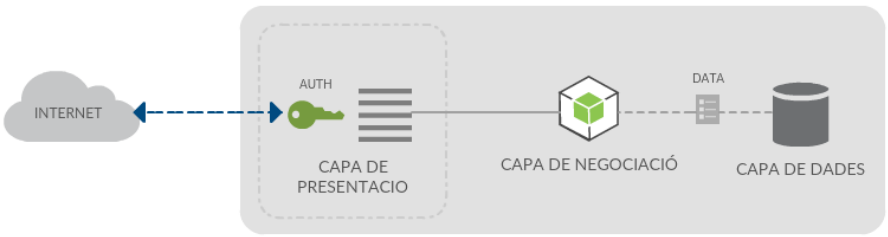
Una de les principals característiques del sistema de monitorització és la dispersió dels sistemes client a monitoritzar, per aquest motiu s'ha optat per una arquitectura client-servidor a la qual un component servidor aglutina les tasques de gestió i persistència de dades, alhora que un component client s'encarreguen d'executar les tasques de monitorització a cada un dels sistemes client i reportar-ne els resultats al component servidor.



Al diagrama anterior es pot observar que un altre aspecte força important a considerar alhora de dissenyar el sistema és la seguretat i protecció de la integritat del conjunt. La comunicació entre els components client-servidor es realitza a través d'internet, i per tant, s'han de tenir en compte tots els mecanismes de seguretat necessaris per garantir la seguretat i la protecció dels accessos al sistema i les comunicacions.

5.1.1 Disseny arquitectònic del component servidor

El disseny arquitectònic del component servidor es basa en una model d'arquitectura de tres capes. Aquest model es fonamenta en al separació lògica del processos interns d'una aplicació client-servidor agrupant-los en tres capes diferenciades: una capa de presentació responsable de la interacció amb l'usuari o amb altres processos externs al sistema, una capa de negoci responsable de rebre les peticions de la capa de presentació, processar-les i obtenir les dades resultants a partir de la tercera capa del model, la capa de dades, encarregada de la lògica de dades, la seva gestió i emmagatzemament.



A continuació es descriuen les funcionalitats específiques de cada una d'aquestes capes a l'arquitectura component servidor:

Capa de presentació: Tradicionalment és la capa responsable de la interacció amb els usuaris, i dins l'àmbit de component servidor , també és la responsable de la interacció amb els nodes client. Dins l'arquitectura del servidor, la capa de presentació és defineix amb una API (Application programming interface) que recull les peticions dels nodes client, les filtra i genera les consultes corresponents que transmet a la capa de negoci. També es responsable de recollir les dades resultants de la capa de negoci, donar-los format i transmetent-les als nodes client.
La capa de presentació és la capa més externa del sistema i l'única a la qual els usuaris i processos externs tenen accés, per aquest motiu la capa de presentació integra també totes les funcionalitats d'autentificació i validació d'usuaris.

Capa de negoci: La capa de negoci integra les principals funcionalitats de processament del component servidor. Es comunica amb la capa de presentació d'on rep les

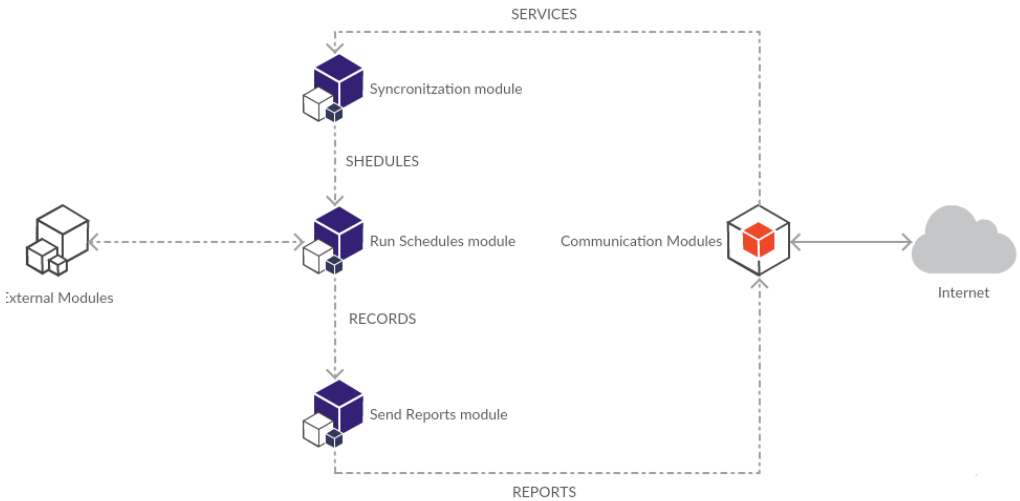
sol·licituds realitzades pels nodes client, les gestiona i realitza les operacions corresponents, comunicant-se amb la capa de dades per entregar o obtenir les dades necessàries.

Capa de dades: És la capa encarregada de la gestió de les dades i de la seva persistència. Dins l'arquitectura del servidor aquesta capa s'encarrega de validar les dades proporcionades per la capa de negoci i emmagatzemar-les a la base de dades. També s'encarrega de realitzar les consultes a la base de dades i retornar els resultats a la capa de negoci.

5.1.2 Disseny arquitectònic del component client

Per al disseny del component client s'ha escollit un model de descomposició modular. En aquest model arquitectònic es divideix el sistema en una sèrie de mòduls diferenciats segons la funcionalitat que implementen, i s'hi defineix per a cada un d'ells una sèrie d'interfases de descriuen les relacions amb la resta de mòduls del sistema.

Al diagrama següent es pot observar la descomposició modular del sistema client:



De l'anàlisi de requisits es desprèn que les funcionalitats principals de l'aplicació client són la comunicació amb el servidor per a l'obtenció de la configuració dels serveis a monitoritzar, la execució de les tasques de monitorització corresponents a cada servei, i l'enviament al servidor del registres obtinguts de l'execució de les tasques de monitorització.

A partir d'aquesta descomposició funcional del sistema client, podem definir quatre mòduls diferenciats així com les seves corresponents interfases:

Mòdul de Comunicacions *(Communication Module)*	
Funcionalitat	Responsable de les comunicacions amb el servidor, la autenticació del client i l'intercanvi de dades
Interfases	Proporciona SERVEIS al mòdul de Sincronització Obté REPORTS del mòdul d'enviament de registres

Mòdul de Sincronització *(Syncronitzation Module)*	
Funcionalitat	Responsable d'obtenir els serveis a monitoritzar i generar les tasques de monitorització corresponents.
Interfases	Obté SERVEIS del mòdul de Comunicació Proporciona SCHEDULES al mòdul d'execució de tasques

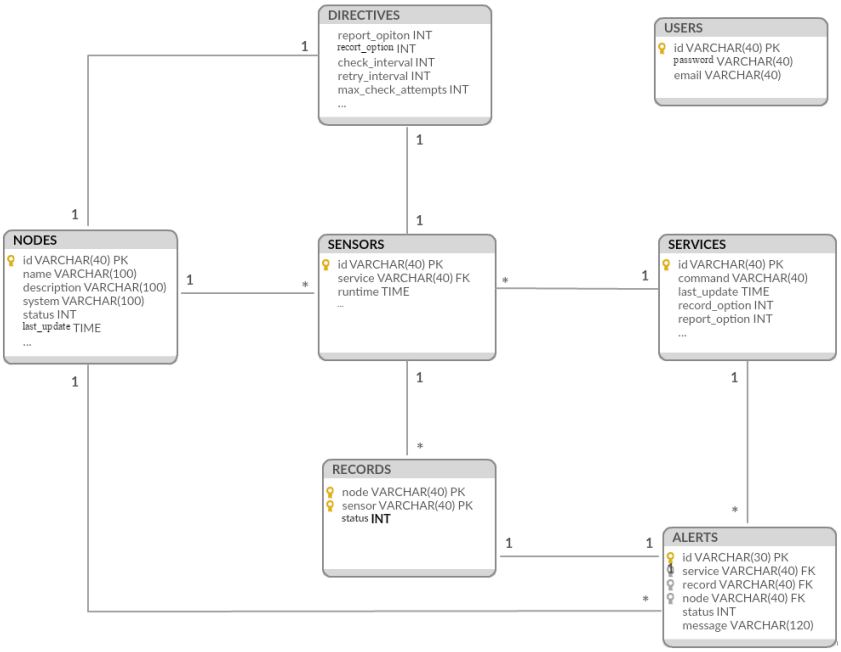
Mòdul d'execució de Tasques *(Run Schedules Module)*	
Funcionalitat	Responsable d'executar les tasques de monitorització segons les seves especificacions
Interfases	Obté SCHEDULES del mòdul de sincronització Proporciona RECORDS al mòdul d'enviament de registres.

Mòdul d'enviament de registres *(Send Reports Module)*	
Funcionalitat	Responsable de l'enviament dels registres resultants de les tasques de monitorització al servidor.
Interfases	Obté RECORDS del mòdul d'execució de tasques. Proporciona REPORTS al mòdul de comunicacions

5.2 Disseny del model de dades

En aquest apartat es defineix el disseny del model de dades a partir dels requisits recollits durant l'anàlisi del sistema.

5.2.1 Model de dades del component servidor



USERS: Un *user* és la representació d'un usuari amb permisos per accedir a l'aplicació. S'identifica mitjançat un correu electrònic i una contrasenya.

DIRECTIVES: Un objecte *directive*, representa un conjunt de directives utilitzades per determinar el comportament d'una tasca de monitorització. A cada sensor de monitorització se li assigna un conjunt de directives, alhora, cada node disposa també d'un conjunt de directives per defecte.

SERVICES: Un *service* és la descripció general d'un servei a monitoritzar. Els serveis es defineixen per la tasca que executen.

NODES: Un *node* representa un sistema client. Cada cop que s'instal·la en un nou sistema l'aplicació client cal registrar-la al servidor, generant-se un nou registre a la base de dades i proporcionant-li un conjunt de directives per defecte. A un node se li

poden assignar tants serveis de monitorització com desitgem, cada una d'aquestes assignacions es defineix com un sensor. Cada un dels sensors disposa d'un estat, el conjunt dels estats dels sensors assignats a un node determina l'estat final del node. Per exemple, la presència d'un sensor en estat CRITIC provocarà que el node també tingui un estat crític.
L'aplicació segueix el model d'estats definit per NAGIOS basat en quatre estats bàsics, SUCCESS, CRITICAL, WARNING i UNKNOWN.

SENSORS: Un *sensor* és el resultat d'un relació entre un node, un servei i un conjunt de directives. Quan s'assigna un servei a un node, també cal proporcionar-li un conjunt de directives que determinaran el comportament de la tasca de monitorització i la forma en que es reporten els registres al servidor, l'associació resultant queda recollida en un sensor. Cada sensor disposa d'un camp on es defineix l'estat del sensor. Aquest camp vindrà determinat per l'estat del darrer registre rebut.

Cada sensor disposen d'una marca de temps que permet identificar el darrer cop que es va actualitzar, els node client utilitzaran aquestes marques de temps per identificar el serveis que han d'actualitzar.

