AMPLIACIÓ DE ITSAT PER A LA GESTIÓ DE SISTEMES WINDOWS

MARCEL HERNANDEZ BERTRAN

Director: Antonio MORENO

Ponent: Xavier MARTORELL

Departament AC

- 23 de Gener de 2015 -

ÍNDEX DEL DOCUMENT

Ampliació de ITSAT per a la gestió de sistemes Windows 1
1. Introducció 4
1.1 Descripció a vista d'ocell del projecte 4
1.2 Objectius del projecte 5
1.3 Planificació inicial 6
1.4 Anàlisi econòmica preliminar 8
2. Especificació 9
2.1 Requeriments funcionals 9
2.1.1 Requeriments relacionats amb l'alta de màquines Windows 9
2.1.2 Requeriments relacionats amb la integració no interactiva de Windows 10
2.1.3 Requeriments relacionats amb la integració interactiva de Windows 12
2.2 Requeriments no funcionals 13
3. Anàlisi d'antecedents i factibilitat 14
3.1 Solucions considerades per a la integració no interactiva de Windows 14
3.2 Solucions considerades per a la integració interactiva de Windows 15
3.2.1 Alternatives 15
3.2.2 Criteris de decisió 16
3.2.3 Taula Comparativa 17
3.2.4 Elecció: Guacamole 17
3.2.5 Què és Guacamole? 17
4. Desenvolupament tècnic 20
4.1 Disseny 20
4.1.1 Alta de màquines Windows 20
4.1.2 Integració de sistemes no interactius amb Windows 20
4.1.3 Integració de sistemes interactius amb Windows 21
4.2 Implementació 24
4.2.1 Alta de màquines Windows 24
4.2.2 Integració dels sistemes no interactius amb Windows 25
4.2.3 Integració dels sistemes interactius amb Windows 31
5. Proves i validació 38
5.1 Donar d'alta les dues màquines al sistema ITSAT 38
5.2 Comprovar funcionament de les comandes d'administració 38

5.3 Comprovar funcionament de l'escriptori remot 38
6. Planificació final 41
7. Anàlisi Econòmica 42
8. Conclusions 43
8.1 Assoliment dels objectius proposats 43
8.2 Ampliacions i millores 43
8.2.1 Suportar Network Level Authentication 43
8.2.2 Fer funcionar Guacamole sobre WebSocket en comptes de HTTP 43
8.2.3 Extensió PHP de SAMBA i WINEXE 44
8.2.4 Logging de sessions interactives 44
Annex A: Tecnologies destacades 45
A.1 PHP 45
A.2 Smarty 45
A.3 Javascript 45
A.4 X.509 45
Annex B: Mostres de codi 46
B.1 Telnet.php 46
B.2 SMBCIFS.php 56
B.3 guacamole-service.php 60
B.4 ITSATDesktopHTTPTunnelServlet.java 62

1. INTRODUCCIÓ

ITSAT és l'acrònim de IT-System Administration Tools i com el nom indica, és un conjunt d'eines d'administració de sistemes en entorns IT dissenyades i empaquetades per a l'ús d'administradors de sistemes amb l'objectiu d'incrementar directament l'eficiència del seu treball i obtenir un entorn de treball amb l'objectiu final de millora del servei. A continuació s'enumeren els objectius principals d'ITSAT:

- Incrementar disponibilitat de serveis.
- Millorar dels ratis de resposta i resolució d'incidències i problemes.
- Increment en la qualitat del servei.
- Reduïr el temps d'adaptació dels administradors de sistemes recen incorporats.

En definitiva, disminuïr el cost d'administració de serveis. Aquest projecte de final carrera treballarà sobre aquesta eina, per tal de dotar-la de noves funcionalitats.

Abans d'abordar aquest PFC, ITSAT només contemplava la gestió de màquines Linux. La seva finalitat és ampliar el seu abast per poder gestionar també màquines basades en sistemes Windows. Per aconseguir això s'ha optat per integrar Guacamole a ITSAT. Guacamole és un projecte de codi obert que té per objectiu proveïr connexions interactives mitjançant els protocols RDP i VNC. Amb això s'intentarà donar el mateix grau de suport i funcionalitat que actualment ITSAT proporciona per màquines basades en Linux.

1.1 DESCRIPCIÓ A VISTA D'OCELL DEL PROJECTE

ITSAT és un sistema web basat principalment en l'stack LAMP[1], és dir es tracta d'una solució muntada sobre les tecnologies Linux[2], Apache[3], MySQL [4]i PHP[5] (Annex A.1).

Aquesta part del sistema és l'encarregada de tasques fonamentals com ara l'autenticació i autorització d'usuaris, la interacció amb bases de dades i el processament i servei dels assets de la pàgina web (HTML/CSS/JS) (Annex A.3).

La part del sistema que s'encarrega d'establir conexions remotes amb màquines Unix a través d'una consola virtual està basada en el projecte GateOne[6], implementat en Python[7].

El sistema també disposa d'una sèrie d'utilitats al backend per dur a terme les mateixes tasques que permet fer l'aplicació web per mitjà de comandes de consola. Aquestes comandes estàn implementades mitjançant scripts de bash, i algunes parts del propi codi PHP també s'interactua amb scripts d'aquesta mena.

[1] http://en.wikipedia.org/wiki/LAMP_%28software_bundle%29
[2] http://en.wikipedia.org/wiki/Linux
[3] http://httpd.apache.org/
[4] http://www.mysql.com/
[5] http://php.net/
[6] http://liftoffsoftware.com/Products/GateOne
[7] https://www.python.org/

1.2 OBJECTIUS DEL PROJECTE

A grans trets l'objectiu del projecte és ampliar ITSAT per tal de suportar la gestió de màquines Windows al mateix nivell que Unix. Aquest objectiu es pot dividir en 3 grans subobjectius:

- Permetre l'alta de màquines gestionades Windows a ITSAT
- Integració de Windows en sistema no interactiu
- Integració de Windows en sistema interactiu

Permetre l'alta de màquines no requereix cap més explicació específica, per tant passo directament a explicar en què consisteix la integració als sistemes "interactiu" i "no interactiu".

La integració no interactiva fa referència a la integració de les comandes disponibles per a l'administrador d'ITSAT i la interfície web. Tant les comandes d'administració com la interfície web permeten realitzar certes operacions sobre les màquines gestionades. Al seu torn els dos sistemes deleguen la feines a les mateixes classes PHP de backend que són les que realment s'encarreguen de gestionar les conexions i executar aquestes tasques. Integrar Windows en el sistema no interactiu implica adaptar el backend per suportar com a mínim:

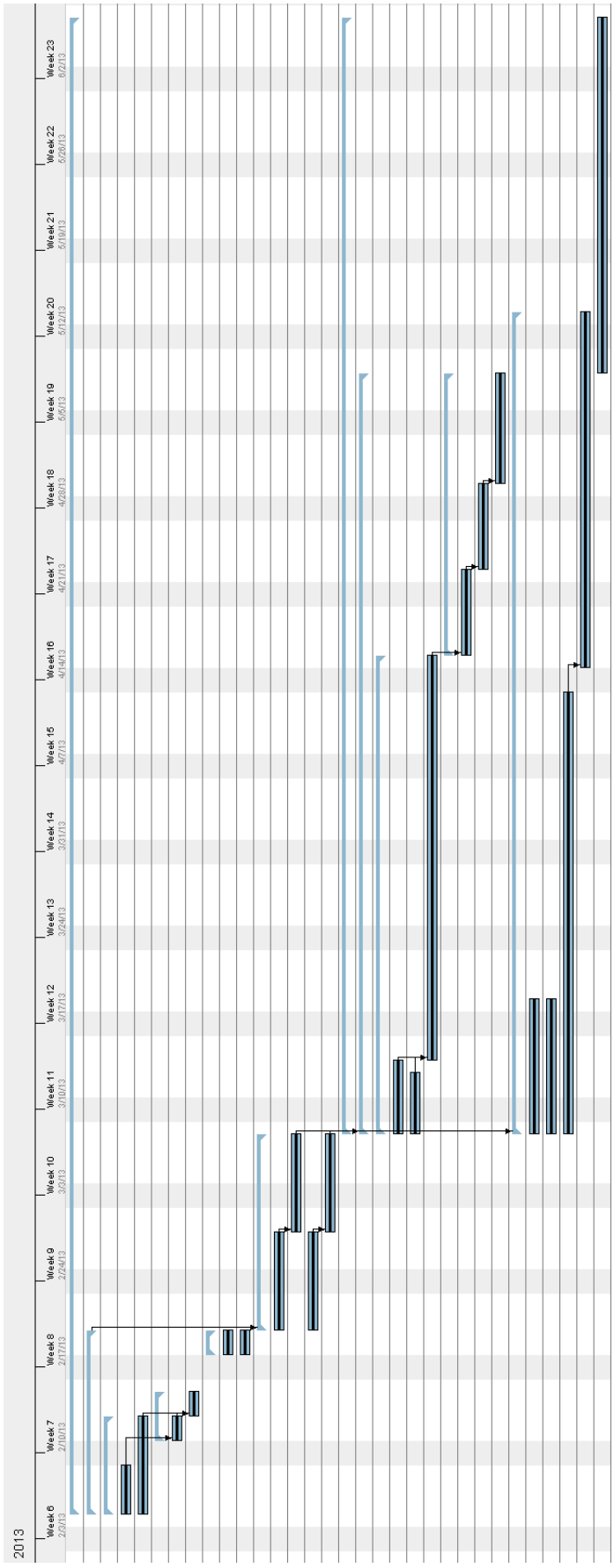
- Conexions remotes a sistemes Windows
- Transmissió de fitxers entre el servidor i la màquina remota
- Execució de comandes Windows (MS-DOS)

Per a les màquines Linux ITSAT disposa també a la seva interfície web d'un terminal interactiu basat en el projecte GateOne que permet treballar amb aquestes màquines des del navegador web talment com si s'hagués obert una conexió ssh amb una consola convencional. Aquesta seria la integració interactiva d'ITSAT amb màquines *Linux*. Per a les màquines Windows ens proposem anar encara més enllà i integrar un escriptori remot complet a la interfície web que permetrà veure el propi escriptori de Windows de les màquines gestionades al navegador. En això consisteix la integració interactiva de Windows.

1.3 PLANIFICACIÓ INICIAL

A continuació s'exposa la planificació del projecte, desglossada per tasques mitjançant la tècnica dels diagrames de Gantt.

Name	Dura...	Begi...	End date
Part A: Afegir suport màquines Windows a ITSAT	88	2/5/13	6/6/13
Estudi Previ (Introducció)	11	2/5/13	2/19/13
Contacte amb windows, etapa de Documentació	6	2/5/13	2/12/13
Familiarització amb SAMBA	4	2/5/13	2/8/13
Familiarització amb Guacamole	6	2/5/13	2/12/13
Proves de concepte	4	2/11/...	2/14/13
Configuració d'un SAMBA per conectar a una màquina Windows des de Li...	2	2/11/...	2/12/13
Muntar màquina aïllada i fer proves d'execució i transferències amb SAMBA	2	2/13/...	2/14/13
Anàlisi de sistemes windows en un entorn d'Administració centralitzat	2	2/18/...	2/19/13
Identificar diferents situacions en les que pot estar una màquina Windows	2	2/18/...	2/19/13
Quines credencials caldràn?	2	2/18/...	2/19/13
Disseny Funcional de ITSAT amb Guacamole i SMB-cif	12	2/20/...	3/7/13
Diagrama funcional de sistema NO interactiu (identificació i descripcio de fun...	6	2/20/...	2/27/13
Disseny dels test de validació de sistema NO Interactiu	6	2/28/...	3/7/13
Diagrama funcional de sistema Interactiu (identificació i descripcio de funcion...	6	2/20/...	2/27/13
Disseny de test de validacio de sistema interactiu	6	2/28/...	3/7/13
Implantacio de canvis per integrar SAMBA	65	3/8/13	6/6/13
Integració (base) de windows en sistema NO interactiu	44	3/8/13	5/8/13
Afegir protocol CIFN a la execució remota	27	3/8/13	4/15/13
Revisar interfície iremote.inc i crear classe smbcifn.inc amb la implem...	4	3/8/13	3/13/13
Revisar com identificar passwords de workgroup/domini AD en la estr...	3	3/8/13	3/12/13
Integrar smbcifn.inc amb authcache.inc i gestió de passwords	23	3/14/...	4/15/13
Etapa de Testeig No interactiu	17	4/16/...	5/8/13
Proves Inicials	5	4/16/...	4/22/13
Proves extenses	5	4/23/...	4/29/13
Proves completes (insertar noves màquines i poder executar-hi quals...	7	4/30/...	5/8/13
Integració de Windows en sistema INTERACTIU	47	3/8/13	5/13/13
diseny de equivalent a rcpsh de gateone (integració amb CID)	7	3/8/13	3/18/13
Scripts de Intal.lació i Manual de compilacio	7	3/8/13	3/18/13
Estudi i implementacio del client JavaScript	26	3/8/13	4/12/13
Testeig de sistema Interactiu	21	4/15/...	5/13/13
Integració d'infraestructura de serveis de ITSAT amb serveis Windows	21	5/9/13	6/6/13


2013
Week 6
2/3/13
Week 7
2/10/13
Week 8
2/17/13
Week 9
2/24/13
Week 10
3/3/13
Week 11
3/10/13
Week 12
3/17/13
Week 13
3/24/13
Week 14
3/31/13
Week 15
4/7/13
Week 16
4/14/13
Week 17
4/21/13
Week 18
4/28/13
Week 19
5/5/13
Week 20
5/12/13
Week 21
5/19/13
Week 22
5/26/13
Week 23
6/2/13

1.4 ANÀLISI ECONÒMICA PRELIMINAR

Aquestes són els diferents ítems de despesa que s'han pogut identificar abans de començar el projecte i que són estrictament necessaris per dur-lo a terme:

Ítem	**Tipus despesa**	**Justificació**	**Cost Estimat**
Lenovo ThinkPad SL510	Hardware	Màquina de desenvolupament proporcionada per l'empresa.	~200 € (prorratejat)
VMWare Workstation 9	Software	Software de virtualització per aixecar màquines gestionades. Permetrà estalviar un gran cost en hardware.	225 €
Llicència WinXP multiequip	Software	Llicències per les màquines virtuals.	750 €
Llicència Win7 multiequip	Software	Llicències per les màquines virtuals.	1300 €
Suite Office	Software	Per redactar la memòria.	540 €
Project Manager	Personal	Direcció del projecte durant 3 mesos.	~ 30.000 €
Desenvolupador	Personal	Desenvolupament durant 3 mesos.	~ 20.000 €

El cost estimat del projecte és per tant d'uns 53.815 € i la seva durada no hauria d'excedir els 4 mesos.

2. ESPECIFICACIÓ

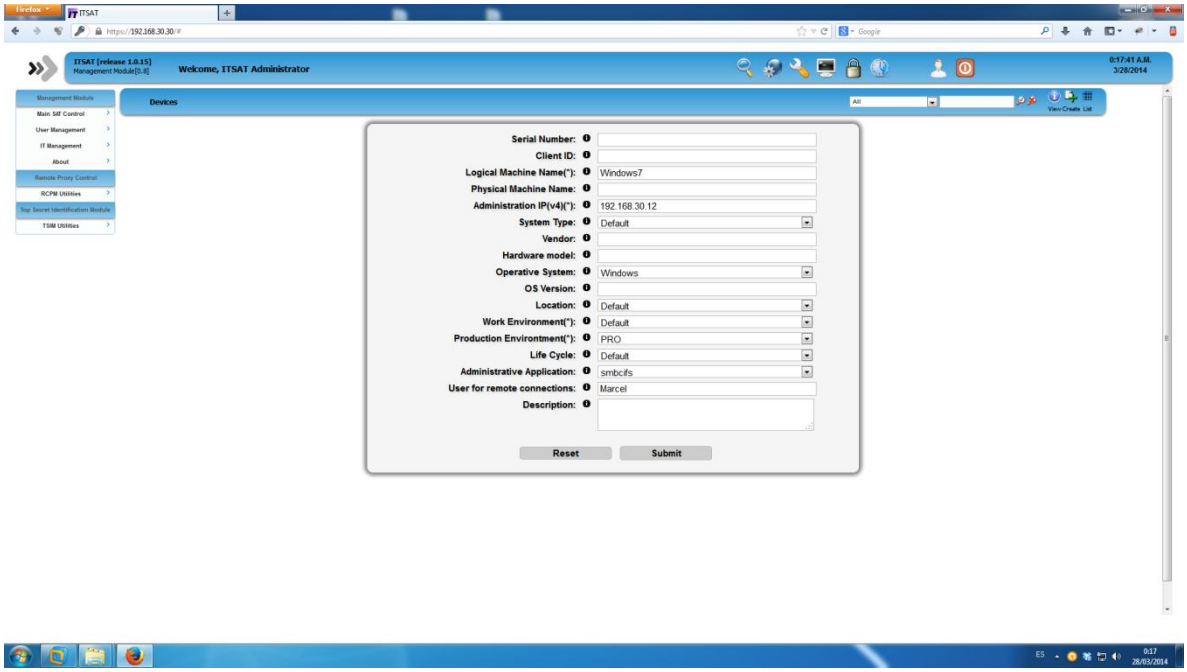
En aquest apartat explicarem tota la informació rellevant per entendre els components d'ITSAT que caldrà adaptar, i definiriem els criteris d'acceptació per poder considerar els objectius descrits anteriorment com a assolits. Al seu torn aquests criteris de decisió (o requeriments) es poden dividir en dos subtipus: requeriments funcionals (referents al projecte) i no funcionals (propietats o característiques que ha de tenir la solució).

