luporum, configuració UCI mitjançant una API REST

Memòria del Projecte de Final de Màster de Programari Lliure
Desenvolupament d'aplicacions

Autora: Mònica Ramírez Arceda
Consultor: Gregorio Robles Martínez
Tutor extern: Víctor Oncins Biosca

Resum

L'empresa **Routek SL** manté una distribució anomenada ***qMp*** basada en la distribució ***OpenWRT***. Tant la distribució **OpenWRT**, com l'ampliació de Routek SL ***qMp*****,** disposen d'un sistema de configuració anomenat *Unified Configuration Interface* (***UCI***). Una de les característiques d'aquest sistema és que les configuracions dels programes que l'usen segueixen una mateixa sintaxi.

El projecte ***luporum*** descrit en aquesta memòria explica com s'ha realitzat el desenvolupament d'una API REST que facilita que es pugui gestionar aquesta configuració remotament, fent peticions HTTP.

El projecte ***luporum*** està dividit en dos subprojectes: el primer subprojecte, de nom ***lupus***, és el projecte principal i tracta el desenvolupament de l'API. El segon subprojecte, de nom ***lupetto***, és un exemple de client de ***lupus***. En aquest cas es tracta d'un client web molt senzill que permet usar les funcionalitats més bàsiques de ***lupus***.

En ambdós subprojectes es detalla la planificació, l'anàlisi, la implementació i la distribució de les dues aplicacions.

Tot el codi implementat és programari lliure així com totes les eines usades per al desenvolupament del projecte.

Índex de continguts

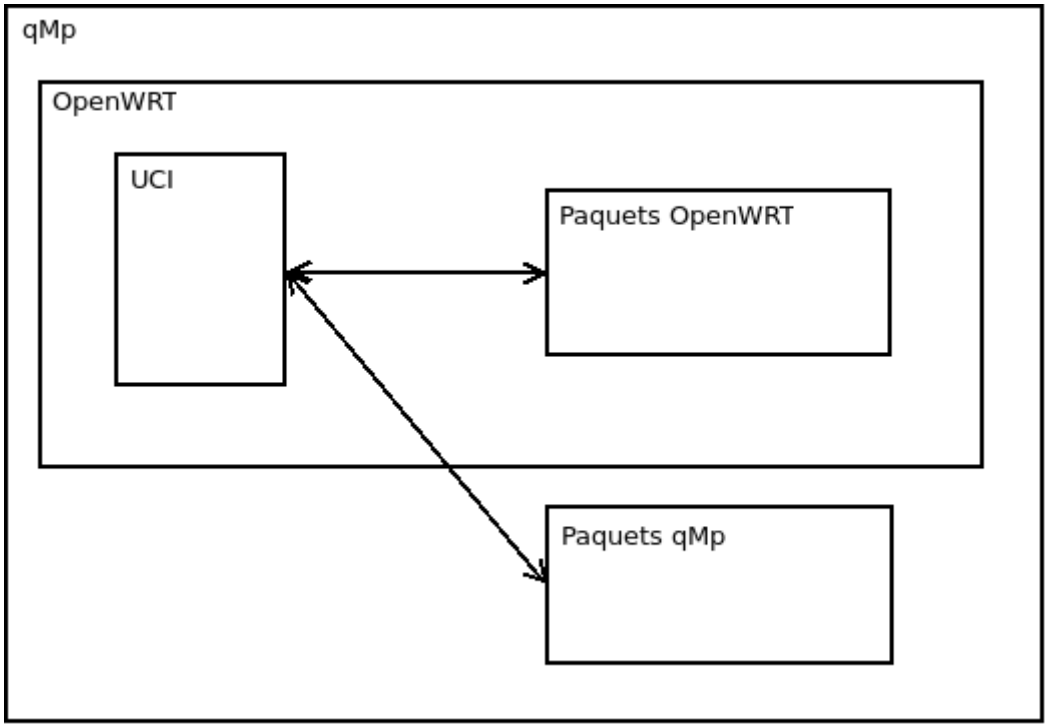
1 Introducció

L'empresa Routek S.L. es dedica al desenvolupament i implementació de projectes telemàtics. Està especialitzada en el disseny, instal·lació i manteniment de xarxes de comunicacions sense fils (WiFi). La seva pàgina web, on es pot veure la filosofia de l'empresa i els seus projectes actuals és http://routek.net/

Un dels projectes clau de l'empresa Routek S.L. és el projecte *qMp* (*Quick Mesh Project*). *qMp* és un firmware, basat en la distribució *OpenWRT*, que permet desplegar xarxes mesh de manera automatitzada i senzilla. Es pot trobar més informació d'aquest projecte a la seva pàgina web: http://qmp.cat

Els fitxers de configuració de *qMp* utilitzen el sistema *UCI* (*Unified Configuration Interface*), el sistema de configuració unificat d'*OpenWRT.*

En el següent esquema es veuen representats tots els actors exposats: *qMp* com a distribució que amplia la distribució *OpenWRT* i tant els paquets d'*OpenWRT* com els afegits per *qMp*, que interactuen amb el sistema *UCI*.



Tot i que *OpenWRT* i *qMp* disposen d'una web per tal de configurar els seus dispositius, anomenada *LuCI*, es té la necessitat de poder desenvolupar més clients (webs externes, aplicacions per a dispositius mòbils, centres de gestió...) que permetin administrar de forma senzilla aquests dispositius.

1.1 Objectius

Amb el desenvolupament del projecte *luporum* es volen aconseguir els següents objectius:

- **Gestió del sistema UCI:** L'objectiu principal del projecte és desenvolupar un mecanisme simple i de fàcil ús per tal de poder gestionar la configuració dels dispositius des d'aplicacions client remotes.
- **Configuració multidispositiu:** Desenvolupar una aplicació que permeti configurar més d'un dispositiu simultàniament. És a dir, que des d'una única aplicació es pugui observar i modificar la configuració de diversos dispositius, sense haver d'accedir a una aplicació

diferent per a cada dispositiu.

1.2 Motivació personal

Escollir un projecte de desenvolupament no és una tasca fàcil, hi ha molts factors a tenir en compte. Els principals motius pels quals he escollit el desenvolupament de *luporum* com a Treball de Final de Màster són els següents:

- **Treballar amb una empresa que desenvolupa programari lliure:** tot i que el programari lliure té molt més renom que fa uns anys, són poques les empreses que desenvolupen exclusivament programari lliure i obren els seus projectes a la comunitat. Moltes usen productes de programari lliure, però poques tenen la filosofia d'usar i crear tots els seus productes amb llicències lliures. Routek SL és una d'aquestes poques empreses amb qui he tingut el gust de col·laborar durant aquests mesos.
- **Aprendre diverses tecnologies**: abans del desenvolupament d'aquest projecte mai havia treballat amb dispositius embedded. Era una motivació aprendre quines eines i llenguatges s'usen en el procés de creació d'aplicacions per a aquest tipus de dispositius amb recursos limitats.
- **Continuïtat del projecte:** molts cops els projectes de final d'estudis, no tenen continuïtat, és a dir, no es segueixen desenvolupant després de la seva presentació. En aquest cas, com que l'empresa Routek SL té força interès en poder oferir aquest producte, existeixen força possibilitats que *luporum* segueixi sent desenvolupat i millorat després de la seva entrega com a treball de final de màster. Això és per a mi una gran motivació, ja que saber que el projecte serà usat i mantingut genera molta il·lusió.

1.3 Estructura de la memòria

La memòria està estructurada en els següents apartats:

- **Introducció:** En aquest apartat es presenta el projecte *luporum* contextualitzant d'on surt la idea del projecte i quins són els seus objectius principals.
- **Fonaments teòrics:** En aquest apartat es descriuen diversos temes essencials per tal d'entendre el projecte. S'expliquen totes aquelles nocions tècniques que cal saber.
- **El projecte *luporum*:** En aquest apartat es descriu el projecte en si: quina ha estat la motivació principal per tal de dur-lo endavant i una descripció detallada d'allò que es vol aconseguir. Es descriuen els dos subprojectes que composen *luporum*: *lupus* i *lupetto*.
- **Planificació:** En aquest apartat es detalla la planificació del projecte. Es detallen les tasques que s'han de dur a terme i en quines dates s'han d'haver aconseguit cadascuna d'elles.
- **Anàlisi:** En aquest apartat es descriu l'anàlisi, funcional i no funcional, dels dos subprojectes, *lupus* i *lupetto*.
- **Implementació:** En aquest apartat es detalla la implementació dels dos subprojectes, *lupus* i *lupetto*, especificant també quins han estat els entorns de desenvolupament i de producció.
- Distribució: En aquest apartat s'especifica com es distribuiran les dues aplicacions del projecte, *lupus* i *lupetto*, a l'usuari final.
- **Conclusions:** En aquest apartat es descriuen les conclusions a les quals s'arriba al final de l'elaboració del projecte.

2 Fonaments teòrics

2.1 OpenWRT

OpenWrt és una distribució GNU/Linux destinada a ser usada en dispositius *embedded*, normalment en *routers*. No es tracta d'un *firmware* estàtic sinó que es tracta d'un sistema de fitxers complet editable i disposa d'un sistema de gestió de paquets. Això permet personalitzar *firmwares*, instal·lant i configurant tots aquells programes que es necessitin.

Per tal de veure quins són els dispositius compatibles amb *OpenWRT*, es pot consultar la següent web: http://wiki.openwrt.org/toh/start

2.2 Configuració d'OpenWRT

Per tal de gestionar i configurar un sistema OpenWRT principalment es disposa de dos mètodes:

- **Línia de comandes:** com tot sistema GNU/Linux, es pot entrar a un dispositiu via ssh i gestionar el sistema a partir de la shell del dispositiu. Per realitzar canvis permanents es poden editar els fitxers o usar la comanda *uci*.
- **LuCI:** es tracta d'una aplicació web instal·lada en cada dispositiu que permet configurar-lo. S'hi pot accedir a partir de qualsevol navegador que estigui connectat a la xarxa del dispositiu. Per a més informació respecte l'aplicació web *LuCI* es pot consultar la seva web: http://luci.subsignal.org/

2.3 El sistema de configuració UCI

El principal sistema de configuració dels programes disponibles amb *OpenWRT* s'anomena ***UCI*** (*Unified Configuration Interface*). L'objectiu principal d'*UCI* és unificar la configuració de tots els programes: emmagatzema els valors de configuració de tots els programes en un únic directori i usa el mateix format en tots els fitxers.

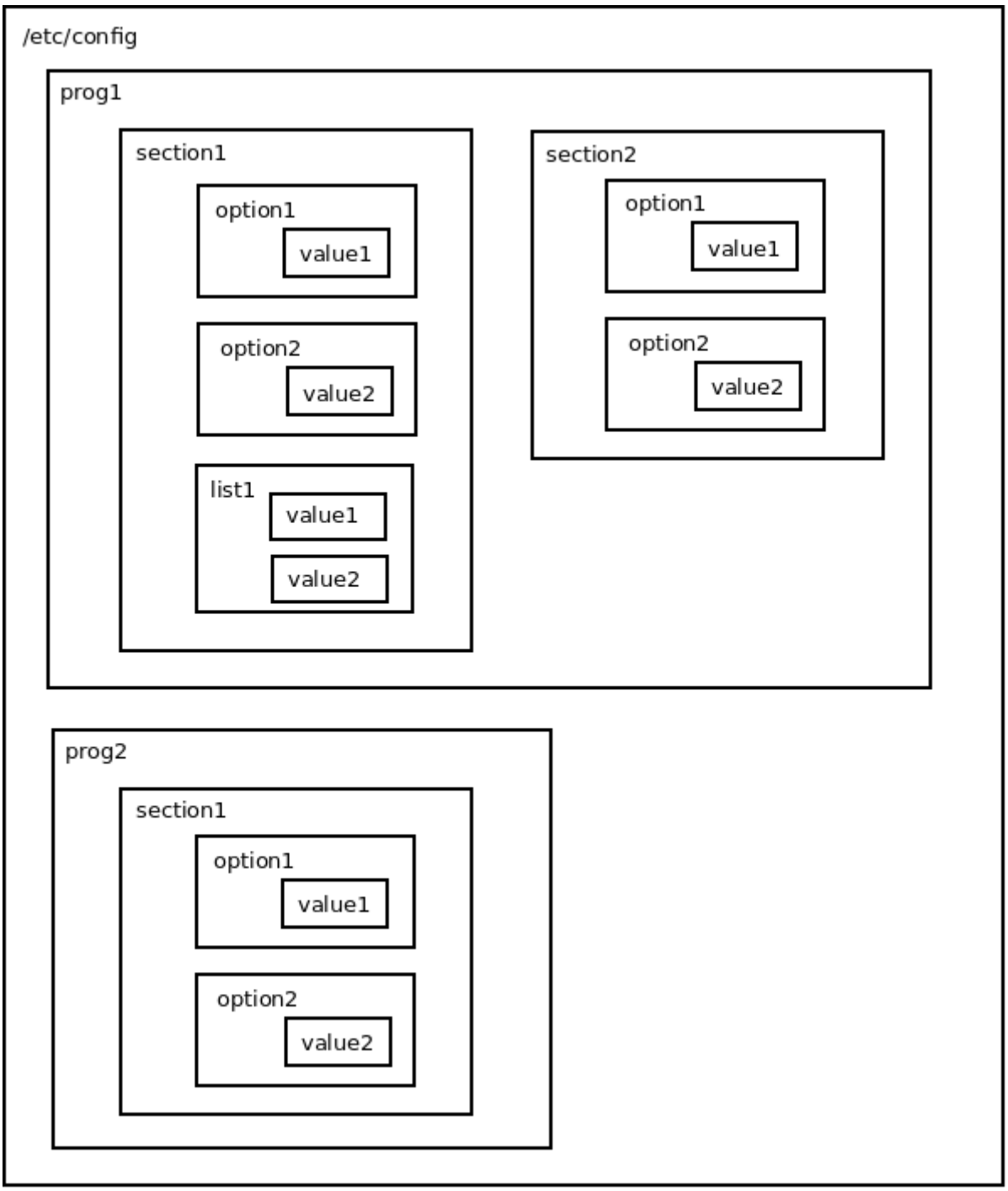
En general, els programes usen els seus fitxers de configuració per emmagatzemar els valors de diferents paràmetres que necessiten durant la seva execució. La sintaxi d'*UCI* permet desar aquests valors classificant-los en diferents seccions i permetent que un paràmetre tingui més d'un valor.

Així doncs, els programes que usen el sistema *UCI* tenen un fitxer de configuració al directori `/etc/config`. Els fitxers ubicats en aquest directori tenen el següent format:

```
config type1 'section_name1'
        option option_name1 'value1'
        option option_name2 'value2'
        list list_name3 'value31'
        list list_name3 'value32'
        list list_name3 'value33'
        ...
config type2 'section_name20'
        ...
config type2 'section_name21'
        ...
config type3
        ...
```

Com es veu en la sintaxi anterior, tot fitxer de configuració està dividit en seccions que sempre tenen un tipus i poden tenir o no un nom. A cadascuna d'aquestes seccions es permet desar els paràmetres del programa amb els seus valors.

El següent esquema mostra més clarament la jerarquia d'aquesta sintaxi:



Per acabar d'entendre cadascun dels conceptes de la sintaxi dels fitxers del sistema *UCI*, es defineixen a continuació les paraules clau més importants:

- `section`: els fitxers de configuració estan dividits en seccions. Aquestes poden tenir un nom, que posarem com a segon paràmetre de la línia que defineix la secció o poden no tenir nom. Si una secció no té nom, es diu que és una secció anònima. De fet, quan una secció és anònima, el sistema *UCI* assigna un nom intern a aquesta secció per tal que tingui un identificador únic.
- `config`: cada secció comença per la paraula clau `config`.
- `type`: tota secció té un tipus amb un nom concret, que és obligatori. El tipus és el primer paràmetre de la línia que defineix la secció. Es posa immediatament després de la paraula clau `config`.
- `option`: cada secció pot tenir diferents opcions. Les opcions són els paràmetres del programa

amb els seus valors. Cada `option` té un nom i un únic valor. La sintaxi és la següent:

```
option option_name 'value'
```

on *`option_name`* és el nom del paràmetre i *`value`* el valor d'aquest paràmetre.

- `list`: les seccions també poden tenir paràmetres amb més d'un valor. En aquest cas, enlloc de definir una *`option`* s'ha d'usar la paraula clau `list`. Per definir una `list` se li ha de donar un nom i s'ha d'escriure una línia per a cadascun dels seus valors. La sintaxi és:

```
list list_name 'value1'
list list_name 'value2'
list list_name 'value3'
...
list list_name 'valuen'
```

Finalment, es mostren dos exemples de fitxers de configuració per acabar de fer més entenedora l'explicació teòrica:

- `/etc/config/network` que configura els paràmetres de xarxa d'un dispositiu:

```
config interface 'loopback'
        option ifname 'lo'
        option proto 'static'
        option netmask '255.0.0.0'
        option ipaddr '127.0.0.1'

config interface 'lan'
        option ifname 'eth0'
        option proto 'static'
        option netmask '255.255.255.0'
        option ipaddr '192.168.2.5'
```

En aquest exemple es tenen dues seccions, de nom `loopback` i `lan` respectivament. Ambdues són del mateix tipus, anomenat `interface`. Per a cada secció hi ha definides diverses opcions, cadascuna amb un únic valor.

