Infraestructura de Clau Pública (PKI) per a la gestió d'historials mèdics.

Daniel Lerch
Enginyeria en Informàtica

Consultor: Jordi Castellà Roca
Gener 2008

Resum

L'objectiu d'aquest projecte, emmarcat en l'àrea de la Seguretat Informàtica, és implementar un esquema criptogràfic, que garanteixi els requisits de seguretat d'un sistema de gestió d'historials mèdics, a través d'una xarxa de comunicacions.
Per aconseguir aquests objectius necessitarem garantir els següents requisits:

Autenticitat: S'ha de poder provar l'autenticitat de la informació del sistema.
Confidencialitat: S'ha de garantir la confidencialitat dels historials mèdics dels pacients.
Integritat: S'ha de garantir en tot moment la integritat de les dades del sistema.
No-Repudi: Si un usuari realitza una acció no ha de poder negar que l'ha realitzada.

Ho farem mitjançant una PKI implementada a partir de la API criptogràfica OpenSSL.

El projecte es desenvolupa íntegrament amb eines OpenSource, cada una de les quals ens permetrà assolir certs objectius:

OpenSSL:

OpenSSL és una eina per a criptografia que permet crear una CA, fer servir el protocol SSL, xifrar mitjançant els protocols més coneguts o utilitzar algoritmes de hash, entre d'altres. Però OpenSSL és al mateix temps una llibreria que permet utilitzar totes aquestes característiques des del llenguatge de programació C.
En el projecte s'utilitza aquesta llibreria per a implementar la PKI.

libxml:

La llibreria libxml ens permet manipular XML des del llenguatge de C d'una manera senzilla. S'utilitza en el projecte per a manipular els documents XML que es fan servir per a emmagatzemar y transmetre les dades del sistema.

Apache/gSoap:

Els aplicatius es comuniquen amb el repositori mitjançant WebServices. Aquests s'implementen mitjançant la llibreria gSoap i corren sobre el servidor web Apache.

MySQL:

Les dades queden emmagatzemades en el sistema gestor de bases de dades MySQL.

GTK+:

GTK+ es la llibreria gràfica base de l'escritori GNOME utilitzat àmpliament en sistemes operatius Open Source. S'utilitza en el projecte per a desenvolupar la interfície gràfica de les aplicacions.

Paraules clau:

- PKI
- Autenticitat
- Integritat
- Confidencialitat
- No-repudi
- Clau Pública
- Clau Privada
- Seguretat
- Criptografia
- WebServices
- OpenSSL
- MySQL
- GTK
- XML
- libxml
- Linux
- historials mèdics
- GNU
- gSoap.

Índex de continguts

Índex de figures

Capítol 1: Introducció

1.1.- Justificació del PFC i context en el qual es desenvolupa

És habitual, en els temps que corren, que tot tipus de servei hagi vist la seva rèplica a Internet. Hem passat del correu tradicional al correu electrònic, del joc al casino al joc online, fem la declaració de la renta per Internet i fins i tot utilitzem la Xarxa per a estudiar una carrera. No hi ha dubte de que les comunicacions en xarxa han donat lloc a tot un nou ventall de possibilitats.

Moltes d'aquestes operacions en xarxa necessiten garantir certs requisits com la cofidencialitat, l'autenticitat, la integritat de les dades i el no-repudi (o impossibilitat de que un usuari negui haver fet una operació). Les matemàtiques ofereixen tota una sèrie de tècniques y protocols que permeten assolir aquests objectius: els anomenem Criptografia.

La gestió d'historials mèdics des de la Xarxa és un nou servei que permetrà als pacients poder consultar el seu expedient mèdic des de casa, i als metges fer consultes inserir noves visites. Això obrirà noves portes a la consultoria mèdica com per exemple, un millorat servei a domicili o la possibilitat d'atendre pacients des de qualsevol ubicació, disposant únicament d'una connexió a Internet.

Però la implementació d'un servei d'aquestes característiques requereix unes fortes mesures de seguretat, doncs no hem d'oblidar que les dades de caràcter mèdic són altament confidencials, com indica la Llei Orgànica de Protecció de Dades (LOPD). Caldrà garantir doncs, que els historials mèdics només són modificats pel personal qualificat i que les dades només són accedides pel el personal mèdic o el propi pacient.

En aquest projecte es farà ús d'una Infraestructura de Clau Pública (PKI) per assolir els requisits esmentats.

1.2.- Objectius del PFC

L'objectiu principal d'aquest projecte és implementar un esquema criptogràfic, que garanteixi els requisits de seguretat d'un sistema de gestió d'historials mèdics, a través d'una xarxa de comunicacions. Aquests requisits són la **confidencialitat** de les dades de l'historial mèdic, la seva **autenticitat**, la seva **integritat** i el **no-repudi** en la inserció de noves visites. Per tant, l'objectiu principal del projecte es pot resumir com la implementació d'una Infraestructura de Clau Pública (PKI) per a la gestió d'historials mèdics.

Com a resultat final de la implementació hem d' obtenir un conjunt d'aplicacions que ens permetin treballar amb el sistema de manera remota. Així doncs, podem desglossar el projecte en els següents objectius:

Objectiu 1: PKI per a la gestió d'historials mèdics

El primer objectiu es el desenvolupament d'una PKI o Infraestructura de Clau Pública que ens permeti complir els requisits de confidencialitat, autenticitat, integritat i no-repudi esmentats anteriorment. Aquesta ha de poder ser utilitzada pels diferents actors del sistema: Gestor, Pacient i Metge.

Objectiu 2: Gestor del sistema com a Servidor

Com és lògic, en una aplicació d'aquest tipus, els clients (Pacient i Metge) no estaran a la mateixa màquina que el servidor (Gestor), pel que s'haurà de desenvolupar el projecte en base a una arquitectura client/servidor. Els clients faran peticions al Gestor que, per tant, agafa el paper de servidor. Així, necessitarem implementar el Gestor del sistema com a Servidor. Ho farem mitjançant serveis web.

Objectiu 3: Repositori central d'historials mèdics

El sistema necessita d'un repositori central on es guardaran els historials mèdic i certes dades de Pacients i Metges de forma segura. Al repositori només hi podrà accedir el Gestor del sistema, en cap cas ho faran Pacients i Metges.
Implementarem el repositori mitjançant un Sistema Gestor de Bases de Dades.

Objectiu 4: Accés remot al Servidor.

Del punt anterior es desprèn que Pacient i Metge hauran d'accedir de forma remota al servidor. Donat que treballem amb serveis web, tots dos hauran d'implementar la part de client del servei web per a poder accedir a les funcionalitats del projecte.
A més haurem de dissenyar un format basat en el llenguatge de marques XML per a poder representar les dades que es transmeten.

Objectiu 5: Funcionalitats del sistema.

El projecte ofereix certes funcionalitats que el Gestor, com a servidor, ha d'implementar i que Pacient i Metge com a clients, han de saber utilitzar. Aquestes són les següents:

Pacient:
- Consulta del seu historial mèdic.

Metge
- Consulta del llistat de pacients assignats.
- Consulta de l'historial mèdic d'un dels seus pacients.
- Inserció d'una nova visita mèdica a l'historial d'un dels seus pacients.

Objectiu 6: Interfície de treball per al Pacient.

El Pacient haurà de disposar d'una interfície gràfica que implementi les funcionalitats esmentades. Concretament, la consulta del seu historial mèdic.

Objectiu 7: Interfície de treball per al Metge.

El Metge haurà de disposar d'una interfície gràfica que implementi les funcionalitats esmentades. Concretament, la consulta del llistat de pacients, la consulta d'historials mèdics i la inserció de noves visites.

1.3.- Enfocament i mètode seguit

El projecte es divideix en diferents fases que permeten un desenvolupament incremental en el que es van assolint fites. Cada fita correspon al desenvolupament d'un petit mòdul (documentació. codificació, proves, tests unitaris...) i la integració amb el mòdul anterior. Aquestes fases es corresponen a les descrites a la planificació del projecte (veure punt següent).

Fases del PFC:
1. Instal·lació de l'entorn de treball
2. Protocols criptogràfics
3. Representació de dades
4. Comunicacions
5. Aplicació Pacient
6. Aplicació Metge
7. Detalls Finals
8. Lliurament Projecte

1.4.- Planificació del projecte

El projecte es situa en el segon semestre del curs 2007-2008. A continuació es mostra una planificació detallada des de l'inici fins al 07 de Gener, data en que es lliurarà el PFC finalitzat.

F.1 Planificació

Fites	Descripció
30 Setembre 2007	**PAC 1: Instal·lació de l'entorn de treball**
	Planificació
	Documentació
	Instal·lar entorn de treball: OpenSSL, MySQL, Apache HTTPD, gSOAP, libxml, GTK+.
21 Octubre 2007	**PAC 2: Protocols criptogràfics**
	Documentació
	Disseny protocols
	API: Protocols criptogràfics
04 Novembre 2007	**PAC 3: Representació de dades**
	Documentació
	Disseny documents
	API: Documents XML
	Disseny Base de Dades
18 Novembre 2007	**PAC 4: Comunicacions**
	Documentació
	Implementació serveis web.
02 Desembre 2007	**PAC 5: Aplicació Pacient**
	Documentació
	Aplicació Pacient
16 Desembre 2007	**PAC 6: Aplicació Metge**
	Documentació
	Aplicació Metge
30 Desembre 2007	**PAC 7: Detalls finals**
	Documentació
	Desenvolupament
07 Gener 2008	**PAC 8: Lliurament Projecte**
	Documentació
	Preparació paquet .deb
	Proves finals

1.5.- Productes obtinguts

Com a resultat del projecte es lliurarà un paquet .deb, instal·lable en sistemes Debian GNU/Linux o distribucions basades en aquest, com per exemple Ubuntu Linux (Distribució Oficial del projecte). Aquest paquet instal·larà tres aplicacions, dues de les quals (Metge i Pacient) es podran utilitzar a través d'una interfície gràfica GTK+, la tercera (Gestor) funcionarà com a servidor. L'aplicatiu Gestor disposarà d'un repositori (Base de dades MySQL) amb tots els historials mèdics i permetrà la seva gestió de forma segura. L'aplicatiu pacient permetrà als pacients consultar les dades del seu historial i l'aplicatiu metge permetrà als metges consultar i inserir visites als l'historials dels pacients.

Després d'instal·lar el paquet .deb i seguir unes petites instruccions de configuració, es podran llençar els aplicatius “pacient” i “metge” per a fer servir el programari.

1.6.- Breu descripció dels capítols de la memòria

Capítol 2: Gestió d'historials mèdics
Descripció de les necessitats del projecte, com s'afronten, protocols utilitzats, etc.

Capítol 3: Representació de les dades de transport
Disseny dels documents XML utilitzats.

Capítol 4: Emmagatzematge de dades
Disseny de bases de dades i implementació amb MySQL.

Capítol 5: Protocol de comunicació
Ús de WebServices com a protocol de comunicació.

Capítol 6: Accés a dades de configuració
Cóm emmagatzemar i accedir a dades de configuració de l'aplicatiu.

Capítol 7: La interfície gràfica
Implementació de les interfícies gràfiques mitjançant GTK+.

Capítol 8: Possibles millores
Comentaris sobre desenvolupaments futurs.

Capítol 9: Conclusions
Conclusions finals.

Capítol 2: Gestió d'historials mèdics

2.1.- Les eines del Projecte

Com s'ha comentat en el capítol anterior, el projecte que ens ocupa disposa d'uns requisits de seguretat elevats. Per poder-los afrontar amb èxit necessitarem una llibreria criptogràfica capaç d'implementar una Infraestructura de Clau Pública. Existeixen diverses llibreries que podem fer servir, com poden ser IAIK (http://jce.iaik.tugraz.at/) per al llenguatge Java o Crypto++ (http://www.cryptopp.com) per al llenguatge C++. Però sense cap mena de dubte, la que més s'utilitza en entorns de programari lliure es OpenSSL (http://openssl.org). Aquesta llibreria la fan servir aplicacions força conegudes com el servidor web Apache (http://www.apache.org) o el servidor de shell segura OpenSSH (http://openssh.org) i és gairebé, la opció a triar per defecte en tota aplicació GNU desenvolupada en llenguatge C. Per aquests motius, s'ha triat la llibreria OpenSSL per al projecte.

Seguint amb la idea d'un desenvolupament basat en una plataforma lliure, el sistema operatiu triat és un GNU/Linux. Com a distribució de GNU/Linux hi han moltes opcions vàlides. El projecte es provarà en distribucions basades en Debian (http://debian.org) i distribucions basades en Fedora/Redhat (http://redhat.com) ja que són les més utilitzades. Però com que és apropiat triar una distribució “oficial” en la que implementar el projecte, s'optarà per fer-ho amb Ubuntu (http://ubuntu.org) basada en Debian. D'aquesta manera el programari desenvolupat es pot lliurar empaquetat en un paquet Debian (.deb).

El programari requereix d'una interfície gràfica que permeti la manipulació de les dades d'una manera còmoda. Actualment dos sistemes d'escriptori dominen el mon dels sistemes operatius lliures, aquests són GNOME (http://gnome.org) i KDE (http://kde.org).
El primer es basa en la llibreria GTK+(http://gtk.og) mentre que el segon ho fa en la llibreria Qt (http://trolltech.com/products/qt). Mentre que la llibreria Qt està desenvolupada en C++, la llibreria GTK+ ho està en C, al igual que la llibreria OpenSSL. Donat que aquest serà el llenguatge de programació utilitzat, la llibreria més adequada serà la GTK+.

La manipulació de dades es farà en format XML. Com que el mateix sistema GNOME disposa d'una llibreria XML perfectament integrada, farem ús d'aquesta. La llibreria es coneix com libxml (http://xmlsoft.org).