2.1 REQUERIMENTS FUNCIONALS

2.1.1 REQUERIMENTS RELACIONATS AMB L'ALTA DE MÀQUINES WINDOWS

El sistema ITSAT està organitzat en diferents mòduls lògics de tal manera que cadascún d'ells s'encarrega d'una tasca específica. Aquesta divisió s'aprecia clarament a la interfície web, on les vistes associades a cada mòdul estan separades per icones a la barra superior.

El mòdul MNGM (Management, icona clau anglesa) és l'encarregat de donar d'alta màquines al sistema, esborrar-les i mostrar els seus detalls. El primer requeriment funcional del projecte serà permetre l'alta de màquines Windows a través de la interfície de registre de noves màquines.



Com s'aprecia a la imatge el camp Operative System és un desplegable que ha de permetre la selecció d'un element de la llista que digui "Windows".

Un altre camp rellevant és Administrative Application. Aquest camp determina quin tipus de conexió o protocol de comunicació ha de utilitzar ITSAT per gestionar la màquina. Abans d'iniciar el projecte la única opció era ssh, que les màquines Windows no suporten per defecte. Per tant un altre requeriment és implementar una nova "aplicació administrativa" que permeti comunicar-se amb Windows, i poder-la seleccionar en aquesta vista (en la imatge es llegeix smbcifs, que és el nom d'una de les implementacions que s'han fet).

Per acabar, és important que aquesta informació (el tipus de sistema operatiu) es guardi al lloc adient de la base de dades. La informació pertinent a les màquines gestionades es guarda a la taula SATmachines de la base de dades itsat. El seu schema és el següent:

```
CREATE TABLE `SATmachines` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `serialnumber` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,
  `clientid` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,
  `name` varchar(100) COLLATE utf8_unicode_ci NOT NULL DEFAULT '',
  `fname` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,
  `vendor` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,
  `ipadm` varchar(100) COLLATE utf8_unicode_ci NOT NULL DEFAULT '0.0.0.0',
  `os` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,
  `osver` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,
  ...,

  `hasagent` tinyint(1) DEFAULT '1',
  PRIMARY KEY (`id`),
  ...,

) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

Doncs bé, és imprescindible que a partir d'ara es guardi un nou identificador de sistema operatiu a la columna 'os' de cada màquina, ja que aquesta informació és en la que es basaràn la resta de sistemes que interaccionen amb màquines gestionades per saber com han de tractar amb elles.

2.1.2 REQUERIMENTS RELACIONATS AMB LA INTEGRACIÓ NO INTERACTIVA DE WINDOWS

Per tal d'entendre aquests requeriments s'explicarà el funcionament de les principals comandes d'administració d'ITSAT.

- sat-sexec: Serveix per executar scripts en alguna de les màquines gestionades. La màquina es referencia pel nom amb que està registrada al sistema ITSAT. No cal proporcionar la IP ni cap tipus de credencial, tota aquesta informació necessària per establir la comunicació ja està guardada a la base de dades d'ITSAT i s'agafa d'allà.

```
# sat-sexec –m oldtimes /home/everis/scripts/win/do_backup.bat
[oldtimes] Backing up home folder.....
[oldtimes] Done.
#
```

- sat-rexec: Permet executar comandes arbitràries en alguna de les màquines gestionades.

```
# sat-rexec –m oldtimes "dir C:\Users"
[oldtimes] Volume in drive C has no label.
```

```
[oldtimes] Volume Serial Number is 5ADC-D664
[oldtimes]
[oldtimes] Directory of C:\Users
[oldtimes]
[oldtimes] 22-Mar-13  21:55    <DIR>          .
[oldtimes] 22-Mar-13  21:55    <DIR>          ..
[oldtimes] 10-Jan-12  13:12    <DIR>          Administrator
[oldtimes] 12-Jul-13  00:49    <DIR>          everis
[oldtimes] 19-Jul-14  13:20    <DIR>          Public
[oldtimes] 08-Jan-12  20:10    <DIR>          UpdatusUser
[oldtimes]               0 File(s)              0 bytes
[oldtimes]               6 Dir(s)  154,449,207,296 bytes free
#
```

- sat-rget: Permet recuperar fitxers d'alguna de les màquines gestionades i portar-los al servidor ITSAT.

```
# sat-rget –m oldtimes –r "C:\Users\everis\document.docx" –l "/home/everis/Documents/"
[oldtimes] OK
#
```

- sat-rput: Permet enviar fitxers del servidor ITSAT a alguna de les màquines gestionades.

```
# sat-rput –m oldtimes –l "/home/everis/Documents/document.docx" –r "C:\Users\everis"
[oldtimes] OK
#
```

- sat-remotedo: Aquesta és la comanda d'administració "reina". No només permet fer qualsevol operació de les que s'han descrit fins ara sinó que la pot fer sobre més d'una màquina a la vegada, i es poden utilitzar diferents filtres per a tal efecte (per exemple: llistes de noms, filtrar per grups de màquines, per sistema operatiu...).

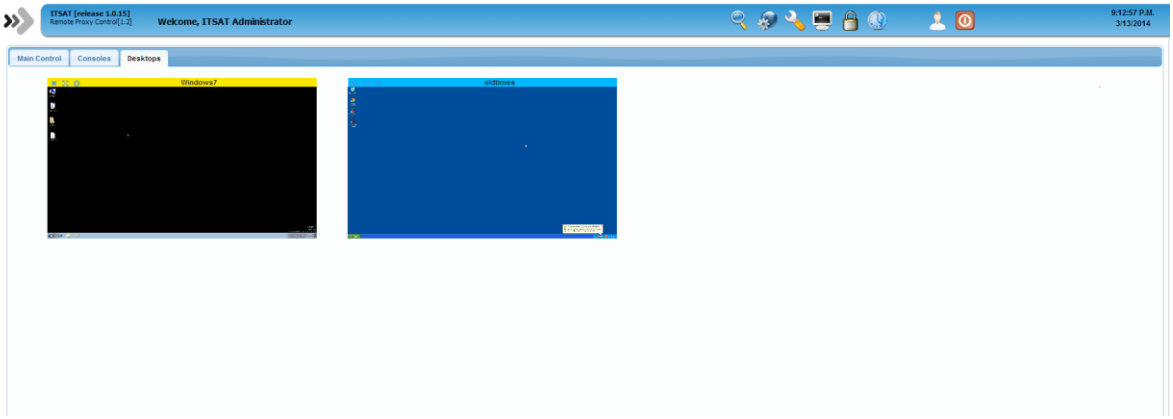
En aquesta comanda el flag –m retorna un llistat de totes les màquines gestionades al sistema, per ajudar a situar-se. També he inclòs un parell d'exemples d'ús:

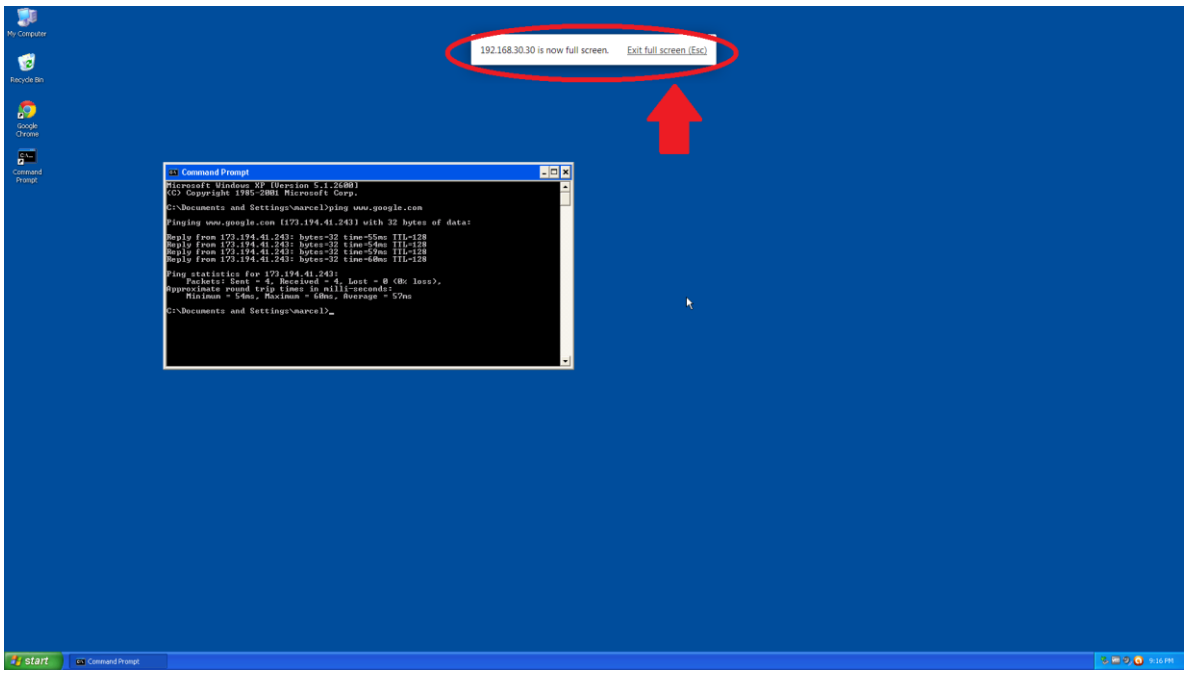
```
# sat-remotedo -m
#ID          WorkEnv(#id)     Op.System      Prod.Env            IP   Machine Name
=============================================================
811            ALL(#0)        Linux          PRO          1.1.1.1          test
806          Default(#1)       Linux          PRO       172.28.57.162     itsatdev
816          Default(#1)       Linux          PRE        10.10.10.10       sg-01
818          Default(#1)       Linux          PRE        10.10.10.20       sg-02
820            ALL(#0)        Windows        PRO        10.10.10.30      oldtimes
824            ALL(#0)        Windows        PRO        10.10.10.40      windows7


Executa la comanda "ipconfig /all" sobre totes les màquines Windows d'ITSAT
# sat-remotedo –S Windows –c "ipconfig /all"