Es veu doncs que la interfície de xarxa de nom `loopback` té com a referència interna `lo`, és estàtica, la seva adreça IP és 127.0.0.1 i la màscara de xarxa és la 255.0.0.0.

En canvi, la interfície de xarxa de nom `lan` té com a referència interna `eth0`, és també estàtica, la seva adreça IP és 192.168.2.5 i la màscara de xarxa és la 255.255.255.0.

- `/etc/config/system` que configura els paràmetres del sistema d'un dispositiu:

```
config system
        option timezone 'UTC'
        option log_file '/var/log/messages'
        option hostname 'OpenWrt'

config timeserver 'ntp'
        option enable_server '0'
        list server '0.openwrt.pool.ntp.org'
        list server '1.openwrt.pool.ntp.org'
        list server '2.openwrt.pool.ntp.org'
        list server '3.openwrt.pool.ntp.org'
```

En aquest segon exemple, hi ha dues seccions. La primera és una secció anònima de tipus `system`. Aquesta secció té tres opcions que defineixen: la zona horària, en aquest cas UTC, on es desaran els fitxers de log, en aquest cas a /var/log/messages i el nom del dispositiu, que en aquest cas és OpenWrt.

La segona secció és de tipus `timeserver` i té per nom `ntp`. Té una opció que permet establir si el servidor està habilitat o no. A més, té una llista, és a dir el paràmetre de nom `server` té més d'un valor, concretament té 4 valors: `0.openwrt.pool.ntp.org`, `1.openwrt.pool.ntp.org`, `2.openwrt.pool.ntp.org` i `3.openwrt.pool.ntp.org`, que permeten establir diferents servidors de temps.

2.4 Els camins UCI

Per accedir al valor d'una opció s'ha d'establir un camí fins a l'opció. Aquest camí es pot establir o bé usant el nom de la secció, si no és una secció anònima, o usant el tipus i especificant en quina posició està aquest.

Es mostra a continuació un exemple per tal d'entendre els camins UCI. Donat el següent fitxer de configuració:

```
# /etc/config/foo
config bar 'first'
        option name    'Mr. First'
config bar
        option name    'Mr. Second'
```

Per accedir a la primera opció es poden establir els següents camins:

```
# Mitjançant el nom de la secció
foo.first.name         # fitxer de configuració foo -> secció first -> opció name
# Mitjançant el tipus de la secció, posant @nom_tipus[posició]
foo.@bar[0].name       # fitxer de configuració foo -> primera secció de tipus bar -> opció name
```

Per accedir a la segona opció només es pot fer a partir del tipus, ja que es tracta d'una secció anònima:

```
foo.@bar[1].name       # fitxer de configuració foo -> segona secció de tipus bar -> opció name
```

2.5 La comanda `uci`

Tots els fitxers de configuració són editables, però es pot usar la comanda `uci` per tal de gestionar-los. A continuació es mostren alguns exemples d'utilització d'aquesta comanda:

- **uci show**: mostra les opcions d'un fitxer de configuració

```
root@OpenWrt:~# uci show httpd        # Mostrar les opcions del fitxer /etc/config/httpd
httpd.@httpd[0]=httpd                 # La primera secció de tipus httpd
httpd.@httpd[0].port=80               # La primera secció de tipus httpd té una opció de nom
                                      port i valor 80
httpd.@httpd[0].home=/www             # La primera secció de tipus httpd té una opció de nom
                                      home i valor /www
```

- **uci get**: obté el valor d'una opció

```
# Obtenir el valor de l'opció port de la primera secció de tipus httpd
root@OpenWrt:~# uci get httpd.@httpd[0].port
80      # El valor és 80
```

- **uci set**: estableix el valor d'una opció

```
# En el fitxer de configuració httpd, dins la secció main, establim que l'opció listen_http
tingui valor 8080
root@OpenWrt:~# uci set uhttpd.main.listen_http=8080
```

- **uci add**: crea una secció anònima

```
# Crear una secció anònima en el fitxer de configuració playapp de tipus blah
root@OpenWrt:~# uci add playapp blah
```

- **uci delete**: esborra una opció

```
# Esborrar l'opció server de la secció ntp del fitxer de configuració system
root@OpenWrt:~# uci delete system.ntp.server
```

- **uci commit**: aplica els canvis

```
# Aplica els canvis fets al fitxer uhttpd
root@OpenWrt:~# uci commit uhttpd
```

Per a més usos de la comanda uci i els seus paràmetres, es pot consultar la documentació oficial: http://wiki.openwrt.org/doc/uci

3 El projecte luporum

El projecte *luporum* pretén oferir una altra via per configurar els fitxers del sistema *UCI*. La idea general és que puguem fer peticions al dispositiu des d'aplicacions externes per tal de gestionar tots els valors del sistema *UCI*.

3.1 Motivació del projecte

Com ja s'ha comentat a l'apartat 2.2, *OpenWrt* disposa de dues maneres de configurar els programes que usen *UCI*, via consola o via l'aplicació web *LuCI*. Aquests dos mètodes tenen dues mancances:

- **No es pot configurar més d'un dispositiu a la vegada:** tant si s'entra a un dispositiu via *ssh* com si s'accedeix a l'aplicació web *LuCI*, només es poden gestionar els valors de configuració d'aquell dispositiu. Si s'estan administrant diversos dispositius, es veu la necessitat de poder modificar tots aquests dispositius alhora des d'una mateixa aplicació, enlloc d'haver de configurar-los d'un en un.

  Existeix una altra aplicació, `ruci`, que centralitza les configuracions dels dispositius en un repositori `git`. Aquesta solució fa que es pugui configurar més d'un dispositiu alhora, però obliga a fer-ho d'una única manera, amb un repositori `git`. Per tant no aporta la versatilitat que es desitja. Per a més informació respecte `ruci`, es pot consultar la seva web: https://github.com/libre-mesh/ruci

- **Si s'incorpora un nou programa gestionat amb UCI, l'aplicació web LuCI no l'incorpora automàticament:** LuCI és una aplicació web molt potent, però quan s'incorpora un nou programa, s'ha d'esperar a que algun desenvolupador incorpori la seva gestió a l'aplicació. Seria desitjable que només tenir un fitxer de configuració nou, l'aplicació el detectés i el permetés configurar automàticament.

Aquestes dues mancances donen peu a pensar una altra filosofia d'accés a la informació de la configuració *UCI*. La idea és crear una API REST que permeti fer peticions demanant informació d'aquests fitxers de manera que s'hi pugui accedir de manera externa.

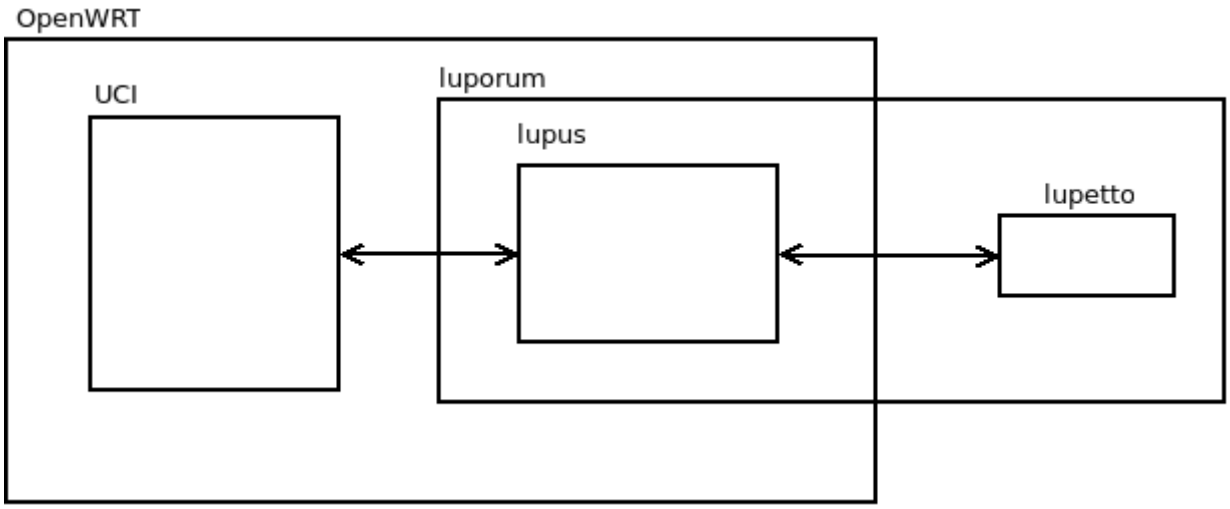
3.2 Descripció del projecte

Per tal de suplir les dues mancances comentades a l'apartat 3.1, es decideix desenvolupar una aplicació que permeti modificar valors del sistema UCI de manera genèrica i que permeti aplicar una mateixa configuració a diversos dispositius alhora.

Per tal d'aconseguir aquests dos objectius es desenvoluparà una API REST que serà instal·lada en cadascun dels dispositius i que permetrà, entre d'altres, obtenir les configuracions de cadascun dels dispositius i establir-hi nous valors. A partir d'aquesta API, es poden desenvolupar clients que facin peticions a cadascuna de les APIs i així configurar cadascuns dels dispositius amb els mateixos valors o no, a discreció del client.

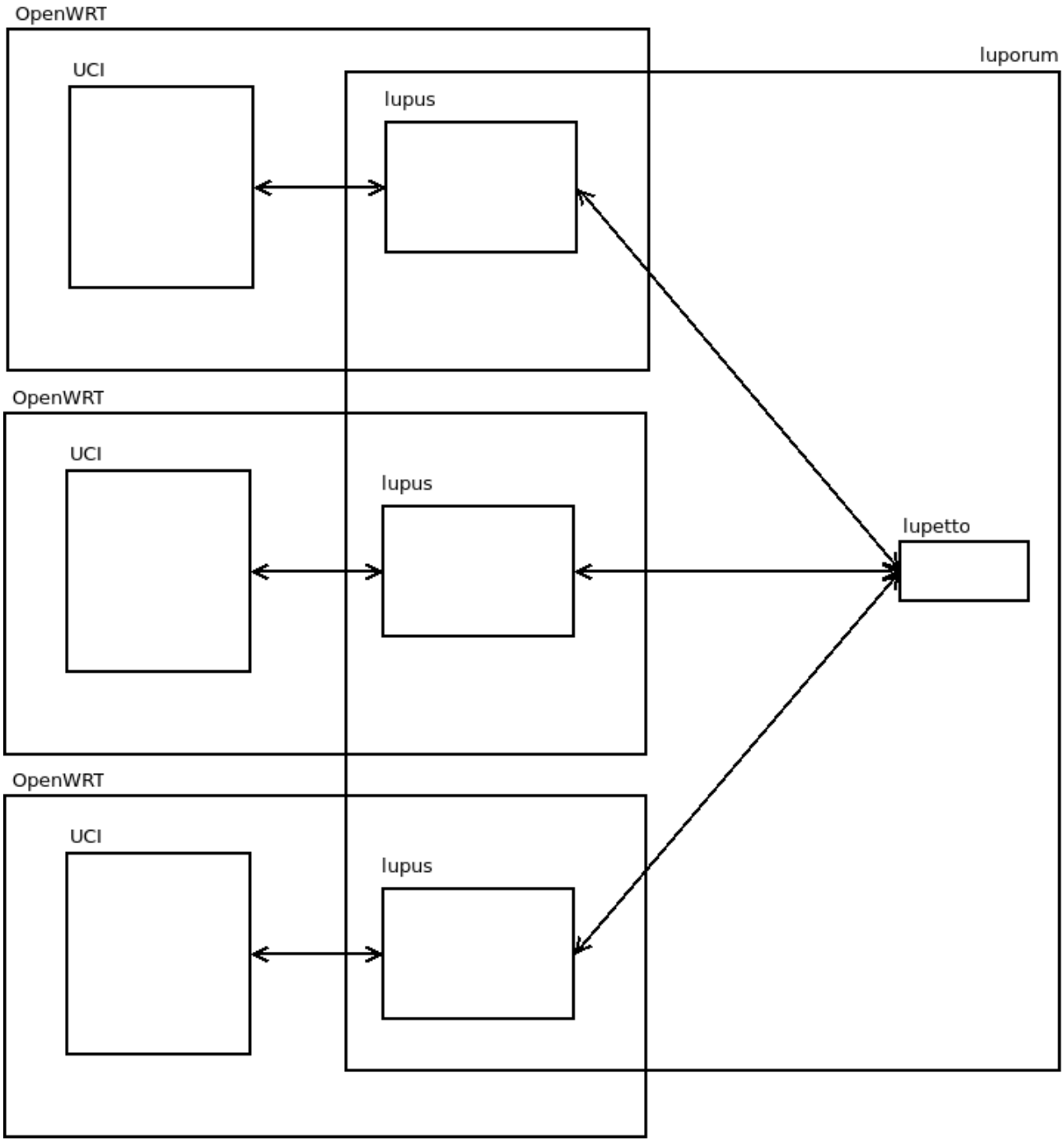
El projecte *luporum* es divideix en el desenvolupament de dues aplicacions: el subprojecte *lupus*, una API REST que farà la funció de servidor i estarà instal·lada en cadascun dels dispositius i el subprojecte *lupetto,* una aplicació client d'exemple que gestionarà aquests dispositius a partir de fer peticions a les APIs de cada dispositiu.

Si es disposa d'un únic dispositiu, es pot representar el sistema *luporum* amb el següent esquema:



És a dir, es té instal·lat *lupus* en el sistema *OpenWRT* que es comunica internament amb els valors del sistema *UCI*. *lupetto*, instal·lat en un ordinador o un altre dispositiu com a client remot, farà peticions a *lupus* obtenint o modificant la informació de la configuració.

Si es disposa de diversos dispositius, es pot representar el sistema *luporum* amb el següent esquema:



És a dir, *lupetto*, que tal i com s'ha dit anteriorment està instal·lat en un altre ordinador o dispositiu remot, es comunica amb els diversos *lupus* instal·lats als altres dispositius, de manera que es poden gestionar les configuracions de diversos dispositius des d'una mateixa aplicació.

4 Anàlisi

A continuació es detallen els requisits que han de complir tant l'API REST *lupus* com el client *lupetto*.

4.1 Anàlisi funcional

L'anàlisi funcional proporciona quins són els requisits que es demanen a ambdues aplicacions. És a dir, s'especifiquen les funcionalitats bàsiques que han de complir des del punt de vista de l'usuari.

4.1.1 lupus

lupus és l'API REST, el subprojecte principal de *luporum,* i és l'aplicació que permet gestionar tota la informació del dispositiu on està instal·lada. Algunes de les funcionalitats són similars a l'ordre de la línia de comandes `uci`. S'ha decidit emular la comanda `uci` per tal de facilitar l'aprenentatge als coneixedors d'*UCI*.

Els requeriments del sistema s'expressen en forma d'històries d'usuari:

Història d'usuari	**Autenticació**
Número	1
Descripció	Com a administrador d'un node, vull poder definir un mètode d'autenticació per tal que els clients de l'API REST s'hagin d'autenticar per fer una petició.
Validació	Un cop he establert un mètode d'autenticació, l'usuari s'ha d'autenticar en fer la petició. Si no s'identifica o ho fa incorrectament no podrà fer la petició i obtindrà un error. Si s'autentica correctament, podrà fer la petició.

Història d'usuari	**Obtenir els noms dels fitxers de configuració**
Número	2
Descripció	Com a usuari de *lupus*, vull obtenir els noms dels fitxers de configuració d'un node.
Validació	S'obté un llistat amb els noms dels fitxers de configuració. Si es produeix algun error s'enviarà a l'usuari.

Història d'usuari	**Obtenir els noms dels fitxers de configuració amb les seves seccions i opcions**
Número	3
Descripció	Com a usuari de *lupus*, vull obtenir els noms dels fitxers de configuració d'un node i que es detalli les seves seccions i opcions per a cadascun d'ells.
Validació	S'obté un llistat amb els noms dels fitxers de configuració amb les seves seccions i les seves opcions. Si es produeix algun error s'enviarà a l'usuari.

Història d'usuari	Obtenir les seccions i opcions d'un fitxer de configuració
Número	4
Descripció	Com a usuari de *lupus*, vull enviar el nom d'un fitxer de configuració i obtenir les seves seccions i opcions.
Validació	S'obté un llistat amb els noms dels fitxers de configuració amb les seves seccions i les seves opcions. Si es produeix algun error s'enviarà a l'usuari.

Història d'usuari	Afegir una o més seccions anònimes
Número	5
Descripció	Com a usuari de *lupus*, vull enviar el nom d'un fitxer de configuració i un tipus i que s'afegeixi una secció anònima del tipus donat en el fitxer de configuració donat. Vull poder enviar més d'un tipus i fitxer de configuració i que se m'afegeixin les seccions anònimes pertinents.
Validació	S'afegeixen les seccions anònimes en els fitxers de configuració pertinents. Si es produeix algun error s'enviarà a l'usuari.