Com a base de dades per a emmagatzematge farem servir MySQL (http://mysql.com) la més utilitzada en el mon del programari lliure.

Finalment, hem optat per fer servir webservices per a comunicar els diferents components que formen part del projecte: L'aplicatiu client, l'aplicatiu metge i el gestor. Com a llibreria que permeti desenvolupar webservices en C la més coneguda i madura es gSoap (http://www.cs.fsu.edu/~engelen/soap.html). Aquesta es farà servir conjuntament amb el servidor web Apache.

2.2.- Documentació

El codi font es documenta amb comentaris tipus Java, els més estesos. D'aquesta manera es poden fer servir eines de documentació automàtica.
Així, per a complementar la documentació inclosa en la memòria també s'afegeix en el producte final la documentació del codi font desenvolupat generat a partir de l'eina Doxygen (http://www.stack.nl/~dimitri/doxygen/).
Aquesta documentació s'inclourà en el paquet .deb resultant en format html.

2.3.- Arquitectura

Els aplicatius segueixen una arquitectura en tres capes ens la que una primera capa formada per l'aplicatiu Pacient o l'aplicatiu Metge soliciten certes operacions a una segona capa, l'aplicatiu Gestor. La tercera capa està formada per la base de dades en la que s'emagatzemen els historials mèdic. La comunicació entre la primera i la segona capa es fa mitjançant webservices, mentre que la comunicació entre la segona i la tercer capa es fa mitjançant sockets TCP.

2.4.- Requisits de seguretat

Les dades mèdiques d'un pacient són altament confidencials, de manera que els aplicatius han de garantir tota una sèrie de mesures de seguretat que descrivim a continuació.

S'ha de garantir la **confidencialitat** de les dades. Pel que només podrà accedir a l'historial mèdic d'un pacient, el propi pacient i el seu metge. Per a poder-ho garantir serà necessari un control d'accés a les mateixes, així com un emmagatzematge de les dades xifrades, amb la correcta gestió de claus corresponent.

Quan un Pacient o el seu Metge consulten els historials mèdics han d'estar segurs de la seva **autenticitat**. De manera que, els historials estaran degudament signats. Per altra banda, els mecanismes criptogràfics utilitzats han de garantir la seva **integritat**, es a dir, que ningú ha falsificat un historial.

Si un Pacient modifica les dades personal del seu historial mèdic o si el seu Metge afegeix noves dades mèdiques, s'ha de complir el requisit de **no-repudi,** es a dir, que no han de poder negar haver fet aquestes modificacions.

Tots aquests requisits es podran dur a terme mitjançant l'ús duna Infraestructura de Clau Pública. En propers apartats es descriuen els protocols que permeten garantir -los.

2.5.- L'historial mèdic del Pacient

Com hem comentat anteriorment les dades de l'historial mèdic d'un pacient són altament confidencials, pel que les haurem de emmagatzemar degudament xifrades i mantenir-hi un estricte control d'accés.

Per al projecte es farà servir un control d'accés bàsic en el que només es disposa de dos tipus d'accés als historials:

Accés de lectura:
Aquest tipus de privilegis és el que tenen tant el propi pacient com el seu Metge i com el seu nom indica permetrà accedir amb permisos de lectura a l'historial complert.

Accés d'inserció de dades mèdiques:
Aquest tipus de privilegi és el Metge assignat al Pacient i permetrà inserir noves dades mèdiques procedents d'una visita al Pacient.

2.6.- Protocol de consulta d'un historial mèdic

A continuació es mostra el protocol utilitzat per a la consulta de l'historial mèdic d'un pacient. Aquest protocol esta dividit en cinc parts, les quals es van executant alternativament entre l'usuari del sistema i el gestor.

[pki_get_medical_data_U_step1]

Primera fase del protocol que permet obtenir l'historial mèdic d'un pacient.

Part que executaran pacients i metges (U).

1. Es genera un nombre aleatori Ni.
2. Xifrar Ni i IDu (id de l'usuari) amb la clau publica del Gestor.
3. Guarda pub_G[Ni,IDu] a md_step1.

[pki_get_medical_data_G_step2]

Segona fase del protocol que permet obtenir l'historial mèdic d'un pacient.

Part que executarà el gestor (G).

1. Desxifrar md_step1 amb priv_G, obtenint Ni i IDu.
2. Obtenir pub_U a partir de IDu i la DB.
3. Generar un nombre aleatori Ng.
4. Guarda Ni i Ng associats amb l'usuari a la DB.
5. Xifra Ni, Ng i IDg (id d'usuari del gestor) amb pub_U.
6. Guarda pub_U[Ni,Ng,IDg] a md_step2.

[pki_get_medical_data_U_step3]

Tercera fase del protocol que permet obtenir l'historial mèdic d'un pacient.

Part que executaran pacients i metges (U).

1. Desxifrar md_step2 amb priv_U, obtenint Ni', Ng i IDg.
2. Si Ni' != Ni retornar error i finalitzar.
3. Xifrar Ng, 'consulta', ID (id usuari a consultar) amb pub_G.
4. Guardar pub_G[Ng,'consulta',ID] a md_step3.

[pki_get_medical_data_G_step4]

Quarta fase del protocol que permet obtenir l'historial mèdic d'un pacient.

Part que executarà el gestor (G).

1. Desxifrar md_step3 amb priv_G, obtenint Ng', 'consulta' i ID.
2. Obtenir Ng i IDu de la DB.
3. Si Ng'!= Ng retornar error i finalitzar.
4. Si (IDu != ID) i (ID no es pacient de IDu) retornar error i finalitzar.
5. Buscar el historial H corresponent a ID i desxifrar-lo amb priv_G.
6. Xifrar H amb pub_U.
7. Guardar pub_U[H] a md_step4.
8. Esborrar Ng i Ni de la DB.

[pki_get_medical_data_U_step5]

Cinquena fase del protocol que permet obtenir l'historial mèdic d'un pacient.

Part que executaran pacients i metges (U).

1. Desxifrar md_step4 amb priv_U, obtenint H.
2. Verificar la signatura del Gestor amb pub_G.
3. Verificar la signatura del Metge amb pub_M.
4. Guardar H a md_step5.

2.7.- Protocol de consulta dels pacients assignats a un metge

A continuació es mostra el protocol utilitzat per a la consulta del llistat de pacients assignats a un metge. Aquest protocol esta dividit en cinc parts, les quals es van executant alternativament entre l'usuari del sistema (un metge) i el gestor.

[pki_get_patient_list_M_step1]

Primera fase del protocol que permet obtenir una llista de Pacients assignats a un Metge.

Part que executarà el Metge (M).

1. Es genera un nombre aleatori Ni.
2. Xifra Ni i IDu (id de l'usuari) amb la clau publica del Gestor.
3. Guarda pub_G[Ni,IDu] a pl_step1.

[pki_get_patient_list_G_step2]

Segona fase del protocol que permet obtenir una llista de Pacients assignats a un Metge

Part que executarà el gestor (G).

1. Desxifrar pl_step1 amb priv_G, obtenint Ni i IDu.
2. Obtenir pub_U a partir de IDu i la DB.
3. Generar un nombre aleatori Ng.
4. Guarda Ni i Ng associats amb l'usuari a la DB.
5. Xifra Ni, Ng i IDg (id d'usuari del gestor) amb pub_U.
6. Guarda pub_U[Ni,Ng,IDg] a pl_step2.

[pki_get_patient_list_M_step3]

Tercera fase del protocol que permet obtenir una llista de Pacients assignats a un Metge

Part que executarà el Metge (M).

1. Desxifrar pl_step2 amb priv_U, obtenint Ni', Ng i IDg.
2. Si Ni' != Ni retornar error i finalitzar.
3. Xifrar Ng, 'llista_pacients' amb pub_G.
4. Guardar pub_G[Ng,'llista_pacients'] a pl_step3.

[pki_get_patient_list_G_step4]

Quarta fase del protocol que permet obtenir una llista de Pacients assignats a un Metge Part que executarà el gestor (G).

1. Desxifrar pl_step3 amb priv_G, obtenint Ng' i 'llista_pacients'.
2. Obtenir Ng i IDu de la DB.
3. Si Ng'!= Ng retornar error i finalitzar.
4. Si IDu no es metge retornar error i finalitzar.
5. Obtenir llista d'ususaris (L) assignats a un Metge amb IDu de la DB.
6. Signar L amb priv_G, obtenint Sg(L).
7. Xifrar L i Sg(L) amb pub_U.
8. Guardar pub_U[L,Sg(L)] a pl_step4. 9. Esborrar Ng i Ni de la DB.

[pki_get_patient_list_M_step5]

Cinquena fase del protocol que permet obtenir una llista de Pacients assignats a un Metge Part que executaran el Metge (M).

1. Desxifrar pl_step4 amb priv_U, obtenint L i Sg(L).
2. Verificar la signatura del Gestor amb pub_G.
3. Guardar L a pl_step5.

2.8.- Protocol d'inserció de dades a l'historial mèdic

A continuació es mostra el protocol utilitzat per a inserir una visita a l'historial d'un pacient. Aquest protocol esta dividit en quatre parts, les quals es van executant alternativament entre l'usuari del sistema (un metge) i el gestor.

[pki_insert_data_M_step1]

Primera fase del protocol que permet inserir una visita a l'historial mèdic.

Part que executarà el Metge (M).

1. Es genera un nombre aleatori Ni.
2. Xifra Ni i IDu (id de l'usuari) amb la clau publica del Gestor.
3. Guarda pub_G[Ni,IDu] a id_step1.

[pki_insert_data_G_step2]

Segona fase del protocol que permet inserir una visita a l'historial mèdic.

Part que executarà el gestor (G).

1. Desxifrar id_step1 amb priv_G, obtenint Ni i IDu.
2. Obtenir pub_U a partir de IDu i la DB.
3. Generar un nombre aleatori Ng.
4. Guarda Ni i Ng associats amb l'usuari a la DB.
5. Xifra Ni, Ng i IDg (id d'usuari del gestor) amb pub_U.
6. Guarda pub_U[Ni,Ng,IDg] a id_step2.

[pki_insert_data_M_step3]

Tercera fase del protocol que permet inserir una visita a l'historial mèdic.

Part que executaran el Metge (M).

1. Desxifrar id_step2 amb priv_U, obtenint Ni', Ng i IDg.
2. Si Ni' != Ni retornar error i finalitzar.
3. Signar V (la visita) amb priv_M, obtenint Sm(V).
4. Xifrar Ng, V, Sm(V), i 'inserir_visita' amb pub_G.
5. Guardar pub_G[Ng,'inserir_visita', V, Sm(V)] a id_step3.

[pki_insert_data_G_step4]

Quarta fase del protocol que permet inserir una visita a l'historial mèdic.

Part que executarà el gestor (G).

1. Desxifrar id_step3 amb priv_G, obtenint Ng', 'inserir_visita', V i Sm(V).
2. Obtenir Ng i IDu de la DB.
3. Si Ng'!= Ng retornar error i finalitzar.
4. Obtenir IDp (id del pacient) a partir de V.
5. Si IDu no es Metge retornar error i finalitzar.
6. Si IDp no es pacient de IDu retornar error i finalitzar.
7. Verificar la signatura de V amb pub_M.
8. Obtenir T, l'instant de temps actual.
9. Obtenir S, el numero de sèrie de la ultima visita de IDp i fer S=S+1.
10. Signar amb priv_G: Sg[V, Sm[V], T, S+1]
11. Xifrar amb pub_G: pub_G[V, Sm(V)].
12. Signar amb priv_G: Sg(S+1, IDp).
13. Guardar a la DB: pub_G[V, Sm(V)], S+1, T, Sg[V, Sm[V], T, S+1] i Sg[S+1, Idp].

2.9.- Xifrat i Signatura

Els protocols exposats juguen amb un parell de claus: clau pública i clau privada. Per aplicar aquestes claus directament per al xifrat i la signatura digital ens trobem amb alguns problemes que cal resoldre. Per exemple, xifrar un document relativament llarg amb un criptosistema de clau pública és costós i poc segur. De la mateixa manera, signar un document de les mateixes característiques pot donar lloc a signatures excessivament llargues. Per aquests motiu ens veiem obligats a cercar mètodes alternatius que resolguin aquests problemes.

De cara a resoldre el problema del xifrat, em de tenir en compte que la millor manera de xifrar un document llarg consisteix en fer servir un algoritme de xifrat simètric, com pot ser per exemple, el AES o el TripleDes. Així una manera de resoldre el nostre problema consistiria en generar una clau de sessió per a un algoritme simètric, xifrar el document amb aquest algoritme i enviar el resultat junt amb la clau de sessió xifrada per la clau pública (també seria convenient xifrar amb la clau pública alguna cosa més, com el nom de l'algoritme, el mode de xifrat, etc).
Aquesta manera de resoldre el problema es coneix com a sobre digital i existeix un estàndard anomenat PKCS#7 que l'especifica, incloent també la signatura digital. De cara al projecte que ens ocupa, optarem per la utilització de la implementació del PKCS#7 que fa la llibreria OpenSSL. Això ens garantirà la compatibilitat i ens permetrà guanyar temps en el desenvolupament.

De cara a resoldre el problema de la signatura digital la millor opció sol ser realitzar un resum o hash del document que volem signar i aplicar la signatura sobre aquest, directament amb la clau privada. Donada la senzillesa d'aquesta operació, ho farem directament, prescindint del mecanisme utilitzat en el PKCS#7.

2.10.- Implementació amb OpenSSL

Per a la implementació dels protocols criptogràfics s'utilitza la llibreria OpenSSL. Aquesta proporciona una API molt rica que permet manipular informació criptogràfica d'una manera força senzilla.