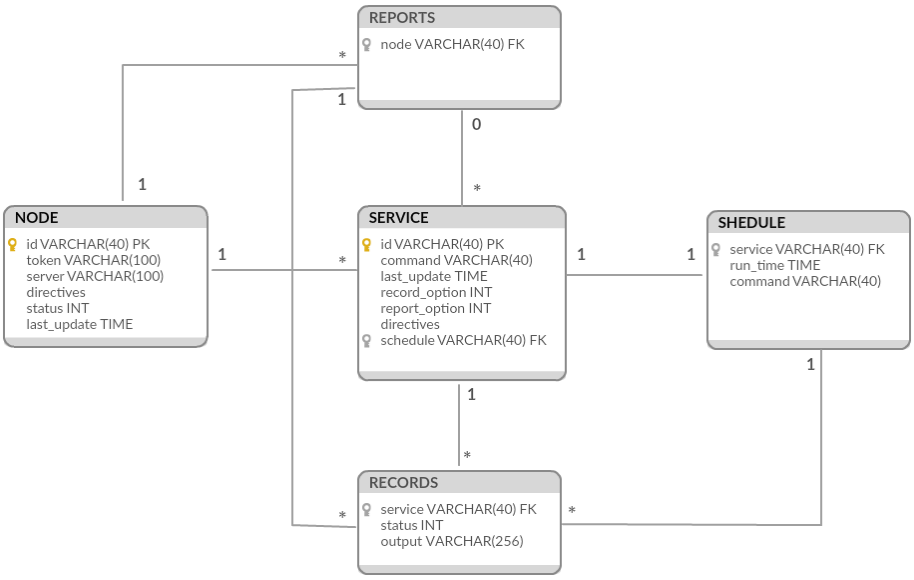
RECORDS: Un r*ecord* representa un registre i és el resultat de l'execució d'una tasca de monitorització en un determinat moment. Conté les dades resultants, així com el seu estat i l'hora d'execució. El component servidor s'encarrega de rebre els registres proporcionats pels nodes client i salvaguardar-los a la base de dades. També n'analitza els seus estats per determinar quan cal generar un alerta o notificació.

Els records contenen dues marques de temps, una que indica l'instant en que es va generar el registre, i un altre, anomenat temps de vida, que utilitzarà al servidor per determinar quan de temps cal emmagatzemar el registre a la base de dades. Un cop excedit el temps de vida, el registre s'elimina de la base de dades.

ALERTS: Representa un alerta de monitorització, les alertes es generen a partir de l'anàlisi dels registres obtinguts del client i en base a les directives especificades per a cada sensor. Les alertes s'emmagatzemen a la base de dades per mantenir un registre del incidents detectats per cada sensor.

5.2.2 Model de dades del component client

A la figura següent es pot veure l' esquema del model de dades de l'aplicació client:



NODE: Conté les dades relacionades amb el node client com ara el seu identificador, el seu token, el seu estat o la darrera vegada que es va actualitzar. També conté una relació les directives que s'aplicaran per defecte al serveis.

SERVICE: Fa referència a un servei a monitoritzar. En un node es poden monitoritzar un únic servei o múltiples serveis i per a cada un dels serveis es genera una tasca de monitorització **(Schedule)**.

SCHEDULE: Representa una tasca de monitorització. Cada servei té la seva corresponent tasca de monitorització que es defineix per la tasca que ha d'executar així com una marca de temps que determina quan s'ha d'executar. Cada cop que una tasca de monitorització s'executa, es genera un nou registre (**record**)

RECORD: Un registre és el resultat de l'execució d'una tasca de monitorització en un determinat moment. Conté les dades resultants, així com el seu estat i l'hora d'execució.

REPORT: Representa un enviament de dades al servidor. Els enviaments de dades es realitzen periòdicament i contenen per a cada servei monitoritzat el conjunt de registres que s'han obtingut durant aquell període. El report també proporciona al servidor dades de l'estat del node.

6. Implementació

6.1 Component Servidor

L'aplicació servidor, identificada amb el nom bnode-core, és la responsable de la definició i gestió del conjunt d'elements del sistema, així com de la gestió de les dades i la seva persistència. El servidor proporciona als nodes clients la configuració dels serveis que han de monitoritzar i recopila de cada un d'ells els registres obtinguts, emmagatzemant-los a la base de dades. A partir de l'anàlisi d'aquests registres, el servidor genera les alertes i notificacions que corresponguin.

6.1.1 Relació de components i llibreries de l'aplicació servidor

bnode-core	
Author:	Copyright (c) 2016-2017 Raul Martínez Díaz
License:	GNU General Public License (GPLv3) https://www.gnu.org/licenses/gpl-3.0.html
Requieres	**Node.js, Express.js** **Node Modules: moongosejs, JsonWebTokenjs, bcrypt-nodejs, pathjs**
Description:	La aplicació bnode-core constitueix el component servidor del projecte. Està desenvolupat en llenguatge Javascript mitjançant el framework Express.js de Node.js.

bnode-management-console	
Author:	Copyright (c) 2016-2017 Raul Martínez Díaz
License:	GNU General Public License (GPLv3) https://www.gnu.org/licenses/gpl-3.0.html
Requieres:	**Angular.js , bootstrap.js, jquery.js**
Description:	La Consola d'Administració és una aplicació independent desenvolupada en Anjular.js que permet gestionar el servidor mitjançant crides a la API .

Components i llibreries desenvolupats per tercers utilitzats a la implementació de l'aplicació servidor:

Node.js – Javascript Development Platform	
Copyright:	Copyright (c) 2016 Node.js Foundation Portions originally © 2016 Joyent https://nodejs.org/es/foundation/
Version:	6.9.2
License	https://raw.githubusercontent.com/nodejs/node/master/LICENSE
Description:	Node.js es una solució de codi obert que proporciona un entorn de desenvolupament basat en llenguatge ECMAScript. NodeJs disposa d'una sèrie de mòduls bàsics i d'una amplia col·lecció de mòduls am b funcionalitats addicionals elaborats per la comunitat. Al present projecte es fa ús d'alguns d'aquests mòduls per a la implementació de funcionalitats avançades: *moongosejs, JsonWebTokenjs, bcrypt-nodejs, pathjs*

Express.js - Fast, unopinionated, minimalist web framework	
Copyright:	Copyright (c) 2009-2014 TJ Holowaychuk <tj@vision-media.ca> Copyright (c) 2013-2014 Roman Shtylman <shtylman+expressjs@gmail.com> Copyright (c) 2014-2015 Douglas Christopher Wilson <doug@somethingdoug.com>
Version:	4.13.4
License	The MIT License https://github.com/expressjs/express/blob/master/LICENSE
Description:	Express és un framework per Nodejs que proporciona una infraestructura per al desenvolupament d'aplicacions web Node.js mínima i flexible.

6.1.2 Arquitectura general del component servidor

El component servidor s'ha implementat mitjançant una API REST que permet les comunicacions amb els nodes client i els usuaris. Aquesta API s'ha desenvolupat mitjançant la plataforma d'execució NodeJS i el framework ExpressJs. Per a la persistència de dades s'ha utilitzat com a gestor de dades MongoDB.

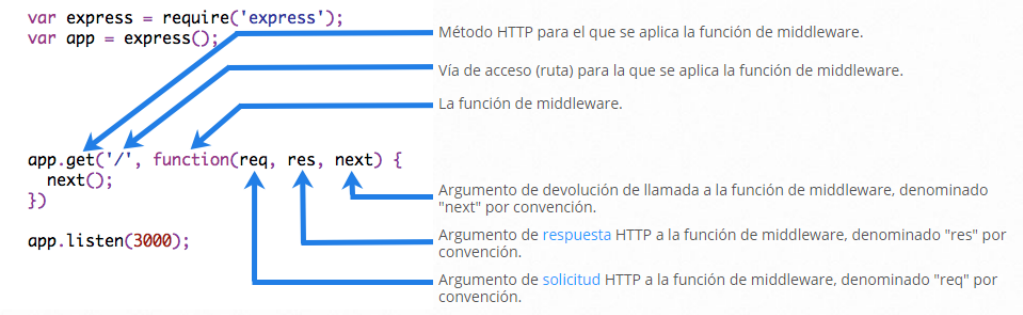
La consola de monitorització web s'ha desenvolupat en AngularJs com un component addicional que permet, mitjançant crides a la API, la gestió i administració del servidor.

6.1.3 API REST

La funcionalitat principal del component servidor és la centralitzar la gestió de totes les tasques de monitorització dels nodes client, permetent assignar i configurar els serveis a monitoritzar per cada un dels node client , així com recopilar els resultats de cada una de tasques de monitorització i dipositar-los en una base de dades pel seu anàlisi. Per poder realitzar aquest intercanvi d'informació entre aplicació client i servidor s'ha dissenyat i implementat una API REST. Mitjançant crides a la API els nodes client obtenen la configuració dels serveis que han de monitoritzar, la API també els permet entregar al servidor els resultats d'aquestes monitoritzacions pel seu emmagatzemament i anàlisi.

Per al desenvolupament de la API REST del servidor s'ha utilitzat el framework ExpressJS (http://expressjs.com/es/) . Aquest framework permet implementar de forma ràpida una infraestructura d'aplicacions web basada en Node.JS mínima i flexible. Incorpora funcionalitats per gestionar les peticions web i enrutar-les dins la nostra aplicació en funció el mètode utilitzat (GET, POST, PUT...).

Un altra característica destacable d'aquest framework és que ofereix la possibilitat d'implementar funcions *middleware*. Aquestes funcions tenen accés a la sol·licitud realitzada (request) i a la resposta(response), així com a la següent funció de la pila. Per exemple, al present projecte s'utilitzen funcions *middleware* per implementar un mecanisme d'autenticació basat en tokens JWT. Aquest mecanisme protegeix les rutes de la API dels accessos no autoritzats.



A la taula següent es detallen els recursos implementats a la API del servidor, així com els mètodes que accepta cada un d'ells:

API – Rutes de gestió /api		
/api/authenticate	POST	Permet identificar un usuari al servidor i obtenir un token d'usuari Requereix usuari i contrasenya Retorna *token* d'usuari
/api/templates	GET	Retorna col·lició de *templates*
/api/templates	POST	Afegeix un nou *template* Requereix objecte *template* en format JSON
/api/templates/:name	GET	Retorn *template* segons el nom donat
/api/services	GET	Retorna col·lició de *services*
/api/services	POST	Afegeig un nou *service* Requereix objecte *service* en format JSON
/api/services/:id	GET	Retorna *service* segons l'identificador donat
/api/services/:id	PUT	Actualitza un *service* segons l'identificador donat Requereix objecte *service* en format JSON Retrona el *service* actualitzat
/api/nodes	GET	Retorna col·lició de *nodes*
/api/nodes/:id	GET	Retorna *node* segons l'identificador donat
/api/nodes/:id	PUT	Actualitza un *node* segons l'identificador donat Requereix objecte *node* en format JSON Retorna el *node* actualitzat
/api/nodes/:id/sensors	POST	Permet afegir un o més sensors al node especificat . Requereix col·lecció de *sensors* en format JSON
/api/records	GET	Retorna col·lició de records
/api/recods/:id	GET	Retorna record segons l'identificador donat

API – Rutes Nodes Client /api/client		
/api/client/	POST	Permet registrar un nou node al servidor. node. Retorna TOKEN de node client
/api/client/:id	GET	Recupera la configuració del node i de tots els sensors que té assignats.
/api/client/:id/heardbeat	GET	Permet verificar que client i servidor estan actius. Retorna la data de la darrera modificació a la configuració del node en format JSON
/api/client/:id/report	POST	Entrega un report al servidor Requereix col·lecció de *records* en format JSON

```
/*
 * Define API Management Routes
 * Protected with User Tokens
 */
app.use('/api', api_router);

/*
 * Define API Client Routes
 * Protected with Node Client Tokens
 */
app.use('/api', api_client_router);
```

Com es pot apreciar al quadre anterior, s'ha diferenciat entre rutes de gestió: aquelles que permeten realitzar tasques de gestió del servidor com afegir i modificar serveis, assignar serveis als nodes, veure els registres... i rutes de client: que són aquelles vinculades a l'activitat dels nodes client. Aquesta diferenciació està motivada, d'una banda pel mecanisme d'autenticació implementat en cada una d'elles, i d'altra banda per permetre la protecció de les rutes de gestió mitjançant filtres IP. Ambdós mecanismes es desenvolupen amb detall als apartats següents.