Copia el fitxer ./certificate.pem al directori /etc/apache/certs de les màquines sg-01 i sg-02
# sat-remotedo –M sg-* put certificate.pem /etc/apache/certs
```

Els requeriments de la integració de Windows amb els sistemes no interactius consisteixen doncs en poder fer servir aquestes comandes executant ordres com les exposades en els exemples anteriors, que ja acceptarien connexions a les màquines amb aquest sistema operatiu.

2.1.3 REQUERIMENTS RELACIONATS AMB LA INTEGRACIÓ INTERACTIVA DE WINDOWS

El requeriment principal per integrar Windows amb el sistema interactiu d'ITSAT passarà per crear una nova pestanya en el mòdul de gestió de màquines on es veuran miniatures dels escriptoris de les màquines Windows gestionades per ITSAT. Clickant un d'aquests escriptoris el navegador ensenyarà l'escriptori a pantalla completa i els inputs del ratolí i el teclat de l'usuari es redirigiran a la màquina per tal d'utilitzar-la en temps real.





2.2 REQUERIMENTS NO FUNCIONALS

Les actuacions dutes a terme per tal d'aconseguir els objectius també han de complir alguns requeriments no funcionals que no estan relacionats directament amb cap funcionalitat d'ITSAT.

- **Seguretat**. ITSAT és una aplicació molt crítica ja que gestiona un gran volum d'informació confidencial (IPs, usuaris i credencials de moltes màquines, entre d'altres). La solució implementada interactua amb algunes d'aquestes dades confidencials, per tant ha de mantenir els estàndards de seguretat més rigorosos.

- **Footprint mínim**. A les màquines gestionades basades en Linux no cal instalar-hi cap "client" ni software addicional per poder ser gestionades. La gestió de màquines Windows també han de tenir aquest footprint mínim. Això s'assolirà comunicant-se amb Windows a través del protocol RDP, que totes les distribucions modernes (XP en amunt) suporten nativament (però s'ha d'activar primer).

- **Arquitectura clara i simple**. ITSAT és un projecte molt madur i complex, on hi han treballat un bon nombre de programadors al llarg dels anys. La solució implementada hauria de millorar la organització del codi dels sistemes adaptats, no empitjorar-lo i fer-lo molt més complex a base de "pegats ad-hoc". Durant el desenvolupament s'han aplicat algunes tècniques de refactorització de codi legacy per assolir aquest objectiu.

3. ANÀLISI D'ANTECEDENTS I FACTIBILITAT

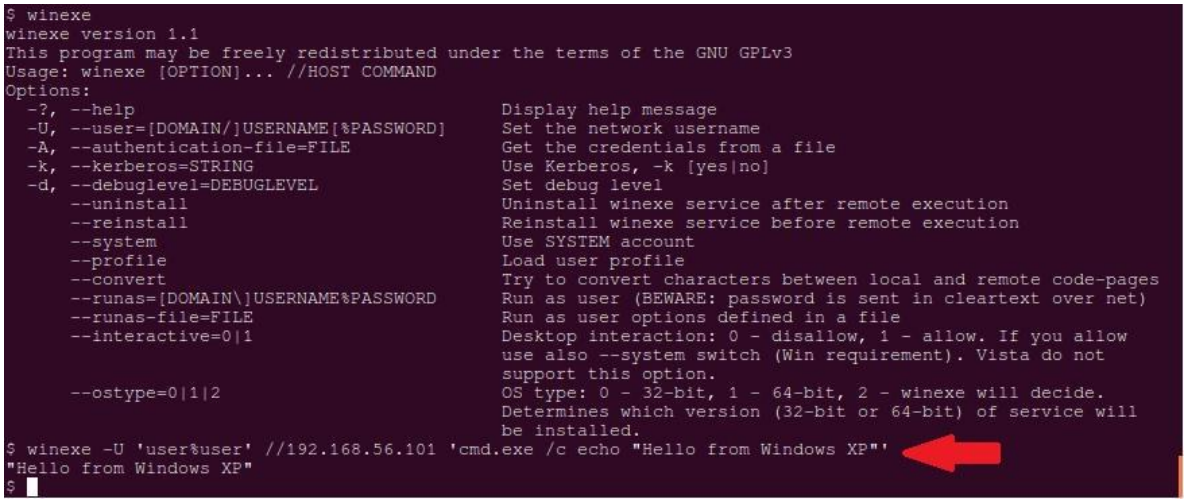
En aquest apartat es farà un breu anàlisi del les diferents tecnologies considerades per dur a terme les integracions interactives i no interactives de màquines Windows.

3.1 SOLUCIONS CONSIDERADES PER A LA INTEGRACIÓ NO INTERACTIVA DE WINDOWS

Pel que fa a l'enviament i recepció de fitxers entre màquines Windows i Unix la opció més clara és samba[8].

Samba és una implementació de codi lliure dels protocols de compartició de fitxers SMB i CIFS de Microsoft. Aquesta aplicació per a sistemes Unix els permet actuar com a clients i/o servidors en xarxes de màquines Windows, així com enviar-se i rebre fitxers.

Per tal d'executar comandes remotament farem servir el programa WinExe[9]. Segons la propia web del projecte: "Winexe remotely executes commands on Windows NT/2000/XP/2003 systems from GNU/Linux". Heus aquí un exemple:



Encara que ens hem limitat a integrar aquesta comanda sense estudiar-ne el codi, una de les dependències de WinExe és precisament la llibreria C de Samba.

[8] https://www.samba.org/
[9] http://sourceforge.net/projects/winexe/

3.2 SOLUCIONS CONSIDERADES PER A LA INTEGRACIÓ INTERACTIVA DE WINDOWS

En aquesta secció es fa una tria de les tecnologies que hem considerat per integrar l'escriptori remot de Windows a l'aplicació d'ITSAT.

3.2.1 ALTERNATIVES

- Remmina (C)[10]

Remmina és un projecte de codi obert basat en la llibreria GTK+. Aquest projecte també es basa en FreeRDP, una implementació lliure del protocol RDP dels mateixos creadors. Es tracta d'una aplicació nativa que permet mostrar l'escriptori de màquines remotes mitjançant el protocol RDP.

- Rdesktop (C)[11]

rdesktop és una altra alternativa força equivalent a Remmina. També és tracta d'una aplicació nativa, i fa més anys que existeix. De fet, FreeRDP va començar com un fork d'aquest projecte per intentar millorar els estàndards del codi.

- Vinagre (C)[12]

Un altre client natiu. A més a més del protocol RDP també suporta els protocols VNC, SSH i SPICE. També té l'avantatge de que vé integrat per defecte al gestor de finestres GNOME des de la versió 2.22, per tant s'adapta perfectament a l'entorn d'execució d'ITSAT, que està basat en Debian.

- Guacamole (C/Java)[13]

Guacamole és un projecte una mica diferent de les alternatives que hem vist fins ara. A grans trets els projecte està basat en dues parts lògiques perfectament separades: el client i el servidor. El servidor és una aplicació nativa escrita en C que s'executa com a dimoni i s'encarrega de gestionar les conexions a les diferents màquines remotes. El client en canvi es tracta d'una aplicació web que es comunica amb aquest dimoni i fa de frontend. Està basat en Java i tecnologies web (HTML, JS).

[10] http://remmina.sourceforge.net/
[11] http://www.rdesktop.org/
[12] https://wiki.gnome.org/Apps/Vinagre
[13] http://guac-dev.org/

3.2.2 CRITERIS DE DECISIÓ

Un cop feta aquesta primera cerca preliminar es va procedir a definir els criteris per decidir quina solució era més adient per integrar-la a ITSAT. Aquests són els punts que van tenir més pes en aquesta decisió:

- Suport pel protocol RDP
  Tot i que no és l'únic protocol disponible per comunicar-se remotament amb màquines basades en sistemes Windows a la pràctica és el més comú, ja que es pot fer servir sense haver de instalar cap software addicional a les màquines remotes. Per tant de cara a ITSAT es va decidir que el requeriment s'assoliria amb només suportar aquest protocol.

- Projecte FOSS (Free and Open Source Software)
  Degut per una banda a la limitació del pressupost, i per l'altra al fet de que ITSAT en sí també és un projecte lliure i de codi obert, es va decidir que la solució emprada també havia de ser FOSS. De fet, la tria preliminar ja comtemplava només projectes d'aquest tipus.

- Amb activitat
  Preferiblement el projecte havia d'estar "actiu", és a dir havia de mostrar una certa continuïtat en l'alliberament de noves versions i de modificacions al codi.

- Bona documentacio
  No cal parlar de la necessitat bàsica de disposar d'una bona documentació completa i actualitzada de tot el que pot oferir el software.

- Suport
  També interessa veure si és factible aconseguir alguna mena de suport directe si sorgeix algun problema important durant la implementació (contacte directe amb un desenvolupador, fòrum de suport, StackOverflow...). Tenint en compte que es tractarà d'una solució de software lliure aquest criteri està bastant influenciat pel grau d'adopció d'aquest software.

- Facilitat d'integració amb ITSAT
  Els requeriments del projecte exigeixen integrar aquest nou component amb el sistema ITSAT ja existent. Això implica bàsicament una integració amb la interfície web i amb el sistema d'autenticació.

3.2.3 TAULA COMPARATIVA

	RDP	FOSS	Actiu	Documentacio	Suport	Integrable
Remmina	✔	✔		✔	✔	
Rdesktop	✔	✔	✔			
Vinagre	✔	✔				
Guacamole	✔	✔	✔	✔	✔	✔

3.2.4 ELECCIÓ: GUACAMOLE

Com es pot veure de manera bastant evident Guacamole és la solució que compleix tots els requeriments de manera més sòlida.

Com totes les altres solucions suporta conexions remotes mitjançant el protocol RDP i és un projecte FOSS. A més a més està en constant desenvolupament, té una documentació molt bona, a més a més de disposar d'un fòrum de suport administrat pels propis desenvolupadors dels projecte. El criteri de més pes però és que serà molt més fàcil d'integrar a ITSAT, especialment pel fet que de la seva interfície està basada en tecnologies web en comptes de ser un executable natiu.

3.2.5 QUÈ ÉS GUACAMOLE?

Guacamole és una aplicació HTML5 que proporciona accés a remot a entorns que suporten els protocols d'escriptori remot VNC i RDP. La seva API es pot utilitzar per implementar altres aplicacions o serveis que vulguin disposar d'aquest cas cas d'ús.

El terme "Guacamole" s'utilitza normalment per referir-se a la l'aplicació web desenvolupada sobre la seva pròpia API. En realitat per això el projecte consta de dues parts ben diferenciades entre sí. Per una banda consta del component Guacamole Server (https://github.com/glyptodon/guacamole-server) que implementa un servidor (no web) que fa d'interfície entre l'aplicació de Guacamole i els protocols d'escriptori remots. Si volguéssim implementar el suport de nous protocols de comunicació els canvis s'haurien de fer en aquest projecte, i s'haurien de poder fer servir utilitzant la API comuna que ja proporciona per a VNC i RDP.

Per altra banda hi ha el projecte Guacamole Client (https://github.com/glyptodon/guacamole-client). Aquest projecte és una aplicació web implementada en Java que en el backend es comunica amb el component del servidor alhora que actua de servidor web, fent de mitjancer entre l'usuari final i el component guacamole-server.

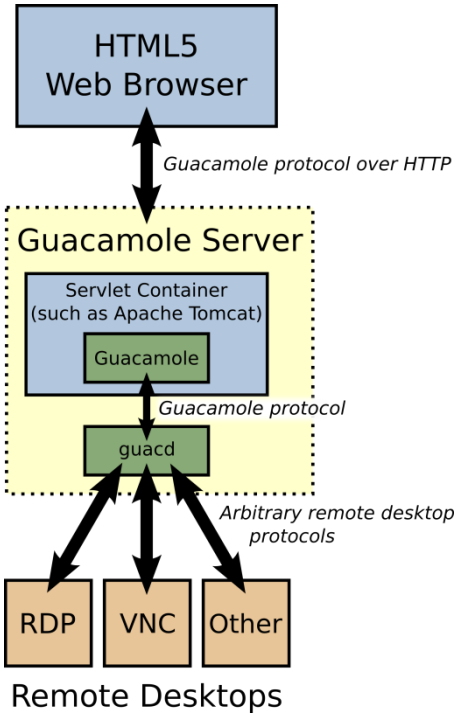
ITSAT utilitzarà el component Guacamole Server "as-is" ja que no cal fer cap adaptació sobre la manera en que s'encapsula el protocol RDP. El component Client sí que es modificarà però només per reduïr-lo a la mínima expressió, ja que l'únic que necessitem és integrar el visor remot a la nostra aplicació actual.

La API de Guacamole proporciona una manera comuna i eficient de transportar la comunicació client-servidor sobre un túnel basat en Javascript (que al seu torn pot estar o no construït sobre WebSocket), i un client Javascript que implementa el protocol i és capaç de renderitzar el visor remot. Aquest és el component més important de cara a integrar Guacamole a ITSAT.

El client Javascript permèt capturar events de ratolí i teclat, i fins i tot disposa d'un teclat propi integrat al visor remot i disposa de "clipboard" per copiar i enganxar text.

ARQUITECTURA

Tal i com ja s'ha explicat, Guacamole no és una aplicació web autocontinguda sinó que està formada de moltes parts. De fet l'aplicació web està pensada per ser tan lleugera com sigui possible.



Els usuaris es connecten al servidor de Guacamole a través del seu navegador web. El client de Guacamole, escrit en Javascript, és servir als usuaris com a part dels assets de la pàgina web, d'igual manera que el contingut HTML i CSS. Un cop carregat, aquest client es connecta de nou al servidor d'on s'ha descarregat mitjançant el protocol de Guacamole.

El backend de l'aplicació web es limita a rebre les comunicacions dels clients que ha anat servint i les redirigeix cap al dimoni guacd, el proxy natiu de Guacamole resultant de compilar el projecte Guacamole Server. Aquest proxy interpreta el contingut del protocol Guacamole i és el que fa la traducció a altres protocols reals com ara VNC i RDP. A partir d'aquí el proxy ja delega en programes de tercers que implementen els protocols reals, i rep les seves respostes, les torna a traduïr al protocol Guacamole i les reenvia cap al backend de l'aplicació web.

Aquesta arquitectura permet implementar un servei totalment agnòstic del protocol real que es fa servir. Ni al client Javascript ni a l'aplicació web els hi cal saber res sobre el protocol real que en realitat s'està fent servir.

EL PROTOCOL DE GUACAMOLE

L'aplicació web no entén cap protocol d'escriptori remot, ni VNC ni RDP ni cap altre protocol "real". L'únic protocol que entén és el protocol de Guacamole, que també és un protocol pensat per renderitzar vistes i transportar events però adaptat al cas d'ús concret de Guacamole.

EL DIMONI GUACD

guacd està en el centre d'aquest esquema arquitectònic, i és l'encarregat de carregar dinàmicament els clients que suporten els diferents protocols d'escriptori remot, als quals anomena "client plugins".

guacd és un "procés dimoni" que s'instal·la conjuntament amb l'aplicació web i s'executa permanentment en background, esperant rebre conexions TCP per part de l'aplicació web en un determinat port. L'executable del dimoni en sí tampoc entén RDP ni VNC, ni tan sols el protocol Guacamole sencer. Només la part necessària per discernir quin protocol real s'ha de suportar per cada conexió que li arriba. Quan ho sap carrega dinàmicament el client plugin que sap fer la traducció entre Guacamole i RDP o Guacamole i VNC, que funciona com a procés independent de guacd i es fa responsable de tot el procés de comunicació entre ell i el backend de l'aplicació web.

Per tal d'uniformar i reduïr la complexitat del codi dels plugins tots ells depenen de la llibreria comuna libguac, escrita en C i que permet fer la comunicació mitjançant el protocol Guacamole una mica més fàcil.

L'APLICACIÓ WEB

La part de Guacamole amb la qual l'usuari final interacciona directament és l'aplicació web.

L'aplicació que vé per defecte en el projecte Guacamole Client ja és molt mínima de per sí. Només implementa un parell de vistes i una capa bàsica d'autenticació basada en usuari/contrasenya.

Com que ITSAT té uns requeriments de seguretat molt més rigorosos la nostra aplicació aprofitarà el client Javascript i les classes de Java bàsiques però proporcionarem el nostre propi mecanisme d'autenticació basat en certificats (el mateix que l'aplicació PHP que tenim fins ara).

Aquesta part de la infraestructura s'englobarà en un subprojecte apart anomenat sat-desktops, que es compilarà apart i es gestionarà amb Maven[14].

[14] http://maven.apache.org/

4. DESENVOLUPAMENT TÈCNIC

Aquest apartat està subdividit en dues parts ben diferenciades: Disseny i Implementació. En la primera explicaré com s'han dissenyat les solucions implementades a partir de les tecnologies vistes fins ara. A Implementació s'explicaràn els detalls de la seva implementació.

4.1 DISSENY

4.1.1 ALTA DE MÀQUINES WINDOWS

A l'hora d'acceptar l'alta de màquines Windows no cal fer cap procés en especial.

Les úniques adaptacions que caldrà fer per acceptar aquestes màquines serà adaptar la vista del registre per afegir un nou element a la llista desplegable de sistemes operatius (OS).

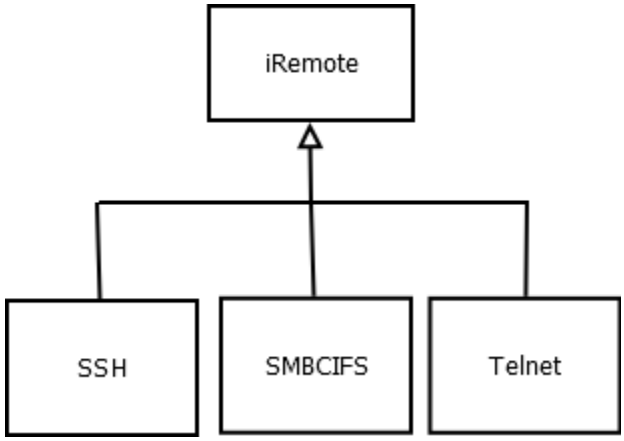
4.1.2 INTEGRACIÓ DE SISTEMES NO INTERACTIUS AMB WINDOWS

Prèviament a aquest projecte a ITSAT existia una sola classe PHP anomenada SSH.php que era l'encarregada de gestionar tots els aspectes de les connexions remotes a les màquines gestionades.

Amb el nou requeriment de suportar màquines Windows aquesta implementació és massa restringida perquè aquesta classe només pot funcionar amb màquines Unix (Windows no suporta nativament el protocol ssh). En comptes d'ampliar aquesta classe i fer-la créixer desmesuradament he apostat per refactoritzar tota la gestió de connexions remotes en un disseny més modular de manera que el codi no sigui innecessàriament molt més complex i tampoc calgui fer adaptacions ad-hoc per tot el projecte.

L'estratègia de refactorització consistirà en crear una nova interfície (iRemote.php) que definirà els mètodes que hauràn d'implementar obligatoriament les classes que gestionin conexions remotes. D'aquesta manera els subsistemes d'ITSAT que fins ara referenciaven la classe SSH.php passaràn a referenciar una simple interfície. Per tal de que això funcioni la classe SSH.php haurà de passar a implementar aquesta nova interfície. Les noves classes que escriuré per donar suport a connexions Windows també implementaran iRemote. D'aquesta manera la resta de sistemes passaran a treballar amb objectes que implementen la interfície iRemote sense saber realment si són objectes SSH.php o d'algun altre tipus, però en qualsevol cas tindràn els mateixos mètodes (els de la interfície) i funcionaràn indistintament de la implementació que hi hagi per sota.

Aquest és el diagrama UML del disseny:



4.1.3 INTEGRACIÓ DE SISTEMES INTERACTIUS AMB WINDOWS

Tal i com s'ha explicat a l'apartat d'Anàlisi d'antecedents i factibilitat, Guacamole és una solució autocontinguda basada en C i Java que no es pot integrar fàcilment en una codebase de PHP com una llibreria més.

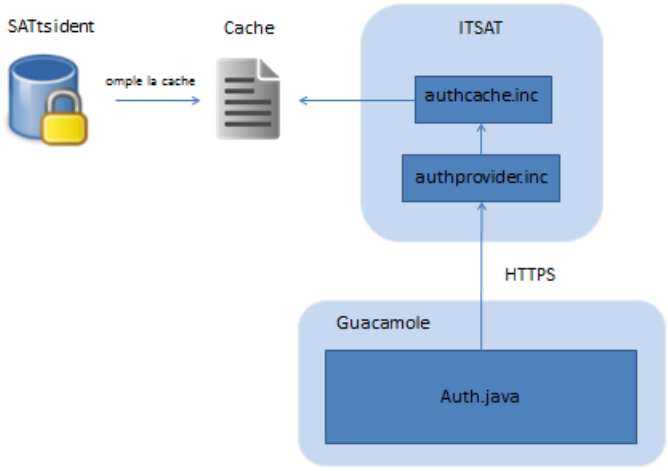
La integració dels escriptoris remots de Guacamole al sistema ITSAT presenta dos reptes importants:

- Accés a les credencials d'autenticació custodiades per ITSAT

  Guacamole disposa del seu propi mecanisme d'emmagatzematge de credencials de connexió (basat en fitxers de text pla en format XML), però en el nostre cas aquestes credencials ja es troben guardades de manera segura a la base de dades d'ITSAT, i Guacamole les haurà de recuperar d'allà.

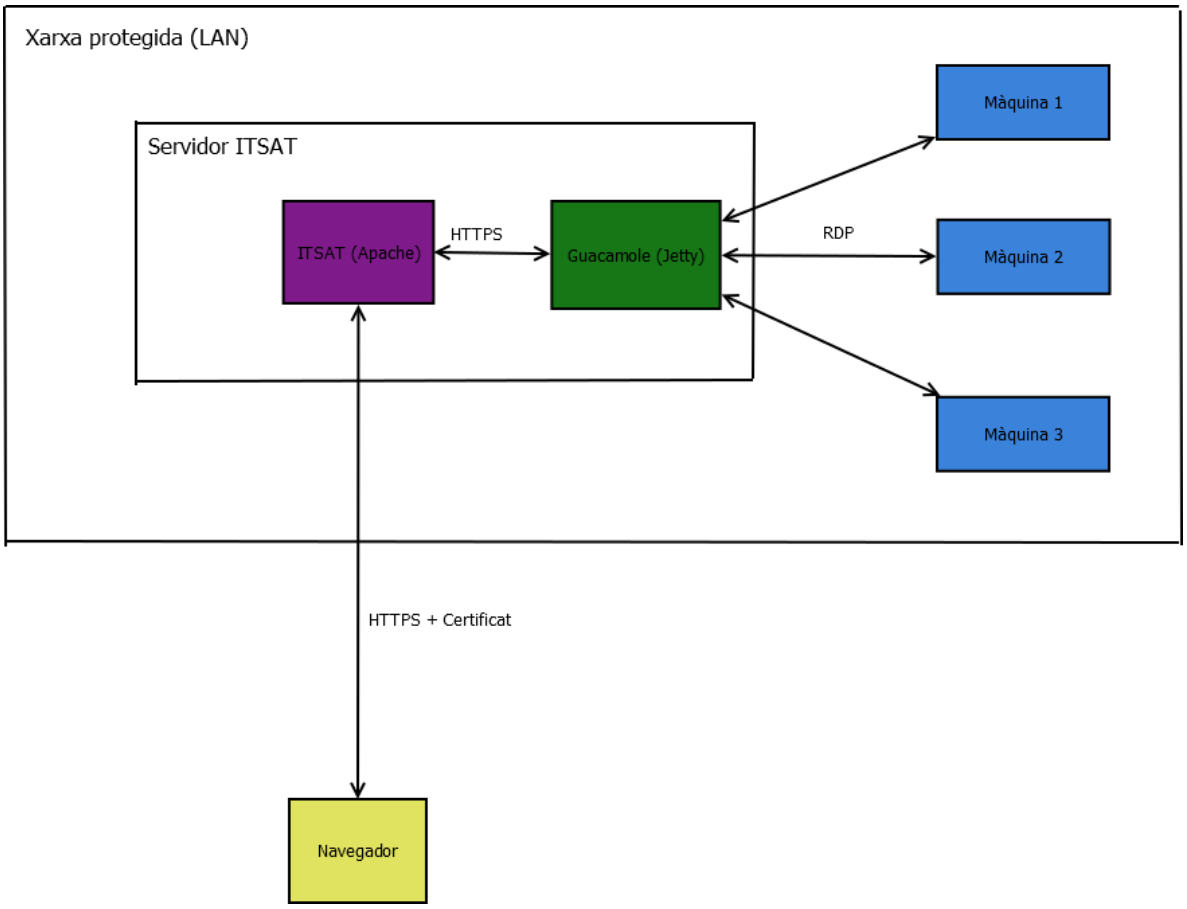
  La solució proposada és crear una nova ruta ITSAT servida per un controlador anomenat guacamole-service.php. Aquest controlador només acceptarà peticions efectuades des del procés de Guacamole corrent dins la mateixa màquina, i s'implementarà de tal manera que li pugui proporcionar totes les dades d'ITSAT que li puguin fer falta.

  El diagrama següent mostra un esquema primerenc d'aquest disseny. La classe authprovider.inc (codi PHP) accedeix a les credencials custodiades a la taula SATtsident de la base de dades mitjançant els mecanismes normals d'ITSAT (que inclou l'ús d'una cache intermitja). Aquest servei/controlador accepta peticions HTTPS del procés de Guacamole.



- Integració amb el sistema d'autenticació d'usuaris d'ITSAT i la vista

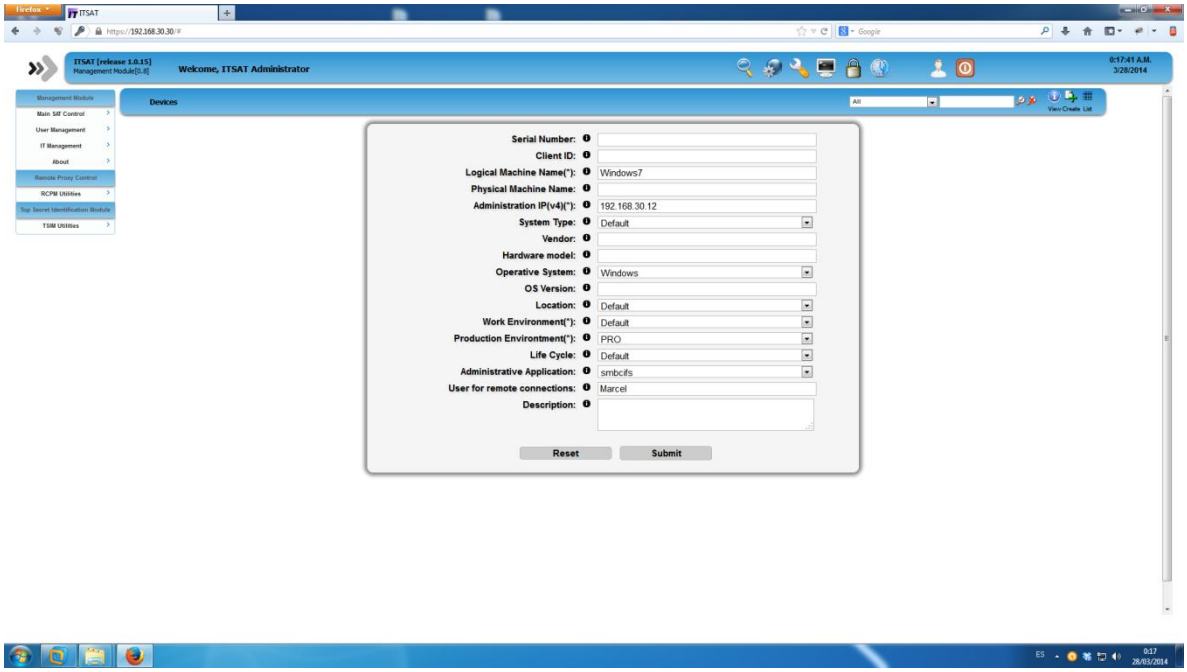
  Una altra consideració molt important a tenir en compte és que l'accés a aquests escriptoris remots ha d'estar protegit pel sistema de credencials d'ITSAT (és a dir no seria acceptable que qualsevol actor sense identificar hi pogués accedir).
  Per tal d'aconseguir aquest efecte la conexió browser <-> guacamole no s'efectuarà directament sinó que passarà a través del servidor Apache que utilitza ITSAT: browser <-> ITSAT (Apache) <-> Guacamole (Jetty). D'aquesta manera s'obliga a l'usuari a identificar-se davant de ITSAT si pretén fer peticions sobre les URLs relacionades amb el funcionament de Guacamole.


Xarxa protegida (LAN)
Servidor ITSAT
ITSAT (Apache)
HTTPS
Guacamole (Jetty)
RDP
Màquina 1
Màquina 2
Màquina 3
HTTPS + Certificat
Navegador

4.2 IMPLEMENTACIÓ

En aquesta secció es detallaran les actuacions concretes que s'han dut a terme per implementar la solució proposada.

4.2.1 ALTA DE MÀQUINES WINDOWS



La definició dels camps que es mostren el formulari de registre es fa a través del fitxer */usr/webapp/mngm/nmgm-machines.inc:*