S'han fet servir principalment tres tipus de dades per a la implementació dels protocols esmentats: Els certificats x509, representat pel tipus de dada “X509”, una estructura que permet emmagatzemar parells de claus anomenada “EVP_PKEY”, un tipus de dades numèric per a tractar nombres molt grans “BIGNUMBER” i el tipus de dada “PKCS7” que permet xifrar fent servir la tècnica coneguda com a 'Sobre Digital”.

Amb tot, s'ha tingut que crear alguns tipus de dades nous que es veuran amb detall en els propers apartats. Són exemples el el tipus de dada “pki_t”, implementat en els arxius “src/pki.h” i “src/pki.c” o el tipus de dada “xmlcrypt_t”, implementat en els arxius “src/xmlcryt.h” i “src/xmlcrypt.c”.

2.11.- Tests unitaris

Com a primera fase de desenvolupament del projecte s'han implementat els protocols criptogràfics. Per a veure que la implementació és correcta, un petit programa situat a “src/test_pki.c” s'encarrega de fer els tests unitaris corresponents.

El programari es lliura en un paquet .deb propi de les distribucions Debian o basades en aquesta, com per exemple Ubuntu. Es pot instal·lar el paquet mitjançant:

$ pkg -i gestor-historials_VERSIO_i386.deb

Aquest paquet instal·larà alguns binaris a /usr/bin així com una còpia de la documentació, del codi font i d'un joc de claus prova a /usr/share/doc/gestor-historials.

De cara a veure el codi font de l'aplicatiu es recomanable mirar la documentació doxygen a /usr/share/doc/gestor-historials/doc/html i a continuació les fonts a /usr/share/doc/gestor-historials/sources.

Finalment, per a realitzar el test unitari, després d'instal·lar el paquet, es pot fer mitjançant la comanda:

$ test_pki

Capítol 3: Representació de les dades de transport

3.1.- Format de representació de dades

El sistema de gestió d'historials mèdic requereix de la manipulació d'estructures complexes de dades. De cara a enviar aquestes estructures a través de la xarxa, una bona manera de representar-les és mitjançant l'estàndard XML. Aquest format, força estès, garantirà la interoperativitat entre diferents tecnologies i protocols.

Per al projecte que ens ocupa necessitarem representar dades de diferents tipus. Per una banda, haurem de representar les dades de transport que es faran servir per a comunicar els aplicatius clients (Aplicatiu Metge y Aplicatiu Pacient) amb el servidor (o Gestor). Per un altra banda, la representació mitjançant XML també ens serà útil per a les dades de l'historial mèdic.

En els següents apartats es defineixen els documents XML que es faran servir en el projecte, així com la manera de fer-los servir per a manipular dades criptogràfiques.

3.2.- Aplicació de criptografia al XML

Els requisits de seguretat del projecte, els quals ja s'han definit en apartats anteriors, requereixen l'ús de criptografia. Donat que les dades transmeses estan representades amb XML el més adequat serà integrar els documents XML que farem servir amb les necessitats criptogràfiques.

Concretament, per a poder assolir els objectius de tota PKI, un document XML s'haurà de poder xifrar, desxifrar, signar i verificar la seva signatura. Totes aquestes característiques les acomplirà el TAD (Tipus Abstracte de Dades) que hem definit per al projecte. Aquest TAD anomenat xmlcrypt_t es defineix en el següent apartat.

3.3.- El tipus abstracte de dades XMLCRYPT

El TAD (Tipus Abstracte de Dades) XMLCRYPT definit en el projecte com “xmlcrypt_t”, representa un document XML que suporta criptografia. Concretament aquest tipus de dades pot ser xifrat i desxifrat, es pot signar amb diferents certificats i es poden verificar les seves signatures.

El TAD XMLCRYPT disposa de les següents operacions:

F.2 TAD xmlcrypt_t

TAD xmlcrypt_t	
xmlcrypt_t * xmlcrypt_alloc ()	Reserva memòria per a un objecte xmlcrypt_t.
void xmlcrypt_free (xmlcrypt_t *obj)	Allibera la memòria reservada per l'objecte.
void xmlcrypt_encrypt (xmlcrypt_t *obj, X509 *cert)	Xifra el contingut del XML mitjançant PKCS#7.
void xmlcrypt_decrypt (xmlcrypt_t *obj, EVP_PKEY *pkey, X509 *cert)	Desxifra el contingut del XML xifrat amb PKCS#7.
void xmlcrypt_sign (xmlcrypt_t *obj, EVP_PKEY *pkey)	Signa el document XML.
int xmlcrypt_verify (xmlcrypt_t *obj, EVP_PKEY *pkey)	Verifica la signatura del document XML.
void xmlcrypt_open_document (xmlcrypt_t *obj, const char *doc, int enciphered)	Inicialitza l'objecte amb un document XML.
void xmlcrypt_create_document (xmlcrypt_t *obj)	Crea un document XML nou.
char * xmlcrypt_get_doc (xmlcrypt_t *obj)	Retorna el document XML en clar.
char * xmlcrypt_get_enciphered_doc (xmlcrypt_t *obj)	Retorna el document XML xifrat.
void xmlcrypt_add_field (xmlcrypt_t *obj, const char *name, const char *value)	Afegeix un camp al XML.
void xmlcrypt_add_field_attribute (xmlcrypt_t *obj, const char *name, const char *attrname, const char *attrvalue)	Afegeix un atribut a un camp del XML.
char * xmlcrypt_get_field_value (xmlcrypt_t *obj, const char *name)	Obté un camp del XML.
char * xmlcrypt_get_attribute_value (xmlcrypt_t *obj, const char *name, const char *attrname)	Obté un atribut d'un camp del XML.

3.4.- Representació de les dades xifrades

Un document XMLCRYPT te dues representacions. Una primera representació en text en clar, dependrà del tipus de dades que conté i es tractarà en els següents apartats. Una segona representació xifrada es tracta a continuació.

Un Exemple de document XMLCRYPT xifrat es la següent:

F.3 XMLCRYPT xifrat

Exemple de representació xifrada de XMLCRYPT

```
<?xml version="1.0"?>
<xmlcrypt encrypt="yes" type="pkcs7">
<content>
MIME-Version: 1.0
Content-Disposition: attachment; filename="smime.p7m"
Content-Type: application/x-pkcs7-mime; smime-type=enveloped-data;
name="smime.p7m"
Content-Transfer-Encoding: base64

MIIDBgYJKoZIhvcNAQcDoIIC9zCCAvMCAQAxggFDMIIBPwIBADCBpzCBmTELMAkG
A1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNlbG9uYTESMBAGA1UEBxMJQmFyY2Vsb25h
MQwwCgYDVQQKEwNVT0MxFjAUBgNVBAsTDVBGQyBTZWd1cmV0YXQxGTAXBgNVBAMU
EENBX1BGQ19TZWd1cmV0YXQxITAfBgkqhkiG9w0BCQEWEmpjYXN0ZWxsYXJAdW9j
LmVkdQIJANj9JE66DWRBMA0GCSqGSIb3DQEBAQUABIGAEmjL0RjfZxQ2kTetJtg3
BG2w5ciHr1ld3e1J5/DXKJro4OO/M4wjYyYSb6wwyO7ngqxlOfBb6L+j2JVM31Mz
/pEQ7ICwKbIO6aNE/Spm/34/upQm8GMI3Sgi2ovG5GaZLGdwz/6iAhLTzR4dEJOQ
fQeSW82Bz3cpCqAOtSYymT0wggGlBgkqhkiG9w0BBwEwFAYIKoZIhvcNAwcECJjC
gSep5Ev9gIIBgBOML8ceVuGxNNTSCdqiqkfPU4HfE3e6BibGaJP5zoQuvnWMTx9E
o7RnhVLnQpq8GB2Pk2ucpwDWhaQNzVaDKFl1K8yWtGpJLuhPhbvsPu3xZQ7aRvxM
K+OJ1YoVooTGWCJI9EyVx1hEX+uGU7LZCEor3/cOrMVERk6Ali1LBqESkBXlJ+yc
d5ABDdmsGZ+KhWG4RwhnZVX145kf1zovwwUktt5NBluryqVBaGeLFeRqzDgwxmMI
BBYcuJPADjSTex6GcDN/iu4TiwUENY+D9GfY/XX88Kg46XHUtXTv6GAaSXmAZUdY
d7t5mdapbveHRKsLCVQyCshR3w5L2JXO/B3NwSmTkKzjgYDfCvI3PoYoEFPq/os9
kODA2lzAtb6xIAgetEsfqbDLB1G1AG0W2zavBSt7j/FFsxftt2rag3BfHn3f3KGP
AN8lf3zJqKR20xK+NEZGJNDokk0Mu8PbhCrvf2pbKzwQMuAU0EdplFoNbhSEt64p
Tp7Fvl+j6BnRdg==

</content>
<sign>
LzpAFj/9sbqHFDT7c8Oub+dnBup+f39VQUJY/jB7SoIH1rk6qSUYX8mxRawrecY8WzhiCNjtQaEGoKl
LJjJ/PkgmH+Ppg13z8dfyfnUW7KspedxMyq3UIv6Xy+LKkxI/k7h+ZhULbicq8k/aMMJkvD42zi2PPu
w61F43FKWkKjM=
</sign>
</xmlcrypt>
```

Com es pot veure en el document XML l'atribut “type” de “xmlcrypt” ens indica que el document està xifrat mitjançant l'estàndard PKCS#7. A continuació, la etiqueta “content” conté el document PKCS#7.

En quant a la signatura veiem que està inclosa dintre de les etiquetes “sign”. Tot i que en aquest cas només se'n mostra una, el TAD XMLCRYPT suporta signatura múltiple.

3.5.- Representació de dades de transport

Les dades de transport que s'envien i es reben en cada transacció tenen un format molt senzill. Estan contingudes dintre d'una etiqueta “xmlcrypt” i cada dada representada és el contingut d'una etiqueta que la descriu. Per exemple, algunes dades comunes en la transmissió de les dades de transport poden ser el nombre aleatori Ni i el identificador d'usuari (veure protocols). Aquestes dues queden representades en el document d'exemple següent:

F.4 Exemple XML dades de transport

Exemple de representació XML de les dades de transport

```
<?xml version="1.0"?>
<xmlcrypt encrypt="no">
<Ni>11132442277852194516559925780395390990194076504783489016328683053942443101608741152415123431832732280003046463462234625973003194984738551314843158524111097</Ni>
<Idu>123</IDu>
</xmlcrypt>
```

La forma presentada en la figura 4 representa el document desxifrat, tal i com hi treballa l'aplicatiu internament. Però en tot el procés de comunicació el document XML viatja xifrat, tal i com s'ha comentat al apartat 3.4.

3.6.- Representació de l'historial mèdic

L' Historial mèdic està format per un conjunt de documents XML separats, cadascun dels quals representa una visita del pacient. Al menys, així es com es representa internament en la base de dades del sistema.
De cara a mostrar el document XML a l'usuari, aquest rep una petita transformació, en la qual s'agrupen totes les visites mèdiques en un sol document XML que representa tot l'historial.

A continuació veiem la representació en clar XML d'un historial mèdic d'exemple:

F.5 Exemple XML historial mèdic

Exemple de representació XML de l'historial mèdic

```
<?xml version="1.0"?>
<historial>
  <visita id="1">
    <idusuari>1</idusuari>
    <data>10/12/2006</data>
    <motiu>Febre</motiu>
    <diagnostic>Infeccio virica</diagnostic>
    <recepta>Paracetamol</recepta>
    <comentaris>Se li dona la baixa laboral</comentaris>
  </visita>
  <visita id="2">
    <idusuari>1</idusuari>
    <data>10/07/2007</data>
    <motiu>Mal de queixals</motiu>
    <diagnostic>Creixement queixals del ceny</diagnostic>
    <recepta>Nolotil</recepta>
    <comentaris>Sense comentaris</comentaris>
  </visita>
</historial>
```

3.7.- Implementació amb libxml

Per a implementar les característiques exposades en els apartats anteriors ha sigut necessària una llibreria amb suport XML per al llenguatge C.

Després d'instal·lar el paquet .deb com en casos anteriors:

$ pkg -i gestor-historials_VERSIO_i386.deb

Quedaran instal·lats en el sistema alguns binaris a /usr/bin així com una còpia de la documentació, del codi font i d'un joc de claus prova a /usr/share/doc/gestor-historials.

De cara a veure el codi font desenvolupat es recomanable mirar la documentació doxygen a /usr/share/doc/gestor-historials/doc/html i a continuació les fonts a /usr/share/doc/gestor-historials/sources.

3.8.- Test unitari

Per a realitzar el test unitari es pot fer mitjançant la comanda:

$ test_xml

Aquesta verifica que el TAD xmlcrypt_t funciona correctament.

Capítol 4: Emmagatzematge de dades

4.1.- El sistema gestor de bases de dades MySQL

MySQL es un dels Sistemes Gestors de Bases de Dades més utilitzats en el mon del programari lliure. Actualment, en la seva versió 5, disposa de totes les característiques necessàries per a un SGBD professional.

En els propers apartats es mostra el disseny de la base de dades utilitzat per al projecte, concretament, el disseny de les taules que ens permeten emmagatzemar certificats, usuaris i historials mèdics.

4.2.- Gestió de Certificats

La gestió de certificats es fa mitjançant al taula CERTIFICATS. Aquesta taula, molt senzilla, s'encarregarà d'emmagatzemar el certificat en format PEM junt amb una referència al usuari al qual pertany.