Història d'usuari	Esborrar una secció d'un fitxer de configuració
Número	6
Descripció	Com a usuari de *lupus*, vull enviar el nom d'un fitxer de configuració i una secció i vull que aquesta s'esborri.
Validació	S'esborra la secció del fitxer de configuració. Si es produeix algun error s'enviarà a l'usuari.

Història d'usuari	Esborrar una opció d'un fitxer de configuració
Número	7
Descripció	Com a usuari de *lupus*, vull enviar el nom d'un fitxer de configuració, una secció i una opció i vull que aquesta s'esborri.
Validació	S'esborra l'opció de la secció del fitxer de configuració. Si es produeix algun error s'enviarà a l'usuari.

Història d'usuari	Obtenir el valor d'una opció
Número	8
Descripció	Com a usuari de *lupus*, vull enviar el nom d'un fitxer de configuració, una secció i una opció i obtenir el valor d'aquesta opció.
Validació	S'obté el valor de l'opció que està a la secció del fitxer de configuració sol·licitat. Si es produeix algun error s'enviarà a l'usuari.

Història d'usuari	Obtenir el tipus d'una secció
Número	9
Descripció	Com a usuari de *lupus*, vull enviar el nom d'un fitxer de configuració i una secció i obtenir el tipus de la secció.
Validació	S'obté el tipus de la secció del fitxer de configuració sol·licitat. Si es produeix algun error s'enviarà a l'usuari.

Història d'usuari	Establir el valor d'una opció
Número	10
Descripció	Com a usuari de *lupus*, vull enviar el nom d'un fitxer de configuració, una secció i una opció i establir el seu valor. Si l'opció no existeix aquesta s'afegeix.
Validació	Es canvia o afegeix el valor de l'opció demanada. Si es produeix algun error s'enviarà a l'usuari.

Història d'usuari	Reiniciar un node
Número	11
Descripció	Com a usuari de *lupus*, vull reiniciar un node.
Validació	El node es reinicia. Si es produeix algun error s'enviarà a l'usuari.

Història d'usuari	Iniciar un servei d'un node
Número	12
Descripció	Com a usuari de *lupus*, vull enviar el nom d'un servei i que aquest s'inicii.
Validació	El servei s'engega. Si es produeix algun error s'enviarà a l'usuari.

Història d'usuari	Aturar un servei d'un node
Número	13
Descripció	Com a usuari de *lupus*, vull enviar el nom d'un servei i que aquest s'aturi.
Validació	El servei s'atura. Si es produeix algun error s'enviarà a l'usuari.

Història d'usuari	Reiniciar un servei d'un node
Número	14
Descripció	Com a usuari de *lupus*, vull enviar el nom d'un servei i que aquest es reinicii.
Validació	El servei es reinicia. Si es produeix algun error s'enviarà a l'usuari.

4.1.2 *lupetto*

lupetto és un exemple d'aplicació client que usa *lupus* per tal d'obtenir informació d'un node. Es tracta d'una web des d'on es poden gestionar diversos dispositius consultant l'API *lupus*, instal·lada a cada dispositiu.

Els requeriments del sistema s'expressen en forma d'històries d'usuari:

Història d'usuari	**Afegir un node**
Número	15
Descripció	Com a usuari de *lupetto*, vull enviar la IP d'un node, per tal de consultar la seva configuració. Si no hi ha cap node afegit es mostren els seus fitxers de configuració amb el seu contingut. Si ja hi ha un node afegit, s'afegiran les configuracions d'aquest node, visualitzant quins valors té cada opció per a cadascun dels nodes afegits (mirar esquema).
Validació	Es mostren els fitxers de configuració d'aquest node. Si es produeix algun error s'enviarà a l'usuari.

Es mostra un esquema del que es vol aconseguir:

En afegir el primer node, es mostren les configuracions d'aquell node:

```
config1
        section11
                option111       node1   value1111
                option112       node1   value1121
                option113       node1   value1131
        section12
                option121       node1   value1111
                option122       node1   value1221
        ...
config2
        ...
config3
        ...
```

Si afegim un altre node, es fusionen els dos fitxers, mostrant els valors que tenen a cada node:

```
config1
        section11
                option111       node1   value1111
                                node2   value1112
                option112       node1   value1121      (el node2 no té aquesta opció)
                option113       node1   value1131
                                node2   value1132
        section12
                option121       node1   value1211
                                node2   value1212
                option122       node1   value1221
                                node2   value1222
                option123       node1   value1231
                                node2   value1232
                option124       node2   value1242 (el node1 no té aquesta opció)

        ...
config 2
        ...
config3
        ...
```

Història d'usuari	**Posar un valor a una opció d'un o més dispositius**
Número	16
Descripció	Com a usuari de *lupetto*, vull poder modificar el valor d'una opció d'un o més dispositius.
Validació	El valor o valors dels dispositius queden modificats.

Història d'usuari	Treure un node
Número	17
Descripció	Com a usuari de *lupetto*, vull poder treure un node ja afegit.
Validació	Ja no es mostraran les configuracions d'aquest node. Si es produeix algun error s'enviarà a l'usuari.

4.2 Anàlisi no funcional

Les tecnologies usades per tal de desenvolupar el projecte *luporum* són les següents:

- **Llenguatge de programació LUA:** l'API es desenvoluparà en aquest llenguatge de programació degut a l'agilitat d'aquest llenguatge, a que forma part de la base l'OpenWRT i a que es disposa d'una llibreria que ens permet gestionar de manera fàcil el sistema UCI, la llibreria libuci-lua. Es detalla més informació respecte aquesta llibreria en l'apartat 5.3.2.1.
- **JSON**: L'intercanvi d'informació es fa el format JSON. Es pot trobar informació d'aquest format en la seva web: http://www.json.org/
- **HTML, CSS i Javascript**: el desenvolupament de l'aplicació client é una web que mostra algunes de les funcionalitats de l'API desenvolupada, com ara la capacitat de poder configurar diversos dispositius alhora.
- **Bootstrap**: en el disseny del client web s'usa el framework Bootstrap, que fa que el desenvolupament de la maquetació i disseny web sigui molt més ràpid. Es detallen els motius d'usar aquest framework en l'apartat 5.4.2.1.
- **jQuery**: en el desenvolupament del client web s'usarà la llibreria de Javascript jQuery, per tal de facilitar i accelerar el desenvolupament de l'aplicació. Es detallen els motius d'usar aquest framework en l'apartat 5.4.2.2.
- **Connexions remotes via SSH**: l'accés per gestionar els dispositius on allotjarem l'API desenvolupada serà via SSH.
- Sistemes de virtualització **libvirt + qemu + kvm:** Per tal de fer les proves necessàries amb els dispositius es treballarà amb màquines virtuals.
- Sistema de control de versions **git**: el projecte es desenvoluparà en un repositori git.
- Ús bàsic de la plataforma de gestió de projectes **Redmine**: l'empresa allotja tots els seus projectes en un servidor usant aquesta aplicació. La URL és la següent: http://dev.qmp.cat/projects

4.3 Planificació

La metodologia àgil de treball Scrum està pensada per a equips de treball. Tot i que aquest projecte es realitza individualment, s'han usat algunes de les seves tècniques per tal d'elaborar el pla de treball.

4.3.1 Product backlog

El *product backlog* és el llistat de tasques a fer, prioritzades i amb un pes que valora si és una tasca més o menys llarga.

Per tal d'elaborar el *product baclkog* es desglossa el projecte en tasques i es valora cadascuna d'aquestes tasques amb un número de la sèrie de Fibonacci, on els nombres cada cop estan més distanciats.

L'avantatja de mesurar les tasques amb nombres de Fibonacci és que quan una tasca és molt llarga, costa molt dir quantes hores necessitem es necessiten per dur-la a terme. És per això que s'usen els nombres de Fibonacci, que fan que no s'hagi d'afinar tant per a tasques llargues. Es pot

assignar un nombre gran, per exemple 100, però no cal que es decideixi entre 100 o 101, ja que molt probablement s'estaria cometent un error de precisió.

El *product backlog* del projecte *luporum* és el següent:

Subprojecte	Tasca	Històries d'usuari	Punts
lupus	Disseny i implementació del nucli de l'API		13
	Obtenir els noms dels fitxers de configuració	2	3
	Obtenir les seccions	3, 4	3
	Afegir seccions anònimes	5	3
	Esborrar seccions i opcions	6, 7	3
	Obtenir el valor d'una opció o el tipus d'una secció	8, 9	8
	Posar un valor o valors d'un paràmetre UCI o afegir seccions no anònimes	10	5
	Autenticació	1	21
	Engegar, parar i reiniciar un servei	11, 12, 13	13
	Confecció del paquet lupus		2
lupetto	Disseny del client web		8
	Afegir un dispositiu	15	8
	Obtenir els fitxers de configuració dels dispositius afegits	15	8
	Mostrar seccions i opcions dels dispositius	15	13
	Posar un valor a una opció d'un o més dispositius	16	13
	Treure un dispositiu	17	5
	Distribució de lupetto		1
	Total		**130**

Cadascuna de les tasques inclou la realització de la seva documentació i la realització de les proves pertinents.

També es pot observar que s'han escrit les històries d'usuari relacionades amb cadascuna de les tasques descrites.

A la taula anterior es pot veure que el projecte *luporum* "pesa" 130 punts. Aquests punts s'hauran de repartir en *sprints*, tal i com es veurà al següent apartat.

4.3.2 Sprints

Un *sprint* és un període de temps on es desenvolupen tantes tasques com es pot. Després de cada *sprint* es queda amb el client, en aquest cas la mateixa empresa Routek SL, i es fa una

demostració de les funcionalitats desenvolupades. Així es pot determinar si allò que s'està desenvolupant és realment allò que el client vol, si s'ha de fer algun canvi, afegir alguna tasca, refer les prioritats...

En l'apartat anterior s'ha confeccionat el *product backlog* del projecte *luporum* i s'ha vist que té un pes total de 130 punts. Els *sprints* s'han fet de 20 dies i s'han desenvolupat uns 30 punts per a cada *sprint*. S'han fet doncs un total de 4 *sprints*.

Com es veurà a les següents taules, els punts no s'han repartit homogèniament, s'ha decidit aquesta distribució per tal de poder desenvolupar primer el projecte *lupus* i després el projecte *lupetto*. Així doncs els primers *sprints* estan una mica més carregats de feina.

Els sprints resultants són els següents:

1er sprint: 8 de març - 27 març

Subprojecte	Tasca	Punts
lupus	Disseny i implementació del nucli de l'API	13
	Obtenir els noms dels fitxers de configuració	3
	Obtenir les seccions	3
	Afegir seccions anònimes	3
	Esborrar seccions i opcions	3
	Obtenir el valor d'una opció o el tipus d'una secció	8
	Posar un valor o valors d'un paràmetre UCI o afegir seccions no anònimes	5
	Total	**38**

2on sprint: 28 març - 16 abril

Subprojecte	Tasca	Punts
lupus	Autenticació	21
	Engegar, parar i reiniciar un servei	13
	Total	**34**

3er sprint: 17 abril - 6 maig

Subprojecte	Tasca	Punts
lupetto	Disseny del client web	8
	Afegir un dispositiu	8
	Obtenir els fitxers de configuració dels dispositius afegits	8
	Treure un dispositiu	5
	Total	**29**

4rt sprint: 7 maig - 26 maig

Subprojecte	Tasca	Punts
lupetto	Mostrar seccions i opcions dels dispositius	13
	Posar un valor a una opció d'un o més dispositius	13
	Confecció del paquet lupus	2
	Distribució de lupetto	1
	Total	**29**

5 Implementació

5.1 Entorn de desenvolupament

5.1.1 Màquines virtuals

Per tal d'emular un dispositiu amb una distribució OpenWRT/qMp s'han usat màquines virtuals amb `libvirt/qemu/kvm`.

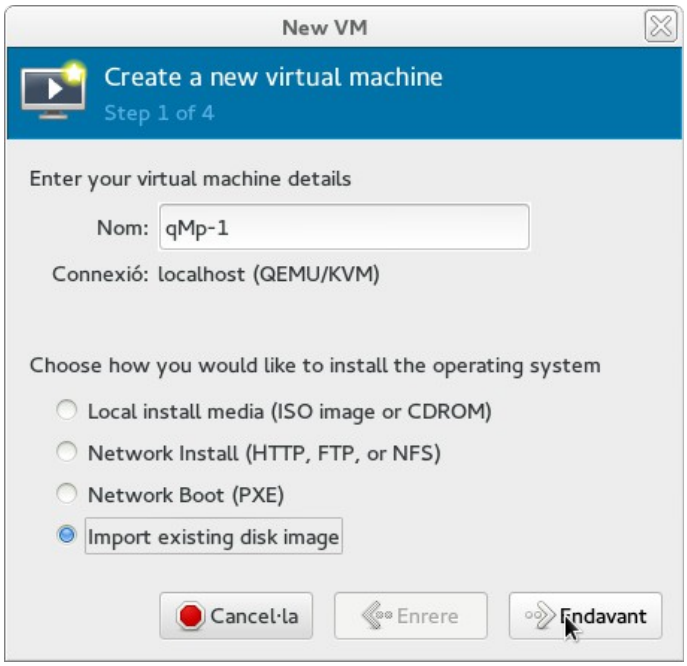
S'han segut les instruccions que es poden trobar a http://qmp.cat/Development, i adapant-ho a les necessitats del projecte:

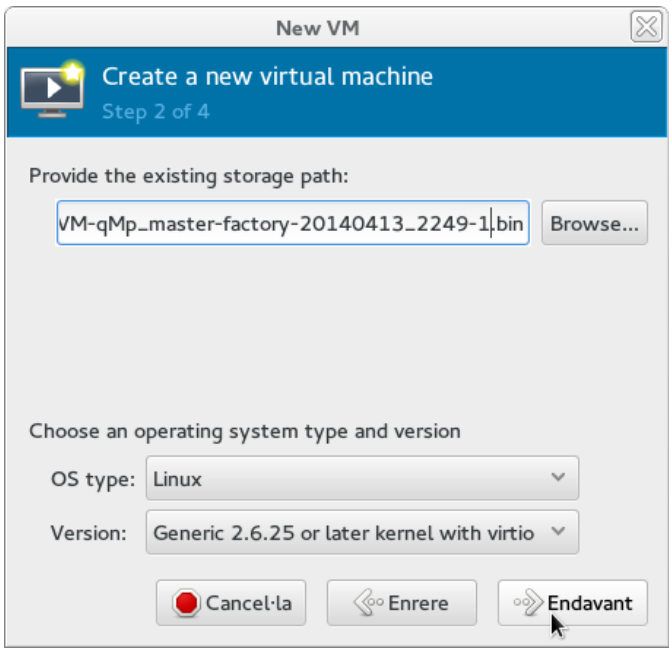
```
$ sudo aptitude install git subversion zlib1g-dev gawk flex unzip bzip2 gettext build-essential
libncurses5-dev libncursesw5-dev binutils cpp psmisc docbook-to-man gcc-multilib

$ git clone git://qmp.cat/qmpfw.git qmpfw

$ make list_targets
ar71xx bullet kvm nsm2 nsm5 pico2 rocket rs rspro tl-2543 tl-703n tl-wr841n-v7 tl-wr841n-v8 tl-842
tl-mr3020 tl-mr3040 tl-mr3040-cam tl-wdr3600 tl-wdr4300 alix x86 vbox vmware wpe72 ath-ib

$ make build T=kvm J=3
```

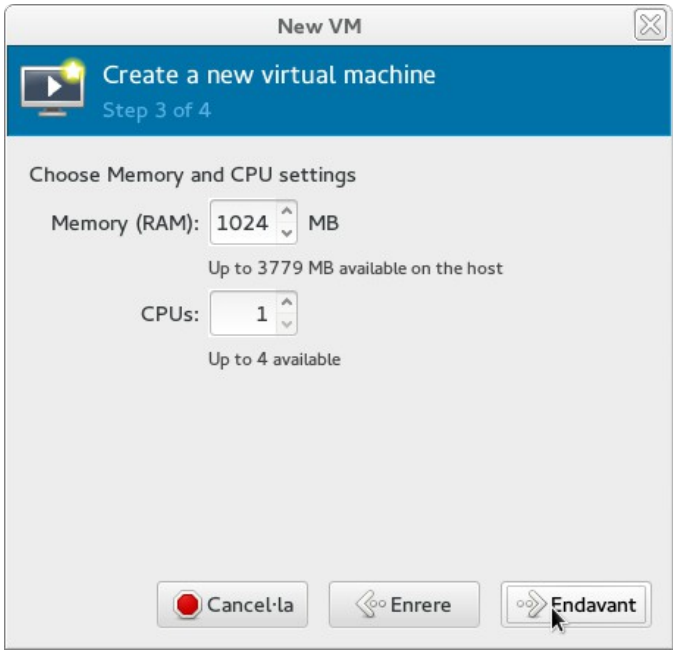
Amb aquestes ordres s'ha aconseguit una màquina virtual *KVM* amb *qMp*. Se'n fa una còpia per tal de mantenir la màquina original i s'importa:

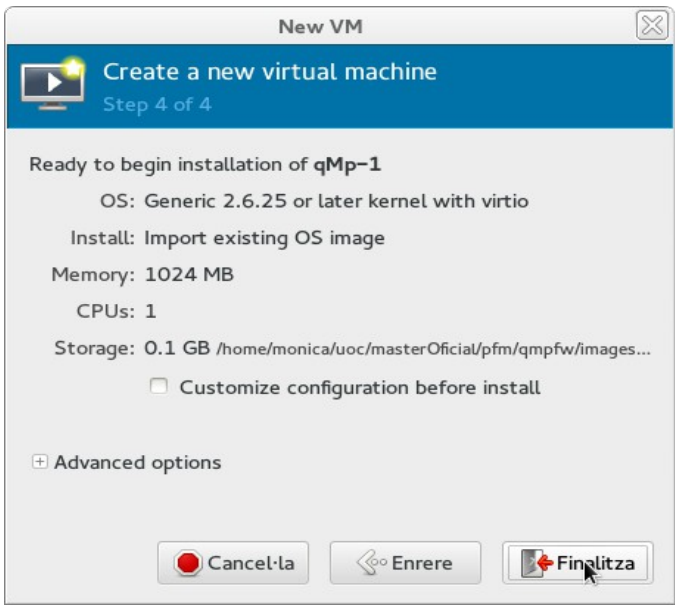
```
cp KVM-qMp_master-factory-20140413_2249.bin KVM-qMp_master-factory-20140413_2249-1.bin
```

La importació es fa a partir de la interfície gràfica `virt-manager`. A continuació es poden veure els passos seguits a partir de captures de pantalla:

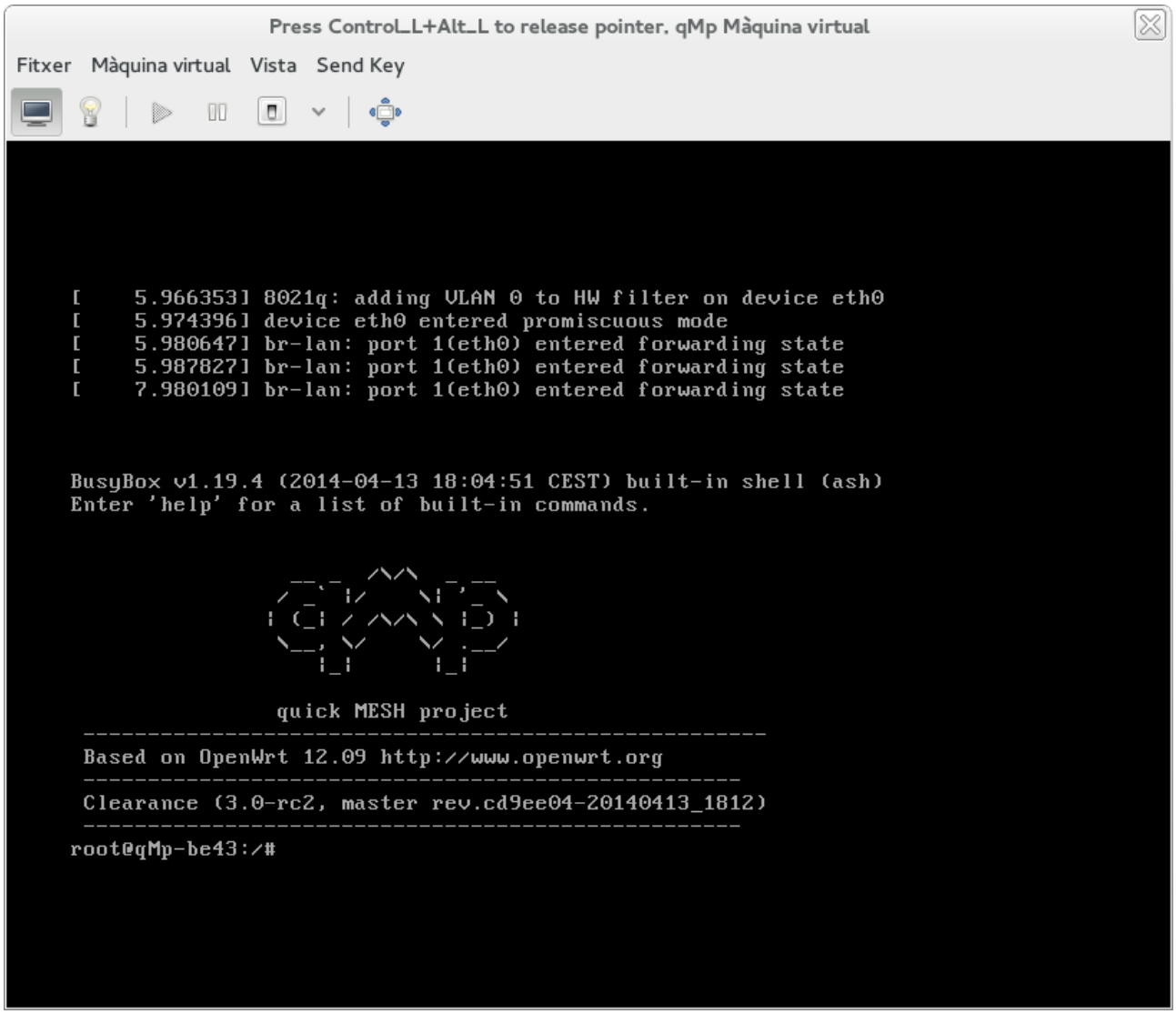



New VM
Create a new virtual machine
Step 2 of 4
Provide the existing storage path:
VM-qMp_master-factory-20140413_2249-1.bin
Browse...
Choose an operating system type and version
OS type:
Linux
Version:
Generic 2.6.25 or later kernel with virtio
Cancel·la
Enrere
Endavant


New VM
Create a new virtual machine
Step 3 of 4
Choose Memory and CPU settings
Memory (RAM):
1024
MB
Up to 3779 MB available on the host
CPUs:
1
Up to 4 available
Cancel·la
Enrere
Endavant



En finalitzar aquest pas, ja s'ha creat la màquina virtual amb qMp:

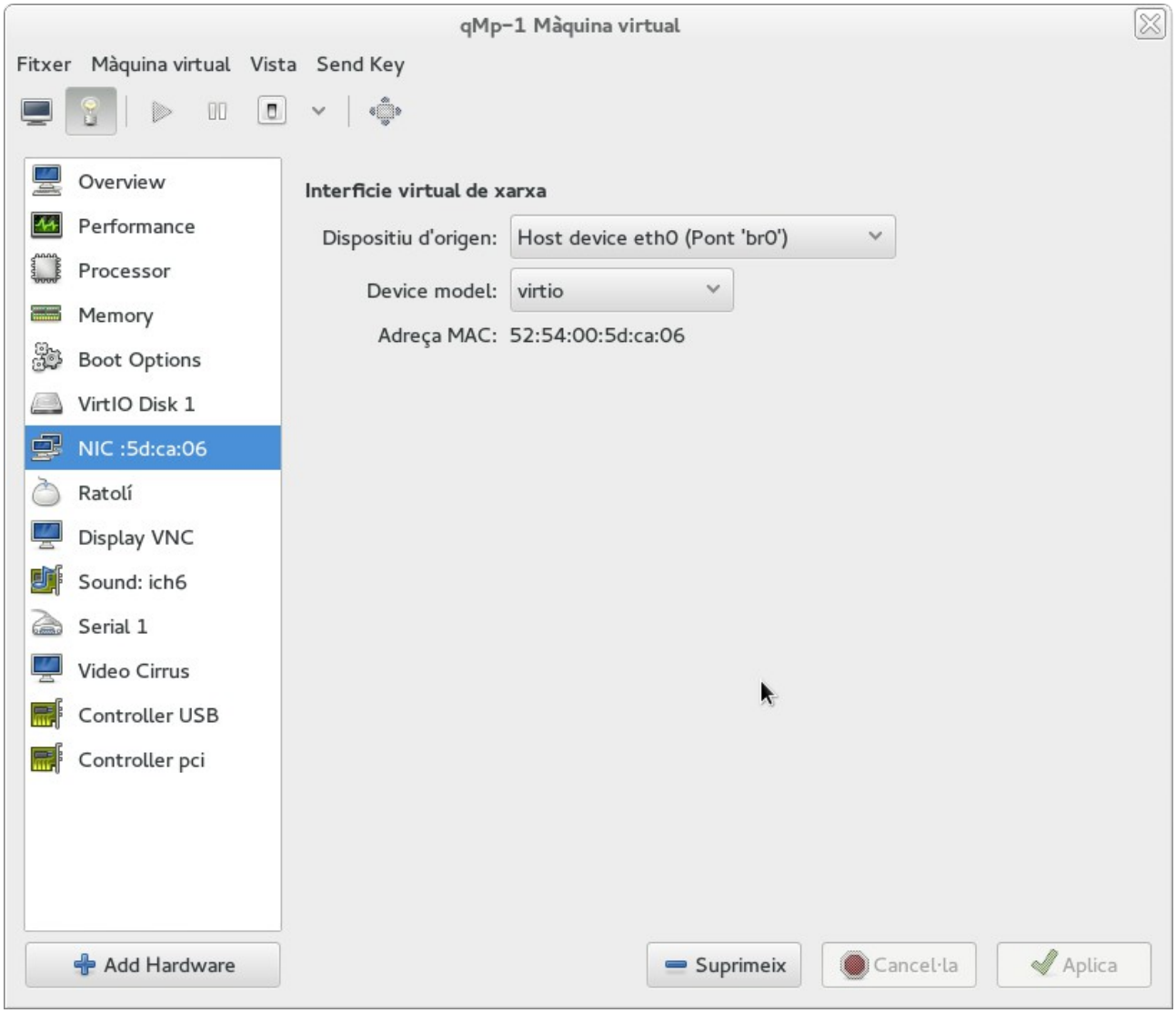


Ara es vol aconseguir comunicar la màquina host amb el node *qMp*.

S'ha de fer a partir d'un bridge. A la màquina host es disposa d'un bridge que connecta la seva interfície de xarxa amb la interfície de la màquina virtual:

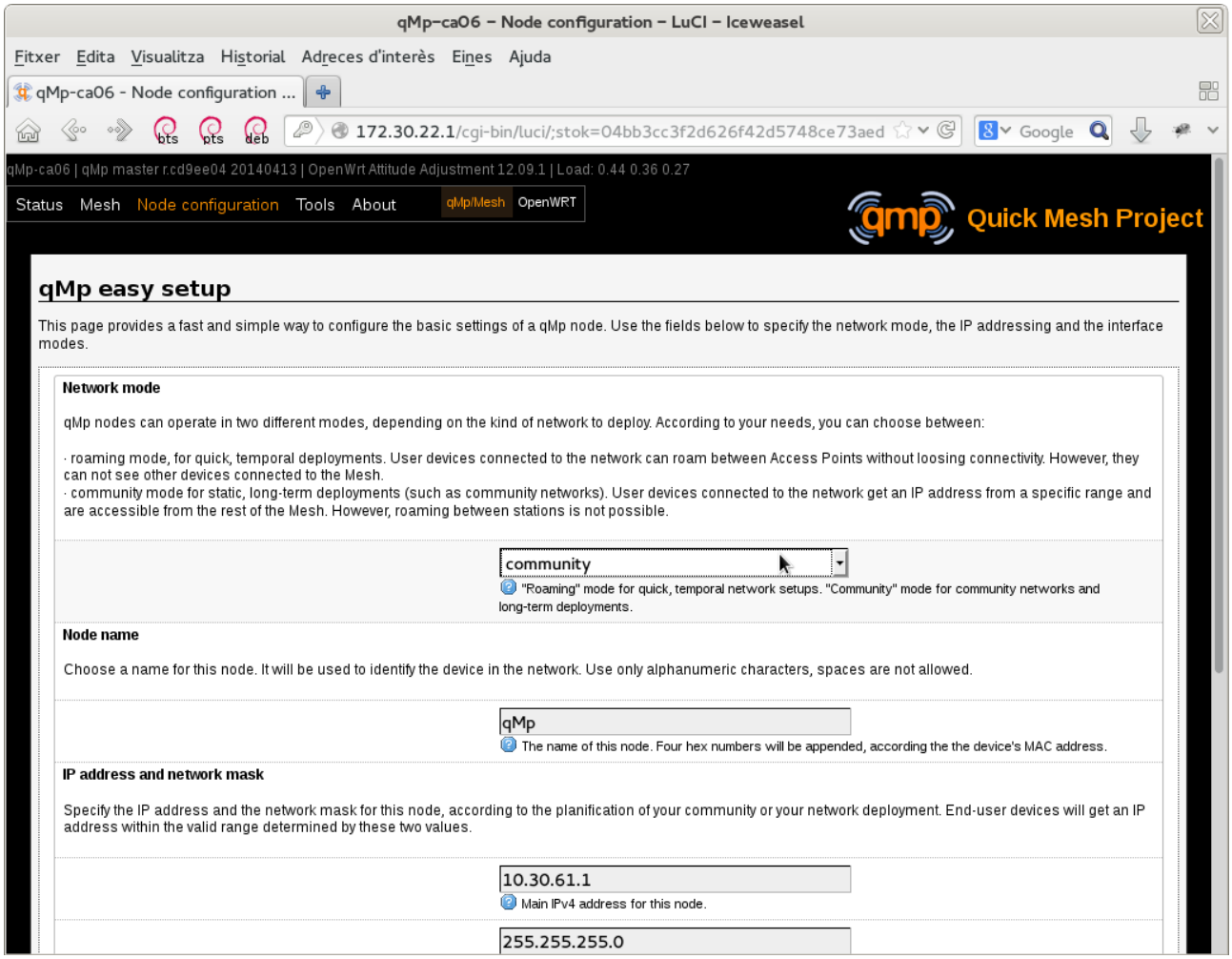
```
$ sudo brctl addif br0 vnet0
$ sudo brctl stp br0 yes
$ sudo brctl show
bridge name    bridge id              STP enabled    interfaces
br0            8000.aa0004000a04      yes            eth0
                                                     vnet0
```

A la configuració de la interfície de xarxa de la màquina virtual, està referenciat aquest bridge (br0):



El node *qMp* per defecte té una adreça IP de la xarxa 172.30.0.0/16, més concretament, té la IP 172.30.22.1. Aquest dispositiu té un servidor DHCP activat i dóna una adreça IP d'aquesta xarxa.

Es configura el node per tal que tingui el mode *community*, ja que és el mode més usat pels usuaris de qMp:



Ara es té una nova adreça en el node, la 10.30.61.1/24. Farem que la màquina host estigui també en aquesta xarxa per tal de que es puguin comunicar:

```
$ sudo ip a a 10.30.61.5/24 dev br0
```

A partir d'ara ja es pot entrar com a *root* via SSH:

```
ssh root@10.30.61.1
```

Per tal d'instal·lar *lupus* s'ha d'executar la següent comanda des d'on es té el codi:

```
make install-remote HOST=10.30.61.1
```

O bé, instal·lar el paquet *lupus*, que conté l'última versió de l'aplicació. Per més detalls respecte l'ús del paquet *lupus*, mirar l'apartat 6.1.

5.1.2 IDE

Per a l'edició del codi s'ha optat per l'IDE Eclipse juntament amb els següents plugins:

- **LDT (Lua Development Tools)**: eines per al desenvolupament de codi amb llenguatge de programació Lua. Més informació a http://www.eclipse.org/koneki/ldt/
- **ShellEd (Eclipse Shell Script Editor)**: editor de shell scripting. Més informació a http://sourceforge.net/apps/trac/shelled/
- **elt (Terminal plug-in for Eclipse)**: terminal incorporada dins d'Eclipse. Més informació a https://code.google.com/p/elt/
- **wtp (Web Tools Platform):** eines per al desenvolupament web. Més informació a http://www.eclipse.org/webtools/

- **jsdt-jquery (jQuery support for JSDT)**: ajuda en la compleció de codi *jQuery*. Més informació a http://code.google.com/a/eclipselabs.org/p/jsdt-jquery/

5.1.3 Gestió del projecte i sistema de control de versions

La gestió del projecte es du a terme mitjançant *Redmine* i el sistema de control de versions usat és *Git*. Es pot obtenir el codi del projecte a partir de les següents URLs:

- Pàgina principal del projecte *luporum*: http://dev.qmp.cat/projects/luporum
- Pàgina principal del subprojecte *lupus*: http://dev.qmp.cat/projects/lupus
- Pàgina principal del subprojecte *lupetto*: http://dev.qmp.cat/projects/lupetto

5.2 Entorn de producció

Quan es volen mostrar els resultats en dispositius reals s'han de seguir els següents passos:

1. Connectar el node a l'ordinador amb un cable de xarxa
2. L'ordinador demana una adreça IP via DHCP
3. Executar `ip r`, el gateway per defecte serà l'adreça IP del node
4. Baixar l'última versió del codi font a l'ordinador:

   ```
   git clone git://qmp.cat/luporum/lupus.git
   ```

5. Generar una clau ssh i copiar-la al node:

   ```
   ssh-keygen
   ssh-copy-id -i ~/.ssh/id_rsa.pub remote-host
   ssh root@remote-host
   cat .ssh/authorized-keys >> /etc/dropbear
   rm .ssh/authorized-keys
   ```

6. El fitxer `opkg.conf` té un error. S'ha de canviar la primera línia on posa .../12.09.1/... per .../12.09/...
7. Des de l'ordinador, s'instal·la l'API:

   ```
   make HOST=remote-host install-remote
   ```

A més de poder seguir els passos anteriors, també s'ha desenvolupat un paquet que conté l'última versió de *lupus*. És molt més senzill instal·lar *lupus* a partir de la instal·lació d'aquest paquet. Per veure més detalls del desenvolupament i l'ús del paquet *lupus*, mirar l'apartat 6.1.