6.1.4 Autenticació d'usuaris i nodes client

Per permetre als nodes remots interactuar amb el servidor, cal publicar la API a internet. Tot i que per a les comunicacions entre els nodes client i el servidor s'utilitza el protocol HTTPS, que implementa xifratge basat en SSL/TLS, aquest mecanisme només garanteix la seguretat de les comunicacions però no protegeix d'accessos no autoritzats al sistema. Per tal de restringir l'accés al sistema, permetent únicament l'accés a usuaris i clients autentificats, s'implementa al servidor un mecanisme d'autentificació basat en JSON Web Token (JWT).

JSON Web Token (JWT) és un estàndard obert (RFC 7519) que s'utilitza per l'enviament d'informació segura que pot ser verificada. Aquest estàndard defineix JWT com:

> *"JSON Web Token (JWT) is a compact, URL-safe means of representing claims to be transferred between two parties. The claims in a JWT are encoded as a JSON object that is used as the payload of a JSON Web Signature (JWS) structure or as the plaintext of a JSON Web Encryption (JWE) structure, enabling the claims to be digitally signed or integrity protected with a Message Authentication Code (MAC) and/or encrypted."*
>
> Extract: http://self-issued.info/docs/draft-ietf-oauth-json-web-token.html

Per a la implementació de JSON Web Token (JWT) al component servidor s'ha utilitzat el mòdul jsonwebtoken (https://github.com/auth0/node-jsonwebtoken). Aquest mòdul distribuït amb llicència MIT i desenvolupat per l'empresa especialitzada Auth0 (https://auth0.com) implementa gran part dels requisits i funcionalitats especificats per l'estàndard RFC 7519.

Al component servidor s'implementen dos tipus de tokens per a la identificació d'usuaris i nodes client. Ambdós tipologies mantenen la mateixa estructura però hi ha diferències sensibles en la seva gestió motivades pel mecanisme d'autenticació utilitzat en cada un d'ells.

```
/*
 * Define API Management Routes
 * Protected with User Tokens
 */
app.use('/api', api_router);

/*
 * Define API Client Routes
 * Protected with Node Client Tokens
 */
app.use('/api', api_client_router);
```

En primer lloc els tokens d'usuari s'utilitzen per a la identificació dels usuaris a la API, aquest tokens es caracteritzen per tindre una data de caducitat curta (24 hores) i per que no s'emmagatzemen en cap moment al servidor. Quan un usuari s'autentifica mitjançant usuari i contrasenya se li proporciona un token d'usuari vàlid durant 24 hores. Les rutes protegides mitjançant aquests tokens únicament comproven si el token és vàlid i és vigent. Aquest mecanisme permet que les consultes a la API del servidor siguin molt més àgils ja que no es requereix de comprovacions addicionals la base de dades.

En segon lloc els tokens de node client s'utilitzen per a la identificació dels nodes client a la API i es caracteritzen perquè a diferència del tokens d'usuari, no tenen data de caducitat. Quan es registra un nou node al servidor s'obté un token d'identificació que el node emmagatzema en un fitxer local per utilitzar-lo en totes les crides al servidor, aquest token es vigent durant tot el cicle de vida del node client. La utilització de tokens sense data de caducitat pot representar un problema de seguretat si el node client es veu compromès i un atacant obté el token emmagatzemat al fitxer. Per aquest motiu el servidor implementa un mecanisme addicional per a la protecció dels tokens de node client, aquesta protecció consisteix en mantenir una taula a la base de dades amb els tokens clients generats i el node al qual van ser assignats, de manera no només es

comprova si el token és vàlid, sinó que també es consulta a la base de dades si està actiu o ha estat bloquejat.

Abans de disposar d'un token vàlid cal autenticar-se al servidor. En el cas dels usuaris aquesta autentificació es produeix mitjançant una crida a /api/authenticate proporcionant usuari registrat i la corresponent contrasenya. En el cas dels nodes client aquesta autenticació només s'ha de realitzar un cop, durant el registre del node des de la aplicació client, consisteix en una crida a */api/clients* proporcionant usuari registrat amb permisos per afegir nodes i la corresponent contrasenya. En ambdós casos si l'autenticació es satisfactòria es retorna el corresponent token.

/api/client/	POST	Permet registrar un nou node al servidor retornant un token de node client.
/api/authenticate	POST	Permet autenticar un usuari al servidor retornant un token d'usuari

La següent captura mostra el middleware de Express.js que permet autentificar els accessos a la API dels usuari mitjançant l'ús de tokens d'usuari:

```
  router.route('/records/:id')

      /*  GET Record by identificator */
      .get( isloggedIn, function(req, res, next) {
          ...
      })
```

```
function isloggedIn(req, res, next) {

    /* Check header for Token */
    var token = req.body.token || req.headers['x-auth-token'];

    /* Decode Token */
    if (token) {

        /* verifies secret and checks exp */
        jwt.verify(token, config.app_secret_users , function(err, decoded) {
            if (err) {
                return res.json({ success: false, message: 'Failed to authenticate token.'
                });
            } else {
                req.decoded = decoded;
                next();
            }
        });

    } else {

        /* No token return error */
        return res.status(403).send({
            success: false,
            message: 'No token provided.'
        });
    }

};
```

6.1.5 Consideracions addicionals de seguretat

Per permetre l'accés dels nodes remots cal que publiquem la API a internet. Per defecte l'aplicació s'executa al port 443 i utilitza un certificat auto signat. En un entorn de producció cal considera l'ús d'un certificat validat per una entitat de certificació comercial per tal de garantir als nodes client l'origen de les comunicacions.

```
/* Use Express framework to publish app on defined port */
https.createServer({
      key: fs.readFileSync('bin/server.key'),
      cert: fs.readFileSync('bin/server.crt'),
      passphrase: key_passphrase
}, app).listen(app_port);
```

Un altre millora de seguretat a destacar és la possibilitat de restringir l'accés a determinades rutes de la API. El servei s'ha dissenyat de manera que es pugui restringir l'accés a determinades rutes de la API en funció del origen de les peticions, així un cop publicada a internet, només certes funcionalitats seran accessibles externament. La recomanació és publicar únicament les rutes necessàries per a la comunicació amb els nodes client (*app_client_router*) i limitar o restringir l'accés a les rutes dissenyades per a la gestió i administració del servidor (*api_router*)

```
  /* Define API Routes */
  app.use('/api', api_router);
  app.use('/api', api_client_router);
```

Mitjançant l'ús del mòdul *express-ipfiler* podem protegir l'accés extern a certes rutes de la API. A la figura següent es mostra una implementació on s'ha utilitzat aquest mòdul per limitar l'accés extern a les rutes de gestió, permetent únicament l'accés des de la xarxa local. Per contra, l'accés a les rutes client no té cap restricció, permetent l'accés il·limitat dels nodes client, tant des de la xarxa local com des de ubicacions externes.

```
var express = require('express'),
ipfilter = require('express-ipfilter').IpFilter;

/* Whitelist the following IPs */
var ips = ['127.0.0.1', '192.168.1.1/24'];

/* Define API Routes with filter protection */
app.use('/api', ipfilter(ips, {mode: 'allow'}), api_router);
app.use('/api', api_client_router);
```

6.1.6 Consola d'Administració Web

La consola d'administració web és un component addicional desenvolupat per oferir a l'usuari una interfícies gràfica (GUI) per a la gestió i administració del servidor . Consisteix en una aplicació front-end SPA (Single-Page Application) desenvolupada mitjançant el framework AngularJS. Aquest framework de Javascript permet el desenvolupament àgil d'aplicacions web front-end basades en el model MVC (Model-vista-controlador).

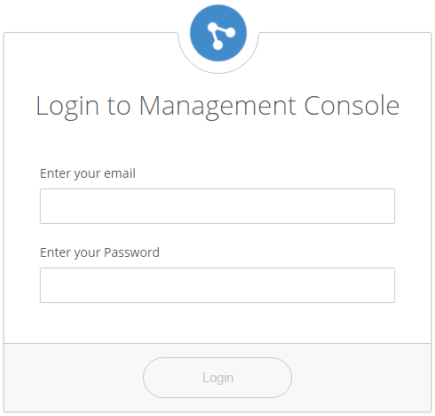
La consola d'administració web permet, mitjançant la API del servidor, gestionar els nodes, crear i assignar serveis o visualitzar els registres. És dins el projecte, el component en el qual menys s'ha treballat limitant-se el seu desenvolupament a les funcionalitats més bàsiques. No obstant això, és el component que en treballs futurs prendrà més rellevància per les possibilitats que ofereix, especialment alhora de presentar a l'usuari les dades obtingudes dels nodes client.

No és objecte d'aquesta memòria final analitzar detalladament la implementació de l'arquitectura MVC (Model-vista-controllador) desenvolupada a la consola web, però si exposar breument algunes de les característiques més destacades.

La consola d'administració web és una aplicació front-end independent que permet administrar i gestionar el servidor mitjançant crides a la API. A la figura següent es pot apreciar part del servei de comunicacions implementat per a l'execució d'aquestes crides:

```
bnodeAPI.getService = function(id) {
  return $http({
    method: 'GET',
    url: config.api + '/api/services/' + id,
    headers: {'x-auth-token': $window.localStorage['auth-token']}
  });
}

bnodeAPI.addService = function(service) {
  return $http({
    method: 'POST',
    data: service,
    url: config.api + '/api/services',
    headers: {'x-auth-token': $window.localStorage['auth-token']}
  });
}
```

Per accedir a les funcions de la API es requereix d'un token d'identificació. S'ha proporcionat a la consola web d'un formulari de identificació que permet a l'usuari autenticar-se a la API i obtenir el corresponent token d'identificació. Un cop l'usuari ha estat validat, el token queda emmagatzemat al navegador fins que l'usuari finalitza la sessió o bé el token expira. A la consola web aquest token s'utilitza d'una banda per a proporcionar o limitar l'accés als diferents components de la consola i d'altra banda s'adjunta a totes les peticions adreçades a la API per tal d'autenticar l'usuari i permetre l'accés als recursos publicats.



Totes les rutes internes de la consola web estan protegides i cal disposar d'un token d'autenticació per poder-hi accedir, si l'usuari no disposa del corresponent token és redirigit al formulari d'autenticació:

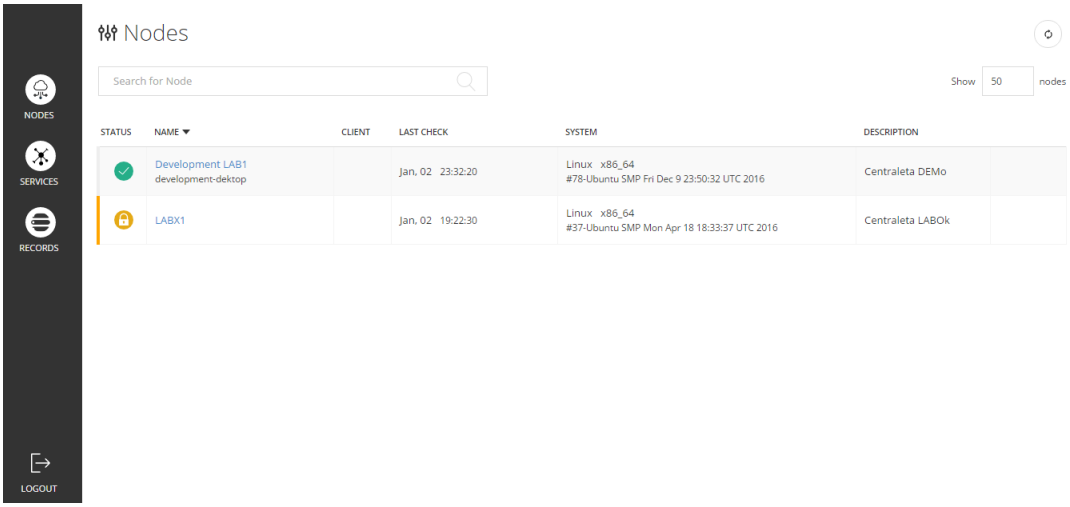
```
        .state('editservice', {
            url: '/app/services/edit/:id',
            templateUrl: 'app/components/services_edit.html',
            controller: 'editserviceController',
            authenticate: true
        })
```

```
$rootScope.$on('$stateChangeStart', function(event, toState, toParams){

      if (toState.authenticate && !authService.isLoggedIn()){

          /* User isn't authenticated */
          $window.location.href = 'login.html';
          event.preventDefault();
      }
});
```