Veiem a continuació la seva estructura:

F.6 Taula de Certificats

TAULA CERTIFICATS	
ID_USUARI	INTEGER (clau primària)
CERTIFICAT	LONGTEXT

4.3.- Gestió d' Usuaris i Metges

La gestió d'usuaris i metges es fa mitjançant una taula anomenada USUARIS. Aquesta taula emmagatzemarà les dades que fa servir un usuari en les transaccions en xarxa.
Aquestes dades corresponen principalment a nombres aleatoris generats durant les transaccions en xarxa i no rebel·len cap informació sensible dels usuaris.
També s'emmagatzema una referència al usuari que actua com a metge.

Veiem a continuació la seva estructura.

F.7 Taula de d' Usuaris

TAULA USUARIS	
ID_USUARI	INTEGER (clau primària)
ID_METGE	INTEGER
Ni	VARCHAR(256)
Ng	VARCHAR(256)

4.4.- Gestió d'historials mèdics

La gestió d'historials es fa mitjançant una taula anomenada HISTORIALS. Aquesta taula emmagatzemarà les dades corresponents a les visites mèdiques de l'usuari. Aquestes dades es guarden xifrades i signades segons les necessitats i no permeten que una persona amb accés il·lícit a la base de dades pugui obtenir informació sobre un usuari i els seu historial.
Cada registre de la taula representa una visita de l'usuari. En aquest registre queda emmagatzemat un identificador seqüencial de visita per usuari, així com una referència a l'usuari. A més es guarda un document degudament xifrat i signat que conté les dades de la visita, així com una marca de temps i la signatura de les dades implicades.

D'aquesta manera es pot detectar si s'ha eliminat o afegit registres, si s'han desordenat o si algú els ha modificat.

Veiem a continuació la seva estructura.

F.8 Taula d'Historials

TAULA HISTORIALS	
ID_VISITA	INTEGER (clau primària)
ID_USUARI	INTEGER (clau primària)
VISITA	LONGTEXT
MARCA_TEMPS	VARCHAR(32)
SIGN1	TEXT
SIGN2	TEXT

4.5.- El tipus abstracte de dades DATABASE

De cara al desenvolupament s'ha creat un TAD (Tipus Abstracte de Dades) anomenat DATABASE, representat en C com a database_t. Aquest tipus, ofereix totes les operacions necessàries per a comunicar-se amb MySQL des del llenguatge C. Aquestes operacions són les següents:

F.10 TAD database_t

TAD database_t	
database_t * database_alloc ()	Reserva memòria per a un objecte database_t.
void database_free (database_t *obj)	Allibera la memòria reservada.
char * database_get_dbuser (database_t *obj)	Obté el nom de l'usuari de la BD.
void database_set_dbuser (database_t *obj, char *user)	Modifica el nom d'usuari.
char * database_get_dbname (database_t *obj)	Obté el nom de la BD.
void database_set_dbname (database_t *obj, char *name)	Modifica el nom de la BD.
void database_set_dbpass (database_t *obj, char *passw)	Modifica el password de la BD.
char * database_get_dbhost (database_t *obj)	Obté l'adreça la BD.
void database_set_dbhost (database_t *obj, char *host)	Modifica l'adreça de la BD.
void database_connect (database_t *obj)	Connecta a la BD.
void database_disconnect (database_t *obj)	Desconnecta de la BD.
resultset_t *database_execquery(database_t *obj, char *fmt,...)	Executa una consulta.
void database_transaction_start (database_t *obj)	Inicia una transacció a la DB.
void database_transaction_commit (database_t *obj)	Finalitza la transacció.
void database_transaction_rollback (database_t *obj)	Desfà la transacció.

4.6.- Implementació amb MySQL

Per a implementar les característiques exposades en els apartats anteriors ha sigut necessària la API C de MySQL. I de cara a la creació de la base de dades i les taules necessàries, s'ha tingut que desenvolupar un script SQL.

Després d'instal·lar el paquet .deb com en casos anteriors:

$ pkg -i gestor-historials_VERSIO_i386.deb

Quedaran instal·lats en el sistema alguns binaris a /usr/bin així com una còpia de la documentació, del codi font i d'un joc de claus prova a /usr/share/doc/gestor-historials.

Al directori /usr/share/doc/gestor-historials/sources hi ha un arxiu gh.sql amb la estructura de la base de dades i algunes dades de prova. Per a que el programari funcioni correctament s'ha de carregar aquestes dades al MySQL.
A més, en aquesta fase es necessari un usuari “uoc” amb contrasenya “uoc” que disposi d'accés sobre la base de dades “gestio_historials” que es crea.

De cara a veure el codi font desenvolupat es recomanable mirar la documentació doxygen a /usr/share/doc/gestor-historials/doc/html i a continuació les fonts a /usr/share/doc/gestor-historials/sources.

4.8.- Test unitari

Per a realitzar el test unitari es pot fer mitjançant la comanda:

$ test_db

Si la comanda te èxit, es pot executar també:

$ test_pki

Aquesta comanda, que ja s'ha fet servir en fases anteriors del projecte fa ús en aquestes alçades, de la base de dades.

Capítol 5: Protocol de comunicació

5.1.- Comunicació entre els elements del sistema

El projecte requereix d'un protocol de comunicació que permeti transmetre dades entre els clients: Pacient i Metge i el servidor: Gestor.

Existeixen diferents protocols que es poden fer servir per assolir aquests objectius. Un podria ser, per exemple, l'ús directe de **Sockets TCP**. Aquest tècnica però, requeriria dissenyar un protocol per sobre de TCP que respongués a les necessitats del projecte, el que podria resultar tant laboriós com innecessari, tenint en compte altres tècniques existents.

Un tècnica molt utilitzada codificada per sobre de TCP es la **crida a procediments remots**. Són força coneguts protocols com RMI que permet fer crides a procediments remots amb Java o RPC que ens permetria fer-ho amb C, llenguatge utilitzat en el projecte.

Però podem anar una mica més enllà i fer ús del protocol **SOAP** i els serveis web. De manera similar als protocols anteriors, SOAP ens permet fer crides a uns procediments remots anomenats serveis web. Aquest cop el protocol funciona mitjançant HTTP (o HTTPS) formant una capa XML per damunt d'aquests. Això ens permet fer servir el protocol a través d'Internet, avantatge força interessant. A més, la possibilitat de fer servir HTTPS ens permet afegir una capa criptogràfica SSL que manegarà el servidor web encarregat de donar els serveis (per exemple Apache) sense fer desenvolupaments addicionals.

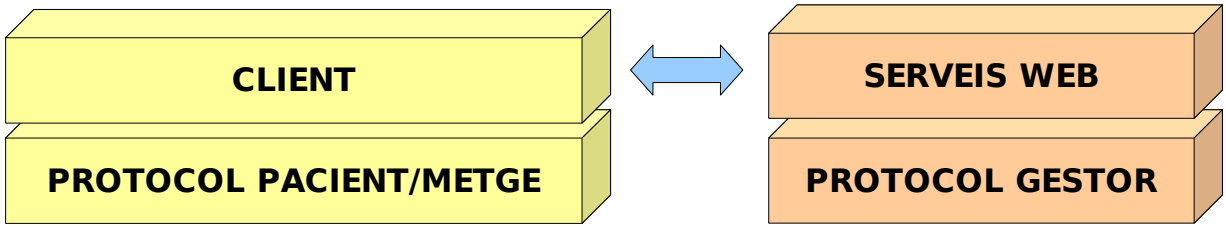
En C una de les llibreries lliures que permeten fer aquest tipus de desenvolupament més utilitzada i madura és gSoap. Com ja s'ha comentat anteriorment, aquesta serà la que farem servir en el projecte.

5.2.- Pacient, Metge y Gestor: Els serveis web

En tota transacció, Pacient i Metge com a clients, necessitaran comunicar-se amb el Gestor, que assoleix el paper de servidor. Per a cada objectiu com pot ser, per exemple, sol·licitar un historial mèdic o consultar una llista de pacients, és necessari realitzar més d'una transacció. Aquestes transaccions o passes que s'han de superar, les hem estudiat per primer cop en el capítol 2: Gestió d'historials mèdics. En aquest però, es veien com un protocol a seguir entre el Pacient/Metge i el Gestor.

Així doncs, els serveis web actuaran com una capa de comunicacions construïda directament sobre els protocols definits. Aquest serveis web seran cridats per Pacient i Metge, tal i com es veu a la figura 11. El client només accedirà a la part de protocol de client mitjançant les llibreries del sistema, doncs per a accedir al Gestor, ho farà mitjançant els serveis web.

F.11 Capa de Serveis Web



5.3.- Disseny dels serveis web

Com que cada servei web te una relació directa amb la part de protocol del Gestor, la seva API serà molt similar. Per la seva naturalesa procedimental, en aquest cas no es recomanable dissenyar un TAD (Tipus abstracte de dades). A continuació es descriuen els diferents Serveis Web i la seva funció.

F.12 Serveis Web

Serveis Web	
int ghws__getMedicalDataStep2(char *xml, char **result);	Fase 2 del protocol per a obtenir l'historial mèdic.
int ghws__getMedicalDataStep4(char *xml, char **result);	Fase 4 del protocol per a obtenir l'historial mèdic.
int ghws__getPatientListStep2(char *xml, char **result);	Fase 2 del protocol per a obtenir la llista de pacients d'un metge.
int ghws__getPatientListStep4(char *xml, char **result);	Fase 4 del protocol per a obtenir la llista de pacients d'un metge.
int ghws__insertDataStep2(char *xml, char **result);	Fase 2 del protocol per a inserir dades en un historial.
int ghws__insertDataStep4(char *xml, char **result);	Fase 4 del protocol per a inserir dades en un historial.

5.4.- Implementació amb gSoap

Per a implementar les característiques exposades en els apartats anteriors ha sigut necessària una llibreria amb suport SOAP per al llenguatge C. La llibreria triada és gSoap, pel que és necessari que estigui instal·lada per a compilar les fonts. Però com que es fa un link estàtic en la compilació, no és necessària per a instal·lar el paquet .deb de gestió d'historials.

Instal·larem el nostre paquet .deb com en casos anteriors.

$ pkg -i gestor-historials_VERSIO_i386.deb

Quedaran instal·lats en el sistema alguns binaris a /usr/bin així com una còpia de la documentació, del codi font i d'un joc de claus prova a /usr/share/doc/gestor-historials.

De cara a veure el codi font desenvolupat es recomanable mirar la documentació doxygen a /usr/share/doc/gestor-historials/doc/html i a continuació les fonts a /usr/share/doc/gestor-historials/sources.

És necessari disposar del servidor web Apache iniciat com a servidor de serveis web. En cas contrari el client no trobarà els serveis. Amb tot, el CGI necessari s'instal·la a /usr/lib/cgi-bin/. Aquesta ruta és correcta per a la majoria de distribucions Linux basades en Debian, però pot no ser-ho per a altres.

Si és una instal·lació nova s'ha de tenir en compte haver realitzat correctament les passes indicades en capítols anteriors, com per exemple la configuració de les bases de dades.

5.5.- Test unitari

Per a realitzar un test unitari es pot fer mitjançant la comanda:

$ test_client

Aquesta verifica que e client i els serveis web estan instal·lats i funcionen correctament.

Capítol 6: Accés a dades de configuració

6.1.- El fitxer de configuració

Per a evitar la necessitat de compilar el codi cada cop que es canvia alguna dada de configuració és necessari fer ús d'un arxiu de configuració. Per la comoditat d'accés que ofereix i per la integració amb la resta del projecte, s'ha decidit que el format d'aquest arxiu de configuració sigui XML. Aquest es troba a "/etc/ghconf.xml" i a continuació se'n dona una mostra.

F.13 Exemple XML historial mèdic

Exemple de representació XML de l'historial mèdic

```
<?xml version="1.0"?>
<conffile>
   <passgestor>uoc0506</passgestor>
   <idgestor>0</idgestor>
   <soapserver>http://localhost/cgi-bin/gh.cgi</soapserver>

   <dbhost>localhost</dbhost>
   <dbuser>uoc</dbuser>
   <dbpass>uoc</dbpass>
   <dbname>gestio_historials</dbname>
</conffile>
```

6.2.- El tipus abstracte de dades CONFFILE

Per a poder accedir a les dades del fitxer de configuració s'ha dissenyat un nou TAD (Tipus Abstracte de Dades). Aquest permet accedir al contingut d'un camp del fitxer de configuració d'una manera senzilla. A continuació es descriuen les seves operacions principals.

F.14 TAD conffile_t

TAD conffile_t	
conffile_t *conffile_alloc()	Reserva memòria per a un objecte conffile_t.
void conffile_free(conffile_t *obj)	Allibera la memòria reservada per a un objecte conffile_t.
void conffile_open_file(conffile_t *obj, const char *path)	Inicialitza l'objecte amb un document XML.
char *conffile_get(conffile_t *obj, const char *name);	Obté un camp del XML.

Capítol 7: La interfície gràfica

7.1.- Introducció al desenvolupament amb GTK+

Quan parlem de sistemes operatius de codi lliure, en especial de GNU/Linux, existeixen dos sistemes d'escritori que es reparteixen gairebé tota la quota d'usuaris. Aquests son GNOME (http://gnome.org) i KDE (http://kde.org). El primer utilitza com a base la llibreria per a desenvolupament gràfic GTK+, mentre que el segon utilitza la llibreria QT.
Una de les característiques que fa especial a GTK+ és que aquesta llibreria és una de les poques que està desenvolupada i es pot programar amb el llenguatge C. La llibreria QT es desenvolupa en C++. Aquesta característica de GTK+ la fa ideal per al nostre projecte.