5.3 lupus

En aquest apartat es descriu com s'ha implementat l'API *lupus*. Quan s'escaigui, a l'inici de l'explicació es mostra un requadre amb la història d'usuari corresponent que implementa. Es poden veure totes les històries d'usuari de lupus a l'apartat 5.1.1

5.3.1 Estructura del codi

Per tal d'implementar el codi s'ha intentat fer-ho de la manera més modular possible i usant el patró de disseny Model-Vista-Controlador.

Per tal de fer-se una idea de com està estructurat el codi, es pot donar un cop d'ull als fitxers que s'han implementat:

```
$ tree
.
|-- AUTHORS
|-- doc
|   |-- lupus-doc.html
|   `-- lupus-doc.markdown
|-- etc
|   `-- config
```

```
|       `-- lupus
|-- INSTALL
|-- LICENSE
|-- Makefile
|-- README
|-- src
|   |-- cgi
|   |   `-- lupus
|   |-- core
|   |   |-- constants.lua
|   |   |-- Controller.lua
|   |   `-- Validator.lua
|   |-- libs
|   |   `-- dkjson.lua
|   |-- main.lua
|   |-- modules
|   |   |-- auth
|   |   |   |-- AuthAction.lua
|   |   |   |-- Authconstants.lua
|   |   |   `-- AuthController.lua
|   |   |-- system
|   |   |   |-- SystemAction.lua
|   |   |   |-- SystemConstants.lua
|   |   |   |-- SystemController.lua
|   |   |   `-- SystemValidator.lua
|   |   `-- uci
|   |       |-- UCIAction.lua
|   |       |-- UCIconstants.lua
|   |       |-- UCIController.lua
|   |       |-- uciutils.lua
|   |       `-- UCIValidator.lua
|   `-- utils
|       |-- files.lua
|       `-- strings.lua
|-- test
|   |-- api
|   |   |-- test_add.sh
|   |   |-- test_all.sh
|   |   |-- test_delete.sh
|   |   |-- test_env.sh
|   |   |-- test_functions.sh
|   |   |-- test_generic.sh
|   |   |-- test_get_configs.sh
|   |   |-- test_get.sh
|   |   |-- testlupus
|   |   |-- test_service.sh
|   |   `-- test_set.sh
|   `-- core
`-- TODO
```

A continuació es detalla els directoris i fitxers mostrats:

- `doc`: documentació tècnica de l'API. La documentació està escrita en format `markdown` i es transforma a format `html` amb la comanda `markdown`.
- `etc`: fitxer de configuració de *lupus*. En aquest fitxer s'estableix quin tipus d'autenticació es vol usar.
- `src`: directori que conté el codi font de *lupus.*
- `src/cgi`: CGI que s'encarrega de rebre les peticions. És la llançadora de l'API.
- `src/core`: directori que conté el controlador genèric de l'aplicació i el validador del format genèric de les peticions.
- `src/libs`: directori que conté les llibreries externes que no es troben empaquetades per a *OpenWRT*. L'única llibreria externa no empaquetada és la llibreria *dkjson*. Es parla més àmpliament d'aquesta llibreria en l'apartat 5.3.2.3
- `src/modules`: directori amb els mòduls de l'API implementats. Els mòduls disposen d'un controlador, que esbrina quina acció es vol fer, un validador que valida que les dades siguin correctes segons la petició que s'està efectuant i un fitxer d'accions on s'implementen les

diverses accions que s'han de fer segons la petició. S'han desenvolupat els següents mòduls:

  - `auth`: autenticació. Determina si el client s'ha autenticat o no.
  - `system`: realitza diverses accions en el sistema.
  - `uci`: realitza les accions pertinents amb els valors del sistema UCI.

- `src/utils`: implementació de diverses funcions genèriques.
- `test`: directori amb diversos shell scripts que avaluen el correcte funcionament de l'API. Aquests scripts fan ús de la comanda `curl`.

5.3.2 Llibreries i dependències externes

Algunes de les funcionalitats necessàries per codificar *lupus* no s'han hagut de codificar des de zero, ja que existeixen llibreries fetes amb LUA que ens les proporcionen.

A continuació s'esmenta perquè són necessàries aquestes llibreries i eines per al desenvolupament de *lupus* i perquè s'han escollit aquestes i no unes altres.

5.3.2.1 Accés al sistema UCI

Per accedir al sistema UCI, es disposa d'una llibreria molt potent anomenada `libuci-lua`. Aquesta llibreria permet executar comandes molt similars a la comanda `uci` de la línia de comandes.

Per exemple, l'equivalent de la comanda

```
uci get system.ntp.enable_server
```

que mostra per pantalla el valor de l'opció `enable_server` de la secció `ntp` del fitxer de configuració `system` és:

```
x = uci.cursor()
value = x:get("system", "ntp", "enable_server")
```

En aquest cas el valor està emmagatzemat a la variable `value`.

Aquesta llibreria té un paquet OpenWRT, de nom `libuci-lua` que ja ve instal·lat per defecte.

5.3.2.2 Servidor web

Per tal de fer peticions a l'API *lupus* s'ha de cridar un CGI que serà executat per un servidor web. La distribució *OpenWRT* incorpora un servidor web senzill per tal de poder oferir l'aplicació web LuCI. S'usa aquest mateix servidor web per executar el CGI que contestarà a les peticions que es facin a l'API.

Aquest servidor web està empaquetat per a *OpenWRT* i el nom del paquet és *uhttpd*. Com ja s'ha copmentat *uhttpd* ja ve instal·lat per defecte a la distribució.

5.3.2.3 *Parsing* i formatació JSON-LUA

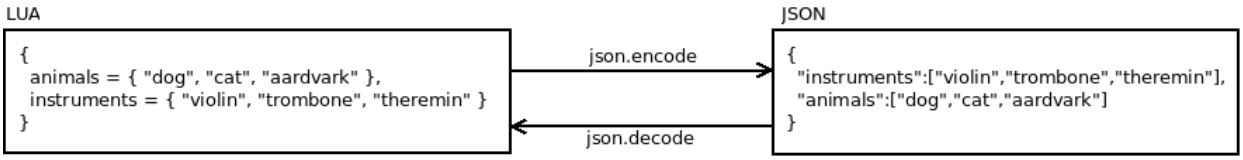
L'intercanvi d'informació entre l'API *lupus* i els possibles clients es fa mitjançant el format JSON. És necessari doncs convertir el format JSON en taules LUA i a la inversa, convertir taules LUA en format JSON.

Aquesta funcionalitat és molt genèrica i per tant, ja hi hagut desenvolupadors que han fet llibreries que ho resolen. Afortunadament, existeixen diverses llibreries LUA amb llicència lliure que proporcionen totes les funcions necessàries. En l'enllaç http://lua-users.org/wiki/JsonModules es disposa d'una comparativa d'aquestes llibreries.

La llibreria escollida a *lupus* és *dkjson*. Els motius pels quals s'ha escollit aquesta llibreria són els següents:

- **Llicència lliure:** *dkjson* està llicenciat sota una llicència MIT/X11.

- **Facilitat d'ús:** el projecte *lupus* necessita només les funcionalitats més simples: codificar (LUA->JSON) i decodificar (JSON->LUA). Amb aquesta llibreria és tan fàcil com usar la funció *json.encode* per passar de LUA a JSON i la funció *json.decode* per passar de JSON a LUA.



  Per a més informació d'ús de la llibreria, es pot consultar la pàgina web de *dkjson*: http://dkolf.de/src/dkjson-lua.fsl/home

- **Senzillesa d'instal·lació:** es tracta d'un únic fitxer que s'incorpora a l'aplicació que l'ha d'usar. Tot i això, seria millor disposar del paquet per a *OpenWRT*.

5.3.2.4 Funcions addicionals del sistema de fitxers

Tot i que LUA porta incorporades algunes funcions relacionades amb el sistema de fitxers, durant el desenvolupament s'ha necessitat incorporar una funcionalitat que no estava en el nucli del llenguatge.

Concretament, s'ha necessitat recórrer un directori obtenint cadascun dels seus fitxers per tal de fer alguna acció en cadascun d'ells. Per tal de fer això s'ha usat la funció `lfs.dir` de la llibreria `luafilesystem`. Per obtenir més informació d'aquesta llibreria es pot consultar la seva pàgina web: http://keplerproject.github.io/luafilesystem/

Afortunadament, existeix el paquet *OpenWRT* per a aquesta llibreria. Per tant la instal·lació és molt senzilla. Només s'ha d'executar la següent comanda:

```
opkg install luafilesystem
```

5.3.3 Format de les peticions

Les peticions que es fan a *lupus* han de seguir un format concret. Algunes peticions es poden fer posant les dades de petició a la URL (GET). D'altres necessitaran que es donin les dades via POST.

Quan s'envien dades via POST aquestes estaran en un fitxer amb format JSON del següent estil:

```
{ "module1_name1":
       { data },
  "module1_name2":
       { data },
  ...
}
```

Per exemple:

```
{ "auth":
       { "password":"lupuspassword"},
  "uci": {
          "network.testset":"valueset",
          "network.valueset.optionset":"optionvalue"}
}
```

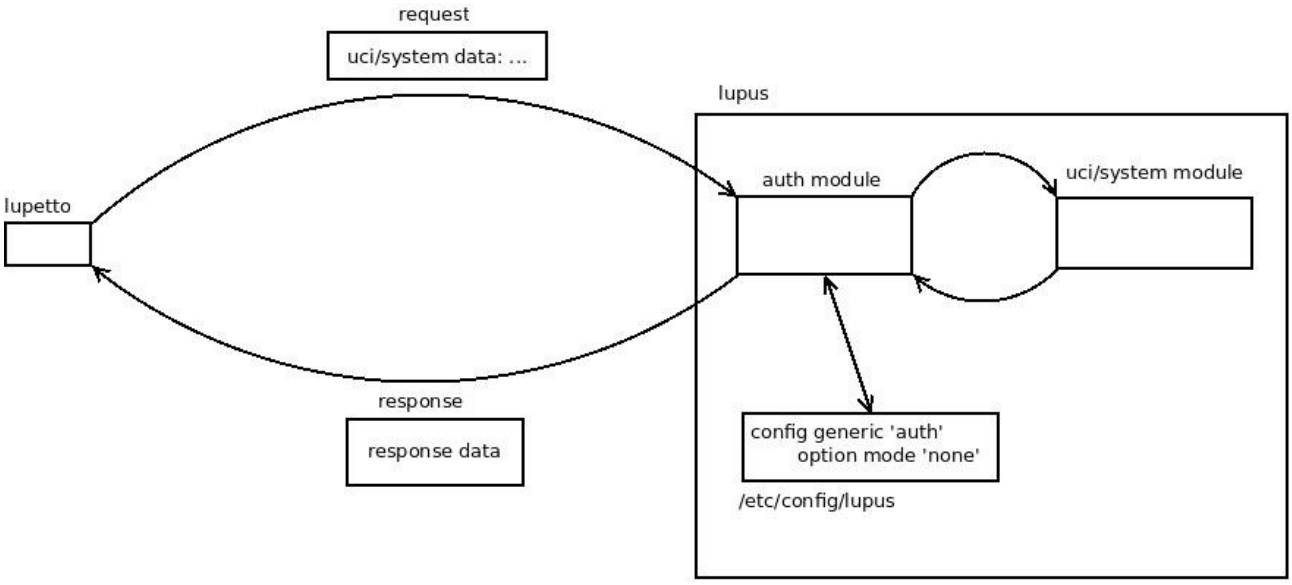
En aquest exemple s'estan passant dades tan al mòdul d'autenticació com al mòdul UCI.

5.3.4 Mòdul auth

Història d'usuari: 1

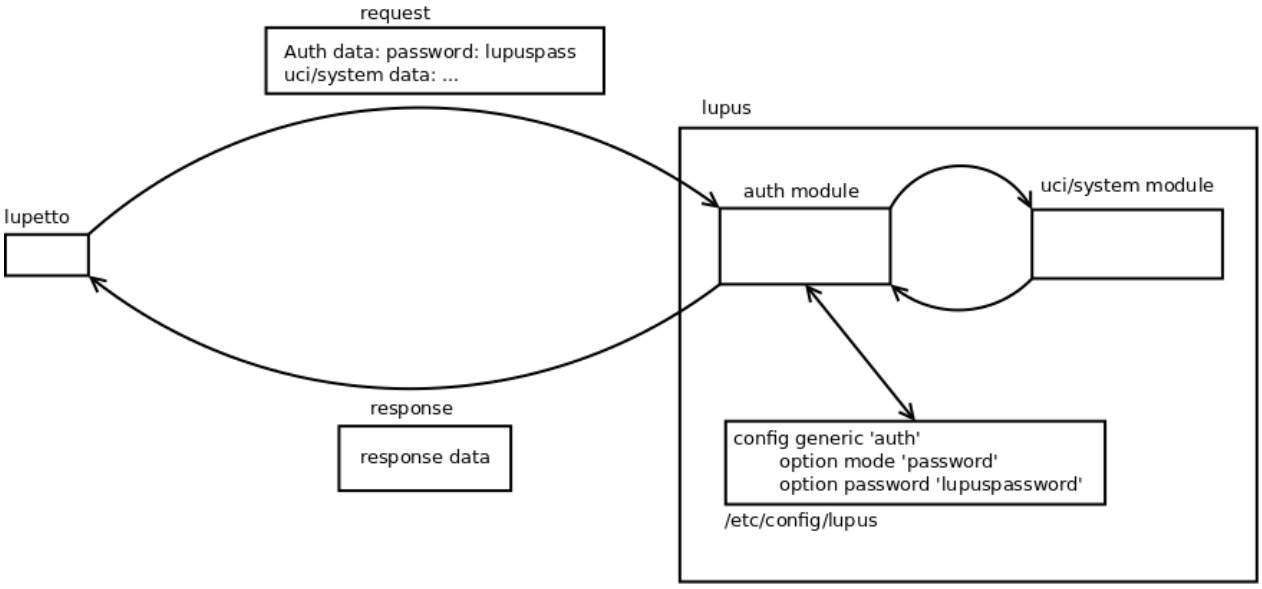
Hi ha diversos mètodes per realitzar una autenticació. El que s'ha intentat en el mòdul **auth** és que sigui molt senzill afegir un nou mètode d'autenticació. S'han elaborat dues autenticacions molt senzilles, a tall d'exemple:

• `none`: No es fa cap tipus d'autenticació. La petició es pot realitzar sense autenticar-se. Compliria el següent esquema:



Si no hi ha autenticació no s'ha de passar cap dada d'autenticació a la petició.

• `password`: Per autenticar-se s'ha de proporcionar una contrasenya dins la petició que ha de coincidir amb la contrasenya desada en el fitxer de configuració de *lupus* (`/etc/config/lupus`).



La petició ha d'incloure les dades relacionades amb l'autenticació. És a dir, s'ha d'enviar la contrasenya dins la petició. El format és el següent:

```
{ "auth":
        { "password":"lupuspassword"}
}
```

5.3.5 Mòdul uci

El mòdul **uci** s'encarrega de totes les peticions relacionades amb l'obtenció de dades dels valors del sistema UCI com de la gestió d'aquestes.

La URL per fer la petició té el següent format genèric:

http://cgi_location/uci/action/action_parameters

on `uci` és el nom del mòdul, `action` l'acció que volem realitzar i `action_parameters` són paràmetres opcionals que els posarem o no segons l'acció que vulguem realitzar.

En els següents apartats s'expliquen cadascuna de les peticions que es poden fer.

5.3.5.1 add

Història d'usuari: 5

Afegeix una o més seccions anònimes dels tipus especificats als fitxers de configuració especificats.

Format URL: `/uci/add`

Les dades s'especifiquen per POST:

```
{
        "uci": {
                "config1.type1"[,
                "config2.type2",
                ...]
        }
}
```

Exemple:

Si s'envien les següents dades:

```
{
        "uci": {
                "network.type1",
                "system.type2"
        }
}
```

S'afegeixen les següents seccions anònimes:

```
# cat /etc/config/network:
...
config type1

# cat /etc/config/system:
...
config type2
```

5.3.5.2 delete

Esborra seccions o opcions d'un fitxer de configuració

Format URL: `/uci/delete`

Les dades s'especifiquen per POST.