```
static private $objdefs = array(
  ...,
  array(
    "label" => "Operative System",
    "desc" => "Device OS",
    "name" => "os",
    "type" => "SELECT",
    "typeparm" => "Linux,AIX,SunOS,HP-UX,Windows",  // <---
    "valid" => "alphanum2",
    "req" => "0",
    "id" => "0",
    "labelid" => "0",
    "view" => "1",
    "list" => "1",
    "edit" => "1"
  ),
  ...
);
```

En quant a la base de dades no ha calgut fer cap modificació. Quan l'usuari selecciona la nova opció "Windows" durant el l'alta d'una nova màquina Windows aquest string és exactament el que s'acaba guardant a la columna "os" de la taula "SATmachines".

4.2.2 INTEGRACIÓ DELS SISTEMES NO INTERACTIUS AMB WINDOWS

Aquesta és la implementació de la interfície iRemote que s'ha debatut a l'apartat de disseny:

Interfície iRemote.php

```
define("ROPEN_MODE_CMD", "0");
define("ROPEN_MODE_SHELL", "1");
define("SHELL_CMD_TIMEOUT", "80000");


define("CAN_CONNECT", "0");
define("CAN_TRANSFER", "1");

/**
 * The iRemote interface specifies the common methods that all protocol-related
 * classes in ITSAT should have.
 */
interface iRemote {

    /**
     * The open method sets up`a new connection to a remote machine using the
protocol modeled by the implementing class.
     * A new CID is created and associated to this connection.
     */
    public function open();

    /**
     * The login method establishes the user that will perform the connection to
the remote machine
     * on behalf of ITSAT
     *
     * @access public
     * @param string $md the format of the credential. It can be a password or a
certificate
     * @param string $user the username which will be used to connect to the
remote machine
     * @param string $credential the password or certificate associated to the
given user
     * @return boolean true if the given credentials are valid and ITSAT
successfully logs in to the remote machine, false otherwise
     */
    public function login($md, $user, $credential);
```

```
    /**
     * The close method terminates the connection to the remote machine,
efectively discarding its CID
     *
     * @access public
     */
    public function close();

    /**
     * The cinfo method fetches the CID associated to the connection
     *
     * @access public
     */
    public function cinfo();

    /**
     * The ping method sends a ping against the remote machine
     *
     * @access public
     */
    public function ping();

    /**
     * The exec method runs a single noninteractive command in the remote
machine
     *
     * @access public
     * @param string $cmd
     * @return string output of the operation
     */
    public function exec($cmd);

    /**
     * The send method sends a local file in the ITSAT server to a remote
machine
     *
     * @access public
     * @param string $lpath path to local file
     * @param string $rpath path in remote machine where the file is to be saved
     * @return string output of the operation
     */
    public function send($lpath, $rpath);

    /**
     * The recv method retrieves a file from a remote machine and stores it in
the ITSAT server
     *
     * @access public
     * @param string $rpath path to the file in the remote machine that is to be
fetched
     * @param string $lpath path in the ITSAT server where the file is to be
saved
     * @return string output of the operation
     */
```

```
    public function recv($rpath, $lpath);

    /**
     * The checkCapability method checks if a given capability is available in
     * the protocol modeled by the implementing class
     *
     * @access public
     * @param string $capability The capability to check
     * @return boolean true if the protocol supports the given capability
     */
    public function checkCapability($capability);
}
```

Els mètodes que obliga a implementar a totes les classes que l'extenguin són els següents:

- open()
Inicia una nova connexió a la màquina remota.

- login()
Presenta un joc de credencials per identificar-se davant una màquina remota. Cal haver obert una connexió prèviament mitjançant open().

- close()
Tanca la connexió.

- cinfo()
Retorna el CID, un identificador que ITSAT utilitza per identificar les connexions a màquines remotes que s'han obert en cada moment. Si la connexió no s'ha obert prèviament ha de retornar NULL.

- ping ()
Comprova si la màquina està "viva". Cal haver obert la connexió primer.

- exec()
Executa una comanda a la màquina remota. Cal haver obert la connexió i haver-se identificat amb login().

- send()
Envia un fitxer a la màquina remota. Cal haver obert la connexió i haver-se identificat amb login().

- recv()
Rep un fitxer de la màquina remota. Cal haver obert la connexió i haver-se identificat amb login().

- checkCapabilities()
Informa de les capacitats que ofereix el protocol, que poden ser CAN_CONNECT i/o CAN_TRANSFER. CAN_CONNECT permet connectar-se i executar comandes remotament, CAN_TRANSFER permet enviar i rebre arxius. Els mètodes associats a aquestes capacitats llencen excepcions si es criden sobre una classe que no les ofereixi.

Les interfícies de PHP tenen la limitació de que no permeten definir atributs obligatoris per a les classes que les han d'implementar.

És per això que la informació per localitzar el host (IP, port TCP) no es modela a la interfície sinó que es deixa al criteri de cada classe implementadora.

La manera més neta de passar aquesta informació a les classes és a través del mètode constructor de la classe.

A continuació es mostra un fragment de la classe SSH.php (la que ja existia) un cop ha passat a implementar la interfície iRemote, i un exemple sobre com s'ha de instanciar i utilitzar.

```php
<?php

/**
 * SSH class
 *
 * This class implements the iRemote interface, thus efectively modelling
 * a connection to a remote machine using the SSH protocol
 *
 * All methods throw Exceptions on error
 */
class SSH extends SATObject implements iRemote
{
    private $host;
    private $port;
    private $timeout;
    private $mode;

    /**
     * The $capabilities attribute lists the capabilities of the protocol
     *
     * @access private
     */
    private $capabilities = Array(CAN_CONNECT);

    public function __construct($host, $port, $timeout = 10, $mode)
    {
        $this->host = $host;
        $this->port = $port;
        $this->timeout = $timeout;
        switch ($mode) {
            case ROPEN_MODE_CMD:
                $this->open("SSH_SHARED");
                break;
            case ROPEN_MODE_SHELL:
                $this->open("SSH_UNSHARED_SHELL");
                break;
            default:
                throw new Exception("SSH:ERROR unhandled mode");
        }
    }

    ... Implementació dels mètodes de la interfície
}
```

```php
// Exemple
$connection = new SSH($machine, 22, 15, 10, $mode);
$connection->open();
$connection->login('password', 'root', 'root');
$output = $connection->exec('ls -l');
```

A partir d'aquí es van poder implementar els protocols Telnet i SMB en classes totalment separades.

Tota i la manca total de seguretat del protocol Telnet es va decidir implementar-lo per donar suport a versions molt antigues de Windows, i sempre tenint en compte el requisit de que el servidor ITSAT s'ha de trobar en el mateix entorn segur que la resta de màquines gestionades. A més a més una altra mancança d'aquest protocol de comunicació és que no permet la transmissió de fitxers, i per tant se'n desaconsella l'ús en la mesura del possible. A continuació es poden veure els detalls de la implementació. La solució es basa en comunicació via sockets de php a la màquina destí sobre el port TCP 23 (el port que utilitza Telnet per defecte, tot i que la classe permet reconfigurar-lo).

Classe Telnet.php (veure annex B.1)

El protocol "definitiu" per Windows s'ha implementat en la classe SMBCIFS. Aquesta solució combina el programari samba[15] per transferir fitxers i winexe[16] per executar comandes de MS-DOS remotament, i és el protocol recomanat per gestionar màquines Windows.

Samba és una implementació lliure dels protocols SMB i CIFS de Microsoft. Són protocols específicament dissenyats per permetre la compartició de fitxers entre sistemes operatius basats en Windows i Unix.

Winexe és un projecte molt menys conegut que permet executar comandes de Windows remotament des d'una màquina Unix i de forma segura.

Classe SMBCIFS.php (veure annex B.2)

Finalment totes aquestes diferents implementacion de la interfície iRemote s'ha encapsulat darrere un classe tipus factoria, que retorna un nou objecte iRemote del tipus desitjat.

En la implementació d'aquesta classe es pot veure que també s'han creat 2 implementacions més addicionals, sshsftp i telnetftp. Com que ssh i telnet no suporten la transmissió de fitxers aquesta mancança es pot suplir a través d'un segon protocol dedicat específicament a això.

Les classes Telnet i SSH segueixen sent útils per les màquines que disposen d'aquests protocols però no de FTP, com podria ser el cas d'un router.

[15] https://www.samba.org/
[16] http://sourceforge.net/projects/winexe/

Finalment s'ha creat una classe factoria (ConPool.php) amb la missió de crear objectes que implementen la interfície iRemote i retornar-los per ser utilitzats. Per exemple, la crida $conPool->openConnecion($machine, 'ssh') retorna un objecte SSH, i la crida $conPool->openConnection($machine, 'smbcifs') retorna un objecte SMBCIFS.

Classe ConPool.php

```php
<?php

include_once("ssh.inc");
include_once("telnet.inc");
include_once("smbcifs.inc");

class ConPool extends SATobject
{
    public $supportedProtocols = array('ssh', 'telnet', 'smbcifs');

    public function openConnection($machine, $protocol = "ssh")
    {
        if (!in_array($protocol, $this->supportedProtocols)) {
            $this->error("Device $machine doesn't suport $protocol
protocol");
            return NULL;
        }
        if ($protocol == "ssh") {
            $rcon = new SSH($machine, "22", 15);
            $this->info("CONPOOL: getting SSH resource  for machine
$machine");

        }
        if ($protocol == "sshsftp") {
            $rcon = new SSHSFTP($machine, "22", 15);
            $this->info("CONPOOL: getting SSH resource  for machine
$machine");

        }
        if ($protocol == "telnet") {
            $rcon = new Telnet($machine, "23", 15);
            $this->info("CONPOOL: getting Telnet resource  for machine
$machine");
        }
        if ($protocol == "telnetftp") {
            $rcon = new TelnetFTP($machine, "23", 15);
            $this->info("CONPOOL: getting Telnet resource  for machine
$machine");
        }
        if ($protocol == "smbcifs") {
            $rcon = new SMBCIFS($machine);
            $this->info("CONPOOL: getting CIFS resource  for machine
$machine");
        }
    }