Desenvolupar interfícies gràfiques sol requerir una gran quantitat de codi a més d'una amplia experiència i capacitat del programador, per a saber com quedarà finalment el producte. Per aquest motiu, gairebé totes les llibreries gràfiques disposen d'eines visuals que permeten dissenyar la interfície sense escriure una sola línia de codi. En GTK+ aquesta eina s'anomena GLADE (http://glade.gnome.org).
En aquest projecte farem servir GLADE com a base per a la construcció de la nostra interfície.

7.2.- Requisits de la interfície gràfica del Pacient.

Els requisits de la interfície gràfica del pacient són reduïts, doncs no disposa de moltes funcionalitats. De fet el pacient l'únic que pot fer és consultar el seu historial.

Per a connectar amb el Gestor i consultar el seu historial, el Pacient necessita la clau pública del Gestor, estar registrat al sistema (això implica que el Gestor disposa del seu certificat) i disposar d'una clau privada. Donat que la clau privada requereix d'una paraula de pas, el Pacient també haurà de proporcionar aquesta dada.

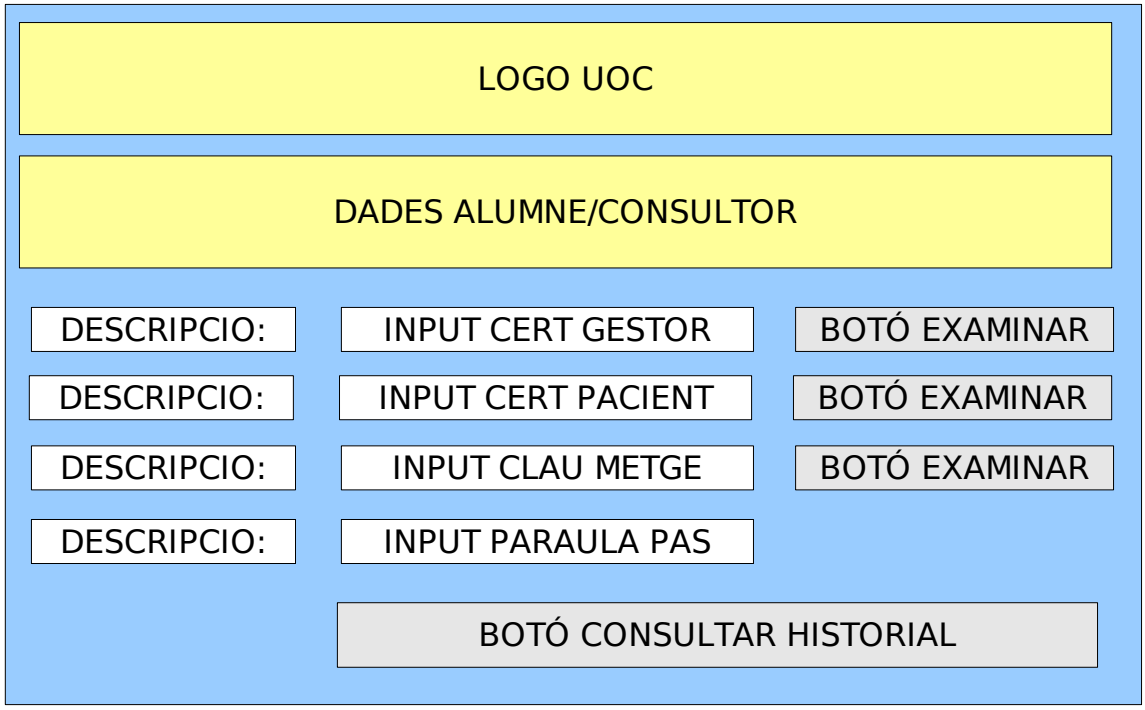
Així doncs, és necessari que al iniciar la interfície gràfica el Pacient proporcioni les següents dades:
- Ruta de la clau pública o certificat del Gestor.
- Ruta de la clau pública o certificat del Metge.
- Ruta de la clau pública o certificat propi.
- Ruta de la clau privada.
- Paraula de pas per accedir a la clau privada.

Una vegada proporcionades aquestes dades, el Pacient només te una opció, que és consultar el historial. Per tant, serà necessari un botó que doni aquesta possibilitat.

7.3.- Disseny de la interfície gràfica del Pacient.

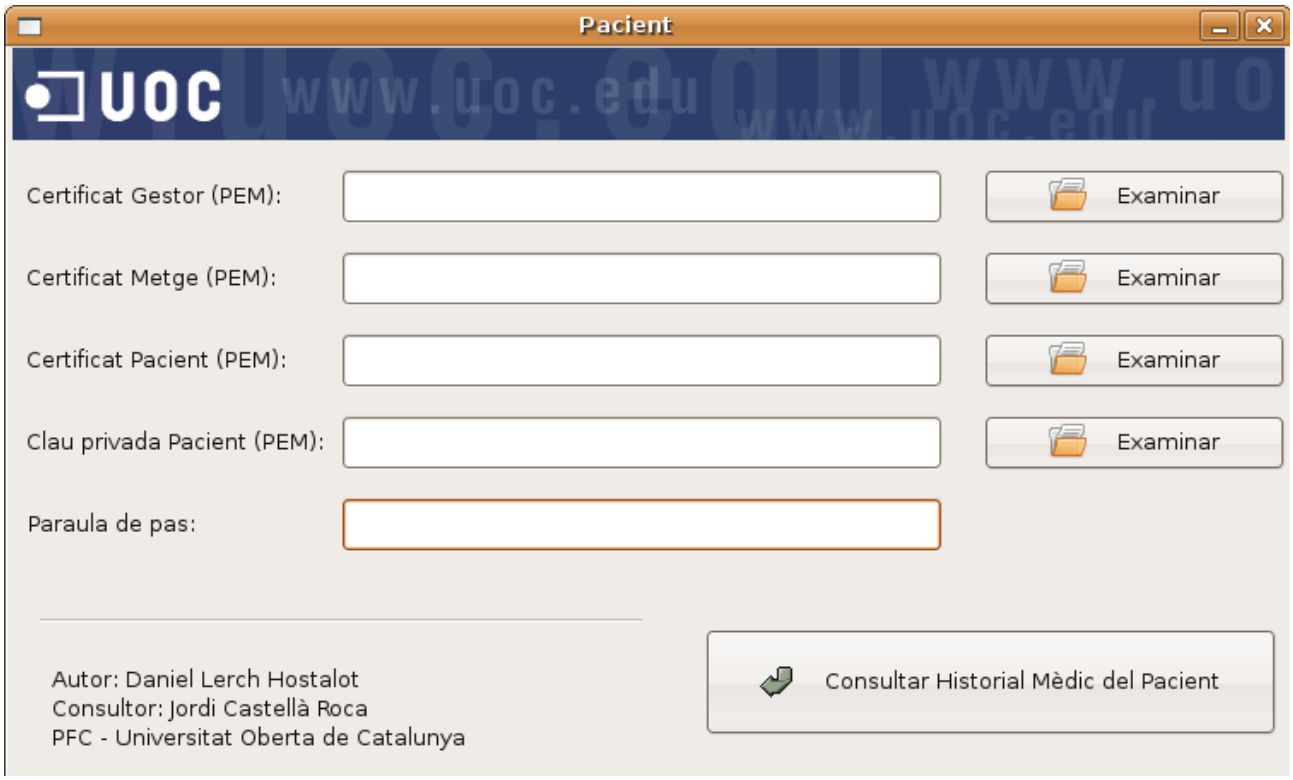
En base als requisits esmentats a l'apartat anterior ja podem desenvolupar la interfície gràfica. Aquesta hauria de disposar d'algunes dades de la Universitat, per exemple un logo, així com de les dades de l'alumne i del consultor. A més, com és lògic, de les entrades necessàries segons els requisits esmentats.
A continuació (Figura 15) és mostra un possible disseny per a la interfície del Pacient.

F.15 Interfície Pacient



La interfície final del Pacient la podem veure a la figura 16.

F.16 Interfície Pacient final



7.4.- Requisits de la interfície gràfica del Metge.

Els requisits de la interfície gràfica del Metge són una mica més extensos que en el cas del Pacient. Aquests inclouen la capacitat d' obtenir els seus pacients, consultar el seu historial i inserir una nova visita mèdica.

Per a connectar amb el Gestor i fer les operacions mencionades, el Metge necessita la clau pública del Gestor, estar registrat al sistema (això implica que el Gestor disposa del seu certificat) i disposar d'una clau privada. Donat que la clau privada requereix d'una paraula de pas, el Metge també haurà de proporcionar aquesta dada.

Així doncs, és necessari que al iniciar la interfície gràfica el Metge proporcioni les següents dades:
- Ruta de la clau pública o certificat del Gestor.
- Ruta de la clau pública o certificat propi.
- Ruta de la clau privada.
- Paraula de pas per accedir a la clau privada.

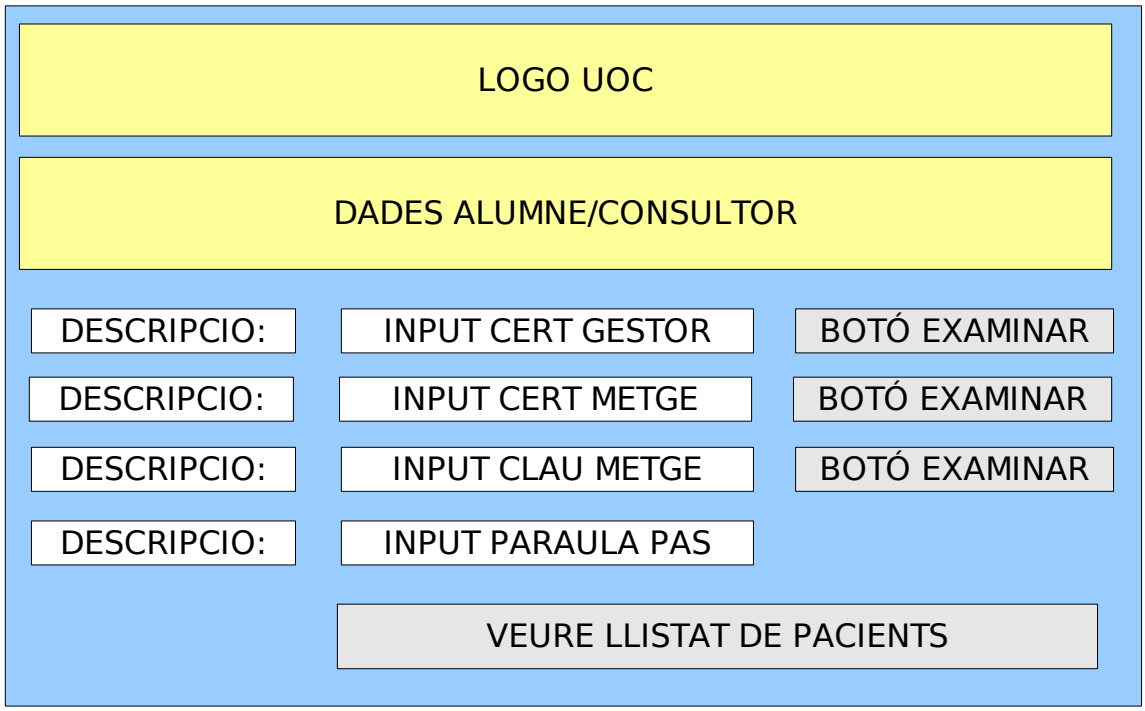
Una vegada proporcionades aquestes dades, el Metge disposa de les següents opcions:
- Consultar un llistat amb els seus pacients.
- Consultar l'historial d'un pacient.
- Inserir la visita d'un pacient.

7.5.- Disseny de la interfície gràfica del Metge.

En base als requisits esmentats a l'apartat anterior ja podem desenvolupar la interfície gràfica. Aquesta hauria de disposar d'algunes dades de la Universitat, per exemple un logo, així com de les dades de l'alumne i del consultor. A més, com és lògic, de les entrades necessàries segons els requisits esmentats.

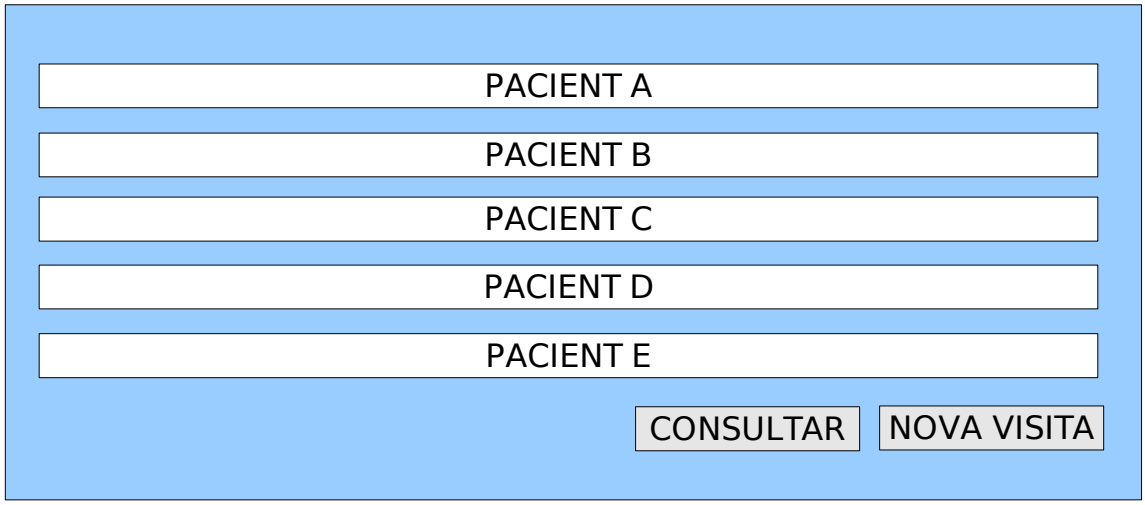
A continuació (Figura 17) és mostra un possible disseny per a la interfície del Metge.