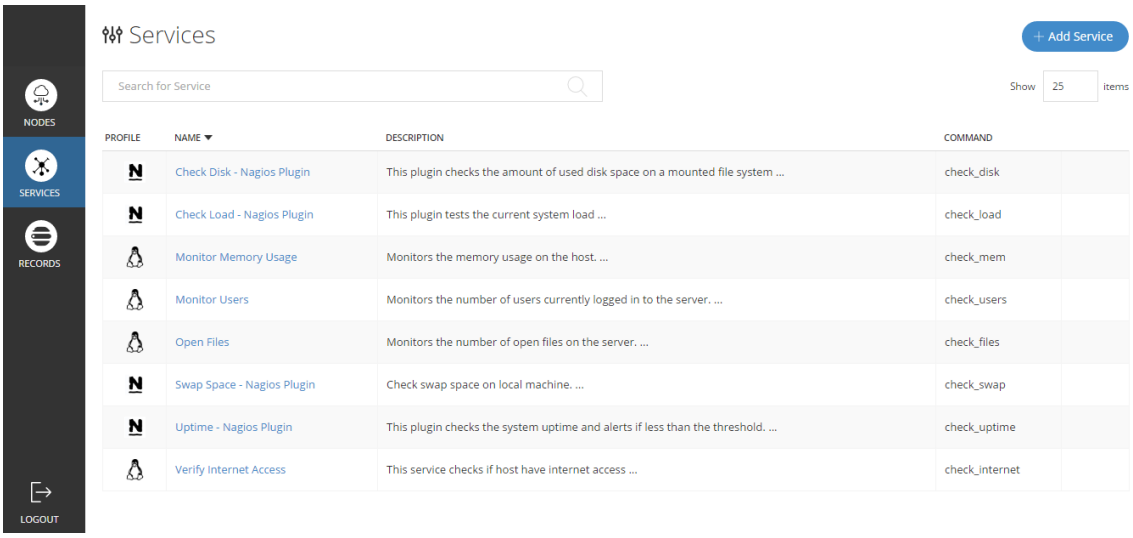
6.1.7 Captures de l'aplicació

En aquest sub apartat es mostren algunes captures de pantalla de l'execució del component servidor. Donat que el component servidor no disposa d'un entorn gràfic les captures que s'adjunten corresponen a l'execució de la Consola Web d'Administració.

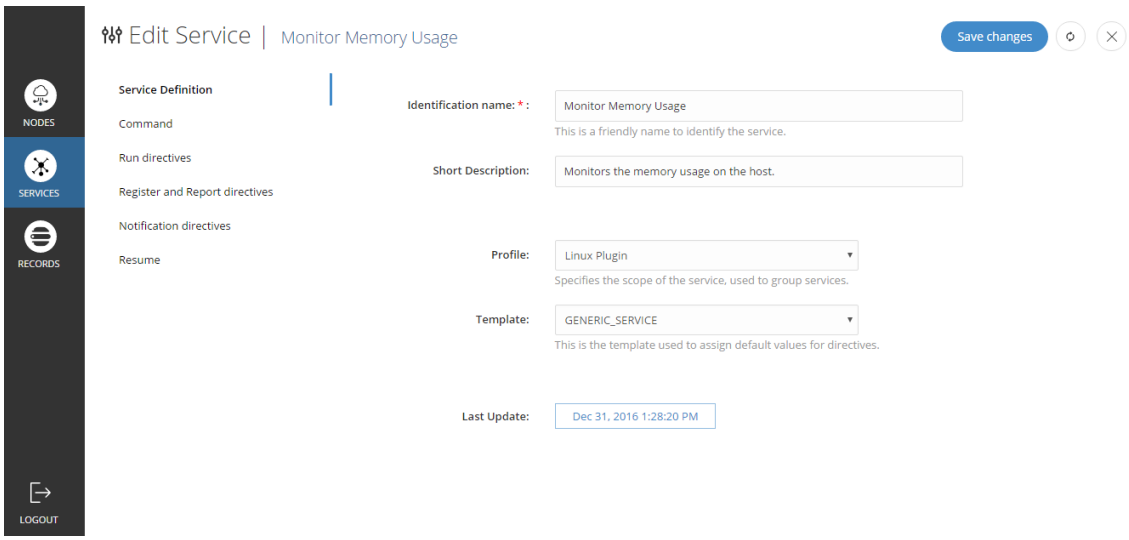
La següent captura mostra el llistat de nodes registrats al servidor. Al llistat de nodes s'hi mostra el nom i l'alies del node, el sistema, la descripció, així com la data de la darrera monitorització. També es mostra l'estat actual de cada un dels nodes:



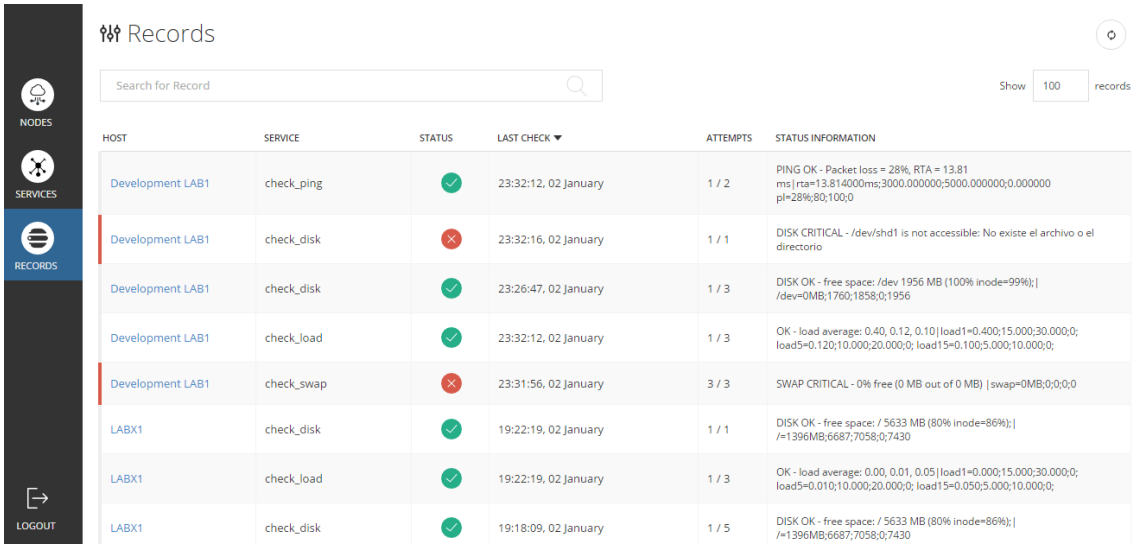
La següent captura mostra el llistat de serveis registrats al servidor. Actualment la consola de gestió web permet afegir nous serveis així com editar i modificar la configuració dels existents.



La següent captura mostra el formulari d'edició d'un servei. L'editor permet configurar la comanda que s'executara així com les diferents directives de monitorització del servei.



La darrera captura mostra els llistat de registres obtinguts a partir de les monitoritzacions definides per cada node. El llistat mostra per a cada un dels registres el node i es servei que l'han generat, la data en la qual es va produir així com el resultat de la monitorització i el seu missatge de sortida:

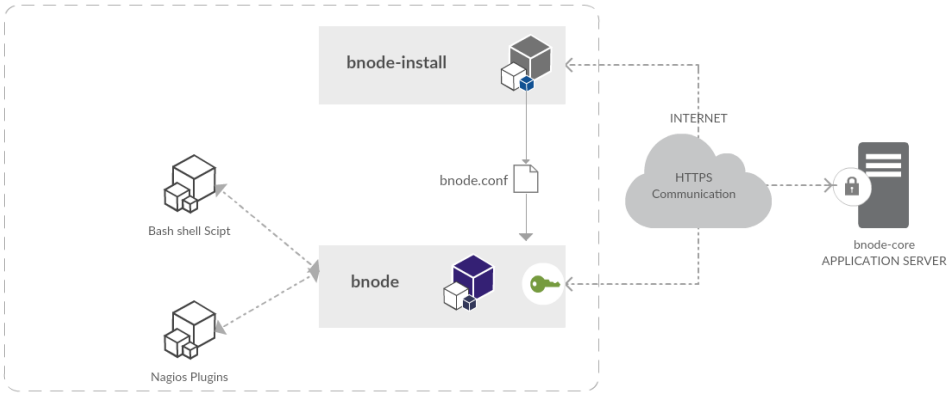


6.2 Components Client

L'aplicació client, identificada amb el nom de codi **bnode**, és la responsable d'executar als nodes client les tasques de monitorització definides al servidor i reportar-ne els resultats.

La seva implementació s'ha realitzat mitjançant el llenguatge de programació ANSI C (C89) a fi de garantir la portabilitat del codi entre diferents distribucions de GNU/LINUX. Consisteix en un servei o *daemon* que s'executa a cada node client. Paral·lelament, s'ha desenvolupar una petita eina de configuració, **bnode-conf**, responsable de realitzar el registre inicial dels nodes al servidor i l'obtenir el token d'identificació.

La aplicació client **bnode** no realitza directament cap tasca de monitorització, més enllà de recopilar algunes dades bàsiques del host, la seva feina consisteix a gestionar l'execució d'una series de tasques de monitorització externes i recollir-ne els resultats pel seu enviament al component servidor. Aquestes tasques externes poden ser utilitats del sistema, scripts Bach Shell, o plugins de Nagios. D'aquesta manera s'assoleix un dels requisits del projecte: la compatibilitat amb els plugins de Nagios existents.



La figura anterior mostra de manera esquemàtica els diferents elements de la solució client. Al propers apartats es descriuen detalladament cada un d'aquests elements i les solucions de desenvolupament que s'hi han implementat.

6.2.1 Relació de components i llibreries de l'aplicació client

La aplicació client resultant la integren dos utilitats, ambdós desenvolupades amb ANSI C (C89) a fi de garantir la portabilitat entre diferents sistemes.

D'una banda, s'ha desenvolupat la utilitat **bnode-conf** o utilitat de configuració, destinada a realitzar la configuració inicial del client i la obtenció del *token* d'autentificació. D'altra banda, la utilitat ***bnode***, que s'executa com a servei o *daemon* als nodes client i que es descriu en detall en aquest mateix document.

En aquest apartat es fa una relació d'aquests components, així com de les llibreries de tercers utilitzades en la seva implementació.

bnode	
Author:	Copyright (c) 2016-2017 Raul Martínez Díaz
License:	GNU General Public License (GPLv3) https://www.gnu.org/licenses/gpl-3.0.html
Uses	**Libpthreads, cJSON, curllib**
Description:	Constitueix el servei principal de l'aplicació client. S'executa com a servei o *daemon* als nodes client i les seves tasques són la comunicació amb el servidor i l'obtenció de la configuració dels serveis a monitoritzar, l'execució de les tasques de monitorització segons la configuració obtinguda, i l'enviament al servidor dels registres resultats de l'execució de les tasques.

bnode-conf	
Author:	Copyright (c) 2016-2017 Raul Martínez Díaz
License:	GNU General Public License (GPLv3) https://www.gnu.org/licenses/gpl-3.0.html
Package:	**cJSON, curllib**
Description:	Utilitat de configuració, permet realitzar el registre inicial del node al servidor, generar el fitxer de configuració **/etc/bnode/bnode.conf** i obtenir el token que permetrà al servei principal autenticar-se al servidor.