    ...
}
```

Amb aquesta refactorització les comandes **sat-rexec**, **sat-rput** i **sat-rget** només han de comprovar amb quin tipus de màquina estan a punt de treballar i demanar el tipus d'implementació desitjat. La resta del codi no ha calgut adaptar-lo significativament.

Per exemple, aquest seria un esquema del codi de la comanda sat-rexec:

```php
<?php

$connPool = new ConPool();

// Informació recuperada de la base de dades
$machine = 'oldtimes';
$protocol = 'smbcifs';

$connection = $connPool->openConnection($machine, $protocol);

// Aquestes crides funcionen amb qualsevol implementació
// de la interfície iRemote, tant si és Windows com Unix.
$connection->open();
$connection->login($credentials);

$connection->exec($command);

$connection->close();
```

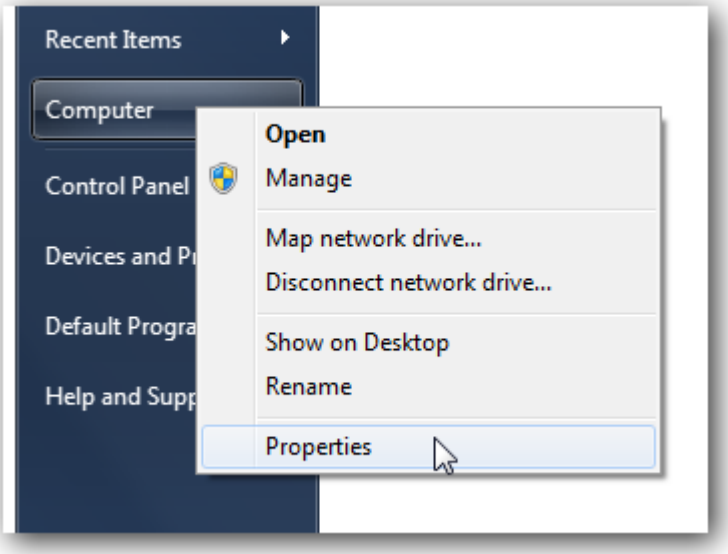
4.2.3 INTEGRACIÓ DELS SISTEMES INTERACTIUS AMB WINDOWS

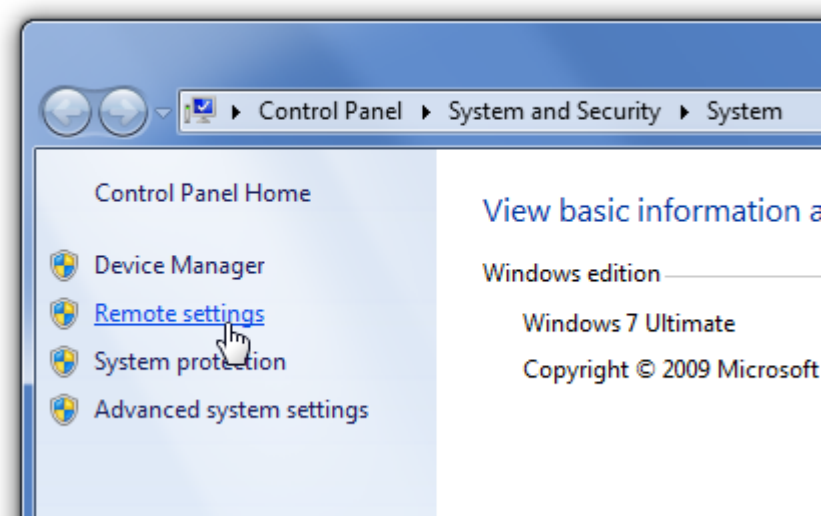
INSTAL·LANT GUACAMOLE

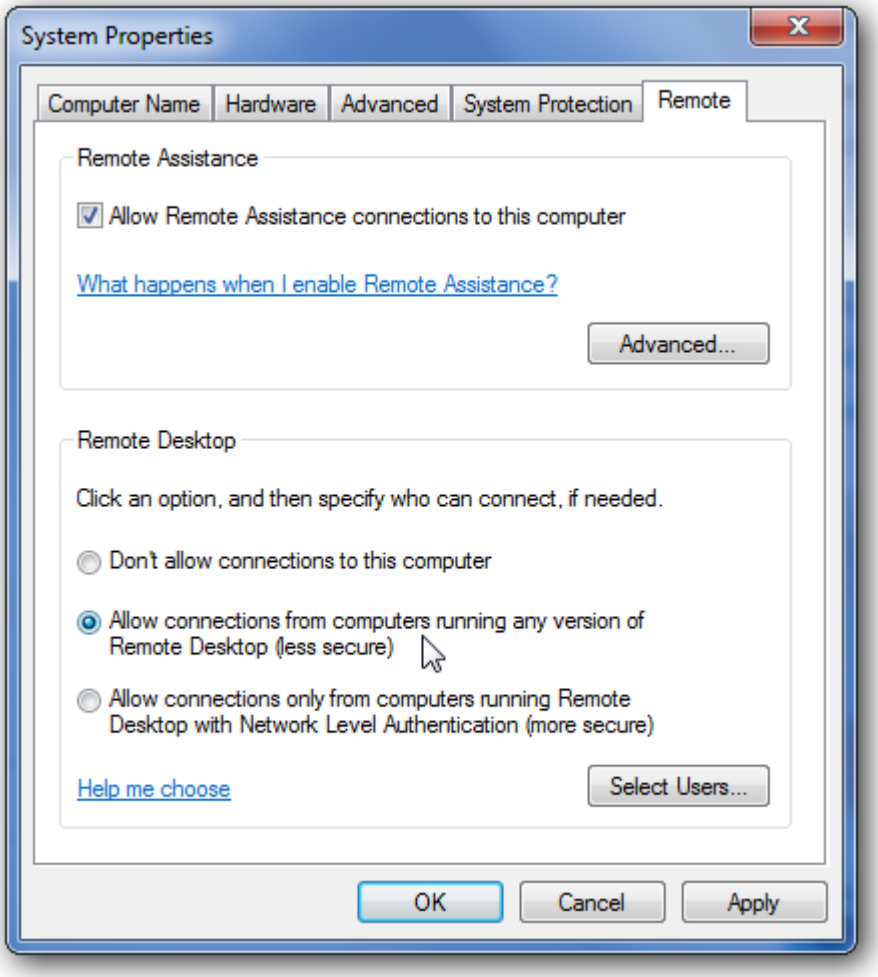
ACTIVACIÓ DEL SERVIDOR RDP A LES MÀQUINES GESTIONADES

Cal tenir en compte que les màquines remotes a les que ens volem conectar també han de tenir instal·lat un servidor RDP al qual el client de Guacamole es pugui conectar. La majoria de versions modernes de Windows ja tenen per defecte un servidor RDP, però en la majoria de casos s'ha d'habilitar manualment.

Aquests són els passos que s'haurien de seguir en un Windows 7 com el que es va fer servir durant el desenvolupament:


Recent Items
Computer
Control Panel
Devices and Pr
Default Progra
Help and Supp
Open
Manage
Map network drive...
Disconnect network drive...
Show on Desktop
Rename
Properties


Control Panel ▸ System and Security ▸ System
Control Panel Home
Device Manager
Remote settings
System protection
Advanced system settings
View basic information a
Windows edition
Windows 7 Ultimate
Copyright © 2009 Microsoft



En l'última fotografia s'aprecia un dels reptes que no va donar temps de resoldre durant el desenvolupament del projecte. La comunicació entre el client plugin de Guacamole i el servidor RDP de la màquina gestionada no treballa en el mode més segur possible ja que Guacamole no suporta Network Level Authentication.

Aquesta mancança es resol doncs mantenint tant la màquina gestionada com el servidor ITSAT en una xarxa local de confiança i correctament securitzada. En aquest cas l'única manera de connectar a la màquina remota des de l'exterior seria passant abans per l'aplicació ITSAT, que sí que està ben protegida amb un sistema d'accés per credencials.

INSTAL·LACIÓ DE GUACAMOLE SERVER AL SERVIDOR ITSAT

INSTAL·LACIÓ PER PAQUETS

Guacamole s'inclou per defecte en la majoria de repositoris de les principals distribucions de Linux. Per desgràcia la plataforma suportada per ITSAT (Debian 6/7) disposa de versions massa antigues, així que el procés d'instal·lació s'ha de fer manualment, compilant els executables a partir del codi font.

INSTAL·LACIÓ MANUAL DE GUACAMOLE SERVER

La instal·lació manual per defecte no inclou el suport del protocol RDP. Per tal d'habilitar el suport al protocol RDP necessitarem instal·lar el paquet **libfreerdp-dev** abans de començar amb el procés de compilació. Aquest paquet conté el codi de la llibreria de desenvolupament FreeRDP, que també és una implementació de codi obert http://www.freerdp.com.

Per compilar els components de C de Guacamole Server cal tenir instal·lat el paquet **build-essentials**, que proporciona el compilador de GCC i altres eines necessàries per al procés de compilació com ara la comanda **make**.

Aquesta és la llista de requeriments sencera per compilar tots els components de Guacamole Server, inclosos els requeriments opcionals:

Nom de la llibreria	Obligatori?	Descripció	Inclosa?
Cairo	Sí	La llibreria libguac utilitza Cairo per renderitzar els gràfics del client.	**Sí**
libpng	Sí	També necessita libpng per crear imatges PNG, que és en el que es basa el visor remot.	**Sí**
FreeRDP	Només per RDP	Llibreria bàsica per habilitar el protocol RDP.	**Sí**
Pango	Només per SSH	Llibreria de fonts necessària per habilitar el plugin de SSH.	**No**
libssh	Només per SSH	Llibreria bàsica per habilitar el protocol SSH.	**No**
libVNCServer	Només per VNC	Llibreria bàsica per habilitar el protocol VNC.	**No**
libpulse	No	Si s'inclou aquesta llibreria es pot habilitar un suport de so experimental utilitzant PulseAudio. Aquesta llibreria és opcional i en tot cas aquest mode només es pot utilitzar en combinació amb el protocol VNC.	**No**

Nom de la llibreria	Obligatori?	Descripció	Inclosa?
libssl	No	Incloure la llibreria libssl li permet al dimoni guacd acceptar connexions securitzades a través de SSL o TLS.	**No***
libvorbis	No	Aquesta llibreria permet utilitzar el format OggVorbis per transmetre so al client final, un format més eficient que el WAV que utilitza per defecte.	**No**

***** El motiu pel qual he decidit no utilitzar connexions securitzades amb TLS entre guacd i l'aplicació web és que d'acord a l'arquitectura del sistema que he dissenyat aquests dos components resideixen dins la mateixa màquina, per tant tota comunicació entre ells es realitza a través d'un socket local. Si l'aplicació web i el dimoni guacd residissin en màquines diferents dins la mateixa xarxa tindria sentit habilitar TLS ja que en aquest cas la comunicació viatjaria sense cap tipus d'encriptació entre elles.

Aquest és l'output de la comanda **configure**. Recordem que segons el procés de compilat estàndard en sistemes basats en Unix aquesta comanda s'encarrega de comprovar que totes les dependències necessàries per la compilació es troben correctament instalades al sistema.

```
$ ./configure --with-init-dir=/etc/init.d
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for...
...

----------------------------------------------
guacamole-server version 0.8.3
----------------------------------------------

  Library status:

    freerdp ..... yes
    pango ..... no
    libssh ..... no
    libssl ..... no
    libVNCServer ..... no
    libvorbis ..... no
    libpulse ..... no

  Protocol support:

     RDP ..... yes
     SSH ..... no
     VNC ..... no

  Init scripts: /etc/init.d

Type "make" to compile guacamole-server.
```

Com a paràmetre opcional li hem passat –with-init-dir al script. Aquest flag indica si també volem que el procés d'instalació crei per nosaltres una sèrie d'scripts de control de Guacamole Server, i a quin directori s'han de guardar. El directori estàndard per guardar aquest tipus d'scripts en la distribució Debian és **/etc/init.d**. A més a més la presència dels scripts en aquest directori també

fan que el procés es posi en marxa tant bon punt arrenca el sistema operatiu, per tant ens dóna el màxim uptime.

Si el procés configure acaba amb èxit només queda executar la comanda make per iniciar el procés de compilat de les fonts, que pot arribar a tardar uns quants minuts:

```
$ make
Making all in src/libguac
make[1]: Entering directory `/home/everis/guacamole/guacamole-server/src/libguac'
...
make[1]: Leaving directory `/home/everis/guacamole/guacamole-server/src/protocols/vnc'
make[1]: Entering directory `/home/everis/guacamole/guacamole-server'
make[1]: Nothing to be done for `all-am'.
make[1]: Leaving directory `/home/everis/guacamole/guacamole-server'
```

Finalment s'instalen els executables a la seva ruta final fent **make install**, i s'actualitza la cache del sistema de llibreries instalades fent **ldconfig**. A diferències de les comandes executades fins ara, aquest parell de passos necessiten permissos de root per poder executar-se, ja que cal fer escriptures en directoris del sistema.

```
# make install && ldconfig
Making install in src/libguac
make[1]: Entering directory `/home/everis/guacamole/guacamole-server/src/libguac'
make[2]: Entering directory `/home/everis/guacamole/guacamole-server/src/libguac'
...
----------------------------------------------------------------------
Libraries have been installed in:
   /usr/local/lib

If you ever happen to want to link against installed libraries
in a given directory, LIBDIR, you must either use libtool, and
specify the full pathname of the library, or use the `-LLIBDIR'
flag during linking and do at least one of the following:
   - add LIBDIR to the `LD_LIBRARY_PATH' environment variable
     during execution
   - add LIBDIR to the `LD_RUN_PATH' environment variable
     during linking
   - use the `-Wl,-rpath -Wl,LIBDIR' linker flag
   - have your system administrator add LIBDIR to `/etc/ld.so.conf'

See any operating system documentation about shared libraries for
more information, such as the ld(1) and ld.so(8) manual pages.
----------------------------------------------------------------------
make[2]: Nothing to be done for `install-data-am'.
make[2]: Leaving directory `/home/everis/guacamole/guacamole-server/src/protocols/vnc'
make[1]: Leaving directory `/home/everis/guacamole/guacamole-server/src/protocols/vnc'
make[1]: Entering directory `/home/everis/guacamole/guacamole-server'
make[2]: Entering directory `/home/everis/guacamole/guacamole-server'
make[2]: Nothing to be done for `install-exec-am'.
make[2]: Nothing to be done for `install-data-am'.
make[2]: Leaving directory `/home/everis/guacamole/guacamole-server'
make[1]: Leaving directory `/home/everis/guacamole/guacamole-server'
#
```

Arribats a aquest punt el dimoni guacd ja es troba instalat i a punt per començar a servir peticions, però encara no està en marxa. Per iniciar-lo només cal reiniciar l'equip o servir-se de la comanda service estàndard en totes les distribucions de Linux basades en Debia:

```
# service guacd start
#
```

INSTALACIÓ DEL WEB SERVER DE GUACAMOLE

L'aplicació Java de Guacamole està localitzada a la ruta develop/extra/java/sat-desktops del projecte principal. Per tal de compilar aquest subprojecte es va crear un script que automatitzés el procés, que compila el codi Java mitjançant Maven i mou el .WAR resultant al directori que el webserver estara rebre'l.