F.17 Interfície Metge



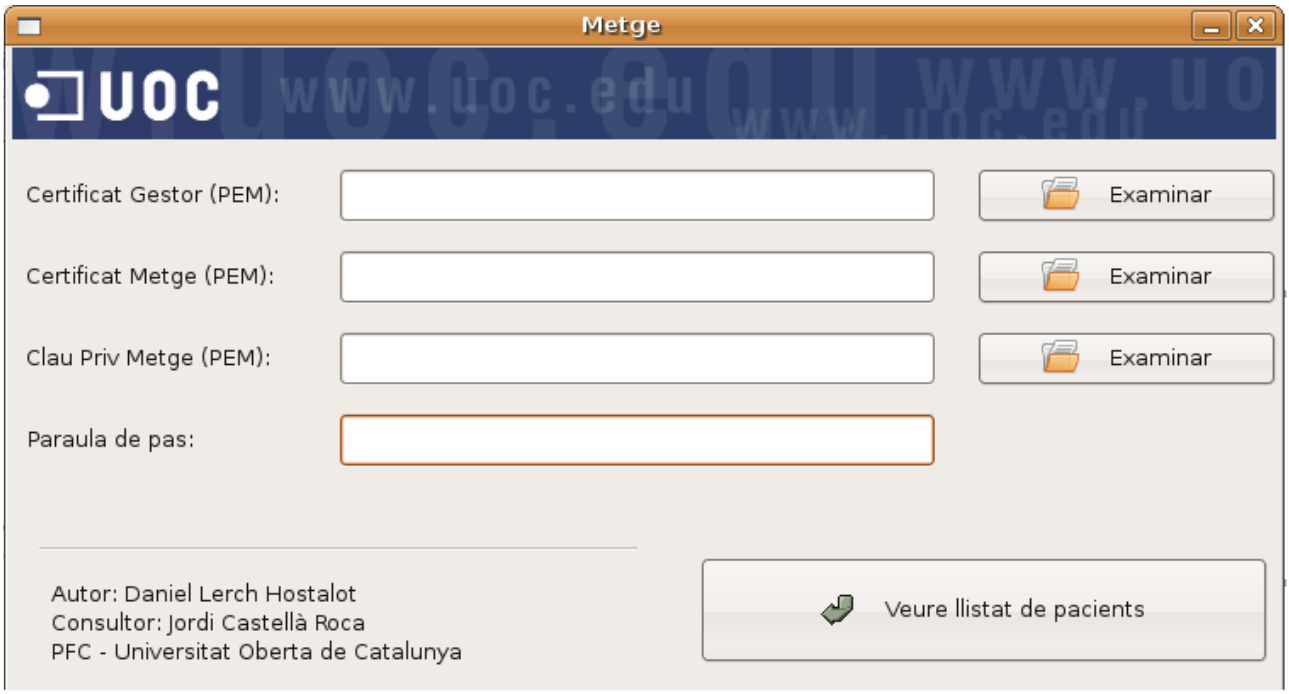
Donat que per a continuar és necessari seleccionar un Pacient, a continuació (Figura 18) és mostra un possible disseny per a la interfície de selecció de pacients.

F.18 Selecció Pacient



La interfície final del Metge la podem veure a la figura 19.

F.19 Interfície Metge final



7.6.- Implementació amb GTK+

Per a implementar les característiques exposades en els apartats anteriors ha sigut necessària la llibreria GTK+. És necessari que aquesta estigui instal·lada per a poder executar l'aplicatiu. Habitualment farà falta també la versió de desenvolupament de la llibreria per a poder compilar les fonts.

Instal·larem el nostre paquet .deb com en casos anteriors.

$ pkg -i gestor-historials_VERSIO_i386.deb

Quedaran instal·lats en el sistema alguns binaris a /usr/bin així com una còpia de la documentació, del codi font i d'un joc de claus prova a /usr/share/doc/gestor-historials.

De cara a veure el codi font desenvolupat es recomanable mirar la documentació doxygen a /usr/share/doc/gestor-historials/doc/html i a continuació les fonts a /usr/share/doc/gestor-historials/sources.

Si és una instal·lació nova s'ha de tenir en compte haver realitzat correctament les passes indicades en capítols anteriors, com per exemple la configuració de les bases de dades o l'inici del servidor web Apache.

7.7.- Test unitari

Donat que estem prop del final del desenvolupament el test unitari correspon directament a la crida de l'aplicatiu Pacient o Metge. Per a fer-ho, s'executarà la comanda:

$ pacient

per a llençar l'aplicatiu Pacient, o la comanda:

$ metge

per a llençar l'aplicatiu Metge.

Aquestes comandes iniciaran els l'aplicatius gràfics corresponents, els quals podrem provar consultant historials mèdics, inserint visites....

Capítol 8: Possibles millores

8.1.- Millores en el Client

Les aplicacions gràfiques de la part del client són força senzilles, però compleixen perfectament els objectius del projecte. De cara a realitzar un programari professional, aquestes estan força lluny del que haurien de ser, pel que s'han de considerar més una proba de concepte que no una aplicació real. En aquest aspecte, les aplicacions corresponents al Pacient i el Metge no necessiten desenvolupaments addicionals. Només una petita correcció que afecta tant al client com al servidor. Aquesta correspon al tractament dels caràcters que no són ASCII, com per exemple els accents.

El problema actual resideix en el fet de que la part del client no tracta aquests caràcters, mentre que el servidor els manipula amb l'API xml de GTK+, la qual en fa una representació especial. Això genera un problema amb les signatures, doncs quan es signa l'historial es fa amb una representació diferent de la que s'utilitza al verificar la signatura. Conseqüentment, la verificació falla. És per aquest motiu que l'aplicació no funciona si s'utilitzen caràcters fora de l'alfabet ASCII. Així doncs, una millora a realitzar consisteix en tractar els caràcters amb UNICODE en tots dos llocs.

Per altra banda, el Gestor, donat que està desenvolupat com a servidor no requereix de gaires necessitats a nivell d'administració. Però si que seria útil disposar d'una aplicació gràfica que permetés donar d'alta Metges i Pacients.

8.2.- Millores en el Servidor

El servidor utilitza webservices a través d'Apache. Això genera alguns problemes que analitzarem a continuació.

El servidor representa la figura de Gestor del sistema. Com a tal, necessita fer servir la clau privada del Gestor i conseqüentment, la seva paraula de pas. La millor manera de obtenir aquesta paraula de pas es sol·licitar-la en el moment d'iniciar el servei web Apache, i a continuació guardar-la de forma segura en memòria. Però donat que el servei web és un CGI, no disposa de la possibilitat d'interactuar amb l'usuari administrador en el moment de l'inici del servei, doncs com hem comentat es fa a través del Apache. Així, la única manera de complir aquest requisit consisteix en desenvolupar un mòdul per Apache que ho faci. Aquesta tasca queda fora de l'abast del projecte, pel que es deixa com una possible millora per al futur. En el sistema actual, el servidor llegirà la paraula de pas directament de l'arxiu de configuració, tot i no ser una forma molt segura de fer-ho.

El cas anterior ens porta a un segon problema que es podria millorar en versions futures, el dels permisos de l'arxiu de configuració. Per a que el servidor pugui llegir l'arxiu de configuració, aquest ha de tenir permisos de lectura per a l'usuari que executa el servidor web Apache. Això te una implicació important en la seguretat, doncs qualsevol aplicació vulnerable que funcionés en el mateix servidor, donaria a un atacant la possibilitat de llegir la paraula de pass de la clau privada del Gestor. Un altre cop, la millora adequada correspon a la comentada en el cas anterior.

Un altra cosa a tenir en compte, és que en aplicacions d'aquest tipus és normal fer servir la part de client en una màquina i la part de servidor en un altra. Per tant, seria convenient en desenvolupaments futurs, distribuir tres paquets per a la instal·lació, un per al Pacient, un per al Metge i l'altre per al Gestor. Cada un d'aquests tindria un arxiu de configuració diferent amb les dades que necessita. No com en el sistema actual on hi estan les dades que fan servir les tres parts. Per exemple, el Pacient i el Metge no han de tenir les dades de connexió a la base de dades o la paraula de pas del Gestor. De la mateixa manera el Gestor no necessita la URL de connexió als serveis web.

Una millora en la seguretat força senzilla consisteix en configurar Apache amb SSL, i modificar els clients en conseqüència. Així, totes les transaccions criptogràfiques disposarien d'una capa de seguretat addicional. De la mateixa manera, fer servir SSL entre els webservices i la base de dades MySQL, donaria la possibilitat de repartir el servei en tres màquines (las segona amb Apache, la tercera amb MySQL) d'una manera segura,

Finalment, un aspecte en referència al identificador d'usuari. En els certificats es fa servir el DNI com a tal, es adir, un conjunt de nombres i una lletra. Per comoditat s'ha fet servir un identificador numèric a la base de dades, que s'ha pres com el DNI de l'usuari sense lletra. Això però, no és una bona opció per aplicacions reals, on ens podríem trobar que els identificadors d'usuari es repeteixen.

Capítol 9: Conclusions

Com a conclusió farem un repàs dels objectius proposats a l'inici del projecte.

El primer objectiu consistia en la implementació d'una PKI que ens permetés assolir els requisits de confidencialitat, autenticitat, integritat i no-repudi. Ho hem fet mitjançant la llibreria OpenSSL i la implementació dels protocols esmentats en els apartats 2.6, 2.7, 2.8. Aquesta PKI havia de ser utilitzada per Pacient, Metge i Gestor, pel que la implementació realitzada ofereix una interfície en llenguatge C, permetent un fàcil accés.

El segon objectiu consistia en desenvolupar les funcionalitats que havia d'oferir el Gestor com a serveis web, de manera que Pacient i Metge hi poguessin accedir. En aquest punt s'ha fet servir el servidor web Apache per a oferir els serveis web. Per a la seva implementació s'ha fet servir la llibreria gsoap. En el capítol 5 s'han estudiat els detalls.

El tercer objectiu era la creació d'un repositori central en el que emmagatzemar els historials mèdics i altres dades necessàries per al correcte funcionament del sistema. L'objectiu s'ha assolit mitjançant la utilització del Sistema Gestor de Bases de Dades MySQL. El disseny de les bases de dades l'hem vist al capítol 4.

El quart objectiu, que consistia en l'accés remot al servidor, es pot dividir en dues part. La primera correspon a la implementació de la part de client dels serveis web. Aquesta, corresponent al desenvolupament amb la llibreria gsoap, s'ha estudiat en detall al capítol 5. La segona part correspon a la representació de les dades que es transmeten i l'hem estudiat en el capítol 3. En aquest punt hem realitzat el disseny de les dades a transmetre basant-nos en el llenguatge XML. Per a poder treballar amb aquest llenguatge em fet servir la llibreria libxml.

El cinquè objectiu consistia en donar certes funcionalitats al sistema: consulta d'historials mèdics, consulta de llistat de pacients d'un metge i creació de noves visites mèdiques. Hem donat aquestes funcionalitats en base als protocols definits als punts 2.6, 2.7 i 2.8. Però l'assoliment d'aquest objectiu forma part de gairebé tot el projecte.

Els objectius sis i set consistien en el desenvolupament de la interfície gràfica per als pacients i els metges que fan servir el sistema. Aquest desenvolupament l'hem fet mitjançant la llibreria gràfica GTK+. En l'apartat 7 podem veure el seu disseny i la interfície obtinguda com a resultat.

Glossari

Aplicació: Eina que obtenim com a resultat del desenvolupament i que ens permetrà accedir a les funcionalitats del projecte.

Autoritat de Certificació: Emet certificats digitals de certes entitats de manera que aquestes es puguin identificar davant d'un tercer. Veure Certificat.

Base 64: Codificació que fa servir 6 bits per caràcter i que es caracteritza per emprar únicament caràcters imprimibles.

Base de dades: Component d'un Sistema Gestor de Bases de Dades que correspon a les dades en si mateixes, aquestes, de tipus persistent.

Certificat: Arxiu que conté les dades que donen fe de l'autenticitat de la persona o entitat que el presenta.

Clau Pública: En criptografia de clau pública o asimètrica, la clau pública correspon a certa informació de caràcter públic que permet xifrar missatges o verificar les signatures fetes per una clau privada. Veure Criptografia de Clau Pública.

Clau Privada: En criptografia de clau pública o asimètrica, la clau privada correspon a certa informació de caràcter privat que permet desxifrar missatges o signarlos. Veure Criptografia de Clau Pública.

Criptografia de Clau Pública: La criptografia de clau pública o asimètrica és aquella que utilitza parells de claus (clau pública / clau privada) de manera que, tot missatge xifrat amb la clau pública només pot ser desxifrat amb la seva clau privada corresponent. Igualment, tot missatge signat per la clau privada només podrà ser verificat amb la clau pública corresponent. En aquest tipus de criptografia les claus públiques poden estar a l'abast de tothom, mentre que les privades, romandran ocultes.

Interfície gràfica: Component visual que permet la interacció de l'usuari amb la máquina.

Hash: Resum que dona lloc a una seqüència de longitud fixa a partir d'unes dades sense importar la longitud d'aquestes. Les seves propietats permeten utilitzar-la per a verificar la integritat de les dades.

MySQL: Programari lliure que implementa un Sistema Gestor de Bases de Dades.

PKCS: Sigles de Public Key Cryptography Standards i són el conjunt d'estàndards definits per a la utilització dels sistemes de clau pública.

PKI: Sigles de Public Key Infraestrucure i Infraestructura de Clau Pública.

Signatura: En criptografia de clau pública és el resultat que s'obté al aplicar la clau privada sobre un document. Només és pot verificar amb la seva clau pública.