Components i llibreries desenvolupats per tercers utilitzats a la implementació de l'aplicació client:

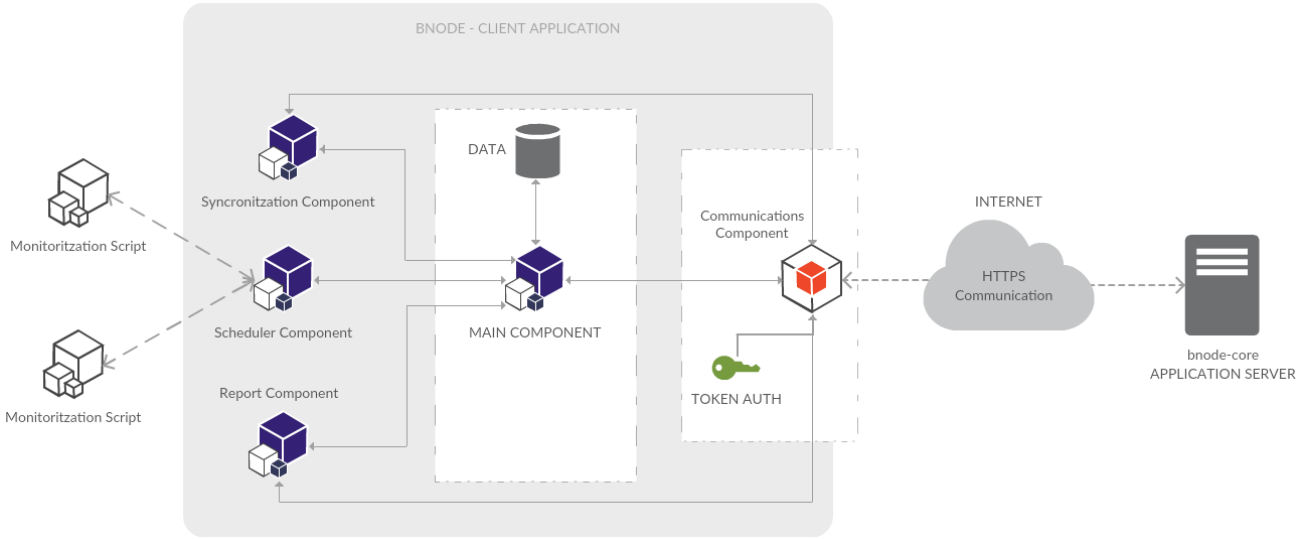
libpthreads - POSIX Threading Library	
Copyright:	Copyright (C) 2016 Free Software Foundation, Inc. https://www.gnu.org/software/hurd/libpthread.html
License	GNU Lesser General Public License https://www.gnu.org/software/libc/manual/html_node/Copying.html
Description:	Conté un conjunt de definicions i rutines que permeten la implementació de fils d'execució segons les definicions de l'estàndard POSIX.

cJSON Library - Ultralightweight JSON parser in ANSI C	
Copyright:	Copyright (c) 2009-2016 Dave Gamble https://github.com/DaveGamble/cJSON
License	MIT License https://github.com/DaveGamble/cJSON/#license
Description:	Aquesta llibreria conté un conjunt de definicions i rutines que permeten la conversió d'estructures de dades ANSI C en cadenes JSON, o viceversa.

Libcurl - the multiprotocol file transfer library	
Copyright:	Copyright (c) 1996 - 2016, Daniel Stenberg, daniel@haxx.se, and many contributors https://curl.haxx.se/libcurl/
License	MIT Style license - License Mixing https://curl.haxx.se/legal/licmix.html
Description:	Curllib és una llibreria que proporciona un extens conjunt de definicions i rutines, permetent el desenvolupament de clients de comunicació per a una ampli nombre de protocols, com ara: FTP, FTPS, HTTP, HTTPS, IMAP, LDAP, LDAPSSCP, SFTP, SMTP ...

6.2.2 Arquitectura general del component client

La figura següent descriu la arquitectura general de la solució client implementada, basada en la descomposició modular de l'aplicació:



- **Main Component:** és el component principal de l'arquitectura, responsable de carregar la configuració del node i generar les estructures de dades. Les seves funcionalitats es descriuen amb detall a l'apartat *6.2.6 Fils d'execució* del present document.

- **Communication Component:** aquest component és el responsable de les comunicacions amb el servidor, l'autenticació del node, i l'intercanvi de dades. Les seves funcionalitat es descriuen amb detall a l'apartat *6.2.4 Comunicació amb el servidor* del present document.

- **Syncronization Component**: aquest component és el responsable de la sincronització de serveis amb el servidor. El component de sincronització executa a intervals regulars de temps un rutina de batec. Aquesta rutina, anomenada internament *heartbeat,* consisteix en contactar amb la API del servidor indicant-li que el node està actiu. Com a resposta el servidor li proporciona una marca de temps que indica la darrera actualització a la configuració del node. Quan aquesta marca de temps no coincideix amb la que el node té emmagatzemada, el component de sincronització sap que s'ha produït una modificació de la configuració al servidor i llença una segona rutina,

backend_sync, que contacta novament amb la API del servidor i obté el llistat de serveis actualitzat, així com les noves directives del node.

A partir de les dades rebudes s' actualitzen les directives del node i cada un dels serveis. Cada servei incorpora la seva pròpia marca de temps, *last_update* utilitzada per determinar si cal actualitzar o no el servei.

Si el llistat de serveis actualitzat presenta un servei que no figura a llistat de serveis del client, aquest s'incorpora al llistats i es genera la corresponent tasca d'execució (*schedule*).

El *heartbea*t no només s'utilitza per determinar si s'ha produït una actualització de la configuració, sinó que també permet al servidor saber si el node està actiu. L'aplicació client defineix un interval de temps entre batec i batec de 10 minuts, tot i que, aquest valor es pot personalitzar per cada node mitjançant la directiva *interval_beat* del node.

- Schedule Component: Per a cada un dels serveis introduïts al sistema es genera una tasca de monitorització anomenada schedule. Aquest tasca es defineix per la marca de temps que indica l'instant en que s'ha de executar, així com per la comanda que cal processar. Cada una d'aquestes tasques s'insereixen en una llista de tasques, ordenada segons la marca de temps que en determina l'execució.

El component *schedule* és l'encarregat d'executar les tasques que figuren al llistat de tasques. El procés és el següent: s'agafa la primera tasca de la llista ordenada i es compara la seva marca de temps amb l'hora actual, si coincideixen o la marca de temps es inferior, el gestor sap que pot executar la tasca associada. Un cop completada la tasca s'actualitza la marca de temps segons l'interval d'execució del servei i s'afegeix de nou la tasca al llistat, ubicant-la en la posició que li correspon segons la nova marca de temps.

Per a l'execució de les tasques de monitorització, el component schedule genera un subprocés responsable de l'execució de la tasca. Per evitar que una tasca pugui bloquejar tota la cua de tasques, s'ha definit un temps màxim d'execució per als subprocesos de monitorització, quan un tasca supera aquest temps, el subprocés es cancel·la i es genera la corresponent alerta. El procés d'execució de les tasques de monitorització es descriu en detall a l'apartat 6.2.7 del present document.

A la figura següent es pot veure una captura de la funció principal d'aquest component:

```
/** \brief          Manages the execution of tasks
 *
 *  \param[in|out]  ptr A pointer to application data structures
 *  \return         Void.
 */
void *
schedule_manager (void *ptr)
{
    args                *p = (args *)ptr;

    schedule_list       *schedules = p->schedules;
    report_list         *reports = p->reports;

    time_t              current;

    /* Infinite Loop */
    for(;;) {

        /* Suspend thread execution for microseconds interval */
        usleep(NODE_PTH_SLEEP);

        if (schedules->head != NULL) {

            /* Check if schedule time is equal or less than now */
            current = time(NULL);
            if ( current >= schedules->head->runtime) {

                /* Run scheduled task */
                host_run_schedule(schedules, reports);
            }
        }
    }

    /* Ends thread execution */
    pthread_exit(NULL);
}
```

- Report component: Cada cop que s'executa una tasca de monitorització es genera un registre amb el resultar de la tasca. El component report és l'encarregat de recopilar aquestes registres i enviar-los al servidor.

La directiva *report_option* de cada servei determina com s'ha de gestionar l'enviament d'aquest registres al servidor:

report_option = 0: enviar segons directiva del host
report_option = 1 : enviar immediatament quan es produeixi un canvi d'estat
report_option = 2: enviar cada cop que es produeixi un nou registre

En els supòsits anteriors 1 i 2, cada nou registre genera un nou *report* i l'afegeix a la llista de reports. El component report comprova cada pocs segons aquest llistat i si hi ha reports pendents els agrupa i els envia al servidor.

Per al supòsit 0, l'enviament de registres es produeix segons la directiva *report_interval* del node. Aquest directiva indica que cada cert període de temps, per

defecte 20 minuts, el component report ha de recopilar tots els registres pendent d'enviar, agrupar-los i enviar-los al servidor en un únic report.

A la figura següent es pot veure una captura de la funció principal d'aquest component:

```
/** \brief          Manages reports to the server
 *
 *  \param[in|out]  ptr A pointer to application data structures
 *  \return         Void.
 */
void *
report_manager (void *ptr)
{
    args                *p = (args *)ptr;

    report_list         *reports = p->reports;
    service_list        *services = p->services;
    struct host         node = p->node;

    time_t              current;
    time_t              next;

    int                 interval = node.directives.report_interval;

    current = time(NULL);
    next = current + interval;

    for(;;) {

        /* Suspend thread execution for microseconds interval */
        usleep(NODE_PTH_SLEEP);

        /* Get current time */
        current = time(NULL);

        /* REPORTS: Directive report_option = 0 */
        if ( current >= next) {

            send_report_all(&node, services );
            next = time(NULL) + interval;
        }

        /* REPORTS: Directive report_option = 1 & 2 */
        if ( reports->head != NULL) {

            send_report (&node, reports);
        }
    }

    /* Ends thread execution */
    pthread_exit(NULL);
}
```

En aquest apartat s'ha definit de manera general el funcionament intern de la aplicació client, als següents apartats es detallen les solucions que s'han implementat per donar resposta als principals reptes del desenvolupament, com son la coherència del model de dades amb l'aplicació servidor, la comunicació amb el servidor i l'intercanvi de dades, l'autentificació de la aplicació client o l'ús de múltiples fils de procés.

6.2.3 Model de dades

Un dels principals reptes alhora de desenvolupar l'aplicació client ha estat traslladar el model de dades de l'aplicació servidor, basat en *col·leccions* i *documents*, al llenguatge C utilitzat en el desenvolupament del client, ja que aquest no incorpora de forma nativa aquest paradigma.

La solució implementada per a la definició d'un model de dades equivalent al model basat en *documents i col·leccions* ha estat la utilització d'estructures de dades *(structs)* per a la definició dels *documents* i llistes encadenades *(liked lists)* per a la definició de *col·leccions.* D'aquesta manera s'ha pogut mantenir la correlació entre el model de dades de la aplicació servidor i el model de dades de la aplicació client, agilitzant la transmissió i intercanvi de dades entre ambdós components.