```
#!/bin/bash

echo "Build .war file"
mvn clean package

echo "Purge old deployments"
rm -rf /opt/itsat/tmp/guacamole/*

echo "Move the .war to an apropiate location"
mv target/sat-desktops.war /opt/itsat/install/scripts/

echo "Create symbolic link to the .war file in $JETTY_HOME/webapps"
ln -sf /opt/itsat/install/scripts/sat-desktops.war /opt/jetty/webapps/sat-desktops.war

echo "Purging old logs"
rm -rf /opt/itsat/var/log/guacamole/*
echo "Restarting ITSAT-guacamole service"
/etc/init.d/ITSAT-guacamole restart

echo "Deployment completed"
```

El projecte **sat-desktops** en sí és un petit projecte web basat en Maven que inclou el codi JS que ITSAT integra a la interfície web, a més a més d'una classe Java que ha de implementar una interfície Java definida per Guacamole per tal de fer efectiu el túnel entre el navegador i el component guacd.

A l'annex B es poden consultar tant el codi Java com el servei guacamole-service.php amb que el codi Java interactua per comunicar-se amb la resta del sistema ITSAT.

5. PROVES I VALIDACIÓ

En el moment de començar el projecte ITSAT no disposava de cap sistema de testeig automatitzat. Tot i que paral·lelament al desenvolupament d'aquest projecte se'n va dur a terme un altre per suportar una mica de testeig, aquest es va centrar bàsicament en cobrir altres parts d'ITSAT més consolidades. En quant a la integració de Windows, tot el testing i les proves de validació es van dur a terme manualment sobre un entorn semireal.

Per tal de dur a terme aquestes proves es van preparar dues màquines virtuals en l'ordinador de desenvolupament, una basada en el sistema operatiu Windows XP i una altra basada en Windows 7. A continuació es detallen alguns d'aquests tests relacionats amb cada àrea del projecte.

5.1 DONAR D'ALTA LES DUES MÀQUINES AL SISTEMA ITSAT

1. Accedir a la aplicació web ITSAT com a administrador.

2. Registrar-les amb el formulari de registre.

3. Comprovar manualment que les seves dades es registren correctament a la BD.

4. Comprovar que apareixen a les vistes, com qualsevol altre màquina Linux/Unix.

5.2 COMPROVAR FUNCIONAMENT DE LES COMANDES D'ADMINISTRACIÓ

1. Accedir al sistema operatiu del servidor ITSAT com a root.

2. Executar sobre cada màquina un script MS-DOS mitjançant sat-sexec.

3. Executar remotament una comanda MS-DOS sobre cada màquina mitjançant sat-rexec.

4. Enviar un fitxer a cada màquina mitjançant sat-rput.

5. Rebre un fitxer a cada màquina mitjança sat-rget.

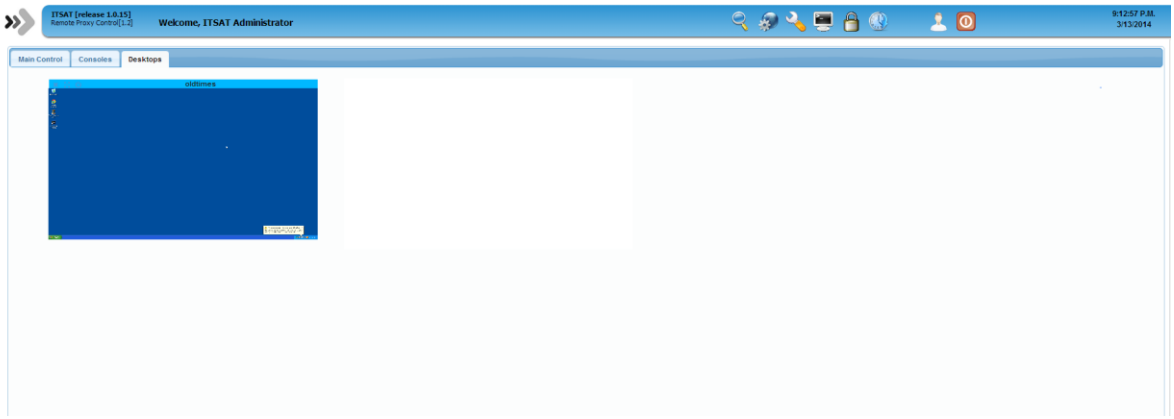
6. Repetir 1-4 utilitzant les diferents modalitats de sat-remotedo, executant cada comanda sobre les dues màquines a la vegada (-M "winxp, win7").

5.3 COMPROVAR FUNCIONAMENT DE L'ESCRIPTORI REMOT

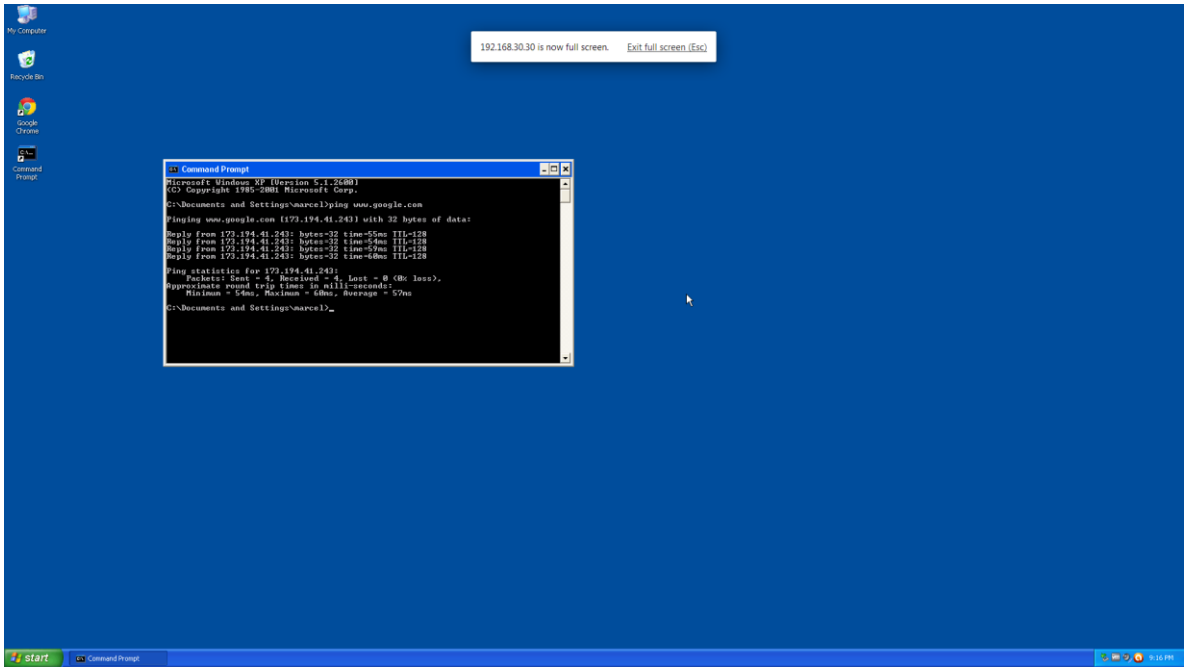
1. Accedir a l'aplicació web ITSAT com a administrador

2. Obrir al mòdul de conexions remotes (icona consola).

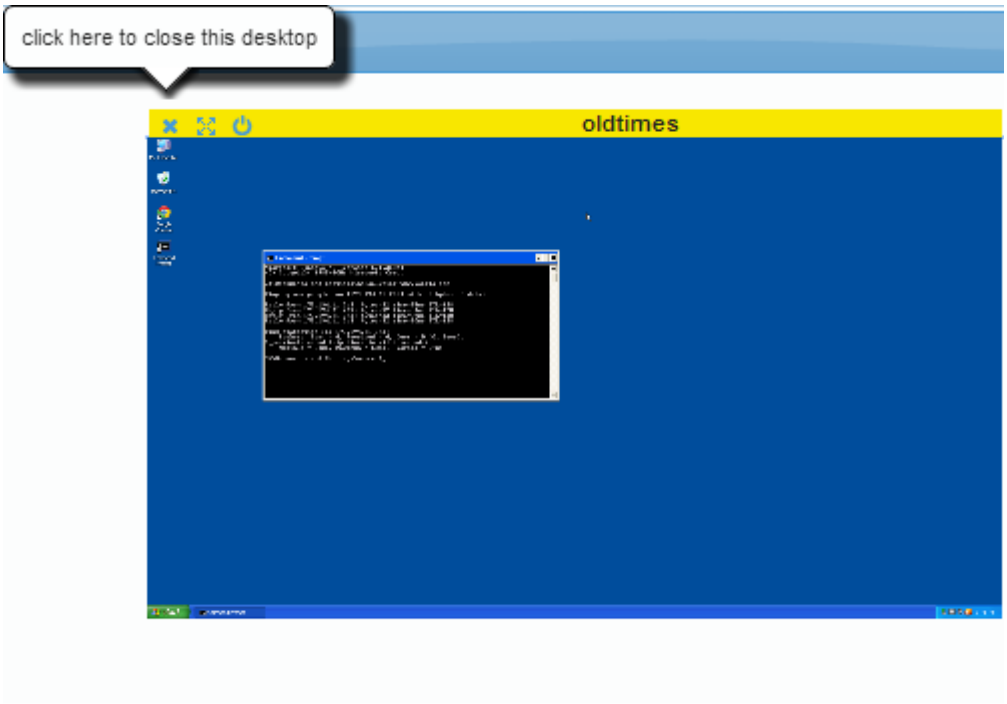
3. Seleccionar l'entrada d'una de les màquines Windows. L'aplicació et redirigeix a la pestanya "Desktops", on apareix una nova miniatura de l'escriptori remot.



4. Clickant la miniatura l'escriptori remot s'ha de obrir a pantalla completa i ha de començar a reaccionar a l'input del teclat i el ratolí.



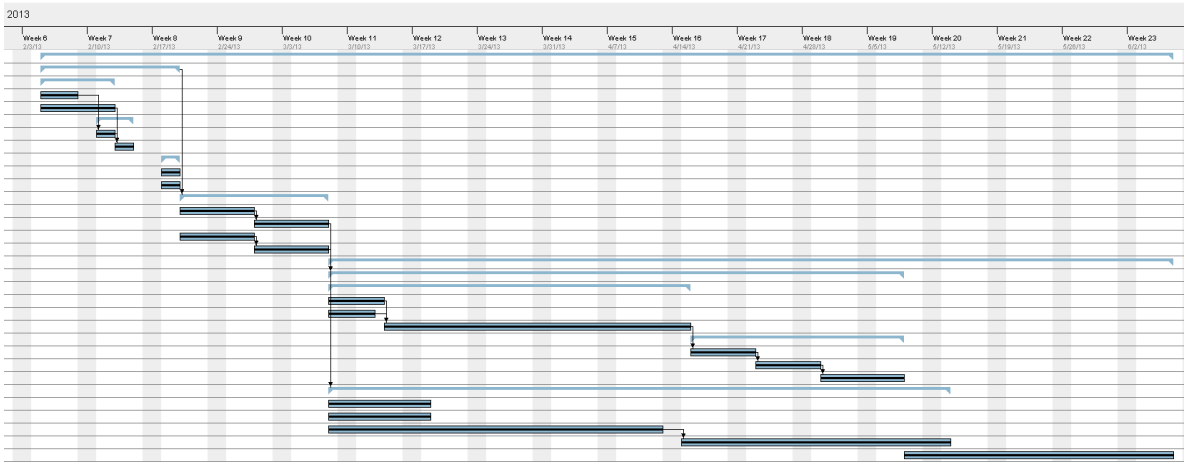
5. Sortir del mode pantalla completa presionant la tecla ESC. Comprovar que ratolí i teclat ja no controlen la màquina remota. Tancar la conexió a la màquina pitjant el botó de tancar, a l'extrem superior esquerre de la miniatura.



6. Repetir els passos 1-5 per a l'altra màquina virtual (Windows 7).

6. PLANIFICACIÓ FINAL

Aquest és l'estat en que va quedar la planificació final. En general els temps es van ajustar a la planificació i el projecte va ser acabat a temps, tot i que la redacció de la memòria i la presentació es van posposar durant un any.



7. ANÀLISI ECONÒMICA

Aquesta secció detalla la despesa real que ha acabat suposant la realització del projecte.

Despesa en Hardware

Lenovo ThinkPad SL510	200 € (valor total 600 €, prorratejat al 33%)

L'única despesa en hardware necessària ha estat la destinada al propi equip usat durant el desenvolupament del projecte.

Despesa en Software

VMWare Workstation 9	225 €
Llicència Windows XP multiequip	750 €
Llicència Windows 7 multiequip	1300 €
Microsoft Office 2013	540 €

Les despeses fetes en aquest apartat han estat la destinades al software de virtualització d'entorns per no haver de menester diverses màquines reals durant el desenvolupament del projecte, a les llicències de Windows per aquestes màquines virtuals i el software d'ofimàtica per redactar la memòria.

Despesa en Personal

Rol	**Dies**	**Hores**	**Preu Hora**	**Cost**
Project Manager	90	360	80 €/h	28.800 €
Programador	90	360	50 €/h	18.000 €

Total

El cost total ha ascendit a uns 50.615 €, que coincideix amb el cost inicial que s'havia estimat (uns 50.000€), ja que l'única despesa susceptible a disparar-se era per retrassos en el desenvolupament, que no n'hi van haver.

8. CONCLUSIONS

Tot seguit s'exposen les conclusions derivades de la realització del projecte.

8.1 ASSOLIMENT DELS OBJECTIUS PROPOSATS

Considero assolits els objectius especificats durant la planificació inicial, és a dir donar suport a màquines Windows dins el sistema ITSAT, si bé és veritat que no s'ha arribat a aconseguir una compatibilitat al 100%.

8.2 AMPLIACIONS I MILLORES

Com hem dit, durant l'execució del projecte es van explorar algunes vies de treball i funcionalitats extra que per algun motiu o altre no es van acabar integrant a ITSAT. Si continués col·laborant en el projecte penso que seria interessant centrar-se en alguna d'aquestes àrees.

8.2.1 SUPORTAR NETWORK LEVEL AUTHENTICATION

NLA és un protocol de seguretat extra per a RDP que en el moment de realitzar aquest projecte Guacamole encara no podia suportar.

8.2.2 FER FUNCIONAR GUACAMOLE SOBRE WEBSOCKET EN COMPTES DE HTTP

El protocol WebSocket és una tecnologia basada en HTML5 que permet establir connexions molt més àgils que HTTP entre el navegador web i el client. Aquest protocol és especialment indicat per a aplicacions web en temps real com ara videojocs, xats o... escriptoris remots. Guacamole suportava nativament WebSockets i de fet es va arribar a implementar una solució que els feia servir i que en general oferia una experiència d'usuari bastant més àgil. Malauradament la versió del servidor Apache de ITSAT que fa de proxy entre Guacamole i l'exterior no suporta la redirecció de connexions WebSocket, només HTTP. Actualitzant el servidor d'Apache a la seva última versió estable (2.4) es podrien redirigir aquest tipus de connexions i es podria fer servir finalment aquest protocol, que s'assembla més a TCP i és molt més indicat per fer aquest tipus de feina.

8.2.3 EXTENSIÓ PHP DE SAMBA I WINEXE

Es va implementar una extensió de PHP (escrita en C i linkada a les llibreries de C de Samba i WinExe) que estenia el propi llenguatge oferint funcions natives per executar comandes i enviar i rebre fitxers a les màquines gestionades. D'aquesta manera s'hagués disposat de funcions PHP com ara smbli_exec(), smbcli_recv(), smbcli_send(), deixant completament de banda els executables externs. Aquesta extensió es va programar a les etapes finals del projecte i no va donar temps de testejar-la prou a fons per integrar-la amb garantíes.

8.2.4 LOGGING DE SESSIONS INTERACTIVES

Es va parlar de capturar les dades de cada sessió interactiva i per fer "replays" a posteriori, a mode de log visual, però no es va arribar a dissenyar cap solució ni planificar la tasca.

ANNEX A: TECNOLOGIES DESTACADES

A.1 PHP[17]

PHP és un llenguatge de programació interpretat dissenyat inicialment per a generar pàgines web amb contingut dinàmic, però des de la seva primera versió l'any 1995 ha evolucionat molt i avui en dia es podria considerar un llenguatge de programació de propòsit general.

A més a més de poder utilitzar-lo com generador de plantilles HTML (el seu propòsit inicial) també suporta connexions a bases de dades, excepcions, programació funcional i programació orientada a objectes.

Les bases d'ITSAT estan programades en PHP, que suposa un 80% del total de línies de codi totals del projecte. Aquestes parts del codi s'encarreguen de tot tipus de responsabilitats, com ara renderitzar les vistes dinàmiques, autenticar els usuaris de l'aplicació o comunicar-se amb la base de dades.

A.2 SMARTY[18]

Smarty és un sistema de plantilles per a PHP. Encara que inicialment PHP es va dissenyar per dur a terme aquesta tasca, a mida que es van anar ampliant les seves capacitats es van començar a gestar altres llenguatges de plantilles com Smarty que només serveixen per escriure codi HTML dinàmic. ITSAT fa un ús extensiu d'aquesta tecnologia per renderitzar les vistes de la interfície web.

A.3 JAVASCRIPT[19]

Javascript és el llenguatge de programació de frontend per excel·lència. Permet fer alteracions en la pròpia estructura de l'HTML en el propi navegador del client, un cop el contingut del servidor ja ha estat servit. Guacamole fa un ús intensiu d'aquesta tecnologia, especialment per tal de renderitzar el monitor remot a la pàgina web mitjançant l'element `<canvas>`.

A.4 X.509[20]

X.509 és un estàndard criptogràfic que defineix les característiques d'una infraestructura de clau pública (*PKI, Public Key Infrastructure* en anglès) que suporti inici de sessió única (*SSO, Single Sign On*) i Privilegi d'Administració d'Infraestructures (*PMI, Privilege Management Infrastructure*).

Aquest estàndard és un dels puntals bàsics sobre els que es basa ITSAT, ja que una de les seves característiques pròpies és que permet accedir als usuaris a l'aplicació automàticament per mitjà de credencials.

Per tal de fer-ho possible els usuaris d'ITSAT s'han d'instal·lar al navegador un certificat expedit pel propi administrador d'ITSAT. El sistema accepta només als usuaris que presenten un certificat en regla, d'aquesta manera no els cal recordar cap contrasenya, mentre que l'administrador reté un gran control sobre l'accés al sistema.

[17] http://php.net/
[18] http://www.smarty.net/
[19] https://developer.mozilla.org/es/docs/Web/JavaScript
[20] https://www.ietf.org/rfc/rfc2459

ANNEX B: MOSTRES DE CODI

B.1 TELNET.PHP