Xifrat: En criptografia de clau pública és el resultat que s'obté al aplicar la clau pública sobre un document. Només és pot desxifrar amb la seva clau privada.

Bibliografia

- Network Security with OpenSSL, Ed O'Reilly. John Viega, Matt Messier & Pravir Chandra.
- Secure Programming Cookbook, Ed O'Reilly. John Viega & Matt Messier.
- Desarrollo de aplicaciones Linux con GTK+ y GDK, Ed Prentice Hall. Eric Harlow.

- OpenSSL, The Open Source toolkit for SSL/TLS – http://openssl.org
- LibXML, The XML Library for GNOME – http://www.xmlsoft.org/
- GSoap, The SOAP C/C++ Web Services – http://gsoap2.sourceforge.net/
- Apache, HTTP Server Project - http://httpd.apache.org/
- GTK+, The Gimp Tookkit - http://www.gtk.org/

Annexos

Annex A: Instal·lació del sistema

La instal·lació que es descriu en els següents apartats considera que es fa servir una distribució Ubuntu Linux 6,06 LTS. Per altres versions de Ubuntu Linux el procés és el mateix, tot i que hi poden haver diferencies en les versions dels paquets. Per altres distribucions de Linux, la documentació és fàcilment adaptable.

Existeixen dos tipus d'instal·lació: La instal·lació a nivell d'usuari correspon al programari necessari per a fer servir l'aplicació i parteix dels binaris ja compilats. La instal·lació per a compilar les fonts és la que permet construir les aplicacions des de zero.

A.1: Instal·lació del sistema a nivell d'usuari

Per a l'emmagatzematge de dades necessitem disposar d'una base de dades. Instal·lem MySQL amb la següent comanda:

```
$ sudo aptitude install mysql-server
$ sudo aptitude install libmysqlclient-dev
$ sudo /etc/init.d/mysql start
```

Per a poder fer servir els webservices haurem d'instal·lar e iniciar el servidor web apache.

```
$ sudo aptitude install apache2
$ sudo /etc/init.d/apache2 start
```

Instal·larem el paquet corresponent al projecte:

```
$ sudo dpkg -i gestor-historials_yyymmdd-x_i386.deb
```

Un cop tot està instal·lat al sistema necessitarem crear una base de dades i inserir-hi dades d'exemple. Ho farem amb les següents comandes:

```
$ sudo mysqladmin create gestio_historials
$ cat /usr/share/doc/gestor-historials/sources/gh.sql | sudo
mysql gestio_historials
```

Per a finalitzar la instal·lació s'ha d'establir l'usuari i la paraula de pas de MySQL a l'arxiu de configuració /etc/ghconf.xml. D'altra banda no funcionarà. En la instal·lació per defecte de MySQL s'estableix l'usuari root sense paraula de pas. Tot i que es pot fer servir, és recomanable posar un usuari diferent. Per a fer-ho, es poden seguir els passos indicats a la documentació de MySQL.
En l'arxiu /etc/ghconf.xml apareix per defecte l'usuari “uoc” amb paraula de pas “uoc”. Si no es vol afegir un nou usuari a MySQL es pot modificar /etc/ghconf.xml per a que tingui com a usuari “root” i paraula de pas “” (sense contingut).

Si desitgem fer servir les aplicacions Pacient/Metge amb un usuari diferent de root, el que és força recomanable, s'hauran de modificar els permisos de /etc/ghconf.xml. En aquest cas, s'ha de tenir en compte que el servidor web Apache també hi ha de poder accedir.

A.2: Instal·lació del sistema per a compilar les fonts

Per a poder compilar les fonts amb èxit és necessari realitzar tots els passos indicats en l'annex A.1.

A part d'aquests necessitem la llibreria SOAP que s'ha utilitzat en el desenvolupament:

```
$ sudo aptitude install gsoap
```

Un cop s'ha instal·lat tot amb èxit entrarem al directori del projecte, on veurem un arxiu Makefile. Aquest ens permet les següents opcions:

Generar la documentació:

```
$ make doxygen
```

Compilar les fonts:

```
$ make
```

Instal·lar el programa (aquest és millor no utilitzar-lo, doncs disposem del paquet .deb):

```
$ sudo make install
```

Generar el paquet .deb:

```
$ make pkg
```

Per a instal·lar el paquet .deb podem executar la següent comanda:

```
$ sudo dpkg -i gestor-historials_yyymmdd-x_i386.deb
```

Per a eliminar-lo:

```
$ sudo dpkg -r gestor-historials
```

Annex B: Joc de proves

Per a poder realitzar proves és necessari disposar d'un conjunt de certificats y claus. Aquests són:

- El certificat del Gestor del sistema y la seva clau privada.
- El certificat de cada Metge y la seva clau privada corresponent.
- El certificat de cada Pacient y la seva clau privada corresponent.

Amb el paquet del projecte s'instal·la un conjunt de certificats y claus de proves al directori "/usr/share/doc/gestor-historials/keys/" amb la seva corresponent Autoritat de Certificació. Per a facilitar el seu ús, les claus estan protegides amb la paraula de pas "uoc0506".

Després d'instal·lar el sistema tal i com s'explica en l'annex A es disposa del Gestor y de un Metge y Pacient de proves donats d'alta al sistema. També s'ha afegit un petit historial mèdic.

B.1: Prova 1: Autenticació

Cada cop que un usuari vol realitzar una acció s'autentica. Aquesta operació la realitzen de la mateixa manera el Pacient y el Metge. Per a les proves farem servir l'aplicació Metge.

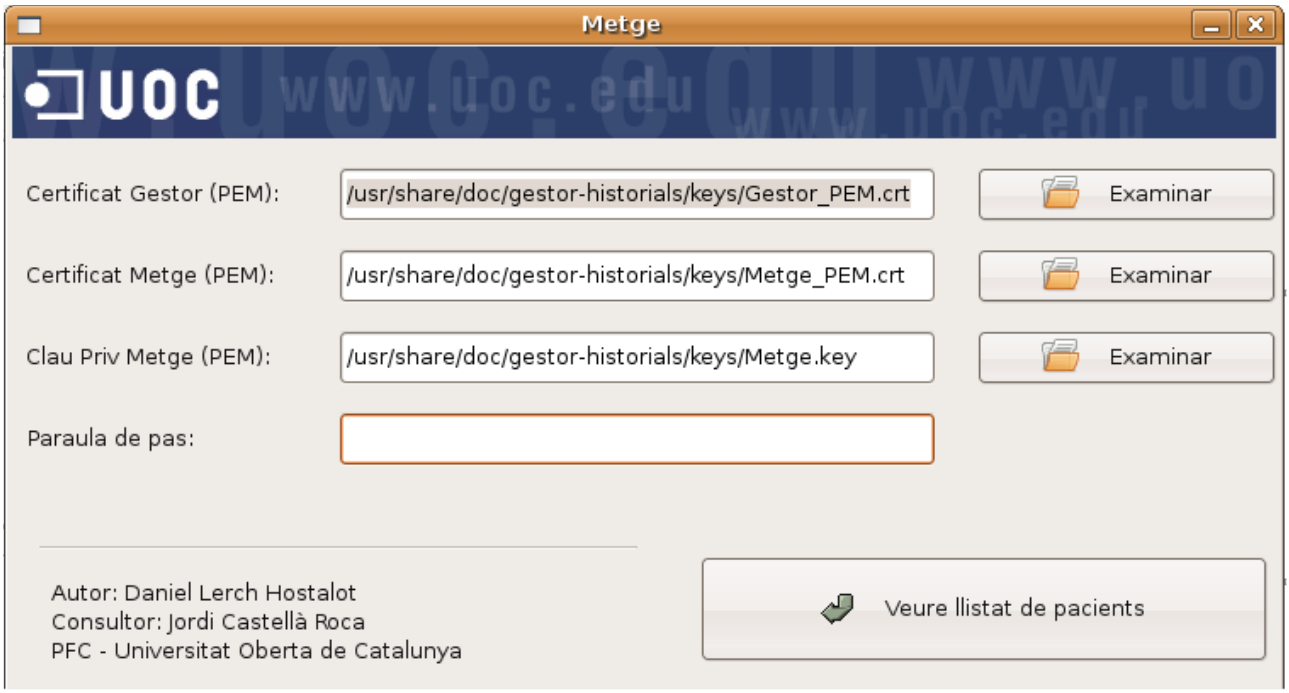
Iniciem l'aplicació Metge amb la comanda:

```
$ metge
```

Recordem que és necessari disposar dels suficients privilegis com per a llegir "/etc/ghconf.xml". En cas contrari, tot i que no és recomanable, podem fer les proves amb:

```
$ sudo metge
```

A continuació l'aplicació ens mostra la interfície gràfica. Aquesta ens demana les claus y certificats necessaris:



Introduïm les dades que ens demana junt amb la paraula de pas de la clau privada i fem clic en el botó "Veure llistat de pacients". Si les dades són correctes podrem accedir al llistat de pacients, però si l'autenticació no és correcta obtindrem un error.

B.2: Prova 2: Un Pacient consulta el seu historial

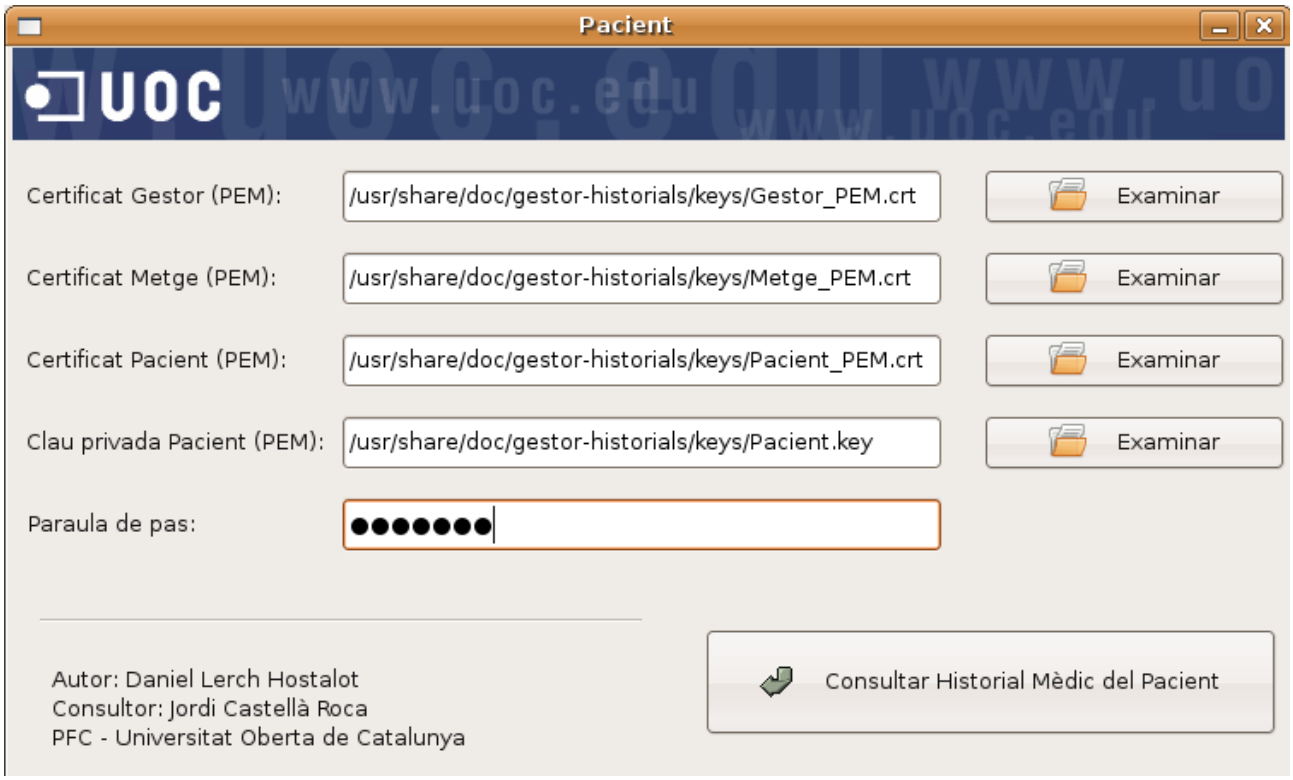
De manera similar al cas anterior un Pacient pot consultar el seu historial. Per afer-ho haurem de llençar l'aplicació Pacient amb la comanda:

```
$ pacient
```