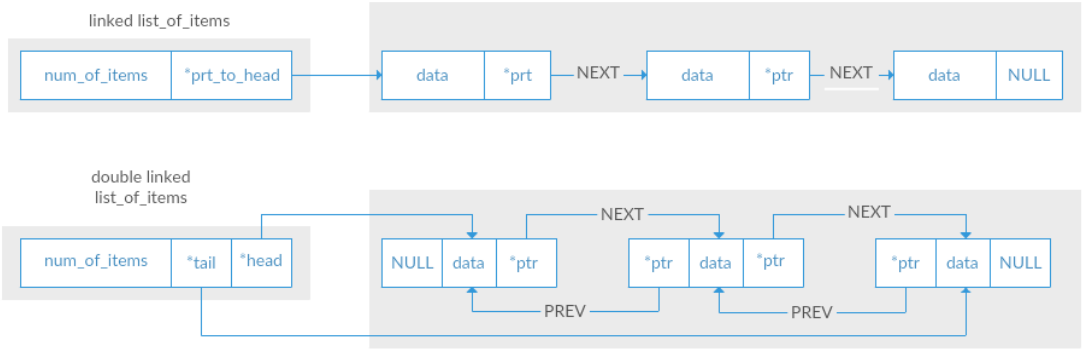
La definició d'un model equivalent als *documents* ha consistit en la utilització d'estructures de dades (structs) sobre les que es defineixen els diferents camps del documents i la utilització de la paraula reservada typedef per proporcionar a la estructura d'un nom alternatiu, o alies. A la figura següent es pot observar la implementació realitzada per definir els documents Schedule i Service, així com les relacions que s'estableixen entre ells.

```
typedef struct _schedule_item {
    unsigned int id;
    int attempts;
    int last_status;
    time_t t;
    service_item *service;
    int cicles;
    struct _schedule_item *prev;
    struct _schedule_item *next;
} schedule_item;
```

```
typedef struct _service_item {
    char *id;
    char *name;
    char *command;
    int use;
    int record_option;
    int report_options;
    int check_interval;
    int retry_interval;
    int max_check_attemps;
    time_t last_check;
    time_t last_update;
    struct _service_item *prev;
    struct _service_item *next;
    schedule_item *schedule;
    record_list *records;
} service_item;
```

Resultava especialment complicada la gestió de la memòria, doncs els nombre de serveis a monitoritzar o la quantitats de registres generats poden variar forma considerable d'un node a un altre, fent inviable la utilització d'arrays . Per aquest motiu s'ha optat per implementar una solució basada en llistes encadenades on cada

element resideix en un àrea de memòria independent i disposa d'un punter cap a l'àrea de memòria que ocupa el següent element de la llista. Aquesta solució permet disposar de llistats de longitud molt variables, ocupant i alliberant memòria segons necessitats. Per contra, s'ha de ser especialment curós en la seva implementació per evitar fuges de memòria (memory lacks).



Per a cada una de les col·leccions utilitzades a la aplicació client (***services, records, reports, schedules***) s'ha implementat un llista encadenada o doblement encadenada, per a la qual s'ha desenvolupat una llibreria que integra les funcions necessàries per crear el llistat, afegir i eliminar ítems, cercar ítems, o eliminar el llistat alliberant-ne tota la memòria:

```
service_list *service_list_create ();
int  service_list_additem (service_list *list, service_item *item);
void service_list_free (service_list *list );
```

A la figura següent es pot apreciar parcialment la solució de llista encadenada implementada per a la gestió de les tasques d'enviament al servidor (***reports***). A l'inici de l'aplicació, es crida la funció ***report_list_create(..)*** que crea un nou llistat de reports buit, cada vegada que es genera un nou report s'afegeix al llista cridat la funció report_list_additem*(...)*

```
/**
 *  Definition of report object and list of reports
 */
typedef struct _report_item {
    service_item *service;
    struct _report_item *next;
} report_item;

typedef struct {
    report_item *head;
    pthread_mutex_t mutex;
} report_list;
```

```
report_list *
report_list_create ()
{
    /* Allocate memory for the new list */
    report_list *list = (report_list *) malloc(sizeof(report_list));
    if(list == NULL)
    {
        syslog (LOG_NOTICE, "Error 12 - Out of memory\n");
        return NULL;
    }
    list->head=NULL;

    /* Initialize the mutex */
    pthread_mutex_init(&(list->mutex), NULL);
    return list;
}


int
report_list_additem (report_list *list, report_item *item)
{
    /* Lock mutex for safe thread */
    pthread_mutex_lock(&(list->mutex));

    if (list->head == NULL)  {
        list->head = item;
    } else {
        item->next = list->head;
        list->head = item;
    }

    /* Unlock mutex for safe thread */
    pthread_mutex_unlock(&(list->mutex));

    return 0;
}
```

D'altra banda, a la figura anterior també es pot apreciar la utilització als llistats dels mecanismes de concurrència Mutex utilitzats per compartir els llistats i les dades que contenen , entre els diferent fils de procés que conformen l'aplicació client . La implementació de fils d'execució es detalla al punt *6.2.6 Fils d'execució*

6.2.4 Comunicació amb el servidor

L'objectiu principal del projecte és el desenvolupament d'un sistema de monitorització basat en un servidor de gestió central on es defineixen els serveis a monitoritzar per cada un dels nodes client. Al servidor també s'hi recopilen i enregistren els resultats de cada una d'aquestes monitoritzacions. Per aquest motiu, un dels aspectes més rellevants de l'aplicació client és la comunicació amb el servidor. En aquest apartat es descriuen les solucions implementades a l'aplicació client per la comunicació amb el servidor i l'intercanvi de dades.

La comunicació entre l'aplicació client i el servidor es realitza mitjançant la API del servidor, on s'hi han definit una sèrie de mètodes específics:

API		
/api/client/	POST	Permet registrar un nou node al servidor retornant el token del nou node.
/api/client/:id	GET	Recupera la configuració del node i de tots els sensors que té assignats. Requereix token d'autenticació
/api/client/:id/heardbeat	GET	Permet verificar que client i servidor estan actius. Retorna la data de la darrera modificació a la configuració del node. Requereix token d'autenticació
/api/client/:id/report	POST	Entrega un report al servidor Requereix token d'autenticació

6.2.4.1 Llibreria de comunicacions libcurl

La llibreria libcurl proporciona un extens conjunt de definicions i rutines que permeten el desenvolupament de clients de comunicació per a una ampli nombre de protocols, com ara: FTP, FTPS, HTTP, HTTPS, IMAP, LDAP, LDAPSSCP, SFTP, SMTP ...

Al desenvolupament del component client s'utilitza aquesta llibreria per a implementar el protocol HTTPS, que utilitza encriptació SSL/TLS per protegir les comunicacions.

A la figura següent es mostra la implementació de la funció *curl_json_request,* aquesta funció és la responsable de gestionar als nodes client les comunicacions amb el servidor

```
char
*curl_json_request(char *url, char *postfields, char *token, int *error)
{
 ...
 TRY
 {
   curl_global_init(CURL_GLOBAL_SSL);
   curl = curl_easy_init();
   ...

   curl_easy_setopt(curl, CURLOPT_URL, url);
```

```
    /* Define headers */
    headers = curl_slist_append(headers, "Accept: application/json");
    headers = curl_slist_append(headers, "Content-Type: application/json");
    headers = curl_slist_append(headers, token);
    curl_easy_setopt(curl, CURLOPT_HTTPHEADER, headers);

    /* Disable verification of the peer certificate */
    curl_easy_setopt(curl, CURLOPT_SSL_VERIFYPEER, 0L);
    curl_easy_setopt(curl, CURLOPT_SSL_VERIFYHOST, 0L);

    /* Add postfiels to queue request */
    if (postfields) {
       curl_easy_setopt(curl, CURLOPT_POSTFIELDS, postfields);
    }

    curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, curl_ response);
    curl_easy_setopt(curl, CURLOPT_WRITEDATA, &curl_result);
    curl_easy_setopt(curl, CURLOPT_TIMEOUT, 20L);

   /* Launch Curl request */
    status = curl_easy_perform(curl);
    if(status != CURLE_OK) {...}

    /* Manage response */
    curl_easy_getinfo(curl, CURLINFO_RESPONSE_CODE, &code);
    if(code != 200) {...}

  } CATCH (ERROR) { ...}

     /* Frees curl allocated memory */
     curl_easy_cleanup(curl);
     curl_global_cleanup();
     curl_slist_free_all(headers);

     return data;
}
```

6.2.4.2 Serialització i deserialització de dades JSON

L'intercanvi d'informació entre l'aplicació client i el servidor es realitza mitjançant el format de dades estructurades JSON. Com s'ha comentat en apartats anteriors a l'apliació client les dades es modelen mitjançant tipus definits amb estructures de dades.

Per a realitzar la conversió entre les estructures de dades JSON i les estructures de dades de l'apliació client s'utilitza la llibreria cJSON de Dave Gamble (https://github.com/DaveGamble/cJSON). Aquesta llibreria conté una sèrie de funcions que permeten decodificar una estructura de dades JSON i convertir-la a tipus

de dades C. També conté funcions que permeten fer l'acció inversa, es a dir, a partir de tipus de dades C generar estructures de dades JSON.
L'apliació client implementa una llibreria de funcions *func_jcon.c* que conté totes les funcions necessàries per serialitzar o decodificar les estructures de dades JSON.

```
/* Estructura de dades que representa un objecte record */
typedef struct _record_item {
      time_t record_time;
      int attempt;
      int status;
      char *output;
      struct _record_item *next;
} record_item;

/* Serialització d'un objecte record en una cadena JSON */
char *record_to_cJSON ( record_item *item) {...}
```

Exemple d'un objecte *record* serialitzat en una cadena JSON:

```
{
   "time" : 1386363036,
   "attempt" : 5
   "status" : 0
   "output" : "CPU Usage 98%|c[cpu]=98%;80;95;0;100"
}
```

6.2.5 Autentificació

El component servidor implementa un protocol d'identificació i validació dels nodes client basat en tokens JSON Web Token (JWT), de manera que s'ha dotat a l'aplicació client de les eines necessàries per a la identificació dels nodes mitjançant aquest mecanisme d'identificació. En aquest apartat es detallen els mecanismes introduïts a l'aplicació client per a la gestió dels tokens JWT.

6.2.5.1 Descripció del procés d'autentificació

Cada una de les peticions que realitzen els nodes cap al component servidor ha d'anar acompanyada d'un token únic que identifica el node client així com els permisos d'accés als recursos del servidor. L'enviament d'aquest token es realitza introduint un registre personalitzat a la capçalera dels paquets HTTPS, d'aquesta manera que qualsevol comunicació amb el servidor va associada al token que identifica el node.

```
"X-Auth-Token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjU4NjdlMD
QxMDFlOGQ5NGMxMTMyNjFjNCIsImlhdCI6MTQ4MzIwMjYyNX0.SPsTcm8YHpPyHVCroxjq
x9hYd2qW4otxcIcoeZc6OoU"
```

Les comunicacions de la aplicació client amb el servidor es realitzen mitjançant les funcionalitats proporcionades per la llibreria *curllib*. Aquesta llibreria ofereix a la seva api les funcions necessàries per a la gestió de les capçaleres HTTP/HTTP i la seva personalització:

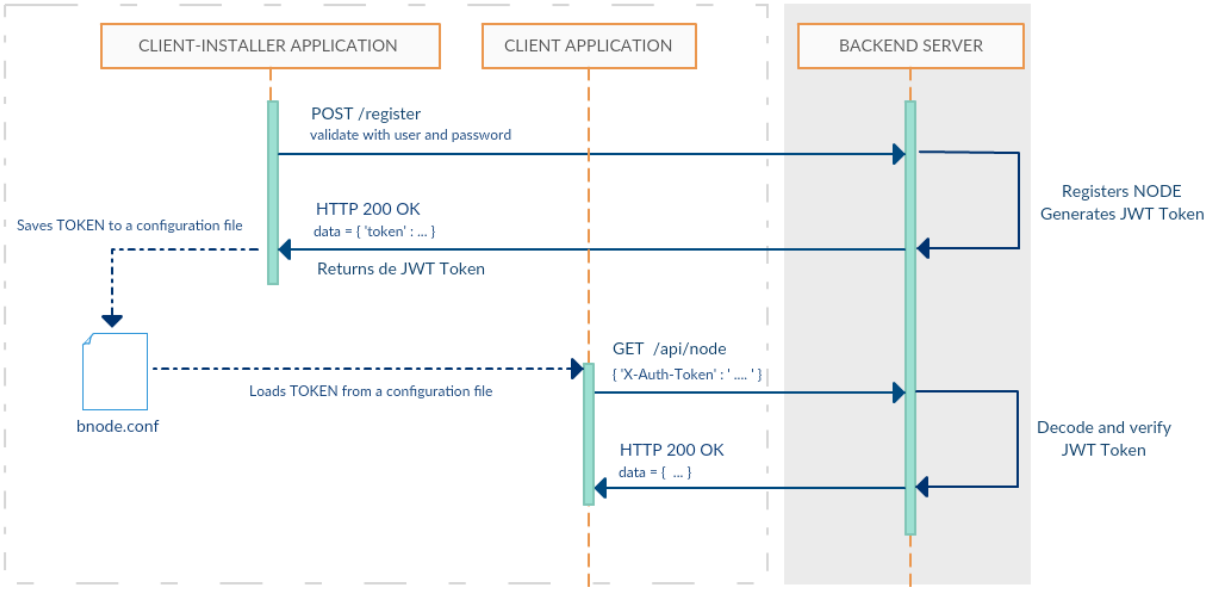
```
/* Define custom headers */
headers = curl_slist_append(headers, "Accept: application/json");
headers = curl_slist_append(headers, "Content-Type: application/json");

headers = curl_slist_append(headers, "X-Auth-Token: eyJhbGciOiJIUzI1Ni
IsInR5cCI6IkpXVCJ9.eyJpZCI6IjU4NjdlMDQxMDFlOGQ5NGMxMTMyNjFjNCIsImlhdCI
6MTQ4MzIwMjYyNX0.SPsTcm8YHpPyHVCroxjqx9hYd2qW4otxcIcoeZc6OoU");

curl_easy_setopt(curl, CURLOPT_HTTPHEADER, headers);
```

Cada node disposa d'un token únic que l'identifica. Aquest token es genera al registrar per primer cop un node al servidor. El procés de registre del node es realitza des de la utilitzat ***bnode-conf*** que constitueix la utilitat de configuració del node. Aquesta utilitat sol·licita a l'usuari l'adreça IP o FQDN del servidor i un usuari/contrasenya amb permisos d'accés al servidor. A partir d'aquestes credencials la utilitat de configuració contacta amb la API del servidor i sol·licita el registre del nou node, obtenint com a resposta el token delnode. Per finalitzar el procés , la utilitat de configuració emmagatzema la URL del servidor i el token del node al fitxer **bnode.conf**

Esmentar que el token es manté durant tot el cicle de vida del node. Es a dir, a diferencia del tokens d'usuari, el tokens dels nodes no caduquen. D'aquesta manera s'evita haver de registrar de nou node cada cop que el token caduca, per contra, el servidor manté un registre dels tokens assignats de manera que es poden bloquejar si un sistema s'ha vist compromès.

6.2.5.2 Esquema de seqüència del procés d'autentificació

Al següent diagrama es pot observar, d'una banda la seqüència de registre del node i obtenció del seu token, i d'altra banda el procés d'identificació al servidor mitjançant el token:



6.2.6 Fils d'execució

Inicialment l'aplicació és va dissenyar implementant un únic fil d'execució, però ben al principi del procés de desenvolupament es va comprovar que aquest model comportava alguns problemes destacables.

La aplicació client realitza tres tasques principals: En primer lloc, comunicació amb el servidor i obtenció de la configuració del serveis. En segon lloc, execució de les tasques de monitorització segons la configuració dels serveis. I en tercer lloc, enviament de resultats de monitorització al servidor. Cada una d'aquestes tasques es realitza segons una marca de temps definida per a cada una d'elles. L'inconvenient d'utilitzar un sol fil d'execució es que totes les tasques comparteixen una mateixa seqüencia d'execució i mentre s'està executant una d'aquestes tasques, no se'n pot executar un altre. Per exemple, cap tasca de monitorització es pot executar mentre el servidor estigui actualitzant els serveis o enviant els resultats de les monitoritzacions al servidor.

Per aquest motiu, i també per les possibilitats formatives que implica, es va decidir implementar un model basat en múltiples fils d'execució, on es destina un fil per cada una de les tasques principals de l'aplicació client, es a dir, un primer fil per a la comunicació amb el servidor i l'obtenció de la configuració del serveis , un segon fil per a la execució de les tasques de monitorització, i un tercer fil destinat a l'enviament dels resultats de monitorització.

6.2.6.1 Descripció dels fils d'execució

Per al desenvolupament dels fils d'execució s'han utilitzat les especificacions definides per l'estàndard POSIX (*Portable Operating System Interface*, https://computing.llnl.gov/tutorials/pthreads), i en concret, la implementació que en la llibreria ***libpthread*** .

En aquest apartat es descriu breument els recursos de la llibreria *libpthread* utilitzats en el desenvolupament de l'aplicació client

```
/* Starts application threads */
pthread_t pth_sync, pth_report, pth_schedule;

...

err = pthread_create( &pth_sync, NULL, sync_manager, params);
err = pthread_create( &pth_schedule, NULL, schedule_manager,
params);
err = pthread_create( &pth_report, NULL, report_manager, params);

...

/* Thread managers */
void *sync_manager( void *params ) { ... };
void *schedule_manager( void *params ) { ... };
void *report_manager( void *params ) { ... };
```

La aplicació s'inicia amb el fil d'execució principal, que llegeix el fitxer de configuració i recupera la configuració inicial del sistema, un cop validada, crea les estructures de dades que es compartiran entre els fils d'execució. El següent pas del fil principal és invocar per a cada nou fil la funció *pthread_create* , proporcionant com arguments cada una de les funcions principals del client, definides com a managers , així com una referència a les estructures de dades compartides.

La funció *pthread_create* crea un nou fil d'execució i n'assigna el control a la funció manager corresponent. A banda, se li es proporciona un punter cap a l'espai de memòria que ocupen les dades estructures de dades compartides, de manera, que tots tres fils tenen accés al mateix espai de dades.

Definició de les estructures de dades compartides:

```
/**
 *  Definition of args
 */
typedef struct _args {
    service_list *services;
    schedule_list *schedules;
    report_list *reports;
    struct client_config client;
} args;

args *params = malloc(sizeof(args));

/* Allocates memory and initializes application data structures */
params->services = service_list_create();
params->schedules = schedule_list_create();
params->reports = report_list_create();
```

Un cop iniciat cada manager s'executa dins un bucle infinit, executant segons marques de temps les tasques que se li han assignat. Per exemple, el fil *report_manager* comprova cada cert interval de temps si hi ha algun registre pendent d'enviar, si hi és, executa la rutina corresponent i l'envia al servidor, per retornar a continuació retornar al punt inicial comprovant i hi ha algun registre pendent d'enviar... Al fil de reports no s'hi genera cap registre, simplement accedeix a l'espai de memòria compartida i recupera els registres que el fil s*chedule_manager* hi haurà desat.

6.2.6.2 Espais de memòria compartida

La principal dificultat alhora de treballar amb múltiples fils d'execució són els problemes que es poden generar derivats de l'accés concurrent a espais de memòria compartida. Per exemple, si un fil d'execució escriu un valor en un registre de memòria mentre un altre, al mateix temps, l'està llegint, els resultats poden no ser els esperats.

Per evitar aquesta mena de problemes i possibilitar l'accés concurrent dels fils d'execució a l'espai de memòria compartida que conté les dades de l'aplicació, l' estàndard POSIX (Portable Operating System Interface) disposa de les variables *Mutex*. Una variable mutex (exclusió mútua) és un mecanisme de sincronització de fils

d'execució que bloqueja temporalment un àrea de memòria compartida, limitant-ne l'accés a un únic fil d'execució.

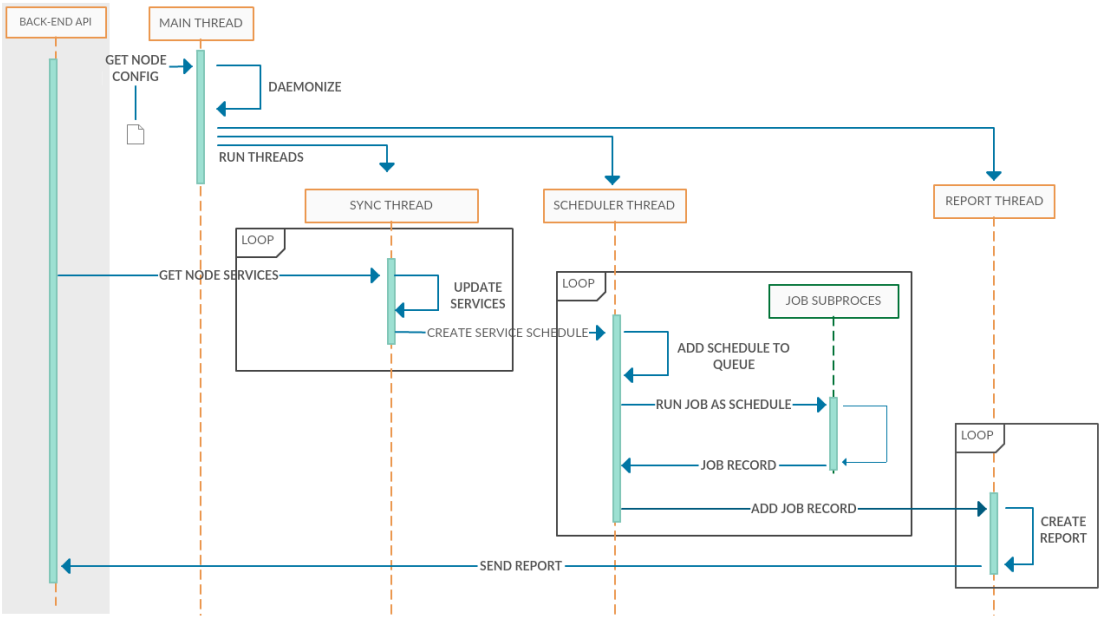
```
typedef struct {
    int count;
    record_item *head;
    pthread_mutex_t mutex;
} record_list;
```

```
record_list *
record_list_create ( )
{
   ...
   /* Initialize the mutex */
   pthread_mutex_init(&(l->mutex), NULL);
   return l;
   ...
}
```

```
void
record_list_additem (record_list *list, record_item *item)
{
    /* Lock mutex for safe thread */
    pthread_mutex_lock(&(list->mutex));

    ...

   /* Unlock mutex for safe thread */
   pthread_mutex_unlock(&(list->mutex));
}
```

6.2.6.3 Esquema de seqüència dels fils d'execució

Al diagrama següent es pot observar la seqüencia d'execució dels tres fils d'execució que integren la aplicació:



6.2.7 Consideracions addicionals de Seguretat

Com ja hem comentat als apartats anteriors la aplicació client es la responsable d'executar les tasques de monitorització als nodes i reportar-ne els resultats al servidor. Cada una d'aquestes tasques de monitorització és, ara per ara, externa a l'aplicació. Una part important d'aquestes tasques externes consisteixen en scripts especialment dissenyats per comprovar l'estat d'algun element, tot i que també s'hi poden trobar crides a comandes i utilitats del sistema.

Aquestes tasques de monitorització, així com els seus paràmetres, son definits al component servidor i traslladats a l'aplicació client per a la seva execució. Això proporciona gran flexibilitat al sistema permetent activar o desactivar serveis, així com definir els seus paràmetres de execució sense necessitat d'accedir als sistemes remots. Malauradament, també pot generar un greu problema de seguretat ja que una intrusió al component servidor pot comprometre també els sistemes client.