Història d'usuari: 6

Per esborrar una secció sencera:

```
{
        "uci": {
                "config.section"
        }
}
```

Història d'usuari: 7

Per esborrar una opció d'una secció:

```
{
        "uci": {
                "config.section.option"
        }
}
```

Exemples:

El següent exemple esborrarà la segona secció de tipus `interface` del fitxer de configuració `network`:

```
{
        "uci": {
                "network.@interface[1]"
        }
}
```

El següent exemple esborrarà l'opció `ipaddr` de la segona secció de tipus `interface` del fitxer de configuració `network`:

```
{
        "uci": {
                "network.@interface[1].ipaddr"
        }
}
```

5.3.5.3 get

Si s'especifica una opció, s'obté el seu valor, si s'especifica el nom d'una secció, s'obté el seu tipus.

Format URL: `/uci/get/get_parameters`

Història d'usuari: 8

Per obtenir el valor d'una opció, es sol·licita una URL amb el següent format:

```
/uci/get/config.section.option
```

i s'obté una resposta amb el següent format:

```
{
        "config.section.option":"value"
}
```

Història d'usuari: 9

Per obtenir el tipus d'una secció, es sol·licita una URL amb el següent format:

```
/uci/get/config.section
```

i s'obté una resposta amb el següent format:

```
{
        "config.section":"type"
}
```

Exemples:

Si es sol·licita la següent URL:

```
/uci/get/system.ntp.enable_server
```

s'obté la següent resposta:

```
{
        "system.ntp.enable_server":"0"
}
```

Si es sol·licita la següent URL:

```
/uci/get/system.ntp.server
```

s'obté la següent resposta:

```
{
        "system.ntp.server":
["0.openwrt.pool.ntp.org","1.openwrt.pool.ntp.org","2.openwrt.pool.ntp.org","3.openwrt.pool.ntp.org"
]
        }
```

Si es sol·licita la següent URL:

```
/uci/get/system.@system[0].hostname
```

s'obté la següent resposta:

```
{
        "system.@system[0].hostname":"OpenWrt"
}
```

Si es sol·licita la següent URL:

```
/uci/get/network.lan.ipaddr
```

s'obté la següent resposta:

```
{
        "network.lan.ipaddr":"192.168.1.5"
}
```

Si es sol·licita la següent URL:

```
/uci/get/network.@interface[2].ipaddr
```

s'obté la següent resposta:

```
{
        "network.@interface[2].ipaddr":"192.168.1.5"
}
```

Si es sol·licita la següent URL:

```
/uci/get/network.loopback
```

s'obté la següent resposta:

```
{
        "network.loopback":"interface"
}
```

5.3.5.4 get_all

Història d'usuari: 3

Donat el nom d'un fitxer de configuració, obté totes les seccions i opcions d'aquest fitxer. Si no es dóna cap nom de fitxer de configuració, s'obtenen tos els noms de fitxer de configuració, amb les seves seccions i opcions.

Format URL: `/uci/get_all`

Història d'usuari: 4

Si es volen obtenir totes les seccions i opcions d'un fitxer de configuració concret, es sol·licita una URL amb el següent format:

```
/uci/get_all/config
```

I s'obté un resultat amb el següent format:

```
{
        "section1":{
        ".name":"section1",
        ".type":"section_type",
        ".index":0,
        ".anonymous":true|false,
        "option1":"value1",
        "option2":"value2",
        },
        "section2":{
                ".name":"sectin2",
                ".type":"section_type",
                ".index":1,
                ".anonymous":true|false,
                "option1":"value1",
                "option2":"value2",
        },
        ...
}
```

Si es volen obtenir totes les seccions i opcions de tots els fitxers de configuració, es sol·licita una

URL amb el següent format:

```
/uci/get_all
```

I s'obté un resultat amb el següent format:

```
{
	"config1":{
		"section1":{
			".name":"section1",
			".type":"section_type",
			".index":0,
			".anonymous":true|false,
			"option1":"value1",
			"option2":"value2",
		},
		"section2":{
			".name":"sectin2",
			".type":"section_type",
			".index":1,
			".anonymous":true|false,
			"option1":"value1",
			"option2":"value2",
		},
		...
	},
	"config2":{
		...
	},
	...
}
```

Exemples:

Si es sol·licita la següent URL:

```
/uci/get_all/system
```

s'obté el següent resultat:

```
{
	"ntp":{
	".name":"ntp",
	".type":"timeserver",
	".index":1,
	"enable_server":"0",
	".anonymous":false,
	"server":
["0.openwrt.pool.ntp.org","1.openwrt.pool.ntp.org","2.openwrt.pool.ntp.org","3.openwrt.pool.ntp.org"
]
	},
	"cfg02e48a":{
	".name":"cfg02e48a",
	".type":"system",
	"hostname":"OpenWrt",
	".index":0,
	".anonymous":true,
	"timezone":"UTC"
	}
}
```

5.3.5.5 get_configs

Història d'usuari: 2

Obté els noms dels fitxers de configuració.

Format URL: `/uci/get_configs`

El resultat obtingut té el següent format:

```
{
	"configs":["filename1","filename2",...]
}
```

Exemple:

Si es sol·licita la següent URL:

```
/uci/get_configs
```

s'obté el següent resultat:

```
        {
                "configs":
["luci","network","firewall","dhcp","system","ucitrack","uhttpd","dropbear"]
        }
```

5.3.5.6 set

Història d'usuari: 10

Donat el nom d'una o més opcions, estableix els seus valors. Si l'opció no existeix, l'afegeix. Si no es dóna una opció, s'afegeixen seccions amb el tipus donat.

Format URL: `/uci/set`

Les dades s'especifiquen per POST.

Les dades enviades tenen el següent format:

```
        {
                "uci": {
                        "config.section.option":"value",
                        "config.type":"sectionname",
                        ...
                }
        }
```

Exemples:

Si s'envien les següents dades

```
        {
                "uci": {
                        "network.loopback.ipaddr": "127.0.0.1"
                        "system.@system[0].hostname": "OpenWrt"
                        "system.ntp.server":
["0.openwrt.pool.ntp.org","1.openwrt.pool.ntp.org","2.openwrt.pool.ntp.org","3.openwrt.pool.ntp.org"
]
                        "system.@system[0].log_file": "/var/log/messages",
                        "network.interface":"wlan0"
                        "network.wlan0.ipaddr":"192.168.1.11"
                }
        }
```

Es modificaran els següents fitxers de configuració de la següent manera:

```
        # cat /etc/config/network:
        ...
        config interface loopback
                option ipaddr '127.0.0.1'

        config interface wlan0
                option ipaddr '192.168.1.11'

        # cat /etc/config/system:
        config system
                ...
                option hostname 'OpenWrt'
                option log_file '/var/log/messages'

        config timeserver 'ntp'
                ...
                list server '0.openwrt.pool.ntp.org'
                list server '1.openwrt.pool.ntp.org'
                list server '2.openwrt.pool.ntp.org'
                list server '3.openwrt.pool.ntp.org'
```

5.3.6 Mòdul system

El mòdul `system` s'encarrega de fer diverses accions relacionades amb la gestió del sistema.

La URL per fer la petició té el següent format genèric:

http://cgi_location/system/action/action_parameters

on `system` és el nom del mòdul, `action` l'acció que volem realitzar i `action_parameters` són paràmetres opcionals que es posen o no segons l'acció que es vulgui realitzar.

En els següents apartats s'expliquen cadascuna de les peticions que es poden fer.

5.3.6.1 reboot

Història d'usuari: 11

Reinicia el sistema.

Format URL: `/system/reboot`

5.3.6.2 restart_service

Història d'usuari: 14

Reinicia un servei del sistema.

Format URL: `/system/restart_service/service_name`

Exemple:

Si s'envia la següent petició:

```
/system/restart_service/firewall
```

es reiniciarà el servei `firewall`.

5.3.6.3 start_service

Història d'usuari: 12

Inicia un servei del sistema.

Format URL: `/system/start_service/service_name`

Exemple:

Si s'envia la següent petició

```
/system/start_service/firewall
```

s'iniciarà el servei `firewall`.

5.3.6.4 stop_service

Història d'usuari: 13

Atura un servei del sistema.

Format URL: `/system/stop_service/service_name`

Exemple:

Si s'envia la següent petició

```
/system/stop_service/firewall
```

s'aturarà el servei `firewall`.

5.3.7 Documentació

L'ús de l'API està especificat (en anglès) al fitxer `doc/lupus-doc.markdown`. Per obtenir un fitxer `.html` que es pugui veure des de qualsevol navegador, es pot fer de dues maneres:

O bé executant:

```
make gen-doc
```

o executant directament la comanda markdown:

```
markdown doc/lupus-doc.markdown > doc/lupus-doc.html
```

5.3.8 Tests

Per tal de fer tests s'han implementat una sèrie de shell scripts que fan un ús intensiu de la comanda `curl`. Per tal d'executar aquests tests, només s'han de canviar les variables que hi ha al shell-script `test_all.sh` i executar-lo.

5.4 lupetto

En aquest apartat es descriurà com s'ha implementat l'aplicació web *lupetto*. Quan escaigui, a l'inici de l'explicació es mostrarà un requadre amb la història d'usuari corresponent que implementa. Es poden veure totes les històries d'usuari de *lupetto* a l'apartat 5.1.2

La implementació de *lupetto* es basa en implementar una aplicació web feta en HTML, CSS i Javascript, usant el framework *Bootstrap* i la llibreria *jQuery*. La idea és que la web es pugui instal·lar a qualsevol lloc, tant en un ordinador com en el mateix node sense haver de fer la instal·lació de cap servidor web.

La web és completament dinàmica. És a dir, a partir de les peticions que es facin a l'API (*lupus*) es crearan totes les pantalles de l'aplicació, depenent dels fitxers de configuració que tingui, les seccions de cada fitxer i les seves opcions.

5.4.1 Estructura del codi

El codi implementat té els següents fitxers:

```
$ tree
.
|-- AUTHORS
|-- css
|   |-- bootstrap.min.css
|   `-- lupetto.css
|-- fonts
|   |-- glyphicons-halflings-regular.eot
|   |-- glyphicons-halflings-regular.svg
|   |-- glyphicons-halflings-regular.ttf
|   `-- glyphicons-halflings-regular.woff
|-- img
|   `-- wolf.svg
|-- index.html
|-- js
|   |-- bootstrap.min.js
|   |-- jquery.min.js
|   `-- lupetto.js
|-- LICENSE
|-- README
`-- TODO
```

A continuació, es detalla aquesta estructura:

- `css`: conté els fitxers CSS de l'aplicació. `bootstrap.min.css` és el fitxer CSS que ens ofereix el framework *Bootstrap* i `lupetto.css` és el que s'ha codificat per tal de personalitzar el disseny gràfic de l'aplicació.
- `fonts`: fonts que ofereix el framework *Bootstrap*.

- `img`: imatges de l'aplicació web *lupetto*.
- `index.html`: fitxer principal de l'aplicació web *lupetto*.
- `js`: fitxers Javascript. S'inclou: `bootstrap.min.js`, del framework *Bootstrap*, `lupetto.js`, codi Javascript propi de *lupetto* i `jquery.min.js`, usat tant per *Bootstrap* com pel codi javascript propi de *lupetto*.

5.4.2 Llibreries externes

Actualment, la majoria d'aplicacions web fetes amb HTML, CSS i Javascript usen llibreries externes per tal d'agilitzar el desenvolupament de l'aplicació ja que ofereixen moltes funcionalitats complexes ja desenvolupades.

5.4.2.1 Bootstrap

Bootstrap no es tracta d'una llibreria sinó d'un *framework* complet. El motiu d'usar-lo ha estat, en gran part, per tal d'estalviar-se tota la gestió de la maquetació de la pàgina web i molta part del disseny gràfic d'aquesta. *Bootstrap* ens ofereix eines per tal de tenir una web amigable sense haver de fer molt esforç.

En el cas de *lupetto*, s'ha usat *Bootstrap* per al següent:

- Maquetació de la pàgina principal.
- Confecció del menú tipus acordió amb els fitxers de configuració dels nodes afegits.
- Estil genèric de l'aplicació.
- Estil dels botons.
- Estil dels formularis.

Per a més informació de Bootstrap es pot consultar la seva pàgina web: http://getbootstrap.com/

5.4.2.2 jQuery

jQuery és una llibreria àmpliament usada per molts desenvolupadors de Javascript. Ofereix una àmplia varietat de funcionalitats i això fa que sigui usada per a tot tipus d'aplicacions.

lupetto usa intensivament *jQuery*. En concret s'usa per:

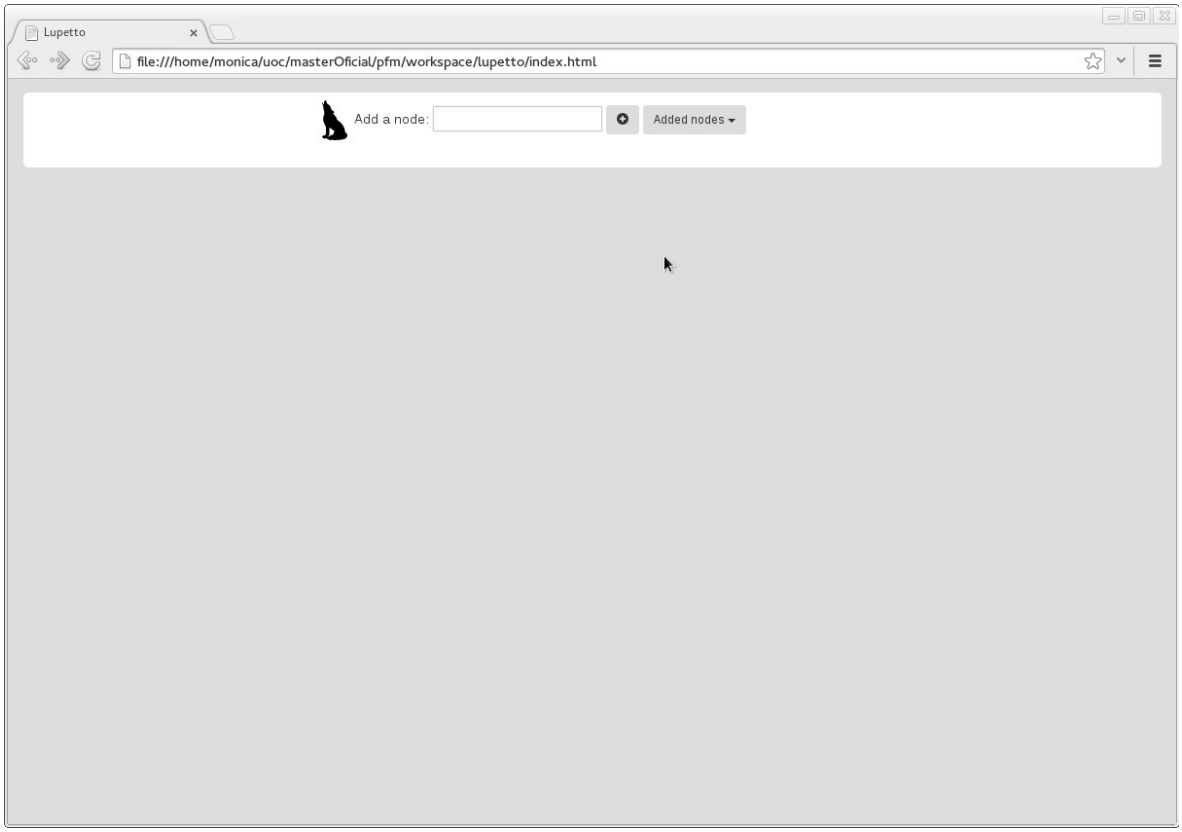
- Recórrer els objectes DOM
- Peticions Ajax

5.4.3 Interfície web

La interfície s'ha dissenyat intentant que sigui accessible des de qualsevol dispositiu. S'ha optat per una interfície molt senzilla, en la qual no calgui cap explicació addicional per tal que l'usuari aprengui a usar-la.