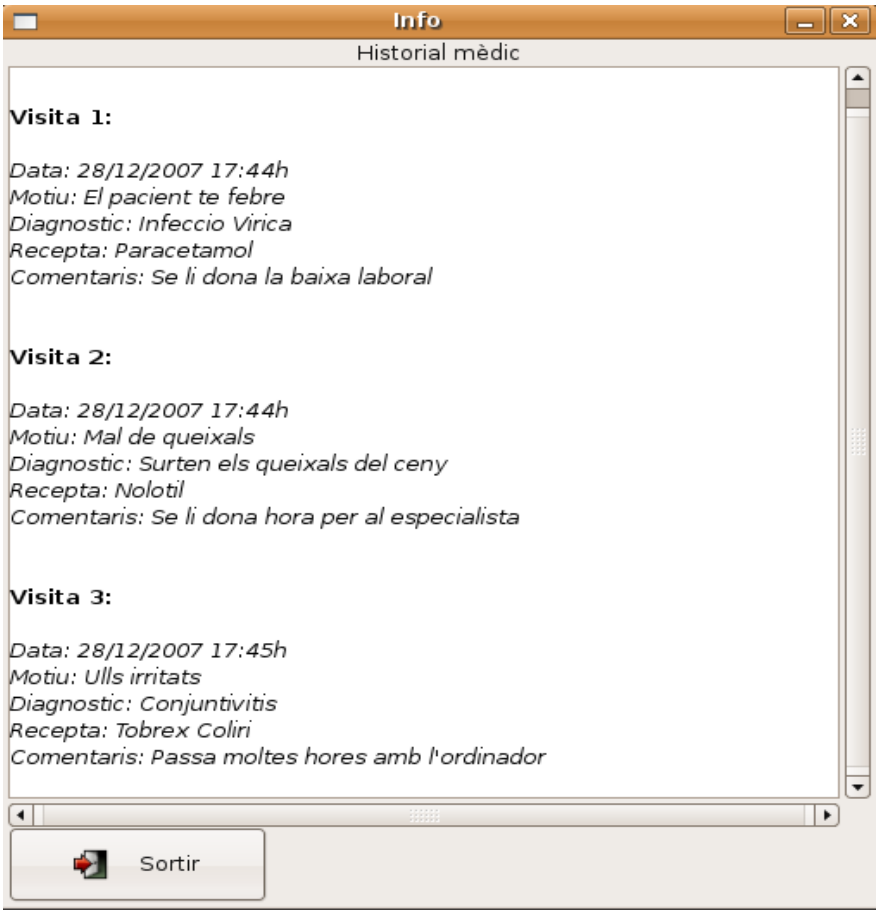
Recordem que és necessari disposar dels suficients privilegis com per a llegir "/etc/ghconf.xml". En cas contrari, tot i que no és recomanable, podem fer les proves amb:

```
$ sudo pacient
```

A continuació l'aplicació ens mostra la interfície gràfica. Aquesta ens demana les claus y certificats necessaris:

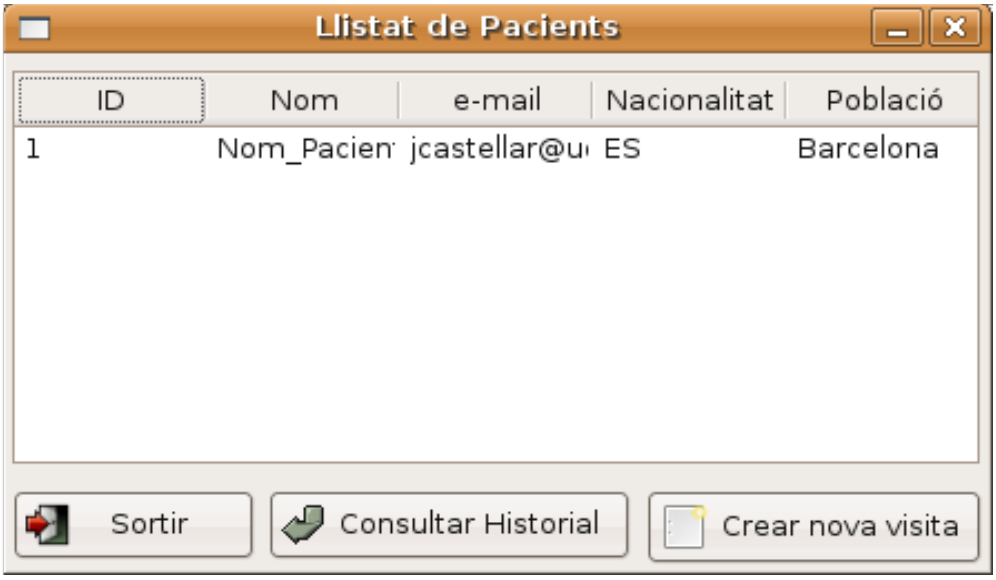


Un cop hem proporcionat les dades que ens demana, podem fer clic al botó “Consultar Historial Mèdic del Pacient”. Si l'autenticació és correcta, obtindrem l'historial, com es mostra a continuació:



B.3: Prova 3: Un Metge consulta la llista de pacients assignats

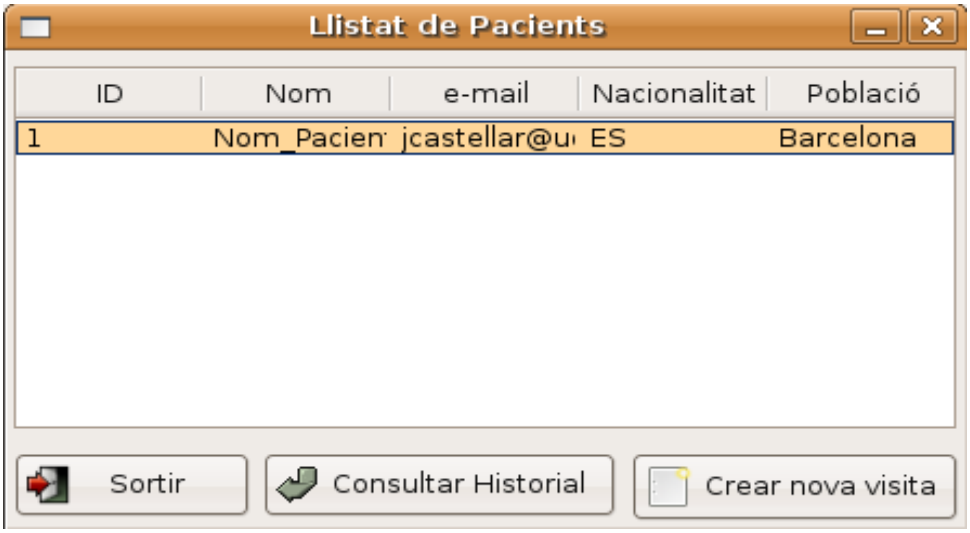
Un cop superada l'autenticació tal i com s'explica a l'annexe B.1 el Metge obté la llista lels seus pacients assignats. El resultat és el següent:

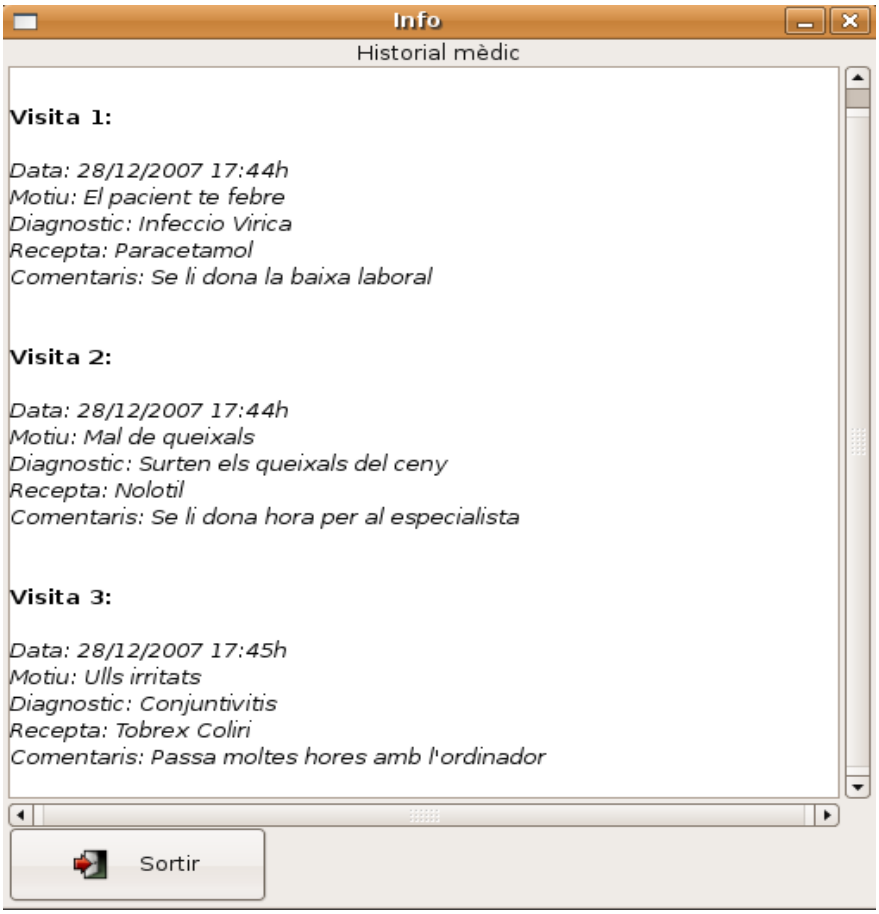


Aquesta llista mostra tots els pacients (en aquest cas només n'hi ha un) amb dades extretes del seu certificat.
En aquest punt es pot seleccionar un Pacient i consultar el seu historial o afegir una nova visita.

B.4: Prova 4: Un Metge consulta l'historial d'un pacient

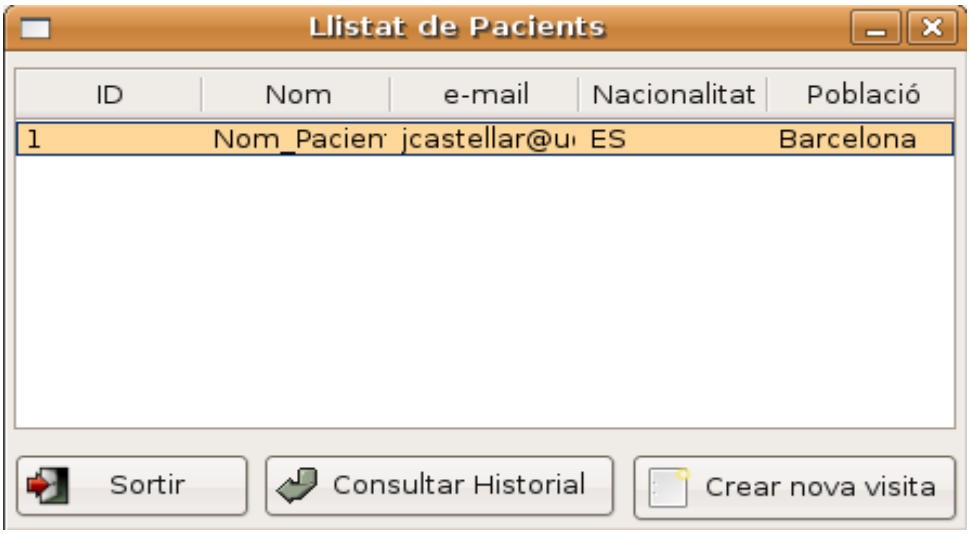
Partint del punt anterior podem seleccionar un pacient i consultar el seu historial mèdic fent clic al botó “Consultar Historial”:



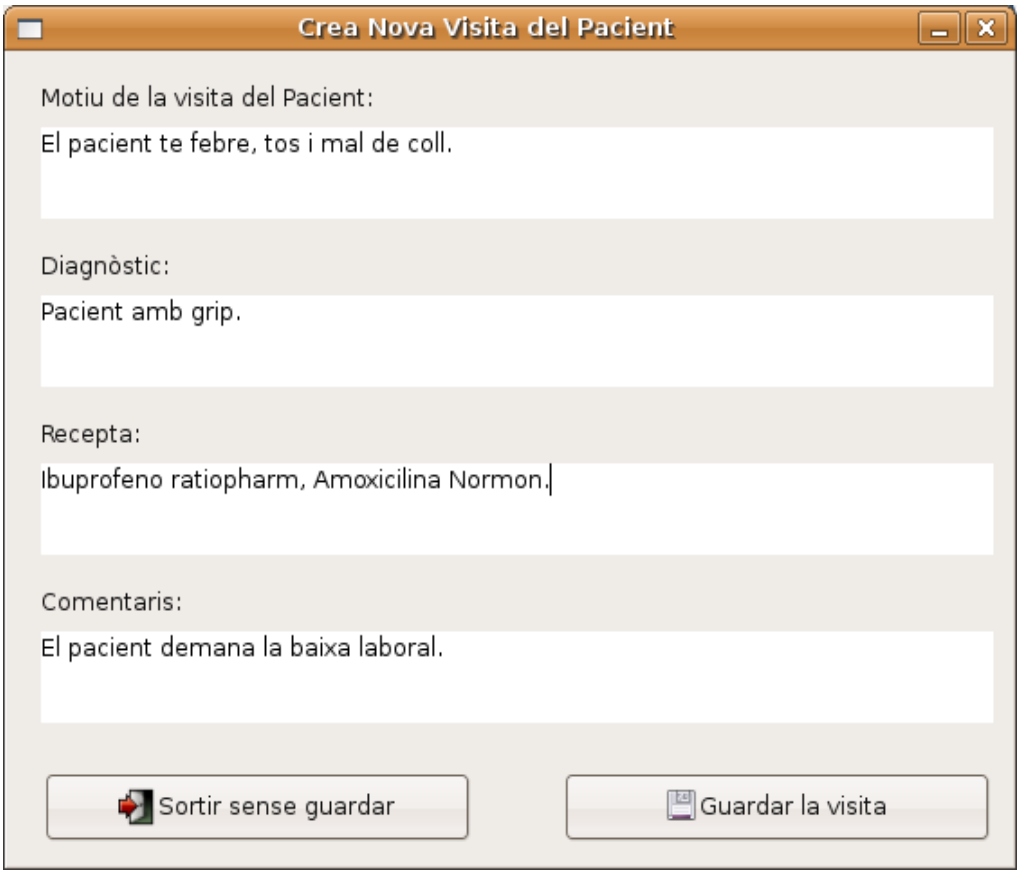


B.5: Prova 5: Un Metge afegeix una visita a l'historial d'un pacient

Finalment afegirem una visita a l'historial.
Igual que en el cas anterior seleccionarem el Pacient, però en aquest cas farem clic al botó “Crear Nova Visita”.



Apareixerà un formulari que haurem d'omplir.



Si en aquest punt tornem a consultar l'historial mèdic del pacient, veurem que s'ha afegit una nova visita.