El servidor disposa del seus propis mecanismes de seguretat per evitar l'accés no autoritzat al sistema, però quan parlem de seguretat mai s'ha de descartar cap possibilitat, per aquest motiu, s'han introduint a la aplicació client un parell de mecanismes de seguretat addicionals:

- Només està permesa la execució de les tasques de monitorització recollides al fitxer services.allow Aquest fitxer, present a cada node client, conté una relació dels scripts i utilitats considerats segurs, així com la seva ubicació al sistema. Cada cop que la aplicació client rep un nou servei a monitoritzar accedeix a aquest fitxer i comprova que la comanda figuri al llistat, si no hi apareix la descarta i genera la corresponent alerta.
- S'ha introduït un sistema de filtrat per evitar que es puguin intruir, mitjançant el paràmetres de les comandes, instruccions perilloses pel sistema. El filtre consisteix a cerca una sèrie de caràcters considerats il·legals, qualsevol servei amb una comanda que utilitzi qualsevol d'aquests caràcters es descarta immediatament es genera la corresponent alerta. A la versió actual de proves aquest caràcter son:

6.2.8 Captures del component client

A la següent figura es pot observar el resultat d'executar la utilitat de configuració **bnode-conf.** En un node que ja es troba registrat al servidor la utilitat ens mostra informació genèrica del node, així com informació detallada sobre l'adreça del servidor i el seu estat.

```
bnode 0.0.2b LINUX
Copyright (c) 2016-2017 Raul Martinez Diaz
Last Modified: 4 December, 2016
License: GPL

-------------------------------------
bnode [ Id 584adec41ea9fa40250045f0 ]
-------------------------------------

Host Name: development-dektop
System:    Linux
Version:   #66-Ubuntu SMP Wed Oct 19 14:12:37 UTC 2016

Server Address: https://192.168.1.240:8443
Server Status: { ONLINE }
Node status:   { registered }
```

A la figura següent es pode apreciar com la API del servidor rep els batecs o *heartbeats* d'un node. També podem observar la capçalera de les peticions, amb el corresponent registre **x-auth-token** que conté el token del node:

```
11 Dec 17:07:02 - Beluganode Core {beluga_core alpha.12.16} is working on port: 8443
{ host: '192.168.1.240:8443',
  accept: 'application/json',
  'content-type': 'application/json',
  'x-auth-token': 'fa8426a0-8eaf-4d22-8e13-7c1b16a9370c',
  'content-length': '164' }
POST /api/heartbeat 200 537.073 ms - 223
GET /api/nodes 200 17.705 ms - 1327
{ host: '192.168.1.240:8443',
  accept: 'application/json',
  'content-type': 'application/json',
  'x-auth-token': 'fa8426a0-8eaf-4d22-8e13-7c1b16a9370c',
  'content-length': '164' }
POST /api/heartbeat 200 10.481 ms - 223
```

7. Conclusions i treballs futurs

7.1 Treball realitzat

Fruït del treball d'estudi i anàlisi de les necessitats de monitorització de l'empresa col·laboradora en les pràctiques externes, s'ha desenvolupat una solució que permet la monitorització remota de tota mena sistemes basats en distribucions GNU/LINUX.

La solució desenvolupada s'integra, segons la arquitectura client-servidor inicialment plantejada, de dos components:

- El component servidor, que rep el nom de *bnode-core*, s'ha desenvolupat amb una arquitectura NodeJS + MongoDB. La funcionalitat d'aquest component és centralitzar la gestió de totes les tasques de monitorització dels nodes client, permetent assignar i configurar els serveis a monitoritzar per cada node, així com recopilar els resultats de cada una de tasques de monitorització i dipositar-los en una base de dades pel seu anàlisi. El component servidor s'integra d'una API REST desenvolupada en NodeJS + MongoDB, així com una consola d'administració desenvolupada en AngularJS. Aquesta consola d'administració interacciona amb la API per configurar els serveis, assignar nodes i llistar els registres de monitorització.

- Components client bnode i bnode-conf desenvolupats en Ansi C. Són els components que s'executen al nodes. La utilitat bnode-conf s'encarrega de la configuració i registre de cada node al servidor, mentre que la aplicació bnode, dissenyada per executar-se com a servei, s'encarrega de descarregar del servidor la configuració dels serveis a monitoritzar, així com crear i executar les tasques de monitorització i enviar-ne els resultats al servidor.

Tot i que inicialment ideada per a la monitorització de sistemes FreePBX i derivats la solució implementada, basada en la compatibilitat amb el model de plugins de Nagios, permet monitoritzar molts altres sistemes i serveis. El desenvolupament del client en llenguatge Ansi C permet que aquest pugui ser compilar i executat a la majoria de distribucions GNU/Linux.

Cal fer una menció especial a la Consola de Monitorització Web. Aquest component desenvolupat en AngularJS permet, mitjançant la API del servidor, gestionar els nodes,

crear i assignar serveis o visualitzar els registres. És dins el projecte, el component en el qual menys s'ha treballat limitant-ne el seu desenvolupament a les funcionalitats més bàsiques. No obstant això, és el component que en treballs futurs prendrà més rellevància per les possibilitats que ofereix.

7.2 Treballs futurs

Tot i que l'aplicació resultant és totalment funcional, encara resta treball abans de plantejar-se ficar-la en producció. Actualment el conjunt d'aplicacions es troba en fase de probes, el que ha permès detectar algunes carències destacables que caldrà considerar a les properes revisions de l'aplicació.

Durant l'anàlisi, disseny i desenvolupament van sorgir moltes funcionalitats addicionals que, bàsicament per motius de temps, no han estat incorporades al projecte presentat, d'altres han quedat incompletes o s'han de refer parcialment. Aquesta és doncs la primera part dels treballs futurs, millorar i polir el codi, així com revisar les funcionalitats existents i implementar les que van quedar al tinter. N'hi ha de tot tipus, temes de seguretat, mancança de funcionalitats a la API, millora en la gestió d'errors, etc...

La segona part dels treballs futurs, fa referència a la Consola de Monitorització web. Un cop el sistema sigui fiable caldrà redefinir i dissenys la Consola Web per treure tot el partit al sistema, no només ha de permetre configurar els serveis i nodes, sinó també ha d'oferir informació detallada dels registres obtinguts, mostrar gràfiques i estadístiques, permetre l'accés multiusuari amb diferents rols, enviament de notificacions de correu mitjançant plantilles, etc ...

7.3 Conclusions

Aquest projecte sorgeix de la necessitat de donar solució a una problema concret, la monitorització de sistemes de telefonia IP (VoIP) remots. Aquest sistemes, basats en FreePBX i els seus derivats es troben distribuïts en un ampli radi d'acció que comprèn clients ubicats a Catalunya, Comunitat Valenciana i Balears. Molts d'aquest sistemes, són gestionats per tercers i resulten de difícil accés pel seu control i monitorització.

Arran de l'oportunitat sorgida en base al present Treball de Final de Màster es va considerar oportú estudiar la viabilitat i desenvolupament d'una eina de monitorització pròpia. Fruït de l'estudi i anàlisi de les necessitats de monitorització de l'empresa col·laboradora s'ha dissenyat i desenvolupat un eina, basada en una arquitectura client-servidor, que permet la gestió remota de les tasques de monitorització, així com recopilar-ne tots els resultats en una base de dades pel seu anàlisi.

L'eina desenvolupada compleix molts dels objectius que es van establir durant la recollida de requisits: És adaptable, de manera que permet definir per a cada node els serveis que s'han de monitoritzar. És segura, doncs les comunicacions entre client i servidor son xifrades mitjançant SSL i alhora s'utilitzen token per a la identificació i validació de nodes i usuaris. És compatible amb els sistemes existents, ja que està dissenyada segons el model de plugins de Nagios permetent-ne la seva execució.

Però també hi ha alguns objectius que no s'ha arribat a assolit plenament: No envia encara notificacions electròniques, ja que es va considerar que per la seva importància era un tema a estudia amb profunditat. La consola de gestió web és encara molt bàsica, només permet les funcions essencials per funcionament del sistema. Resten funcionalitats de la API per implementar així com altres mancances detectades durant la fase de probes.

Moltes de les solucions implementades han resultat perfectament vàlides i els seus resultats satisfactoris, d'altres caldrà revisar-les i redissenyar-les abans de poder ficar l'aplicació en producció. Tot i això, el punt més positiu és que més enllà de l'àmbit merament acadèmic del projecte, el seu desenvolupament continua.

Des del punt de vista de l'aprenentatge, aquest projecte ha estat un instrument molt útil per poder utilitzar i aprofundir en moltes de les eines de programari lliure introduïdes durant el Màster de Programari Lliure. També vull destacar que el desenvolupament del projecte ha motivat una recerca constant en el camp de la monitorització de sistemes que m'ha servit per aprofundir i reforçar els meus coneixement en l'àmbit de la monitorització de sistemes.

8. Referències

Chodorow, Kristina (2013). *MongoDB: The Definitive Guide, Second Edition* (2a ed.). EEUU: O'Reilly Media Inc. ISBN: 978-1-449-34469-9

Green, Brad; Seshadri Shyam (2013). *AngularJS* (1a ed.) EEUU: O'Reilly Media Inc. ISBN: 978-1-449-34485-6

Ornbo, George (2013). *Seams Teach Yourself Node.js in 24 Hours* ["Programación Node.JS"] (1a. Ed.) Madrid: Ediciones Anaya Multimedia ISBN: 978-84-415-3314-1

W.Kernighan, Brian; M.Ritchie, Dennis (1988). *The C Programming Language* (2nd. ed.) EEUU: Practice Hall. ISBN: 0-13-110362-8 / 0-13-110370-9

Crockford, Douglas (2008). *Javascripts: The Goog Parts* (1a ed.) EEUU: O'Reilly Media Inc. ISBN: 978-0-596-51774-8

M. Jones, Microsoft, J. Bradleyl, N. Sakimura (May 2015) *JSON Web Token (JWT)* ISSN: 2070-1721 URL: https://tools.ietf.org/html/rfc7519

Curl Project Documentation
[Data darrera consulta: 30 de Desembre de 2016]
https://curl.haxx.se/docs/projdocs.html

Nagios Core Documentation
[Data darrera consulta: 20 de Desembre de 2016]
https://www.nagios.org/documentation/

Mangal, Agraj *(2014), Authenticating Node.js Applications With Passport*
Envato Tuts [Data darrera consulta: Novembre de 2016]
https://code.tutsplus.com/tutorials/authenticating-nodejs-applications-with-passport--cms-21619

Holmes, Simon (2016), *User Authentication with the MEAN Stack*
Published Feb.2016 on Sitepoint.com [Data darrera consulta: Novembre de 2016]
https://www.sitepoint.com/user-authentication-mean-stack/

Sevilleja, Chris (2015) Authenticate a Node.js API with JSON Web Tokens
Published Apr.2015 on Scotch.io [Data darrera consulta: Novembre de 2016]
https://scotch.io/tutorials/authenticate-a-node-js-api-with-json-web-tokens

9. Llicència

GNU Free Documentation License
Version 1.3, 3 November 2008

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain

any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy

that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license

from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this

License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or
in part into the MMC, (1) had no cover texts or invariant sections, and
(2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

    Copyright (c) YEAR YOUR NAME.
    Permission is granted to copy, distribute and/or modify this document
    under the terms of the GNU Free Documentation License, Version 1.3
    or any later version published by the Free Software Foundation;
    with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
    A copy of the license is included in the section entitled "GNU
    Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts,
replace the "with...Texts." line with this:

    with the Invariant Sections being LIST THEIR TITLES, with the
    Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License,
to permit their use in free software.