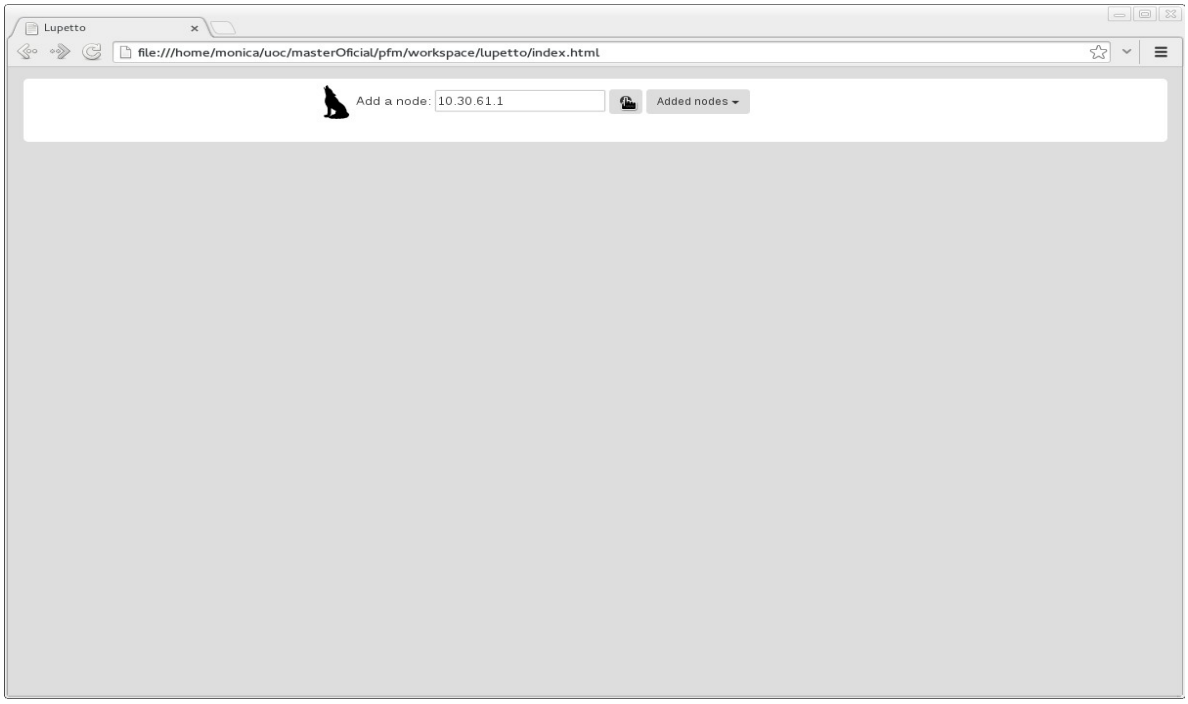
Es mostren a continuació les funcionalitats implementades a partir d'una breu explicació i captures de pantalla.

En obrir l'aplicació web *lupetto*, es disposa d'una interfície molt senzilla que permet afegir nodes:

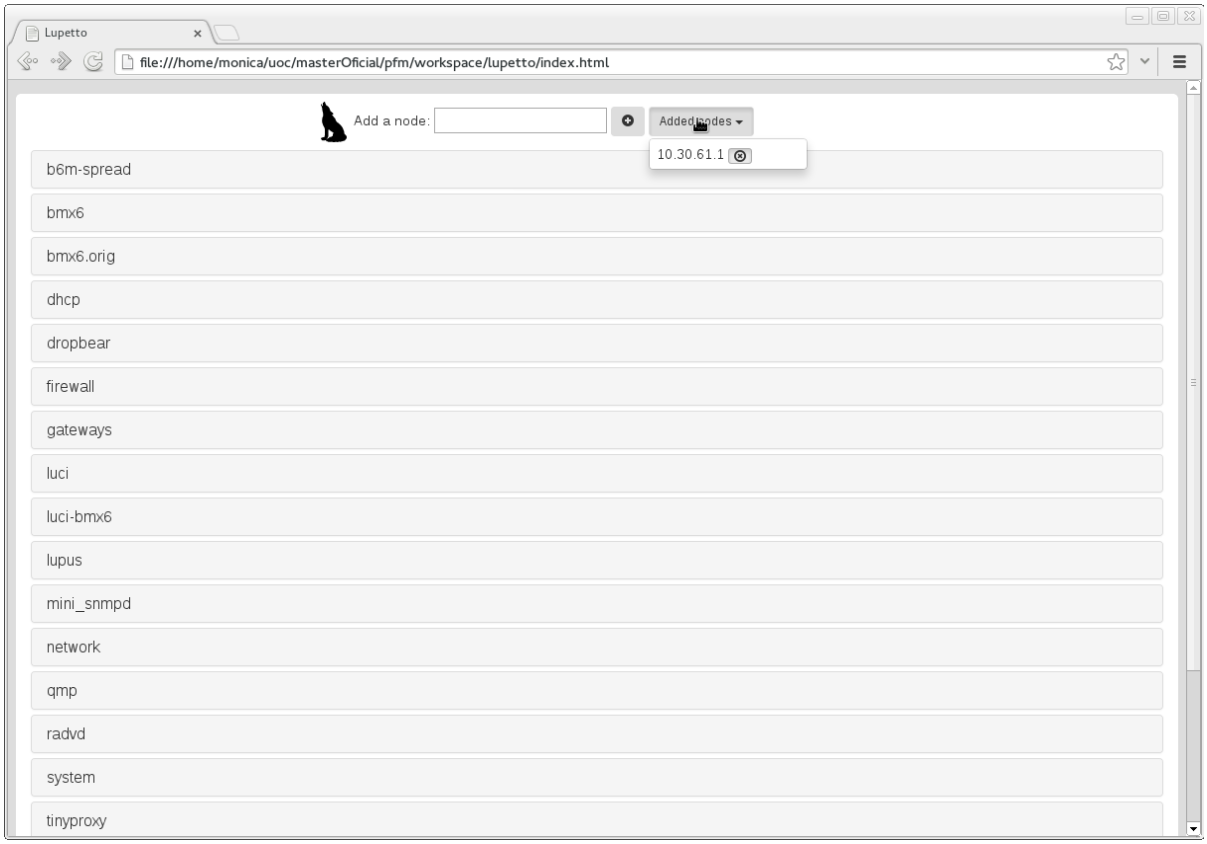


Se n'afegeix un:

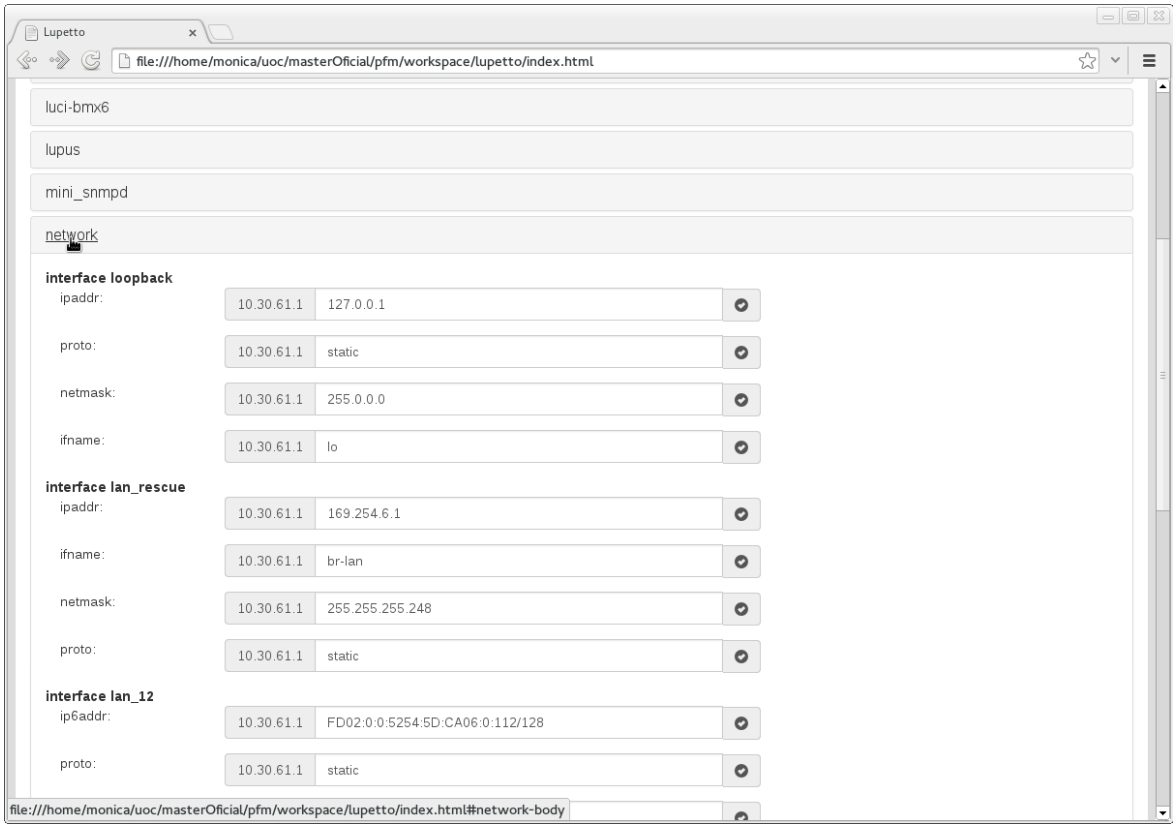
Història d'usuari: 1



Un cop afegit, es mostren tots els fitxers de configuració d'aquest node i s'afegeix a la llista de nodes afegits:

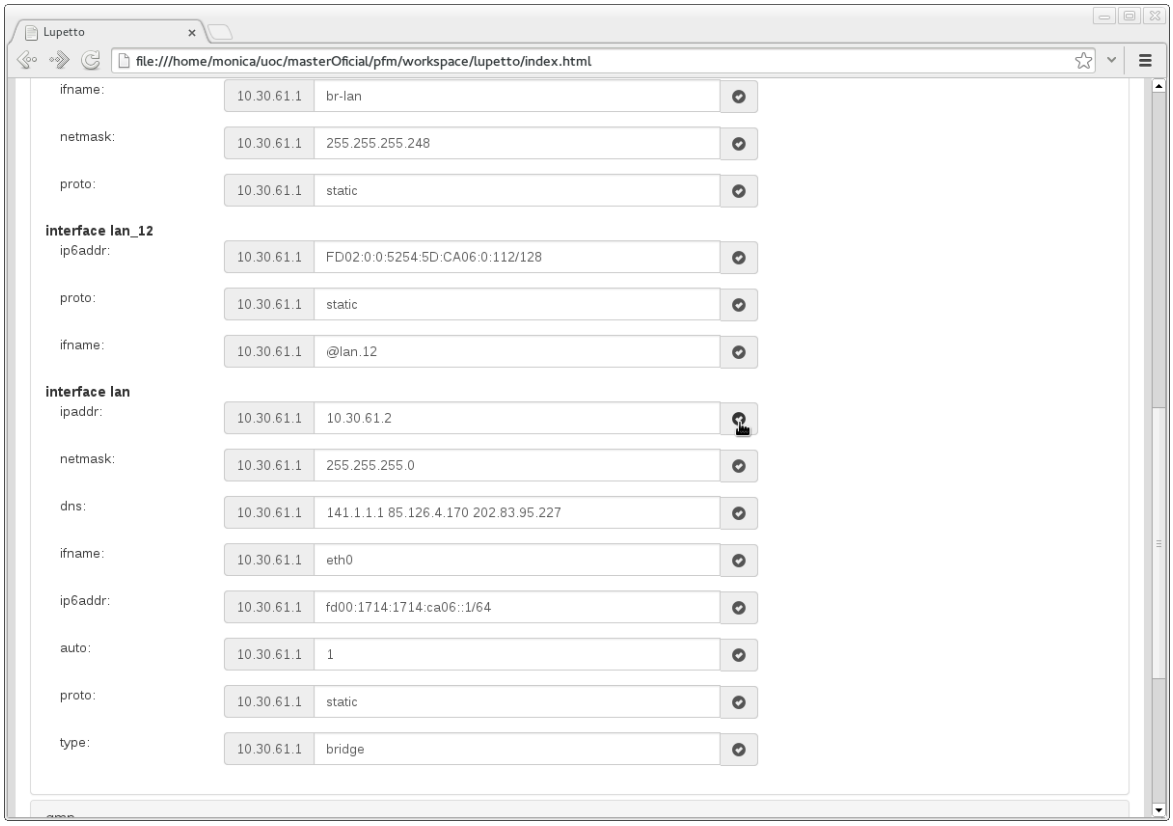


Ara es poden veure les seccions i opcions de cadascun dels fitxers de configuració:

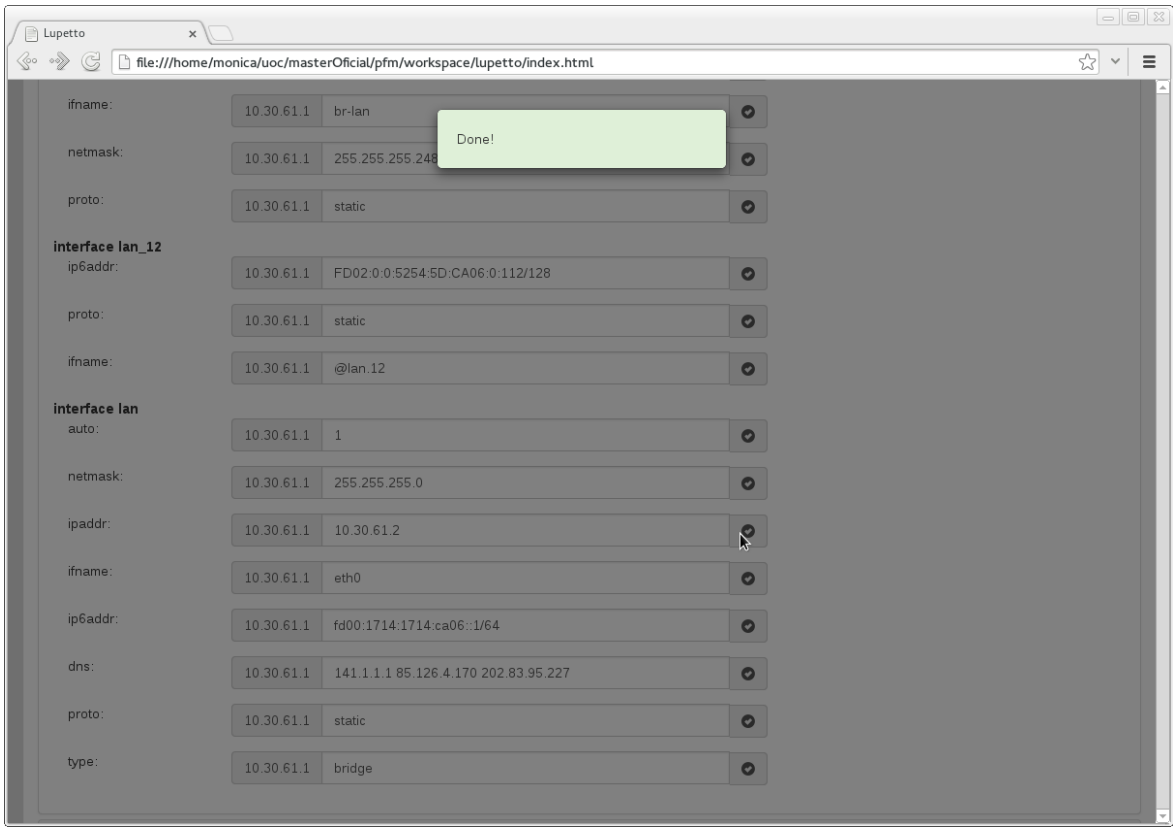


I es poden canviar el valor de qualsevol d'aquestes opcions:

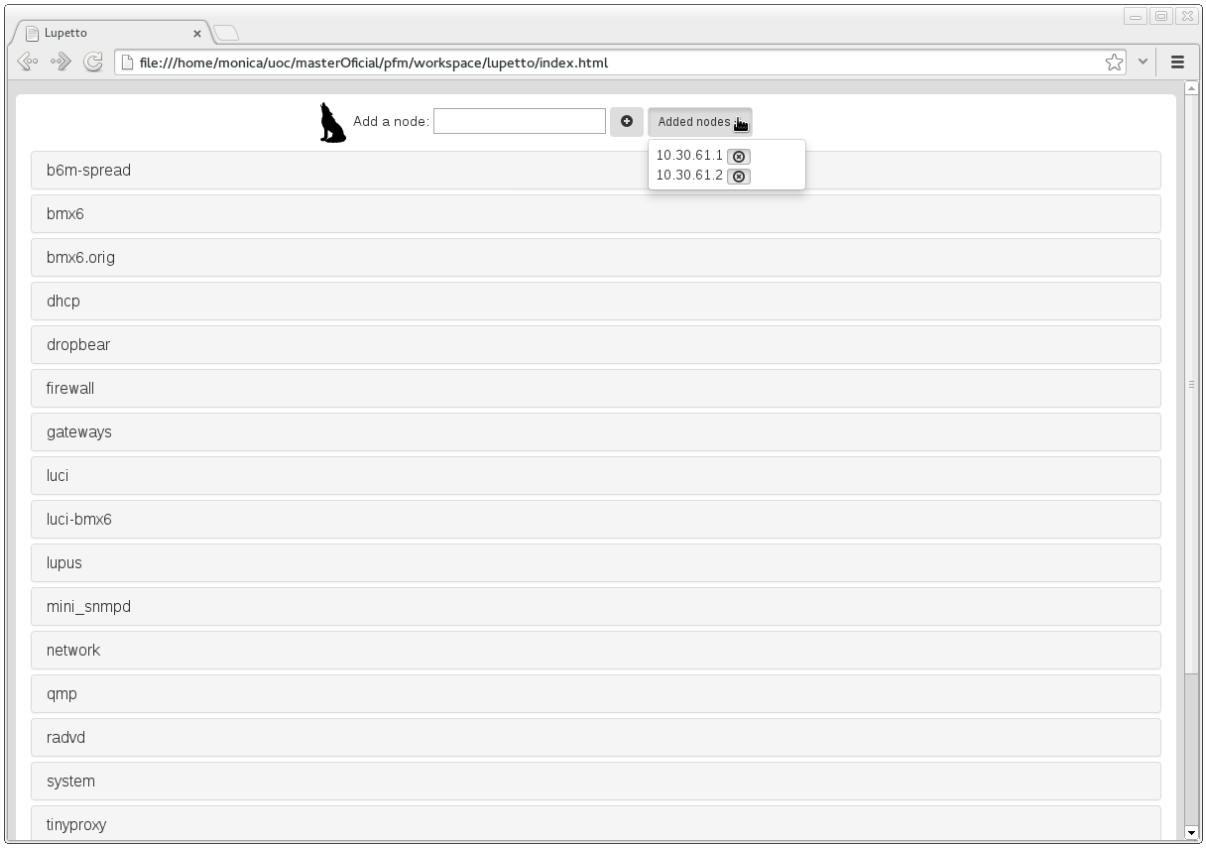
Història d'usuari: 3



Un cop s'ha canviat una opció, es mostrarà un missatge de confirmació, informant que el canvi s'ha realitzat al node:

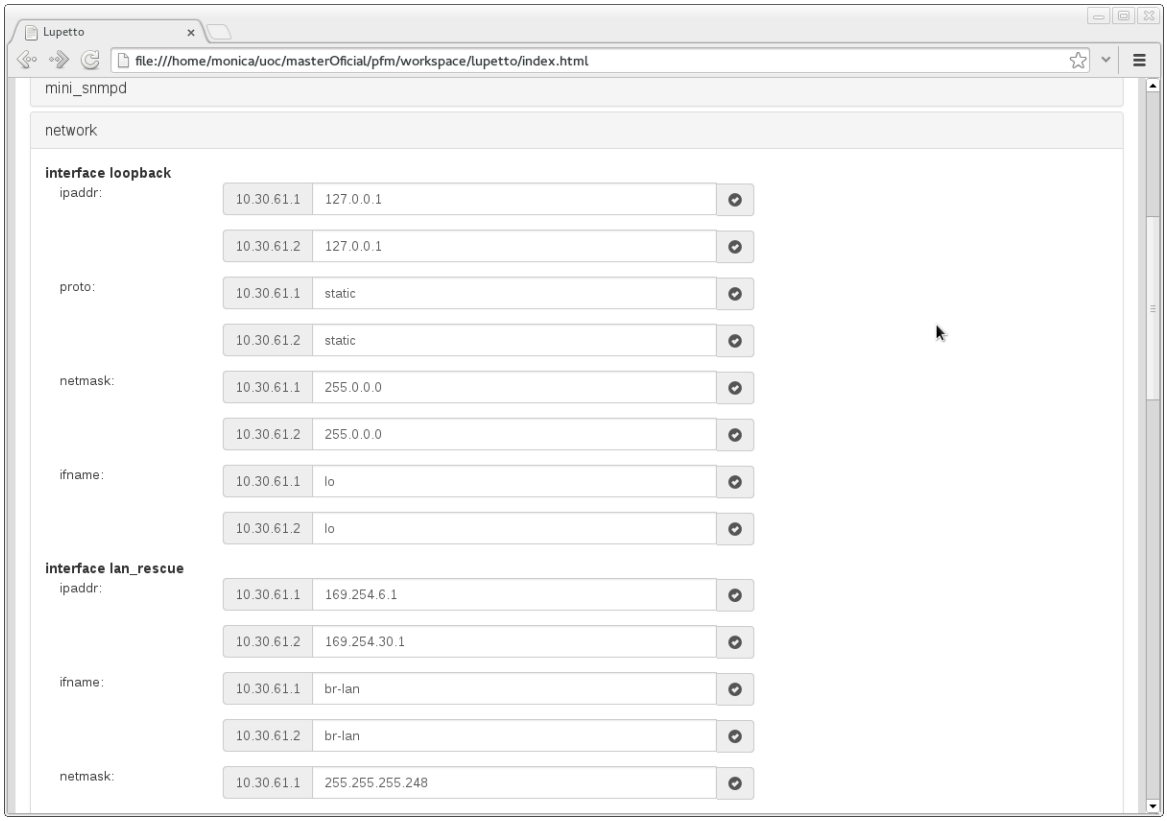


S'afegeix un segon node:



El node ha estat afegit i així es mostra a la llista desplegable. A més, si hi hagués algun fitxer de configuració que no estava en el primer node, apareixeria.

Es fa clic en un fitxer de configuració:



Ara, per a cada opció ens dóna el valor que té aquella opció per a cadascun dels nodes afegits.

Història d'usuari: 4

S'observa que a partir de la llista desplegable dels nodes afegits es pot treure un node prement el botó per tal d'eliminar-lo. *lupetto* actualitzaria tots els valors.

5.4.4 lupetto.js

El fitxer `lupetto.js` és on està codificat el codi que fa que *lupetto* funcioni. S'usa el llenguatge de programació Javascript amb un fort ús de *jQuery*. Es descriuen a continuació cadascuna de les funcions que s'han implementat:

5.4.4.1 addNodeToList

Paràmetres	*node*: la IP d'un node
Retorn	-
Descripció	Afegeix un node (la seva IP) a la llista desplegable on apareixen tots els nodes afegits

5.4.4.2 createButtonOk

Paràmetres	-
Retorn	Un botó
Descripció	Crea un botó amb la icona d'ok

5.4.4.3 createConfigElement

Paràmetres	headingText: nom del fitxer de configuració headingId: identificador de la capçalera del panell bodyId: identificador HTML del cos del panell
Retorn	Un element HTML
Descripció	Crea un panell amb el nom del fitxer de configuració desitjat

5.4.4.4 createOptionElement

Paràmetres	node: la IP d'un node optionKey: el nom d'una opció optionValue: el valor de lopció configKey: el nom del fitxer de configuració sectionKey: el nom de la secció firstNode: booleà que determina si és la primera opció que s'afegeix
Retorn	Un element HTML
Descripció	Crea els elements de formulari necessaris per tal de visualitzar i canviar el valor d'una opció

5.4.4.5 createSectionElement

Paràmetres	sectionKey: el nom de la secció sectionValue: l'objecte amb totes les dades de la secció
Retorn	Un element HTML
Descripció	Crea els elements HTML necessaris per tal de visualitzar el nom d'una secció i el seu tipus

5.4.4.6 findType

Paràmetres	section: un objecte amb tots els camps d'una secció
Retorn	El tipus de la secció
Descripció	Esbrina el tipus d'una secció

5.4.4.7 getConfigs

Paràmetres	-
Retorn	-
Descripció	Afegeix el node escrit per l'usuari i actualitza els noms dels fitxers de configuració, afegint aquells que calguin, si s'escau.

5.4.4.8 getSections

Paràmetres	configName: nom d'un fotxer de configuració
Retorn	-
Descripció	Obté les seccions d'un fitxer de configuració per a tots els nodes afegits. La informació s'obté fent una petició a l'API *lupus* del tipus `/uci/get_all/configName`

5.4.4.9 prepareConfigs

Paràmetres	data: objecte amb les dades d'un fitxer de configuració
Retorn	-
Descripció	Obté tots els noms dels fitxers de configuració i els mostra en un element de tipus acordió.

5.4.4.10 prepareSections

Paràmetres	node: la IP d'un node sectionData: les dades d'una secció configName: nom d'un fitxer de configuració
Retorn	-
Descripció	Obté les seccions del fitxer de configuració de tots els nodes afegits.

5.4.4.11 removeConfigs

Paràmetres	-
Retorn	-
Descripció	Esborra tot el contingut de l'acordió que conté els noms dels fitxers de configuració.

5.4.4.12 removeSections

Paràmetres	configName: el nom d'un fitxer de configuració
Retorn	-
Descripció	Esborra tot el contingut de l'element HTML que conté les seccions d'un fitxer de configuració.

5.4.4.13 runModal

Paràmetres	className: la classe CSS amb la qual volem que apareixi el missatge text: text del missatge
Retorn	-
Descripció	Mostra un missatge a l'usuari. El missatge pot ser informatiu, d'error, d'èxit...

5.4.4.14 setOption

Paràmetres	event: inclou diversos paràmetres: • param1: el nom de l'opció • param2: el valor de l'opció • param3: el node
Retorn	-
Descripció	Estableix un nou valor a l'opció en el node desitjat. Per tal d'establir aquest nou valor es fa una petició a l'API *lupus* del tipus `/uci/set`.

5.4.4.15 showConfigs

Paràmetres	-
Retorn	-
Descripció	Mostra tots els noms dels fitxers de configuració de tots els nodes afegits.

5.4.4.16 showSections

Paràmetres	configName: nom d'un fitxer de configuració
Retorn	-
Descripció	Mostra les seccions d'un fitxer de configuració de tots els nodes afegits. Si un node no té aquella secció, no mostrarà res.

5.4.4.17 updateConfigs

Paràmetres	-
Retorn	-
Descripció	Actualitza els fitxers de configuració existents segons els nodes afegits. Per fer-ho es fa una petició a l'API *lupus* del tipus `/uci/get_configs`.

6 Distribució

6.1 lupus

Per tal de distribuir *lupus*, s'ha confeccionat el paquet amb el mateix nom de manera que sigui molt fàcil instal·lar-lo.

Es pot trobar el codi del paquet a la branca `openwrt` del repositori `git`.

Per afegir-lo als feeds d'OpenWRT, s'ha d'afegir la següent línia al fitxer `feeds.conf`:

```
src-git lupus git://qmp.cat/luporum/lupus.git;openwrt
```

En un futur, el paquet *lupus* vindrà instal·lat a la distribució *qMp* per defecte.

6.2 lupetto

L'aplicació web *lupetto*, com que es tracta d'una aplicació Javascript que no necessita cap servidor web per tal d'executar-se es distribueix com a un fitxer .tar.gz. Per tal de generar el comprimit s'ha d'executar la següent comanda, on es genera el fitxer comprimit excloent els fitxers relacionats amb .git i amb el projecte Eclipse:

```
tar czvf lupetto.tar.gz --exclude .git* --exclude .settings --exclude .project lupetto
```

Per tal de fer-la funcionar només s'ha de descomprimir en el directori on es vulgui instal·lar i obrir el fitxer index.html des d'un navegador. Per descomprimir el fitxer s'ha d'executar la següent comanda:

```
tar xvf lupetto.tar.gz
```

7 Conclusions

En aquest apartat s'exposen diverses conclusions a les quals s'ha arribat després del desenvolupament del projecte *luporum*.

7.1 Valoració personal

En l'apartat 1.1 es descrivien els objectius del projecte luporum. A continuació es valora si s'han assolit aquests objectius:

- **Gestió de fitxers UCI:** Aquest objectiu s'ha assolit bàsicament amb el desenvolupament de *lupus*, que fa una emulació de la comanda `uci`. A través de l'API, s'ha aconseguit fer la gestió bàsica de tots els valors del sistema *UCI*. Amb el client web *lupetto*, tot i que no s'usa tota la funcionalitat de l'API *lupus*, s'aconsegueix mostrar i modificar els valors del sistema *UCI*.
- **Configuració multidispositiu:** Aquest objectiu s'assoleix a partir del client *lupetto*, ja que s'ha aconseguit poder afegir diversos dispositius i modificar els fitxers de configuració de cadascun d'ells des de la mateix aplicació.

En l'apartat 1.2 es detallaven algunes de les motivacions personals per fer el projecte. A continuació es valora si s'han aconseguit satisfer aquestes motivacions:

- **Treballar amb una empresa que desenvolupa programari lliure:** ha estat una satisfacció poder treballar en una empresa on no hi hagut cap problema, de fet s'ha motivat, en desenvolupar una aplicació amb llicència lliure. A més, les eines de desenvolupament usades també són programari lliure, en concret, *redmine* per a la gestió dels projectes i *git* com a repositori de versions.
- **Aprendre diverses tecnologies**: en aquest cas, he tingut l'oportunitat de treballar amb tecnologies fins ara desconegues per a mi fins llavors, com l'ús de dispositius embedded, conèixer la distribució OpenWRT i aprendre i treballar amb el ràpid i lleuger llenguatge de programació LUA.
- **Continuïtat del projecte:** tot i que en finalitzar el projecte, encara no sé si se seguirà mantenint, crec que hi ha força possibilitats, ja que és un objectiu de l'empresa Routek oferir una aplicació de configuració multi-dispositiu. Tot i que *lupus* i *lupetto* són aplicacions que poden ser millorades, crec que són una bon base per aconseguir un producte completament competitiu.

7.2 Què he après

Com ja he comentat en l'apartat anterior, una de les motivacions que tenia per desenvolupar aquest projecte era aprendre noves tecnologies i maneres de treballar. Exposo a continuació amb més detall tot allò que he après:

- **Ús de dispositius embedded:** tot que estava ja habituada a fer servir la línia de comandes i la gestió remota via ssh, la configuració de routers en distribucions mínimes ha estat un bon aprenentatge. T'adones que moltes de les eines que tenim en distribucions per a ordinadors (Debian, Ubuntu, Fedora...), desapareixen en aquest tipus de dispositius, ja que estan limitats en espai i potència. Com a exemple, dir que no tenim les típiques pàgines man, no es disposa d'entorn gràfic (com a molt un aplicatiu web), etc.
- **Distribució OpenWRT:** no havia usat mai abans aquesta distribució. Les dues característiques que més temps he dedicat ha estat en veure la gestió de paquets, amb la comanda `opkg` i el mateix sistema de configuració UCI.
- **Llenguatge de programació LUA:** tot i haver usat diversos llenguatges de programació durant la meva vida professional, mai havia programat amb LUA. Es tracta d'un llenguatge

interpretat, molt lleuger i ràpid. Cal destacar l'ús intensiu que fa de les taules, ja que les usa pràcticament per a tot, fins i tot per simular la programació orientada a objectes. Quant a la sintaxi és força senzilla, similar a la de Python.

7.3 Relació del màster amb el projecte *luporum*

Durant el transcurs del màster de programari lliure de la UOC he tingut la possibilitat d'aprendre diverses eines i tecnologies relacionades amb el món del programari lliure que he pogut aplicar directament en el desenvolupament del projecte *luporum*.

A continuació s'exposen quines han estat aquestes eines:

- **Programació:** en els crèdits de programació he refrescat totes les nocions que cal saber a l'hora e desenvolupar una aplicació i, tot i que he usat un llenguatge diferent als que he après al màster, les nocions bàsiques de programació són sempre les mateixes.
- **Gestió de projectes:** en el màster, se'ns va mostrar com es desenvolupa amb Scrum. Tot i que no s'ha usat Scrm pur, ja que la implementació del projecte l'he desenvolupat sola (amb l'ajuda del meu tutor i d'altres integrants de l'empresa Routek SL), sí que s'han usat algunes de les idees i eines que aporta Scrum (*product backlog*, *sprints*...).
- **Repositoris de versions:** per a la implementació, hem usat *git*, que també va ser part d'un crèdit del màster. Actualment, és impossible pensar en el desenvolupament d'un projecte sense usar un sistema de versions i git és una de les millors eines que hi ha de moment.
- **Ús d'entorns de desenvolupament:** l'ús d'Eclipse també ha estat recomanat en alguns crèdits del màster i m'ha estat de força utilitat durant el desenvolupament de luporum.

7.4 El futur de *luporum*

Quan es va plantejar el desenvolupament del projecte *luporum* dins l'empresa Routek, es va veure com a l'inici del desenvolupament d'un producte que es podria oferir als seus clients. És per això, que hi ha moltes possibilitats que el desenvolupament i manteniment del projecte continuï.

D'altra banda, si el temps m'ho permet, jo mateixa estic interessada en seguir col·laborant en el projecte, en aquest cas, intentant aconseguir que més desenvolupadors i usuaris s'hi sumin per tal d'ampliar i millorar les funcionalitats de les dues aplicacions desenvolupades.

També seria molt interessant que es vegi *lupus* com una font d'informació per al desenvolupament d'altres clients. Una idea seria fer una aplicació per a dispositius amb el sistema operatiu Android que permetés fer el mateix (o més) que l'aplicació web *lupetto*.

A continuació exposo algunes de les millores que es poden fer tant a *lupus* com a *lupetto* en un futur:

Millores de lupus

- Incorporar més peticions, com per exemple, demanar al dispositiu, quins dispositius estan connectats amb ell.
- Millorar la gestió dels errors, oferint missatges més aclaridors.
- Crear el paquet *OpenWRT* per a la llibreria *dkjson*.

Millores de lupetto

- Afegir i esborrar fitxers de configuració
- Afegir i esborrar seccions
- Afegir i esborrar opcions