```
<?php

include_once("iremote.inc");

/**
 * Telnet class
 *
 * This class implements the iRemote interface, thus efectively modelling
 * a connection to a remote machine using the Telnet protocol
 *
 * All methods throw Exceptions on error
 */
class Telnet extends SATobject implements iRemote
{
    const TELNETFTP_ERROR = FALSE;
    const TELNETFTP_OK = TRUE;

    public $host;
    private $port;
    private $timeout;
    private $socket = NULL;
    private $buffer = NULL;
    private $prompt;
    private $errno;
    private $errstr;
    private $curcmd; //needed to parse output and delete it
    private $NULL;
    private $DC1;
    private $WILL;
    private $WONT;
    private $DO;
    private $DONT;
    private $IAC;
    private $NAWS;
    private $global_buffer = '';

    /**
     * The $capabilities attribute lists the capabilities of the protocol
     *
     * @access private
     */
    private $capabilities = Array(CAN_CONNECT);
    private $env_vars = Array();

    //ftp
    public $cid;
    private $user;
    private $passwd;
    public $output;
    private $mode;

    /**
     * Constructor. Initialises host, port and timeout parameters
     * defaults to localhost port 23 (standard telnet port)
     *
     * @param string $host Host name or IP addres
     * @param int $port TCP port number
```

```
     * @param int $timeout Connection timeout in seconds
     * @return void
     */
    public function __construct($host = '127.0.0.1', $port = '23', $timeout = 10,
$mode)
    {

        parent::__construct();
        parent::setLogger("remote");

        $this->host = $host;
        $this->port = $port;
        $this->timeout = $timeout;
        $this->cid = NULL;

        // set some telnet special characters
        $this->NULL = chr(0);
        $this->DC1 = chr(17);
        $this->WILL = chr(251);
        $this->WONT = chr(252);
        $this->DO = chr(253);
        $this->DONT = chr(254);
        $this->IAC = chr(255);

        $this->SE = chr(240); //SubObtionEnd
        $this->SB = chr(250); //SubOptionsInit

        $this->NAWS = chr(31); // Negotiate About Window Size:
https://tools.ietf.org/html/rfc1073
        $this->ECHO = chr(1); //Echo http://www.faqs.org/rfcs/rfc857.html

        $this->mode = $mode;
        $this->open($mode);
        $this->telnetCmdWrite($this->WONT . $this->ECHO);
    }

    private function sendWindowSize($width, $height)
    {
        //width 255
        //height 40
        $this->telnetCmdWrite($this->SB . $this->NAWS . chr(0) . chr($width) .
chr(0) . chr($height) . $this->IAC . $this->SE);
    }

    /**
     * Destructor. Cleans up socket connection and command buffer
     *
     * @return void
     */
    public function __destruct()
    {

        // cleanup resources
        $this->close();
        $this->buffer = NULL;
        $this->global_buffer = NULL;
    }

    /**
     * Attempts Telnet connection to remote host. Returns TRUE if sucessful.
     *
     * @return boolean
     */
```

```
    private function t_open()
    {
        // check if we need to convert host to IP
        if (!preg_match('/([0-9]{1,3}\\.){3,3}[0-9]{1,3}/', $this->host)) {

            $ip = gethostbyname($this->host);

            if ($this->host == $ip) {

                throw new Exception("Cannot resolve $this->host");
            } else {
                $this->host = $ip;
            }
        }

        // attempt connection
        $this->socket = fsockopen($this->host, $this->port, $this->errno, $this-
>errstr, $this->timeout);


        if (!$this->socket) {
            $this->error("Cannot connect to $this->host on port $this->port");
            return self::TELNETFTP_ERROR;
            //throw new Exception("Cannot connect to $this->host on port $this-
>port");
        }

        return self::TELNETFTP_OK;
    }

    /**
     * Attempts FTP connection to remote host. Returns TRUE if sucessful.
     *
     * @return boolean
     */
    private function f_open()
    {
        $this->info("TELNETFTP: opening ftp conection");
        $this->cid = ftp_connect($this->host, 21);
        if ($this->cid) {
            $this->info("TELNETFTP: opened OK");
        } else {
            $this->warn("TELNETFTP: ftp conection Failed!!");
        }
        return $this->cid;
    }

    public function open($mode = 0)
    {
        if ($mode == ROPEN_MODE_SHELL) {
            if ($this->t_open())
                return self::TELNETFTP_OK;
        } else {
            if ($this->t_open() & $this->f_open())
                return self::TELNETFTP_OK;
        }
    }

    /**
     * Closes Telnet (IP socket)
     *
     * @return boolean
     */
```

```
    public function t_close()
    {
        if ($this->socket) {
            if (!fclose($this->socket)) {
                throw new Exception("Error while closing telnet socket");
            }
            $this->socket = NULL;
        }
        return self::TELNETFTP_OK;
    }

    /**
     * Closes ftp
     *
     * @return boolean
     */
    public function f_close()
    {
        if ($this->cid) {
            ftp_close($this->cid);
        }
        return self::TELNETFTP_OK;
    }

    /**
     * Closes ALL
     *
     * @return boolean
     */
    public function close()
    {
        $this->t_close();
        $this->f_close();
        return self::TELNETFTP_OK;
    }

    /**
     * Executes command and returns a string with result.
     * This method is a wrapper for lower level private methods
     *
     * @param string $command Command to execute
     * @return string Command result
     */
    public function exec($cmd)
    {
        $this->curcmd = $cmd;
        if ($this->mode == ROPEN_MODE_CMD) {
            $this->info("TELNETFTP:executing: $cmd");
            $this->write($cmd);
            $this->waitPrompt();
            return $this->getBuffer();
        } else {

            if (!fwrite($this->socket, $cmd) < 0) {
                throw new Exception("Error writing to socket");
            }
            $this->debug("TELNETFTP: writted:($cmd) to telnet output");
            usleep(SHELL_CMD_TIMEOUT);
            $output2 = " ";
            while (strlen($output2) > 0) {
                $this->debug("TERLNETFTP: output: $output");
                $output2 = stream_get_contents($this->socket);
                $output1 .= $output2;
```

```
                echo $output1;
            }
            $this->output = $output1;

            $this->debug("TELNETFTP: stream get $this->output");
            return $this->output;
        }
    }

    /**
     * Attempts login to remote host.
     * This method is a wrapper for lower level private methods and should be
     * modified to reflect telnet implementation details like login/password
     * and line prompts. Defaults to standard unix non-root prompts
     *
     * @param string $md Credential mode.
     * @param string $username Username
     * @param string $password Password
     * @return boolean
     */
    public function login($md, $username, $password)
    {
        $retval = TRUE;

        $this->info("TELNETFTP:login: $username:xxxxxxx");
        if ($md != "USER/PASS") {
            $this->error("trying to login to FTP with no user/password credential
($md)");
            return;
        }

        try {
            $this->setPrompt('login:');
            $this->waitPrompt();
            $this->write($username);
            $this->setPrompt('Password:');
            $this->waitPrompt();
            $this->write($password);
            //\n$ is necessary if we want use variables in our commands
            if ($username == "root") {
                $this->setPrompt('# ');
            } else {
                $this->setPrompt('$ ');
            }
            $this->waitPrompt();
        } catch (Exception $e) {

            //throw new Exception("Login failed.");
            $this->error("TELNETFTP: Telnet Login Failed");
            $retval = FALSE;
        }

        $this->user = $username;
        $this->passwd = $password;

        $this->debug("Telnet: result login $retval");
        if ($this->mode != ROPEN_MODE_SHELL) {
            if ($this->cid)
                $ftp = ftp_login($this->cid, $username, $password);
            if (!$ftp) {
                $this->warn("TELNETFTP: error on FTP login");
                $retval = FALSE;
            }
```

```
        }
        if ($this->mode == ROPEN_MODE_SHELL) {
            stream_set_blocking($this->socket, false);
            stream_set_timeout($this->socket, 5, 0);
            $this->output = stream_get_contents($this->socket);
            echo $this->global_buffer;
            echo $this->output;
        }

        // TODO: No assumir que el SO de sota 鳜 Linux!!!
        $this->env_vars['TEMP'] = '/tmp';
        $this->env_vars['HOME'] = $this->exec('echo $HOME');

        return $retval;
    }

    /**
     * Sets the string of characters to respond to.
     * This should be set to the last character of the command line prompt
     *
     * @param string $s String to respond to
     * @return boolean
     */
    public function setPrompt($s = '$')
    {
        $this->prompt = $s;
        return self::TELNETFTP_OK;
    }

    /**
     * Gets character from the socket
     *
     * @return void
     */
    protected function getc()
    {
        $c = fread($this->socket, 1);
        $this->debug("TELNETFTP: int:" . ord($c) . ":char:" . $c);
        $this->global_buffer .= $c;
        return $c;
    }

    /**
     * Clears internal command buffer
     *
     * @return void
     */
    protected function clearBuffer()
    {
        $this->buffer = '';
    }

    /**
     * Reads characters from the socket and adds them to command buffer.
     * Handles telnet control characters. Stops when prompt is ecountered.
     *
     * @param string $prompt
     * @return boolean
     */
    protected function readTo($prompt)
    {

        if (!$this->socket) {
```

```
            throw new Exception("Telnet connection closed");
        }

        $d = NULL;

        $this->debug("READTO: " . $prompt);

        // clear the buffer
        $this->clearBuffer();

        $until_t = time() + $this->timeout;
        do {
            // time's up (loop can be exited at end or through continue!)
            if (time() > $until_t) {
                throw new Exception("Couldn't find the requested : '$prompt'
within {$this->timeout} seconds");
            }

            $c = $this->getc();

            if ($c === false) {
                throw new Exception("Couldn't find the requested : '" . $prompt .
"', it was not in the data returned from server: " . $this->buffer);
            }
            // Interpreted As Command
            if ($c == $this->IAC) {
                if ($this->negotiateTelnetOptions()) {
                    continue;
                }
            }

            // append current char to global buffer
            $this->buffer .= $c;
            //$this->debug("TELNETFTP: int:".ord($c).":char:".$c);
            // we've encountered the prompt. Break out of the loop
            if ((substr($this->buffer, strlen($this->buffer) - strlen($prompt)))
== $prompt) {
                // Si el prompt 鱳"# " es podria confondre amb un caracter normal
(comentari bash, etc.)
                if ($prompt == "# ") {
                    stream_set_timeout($this->socket, 0, 200000);
                    $d = $this->getc();
                    $smd = stream_get_meta_data($this->socket);
                    stream_set_timeout($this->socket, 5);
                    $this->debug("telnetFTP->readTo(): HI HA HAGUT TIMEOUT? " .
$smd['timed_out']);
                    if ($smd['timed_out']) {
                        // Realment era un prompt
                        return self::TELNETFTP_OK;
                    } else {
                        // Era un caracter normal i s'ha de processar $d com
qualsevol altre
                        if ($d === false) {
                            throw new Exception("Couldn't find the requested : '"
. $prompt . "', it was not in the data returned from server: " . $this->buffer);
                        }
                        if ($d == $this->IAC) {
                            if ($this->negotiateTelnetOptions()) {
                                continue;
                            }
                        }
                        $this->buffer .= $d;
                    }
```

```
                } else {
                    return self::TELNETFTP_OK;
                }
            }
        } while ($c != $this->NULL || $c != $this->DC1);
    }

    protected function telnetCmdWrite($cmd)
    {
        fwrite($this->socket, $this->IAC . $cmd);
    }

    /**
     * Write command to a socket
     *
     * @param string $buffer Stuff to write to socket
     * @param boolean $addNewLine Default true, adds newline to the command
     * @return boolean
     */
    protected function write($buffer, $addNewLine = true)
    {

        if (!$this->socket) {
            throw new Exception("Telnet connection closed");
        }

        // clear buffer from last command
        $this->clearBuffer();

        if ($addNewLine == true) {
            $buffer .= "\n";
        }

        $this->global_buffer .= $buffer;
        if (!fwrite($this->socket, $buffer) < 0) {
            throw new Exception("Error writing to socket");
        }
        $this->debug("TELNETFTP: writted:($buffer) to telnet output");
        return self::TELNETFTP_OK;
    }

    /**
     * Returns the content of the command buffer
     *
     * @return string Content of the command buffer
     */
    protected function getBuffer()
    {
        // cut last line (is always prompt)
        if ($this->mode == ROPEN_MODE_SHELL) {
            $this->debug("TELNETFTP: getbuffer mode SHELL");
            return $this->buffer;
        } else {
            $this->debug("TELNETFTP: getbuffer mode CMD from: $this->buffer");
            $buf = explode("\n", $this->buffer);
            unset($buf[count($buf) - 1]);
            //cut first line if equal to the command sended( we expect always to
get full sended command but sometimes isn't)
            $t_cmd = trim($buf[0]);
            $this->debug("TELNETFTP: comparing first line [$t_cmd/$this-
>curcmd]");
            if (!strcmp($t_cmd, $this->curcmd)) {
                $this->debug("TELNETFTP: unset first line ");
```

```
                unset($buf[0]);
            }
            $buf = implode("\n", $buf);
            return trim($buf) . "\n";
        }
    }

    protected function sendDefaultResponse($c, $opt)
    {
        if (($c == $this->DO) || ($c == $this->DONT)) {
            $this->info("TELNETFTP: sending WONT: $opt ");
            $this->telnetCmdWrite($this->WONT . $opt);
        } else if (($c == $this->WILL) || ($c == $this->WONT)) {
            $this->info("TELNETFTP: sending DONT: $opt ");

            $this->telnetCmdWrite($this->DONT . $opt);
        } else {
            $this->warning('Error: unknown control character ' . ord($c));
            throw new Exception('Error: unknown control character ' . ord($c));
        }
    }

    /**
     * Telnet control character magic
     *
     * @param string $command Character to check
     * @return boolean
     */
    protected function negotiateTelnetOptions()
    {


        $c = $this->getc();
        if ($c != $this->IAC) {
            $opt = $this->getc();
            $this->debug("TELNETFTP:Received control command: " . ord($c) . "
option:" . ord($opt));

            switch (ord($opt)) {

                case 0: //Transmit Binary
                    $this->debug("TELNETFTP:Received control command: " . ord($c)
. " option: Transmit Binary");
                    $this->sendDefaultResponse($c, $opt);
                    break;
                case 1: //Echo
                    $this->debug("TELNETFTP:Received control command: " . ord($c)
. " option: Echo");
                    $this->sendDefaultResponse($c, $opt);
                    break;
                case 3: //Suppress Go Ahead
                    $this->debug("TELNETFTP:Received control command: " . ord($c)
. " option: Suppress Go Ahead");
                    $this->sendDefaultResponse($c, $opt);
                    break;
                case 5: //Status
                    $this->debug("TELNETFTP:Received control command: " . ord($c)
. " option: Status");
                    $this->sendDefaultResponse($c, $opt);
                    break;
                case 6: //Timing Mark
                    $this->debug("TELNETFTP:Received control command: " . ord($c)
. " option: Timing Mark");
```

```
                    $this->sendDefaultResponse($c, $opt);
                    break;
                case 24: //Terminal Type
                    $this->debug("TELNETFTP:Received control command: " . ord($c)
. " option: Terminal Type");
                    $this->sendDefaultResponse($c, $opt);
                    break;
                case 31: //Window Size
                    $this->debug("TELNETFTP:Received control command: " . ord($c)
. " option: Window Size");
                    //$this->sendDefaultResponse($c,$opt);
                    if ($c == $this->DO || $c == $this->DONT) {
                        $this->debug("TELNETFTP: Sendind  option: Window Size");
                        $this->telnetCmdWrite($this->WILL . $this->NAWS);
                        $this->sendWindowSize(220, 24);
                    } else {
                        $this->debug("TELNETFTP: Sendind  Default response $c:
Window Size");
                        $this->sendDefaultResponse($c, $opt);
                    }
                    break;
                case 32: //Terminal Speed
                    $this->debug("TELNETFTP:Received control command: " . ord($c)
. " option: Terminal Speed");
                    $this->sendDefaultResponse($c, $opt);
                    break;
                case 33: //Remote Flow Control
                    $this->debug("TELNETFTP:Received control command: " . ord($c)
. " option: Remote Flow Control");
                    $this->sendDefaultResponse($c, $opt);
                    break;
                case 34: //Linemode
                    $this->debug("TELNETFTP:Received control command: " . ord($c)
. " option: Linemode");
                    $this->sendDefaultResponse($c, $opt);
                    break;
                case 35: //X Display Location
                    $this->debug("TELNETFTP:Received control command: " . ord($c)
. " option: X Display Location");
                    $this->sendDefaultResponse($c, $opt);
                    break;
                case 39: //New Environment
                    $this->debug("TELNETFTP:Received control command: " . ord($c)
. " option: New Environment");
                    $this->sendDefaultResponse($c, $opt);
                    break;
                default:
                    $this->sendDefaultResponse($c, $opt);
                    break;
            }
        } else {
            $this->warn('Error: Something Wicked Happened');
            throw new Exception('Error: Something Wicked Happened');
        }
        $this->global_buffer = "";
        return self::TELNETFTP_OK;
    }

    /**
     * Reads socket until prompt is encountered
     */
    protected function waitPrompt()
    {
```

```
        return $this->readTo($this->prompt);
    }

    public function send($lpath, $rpath)
    {
        $this->error("send() error: This machine doesn't support file
transfers");
    }

    public function recv($rpath, $lpath)
    {
        $this->error("recv() error: This machine doesn't support file
transfers");
    }

    public function cinfo()
    {

    }

    public function ping()
    {
        $this->exec('echo');
    }

    /**
     * Telnet's implementation of the checkCapability method
     */
    public function checkCapability($capability)
    {
        return in_array($capability, $this->capabilities);
    }

    public function get_env_variable($var)
    {
        return $this->env_vars[$var];
    }
}
```

B.2 SMBCIFS.PHP

```
<?php

include_once("iremote.inc");

/**
 * SMBCIFS class
 *
 * This class implements the iRemote interface, thus effectively modelling
 * a connection to a remote machine using the SMB protocol
 *
 * All methods throw Exceptions on error
 */
class SMBCIFS extends SATobject implements iRemote {

    public $host;
    public $workgrp;
    public $drive;
    public $cid;
    public $output;
    private $user;
```

```
    private $passwd;
    private $env_vars = Array();

    /**
     * The $capabilities attribute lists the capabilities of the protocol
     *
     * @access private
     */
    private $capabilities = Array(CAN_CONNECT, CAN_TRANSFER);

    public function __construct($host) {

        parent::__construct();
        parent::setLogger("remote");

        $this->host = $host;
        $this->cid = null;
    }

    public function __destruct() {
        $this->close();
    }

    protected function parse_rpath($rpath) {
        if ($rpath[1] == ":" && $rpath[2] == "\\") {
            $this->drive = strtolower($rpath[0]);
            $rpath = substr($rpath, 3);
        } else {
            $this->drive = 'c';
        }
        $r = str_replace("\\", "\\\\", $rpath);
        return $r;
    }

    protected function parse_lpath($lpath) {
        $pos = strrpos($lpath, "/");
        $dir = substr($lpath, 0, $pos);
        return array(dirname($lpath), basename($lpath));
    }

    protected function parse_cmd_output($output) {
        $strout = "";
        if ($output[0] == "")
            array_slice($output);
        foreach ($output as $line)
            $strout .= $line . "\n";
        return $strout;
    }

    protected function init_env_vars() {
        $allVars = $this->exec('set');

        // TEMP
        $partial = substr($allVars, strpos($allVars, 'TEMP'));
        $this->env_vars['TEMP'] = substr($partial, strpos($partial, '=') + 1,
strpos($partial, "\n") - strpos($partial, '=') - 1);

        // HOME
        $partial = substr($allVars, strpos($allVars, 'PUBLIC'));
        $public = substr($partial, strpos($partial, '=') + 1, strpos($partial,
"\n") - strpos($partial, '=') - 1);
        $this->env_vars['HOME'] = substr($public, 0, strrpos($public, '\\') + 1);
        $this->env_vars['HOME'] .= $this->user;
```

```
    }

    protected function init_temp_dir() {
        $allVars = $this->exec('set');
    }

    public function open($mode) {
    }

    public function close() {

    }

    public function exec($cmd) {
        if (!isset($this->user) || !isset($this->passwd)) {
            $this->error("smbcifs:exec() error: No USER/PASS provided!");
            return;
        }
        $command = "winexe -U $this->workgrp/$this->user%$this->passwd //$this-
>host 'cmd /c $cmd'";
        $output = Array();
        exec($command, $output);
        return $this->parse_cmd_output($output);
    }

    public function login($md, $username, $password) {
        if ($md != "USER/PASS") {
            $this->error("SMBCIFS requires USER/PASS credential");
            return false;
        }
        $auth = explode("\\", $username);
        if (count($auth) == 2) {
            $this->workgrp = $auth[0];
            $this->user = $auth[1];
        } else {
            $this->workgrp = 'WORKGROUP';
            $this->user = $username;
        }
        $this->passwd = $password;

        $this->init_env_vars();
        return true;
    }

    public function send($lpath, $rpath) {
        if (!isset($this->user) || !isset($this->passwd)) {
            $this->error("smbcifs:exec() error: No USER/PASS provided!");
            return;
        }

        if ($rpath == '$HOME') {
            $rpath = $this->env_vars['HOME'] . "\\" . basename($lpath);
        }

        $rpath = $this->parse_rpath($rpath);
        $rfile = explode('\\', $rpath);
        $rfile = $rfile[count($rfile)-1];
        $rpath = substr($rpath, 0, strrpos($rpath, '\\') - 1);
        $lfile = basename($lpath);
        $ldir = dirname($lpath);

        // Save current working directory
        $current_cwd = getcwd();
```

```
        // Move to the local file directory
        chdir($ldir);

        $command = "smbclient '\''\'$this->host'\'$this->drive$ --
workgroup=$this->workgrp --user=$this->user $this->passwd -c 'cd $rpath ; put
$lfile $rfile'";
        $output = Array();
        exec($command, $output);

        chdir($current_cwd);
        return count($output) == 0;
    }

    public function recv($rpath, $lpath) {
        if (!isset($this->user) || !isset($this->passwd)) {
            $this->error("smbcifs:exec() error: No USER/PASS provided!");
            return;
        }

        $rpath = $this->parse_rpath($rpath);
        list($lplace, $lname) = $this->parse_lpath($lpath);

        set_euid_userid();
        // Save current working directory
        $current_cwd = getcwd();

        // Move to destination directory
        chdir($lplace);
        //print "current dir " . getchwd() . "\n";
        // Get the file from the retome location
        $command = "smbclient '\''\'$this->host'\'$this->drive$ --
workgroup=$this->workgrp --user=$this->user $this->passwd -c 'get $rpath
$lname'";
        $output = Array();

        exec($command, $output);

        chdir($current_cwd);
        unset_euid_userid();

        return count($output) == 0;
    }

    public function cinfo() {

    }

    public function ping() {
        $this->exec('echo');
    }

    /**
     * SMBCIFS' implementation of the checkCapability method
     */
    public function checkCapability($capability) {
        return in_array($capability, $this->capabilities);
    }

    public function get_env_variable($var) {
        return $this->env_vars[$var];
    }
}
```

B.3 GUACAMOLE-SERVICE.PHP

```
<?php

require_once('setup-services.inc');
require_once('satacctrl.inc');
require_once('rcpm-cid.inc');
require_once('satmachines.inc');
require_once('satauthcache.inc');

$allowed_actions = Array('connect', 'close', 'forced_close');
$cid = $_GET['cid'];
$action = $_GET['action'];
$cn = $_GET['cn'];

$log->setLogger("remote");
$log->info("GUACAMOLE-SERVICE CONNECT: init");

if ($cid == null || $action == null || !in_array($action, $allowed_actions)) {
    echo json_encode(Array("code" => "ERROR", "msg" => "Invalid call"));
} else {

    $cid_obj = new SSHCID();

    if ($action == 'connect') {

        list($mode, $exp_userid, $namedest, $ipdest, $xport, $os) = $cid_obj-
>getCIDInfo($cid);
        $log->debug("GUACAMOLE-SERVICE:
$mode,$exp_userid,$namedest,$ipdest,$xport,$os");

        if (!isset($userid) || empty($userid) || $userid == "root") {
            $userid = $exp_userid;
            $ac = new SATAcCtrl($userid);
            $ac->setLogger("console");
            $log->debug("RCPSH: setting access-control vars for user: $userid");
        }

        $date = date("Ymd");
        $logfilename = $userid . "-" . rand() . "-" . $date . ".log";

        // check for certificate tampering
        if ($cn != $userid) {
            echo json_encode(Array("code" => "ERROR", "msg" => "Certificate
Common Name does not match this connection's granted user"));
            die();
        }

        //check  user
        $obj = new DB_SATusers();
        $obj->query("SELECT * From SATusers where userid='$userid' and
conectable=1 and active=1 and locked=0");
        if ($obj->N != 1) {

            $log->error("ERROR: usuario $userid no exite en la BBDD de usuarios o
no es un usuario conectable, o no esta activo o esta bloqueado");
            $cid_obj->setStatus($cid, "ABORTED", "");
            echo json_encode(Array("code" => "ERROR", "msg" => "Invalid user"));
```

```
            die();
        }

        //Access control
        $_access = $ac->acCheck("rcpm", "connect", $namedest);
        $log->debug("RCPSH CheckAccess: $namedest: $_access");
        if (!$_access) {
            $log->warn("RCPSH: userid $userid have no granted access to  $ipdest
/ $namedest");
            $cid_obj->setStatus($cid, "ABORTED", "");
            echo json_encode(Array("code" => "ERROR", "msg" => "Forbidden
Access"));
            die();
        }

        // $action->rlogshell($ipdest,$logfilename);
        $mCtrl = new SATmachines();
        $admapp = $mCtrl->getAdmApp($ipdest);
        $ip = $mCtrl->getIPfromAny($ipdest);

        if ($mCtrl->isBlocked($ip)) {
            $this->error("ERROR: la machina $namedest esta bloqueada");
            echo json_encode(Array("code" => "ERROR", "msg" => "Machine
blocked"));
            die();
        }

        $auth = new SATAuthCache();
        list($md, $user, $raw_pass) = $auth->getUserPass($ip);
        if (empty($md) || empty($user) || empty($raw_pass)) {
            $this->userError("Device $machine has not available credentials\n");
            echo json_encode(Array("code" => "ERROR", "msg" => "No
credentials"));
            die();
        } else {
            $user = explode("\\", $user);
            $user = (count($user) == 2)? $user[1] : $user[0];
        }


        //Fix for strange characters on passwords
        $pass = escapeshellcmd($raw_pass);

        $cid_obj->setStatus($cid, "ACTIVE", $logfilename);

        echo json_encode(Array("code" => "OK", "ip" => $ipdest, "user" => $user,
"pass" => $pass));
    } else if ($action == 'close') {
        $cid_obj->setStatus($cid, "FINISHED", "");
        echo json_encode(Array("code" => "OK"));
    } else if ($action == 'forced_close') {
        $sessionID = $cid;
        // 1. Comprovar que qui ha cridat l'script te permis per parar una
conexio [SOLUCIO PROVISIONAL]
        $ac = new SATAcCtrl($cn);
        if (!$ac->acCheck("rcpm", "closesession")) {
            $log->error("ERROR: El usuario no tiene permiso para cerrar
sesiones");
            echo json_encode(Array("code" => "ERROR", "msg" => "Unprivileged
user"));
            die();
        }
```

```
        // 2. A partir del sessionID, aconseguir el cid de la conexio que volem
chapar
        $ses= new DB_SATrcses();
        $ses->query("SELECT cid from SATrcses where rcsesid='$sessionID'");
        if ($ses->N != 1) {
            // Error, no existeix cap entrada amb aquell sessionID. Aixo no
hauria de passar mai pero be...
            $log->error("ERROR: SATrcses no contiene la conexion $sessionID");
            echo json_encode(Array("code" => "ERROR", "msg" => "Invalid
sessionID"));
            die();
        }
        $ses->fetch();
        $cid = $ses->cid;
        // 3. setejar el cid a FINISHED
        $cid_obj->setStatus($cid, "FINISHED", "");
        // 4. Retornar cid
        echo json_encode(Array("code" => "OK", "cid" => $cid));
    }
}
```

B.4 ITSATDESKTOPHTTPTUNNELSERVLET.JAVA

```
package net.sourceforge.guacamole.net.basic;

import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.InputStreamReader;
import java.security.KeyStore;
import java.security.cert.X509Certificate;
import javax.net.ssl.HostnameVerifier;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import javax.servlet.ServletException;

import java.util.Enumeration;

public class ITSATDesktopHTTPTunnelServlet extends GuacamoleHTTPTunnelServlet {

    private Logger logger =
LoggerFactory.getLogger(ITSATDesktopWebSocketTunnelServlet.class);
    private KeyStore keystore;
    private KeyStore truststore;
    private DefaultHttpClient httpclient;

    protected JSONObject getITSATServiceRequest(String UID, String cid, String
action) throws GuacamoleException {
        JSONObject credentials = null;
        try {
            HttpGet get = new HttpGet("https://localhost/rcpm/guacamole-
service.php?action=" + action + "&cid=" + cid + "&cn=" + UID);
            HttpResponse response = httpclient.execute(get);
            HttpEntity entity = response.getEntity();
            System.out.println("----------------------------------------");
            System.out.println(response.getStatusLine());
            if (entity != null) {
                System.out.println("Response content length: " +
entity.getContentLength());
            }
```

```
            BufferedReader rd = new BufferedReader(new
InputStreamReader(response.getEntity().getContent()));
            StringBuilder sb = new StringBuilder();
            String s;
            while ((s = rd.readLine()) != null) {
                sb.append(s);
            }
            EntityUtils.consume(entity);
            s = sb.toString();
            credentials = new JSONObject(s);
        } catch (Exception e) {
            System.out.println("Error:" + e.getMessage());
            e.printStackTrace();
        }
        return credentials;
    }

    @Override
    protected GuacamoleTunnel doConnect(HttpServletRequest request) throws
GuacamoleException {

        String ip = null;
        String user = null;
        String pass = null;
        boolean login_successful = false;
        System.out.println("GUACAMOLE doConnect");

        // Get ID of connection
        final String cid = request.getParameter("cid");
        final String UID = getUID(request);

        try {
            System.out.println("Request from ITSAT");

            JSONObject credentials = getITSATServiceRequest(UID, cid, "connect");
            if (credentials.get("code").equals("OK")) {
                ip = (String) credentials.get("ip");
                user = (String) credentials.get("user");
                pass = (String) credentials.get("pass");
                login_successful = true;
            } else {
                logger.info("guacamole-service.php ERROR: {}",
credentials.get("msg"));
                throw new Exception((String) credentials.get("msg"));
            }


        } catch (Exception e) {
            System.out.println("Error:" + e.getMessage());
            e.printStackTrace();
        } finally {
            httpclient.getConnectionManager().shutdown();
        }

        if (!login_successful) {
            logger.info("Guacamole login aborted.");
            return null;
        }


        HttpSession httpSession = request.getSession(true);
        httpSession.putValue("cid", cid);
```

```
        GuacamoleConfiguration config = new GuacamoleConfiguration();
        config.setProtocol("rdp");
        config.setParameter("hostname", ip);
        config.setParameter("username", user);
        config.setParameter("password", pass);

        String width = request.getParameter("width");
        if (width != null && !width.isEmpty()) {
            config.setParameter("width", width);
        }
        String height = request.getParameter("height");
        if (height != null && !height.isEmpty()) {
            config.setParameter("height", height);
        }
        String colorDepth = request.getParameter("colorDepth");
        if (colorDepth != null && !colorDepth.isEmpty()) {
            config.setParameter("color-depth", colorDepth);
        }

        logger.info("Successful connection from {} to \"{}\".",
request.getRemoteAddr(), cid);

        GuacamoleSocket socket = new ConfiguredGuacamoleSocket(
                new InetGuacamoleSocket("localhost", 4822),
                config);

        // Associate socket with tunnel
        GuacamoleTunnel tunnel = new GuacamoleTunnel(socket) {

            @Override
            public void close() throws GuacamoleException {
                try {
                    JSONObject ret = getITSATServiceRequest(UID, cid, "close");
                    if (ret.get("code").equals("OK")) {
                        logger.info("closing session CID: \"{}\".", cid);
                    } else {
                        logger.info("guacamole-service.php ERROR: {}",
ret.get("msg"));
                        throw new Exception((String) ret.get("msg"));
                    }
                } catch (Exception e) {
                    System.out.println("Error:" + e.getMessage());
                    e.printStackTrace();
                } finally {
                    super.close();
                }
            }
        };

        // Attach tunnel to session and servlet context
        this.getServletConfig().getServletContext().setAttribute(cid, tunnel);
        GuacamoleSession session = new GuacamoleSession(httpSession);
        session.attachTunnel(tunnel);

        return tunnel;
    }
}
```