Esquema criptogràfic

Història Mèdica Electrònica remota i segura

Iñaki Romero López
Enginyeria en Informàtica

Jordi Castellà Roca

Juny 2006

Dedicatòria i Agraïments.

A la meva dona Marina, per recolzar-me i comprendre'm,
A la meva petitona Itziar, per alegrar-me cada dia,
Als meus pares, per ajudar-me en els moments més durs,
A David Muñoz, per mostrar interès
i no vull oblidar-me de totes les persones presents sempre en els meus records.

A tots sincerament, gràcies de tot cor.

Resum.

La inversió en sistemes informàtics i l'ús de les TIC per part de les organitzacions en l'actualitat, és una realitat que va a més. La informació es guarda electrònicament i es distribueix per les xarxes de comunicacions, que ens permeten accedir a un gran volum d'informació ràpidament.

És el cas de la Sanitat pública que organitza les visites dels pacients en format electrònic, guardant aquesta informació en bases de dades, de forma que aquestes són accessibles remotament gràcies a les xarxes de comunicacions, enllaçant als centres, que es troben distribuïts en diferents punts de la geografia.

L'historial mèdic d'un pacient és informació de gran valor que en termes de seguretat està catalogada amb el màxim nivell de seguretat a oferir, per tant s'han de garantir certs aspectes relacionats amb la seguretat.

Ningú a part del pacient o un metge ha de poder veure les dades d'un l'historial. D'altra banda, ningú ha de poder suplantar la identitat en el nostre sistema, fent-se passar per una altre persona. En el cas que un usuari maliciós pogués accedir a la base de dades, s'ha de garantir que aquest no podrà modificar la informació o si més no, detectar que s'ha efectuat una modificació no permesa i per finalitzar cap professional no es podrà desdir del que ha pogut observar anteriorment en un historial mèdic.

Aquest projecte fi de carrera es centra en els mecanismes de seguretat que podem implementar per poder garantir els requisits esmentats anteriorment.

En aquest cas s'implementa una PKI (Infraestructura de Clau Pública), de forma que la informació sensible que hem de protegir estarà xifrada i signada amb certificats digitals. És defineixen les accions i protocols que implementa el sistema per poder assegurar que tenim un producte que accedeix a l'historial mèdic remotament i de forma segura.

Índex de continguts.

Índex d'il·lustracions.

1. Introducció.

1.1. Justificació del PFC i context en el que es desenvolupa.

Les xarxes de comunicacions connecten màquines que es troben situades al voltant de la geografia, permetent-les intercanviar informació de forma ràpida, entre altres coses. Per posar un exemple, l'ús d'Internet ens permet connectar a l'oficina virtual del nostre banc.

Si desitgem realitzar una transferència, l'usuari ha d'estar segur que està treballant amb la seva caixa, doncs amb tècniques de fishing, l'han pogut prendre l'identitat.

Paral·lelament el banc ha d'estar segur de qui és l'usuari, doncs aquest podrà fer una transferència de diners entre comptes.

La utilització de tècniques criptogràfiques ens permeten realitzar totes aquestes operacions de forma segura, garantint que es compleixen els requisits de seguretat sol·licitats.

Un dels casos en que la seguretat és molt important, és en aquelles situacions en que tractem amb dades sanitàries, doncs aquestes per llei pertanyen al màxim nivell de seguretat que s'ha de garantir.

És el cas dels hospitals, centres d'atenció primària, mútues, etc. Aquestes entitats han de garantir la confidencialitat i autenticitat de les dades

Quan un pacient sol·licita un servei, aquest es registrat electrònicament a través d'un programa informàtic. L'usuari primer s'ha de validar en el sistema, si no és així no podrà registrar la petició del pacient. Tota la informació es guarda en una base de dades, de forma que podem consultar les peticions realitzades, posteriorment en el temps.

1.2. Objectius del PFC.

L'objectiu d'aquest PFC és el d'aprofundir en els aspectes relacionats amb la seguretat per poder obtenir un producte segur. La utilització de la criptografia ens ajudarà en aquestes tasques.

S'utilitzarà criptografia de clau pública, juntament amb els certificats digitals dels usuaris del nostre sistema.

Aquest haurà de seguir el model client - servidor. Per un costat tindrem l'aplicatiu client que haurà de ser segur i per un altra banda l'aplicatiu servidor. Aquest s'encarregarà de la gestió de les dades persistents, manipulant-les de forma segura i permetrà peticions al client de forma remota.

Client i servidor hauran d'entendre's en els diàlegs, per això necessitarem definir els protocols que han de seguir, per cada acció a executar.

L'intercanvii de informació entre client i servidor també serà segur, aquestos intercanviaran dades en forma de document XML que viatjarà xifrat i/o signat.

Aquest producte segur haurà de ser persistent, és a dir, ha d'emmagatzemar les dades en un suport que permeti consultar-les independentment del moment en que ho fem.

Serà necessari facilitar la feina als futurs usuaris, desenvolupant interfícies gràfiques.

1.3. Enfocament i mètode seguit.

El desenvolupament del present projecte fi de carrera s'ha descompassat en fases. Aquestes ens permetran realitzar una implementació incremental, és a dir, s'implementa el codi, les proves i la documentació de la fase en qüestió, de forma que al passar a la fase següent, partim de la fase anterior.

S'utilitza la filosofia de test unitari, és a dir, la següent fase s'integrarà amb l'anterior i es provarà.

La implementació es divideix en 7 fases:

1) Estudi de les necessitats del sistema.
2) Disseny de l'esquema criptogràfic.
3) Representació de les dades: XML.
4) Comunicació dels components: RMI.
5) Gestió de la informació: BD.
6) Interfícies: Usuari - Gestor.
7) Documentació.

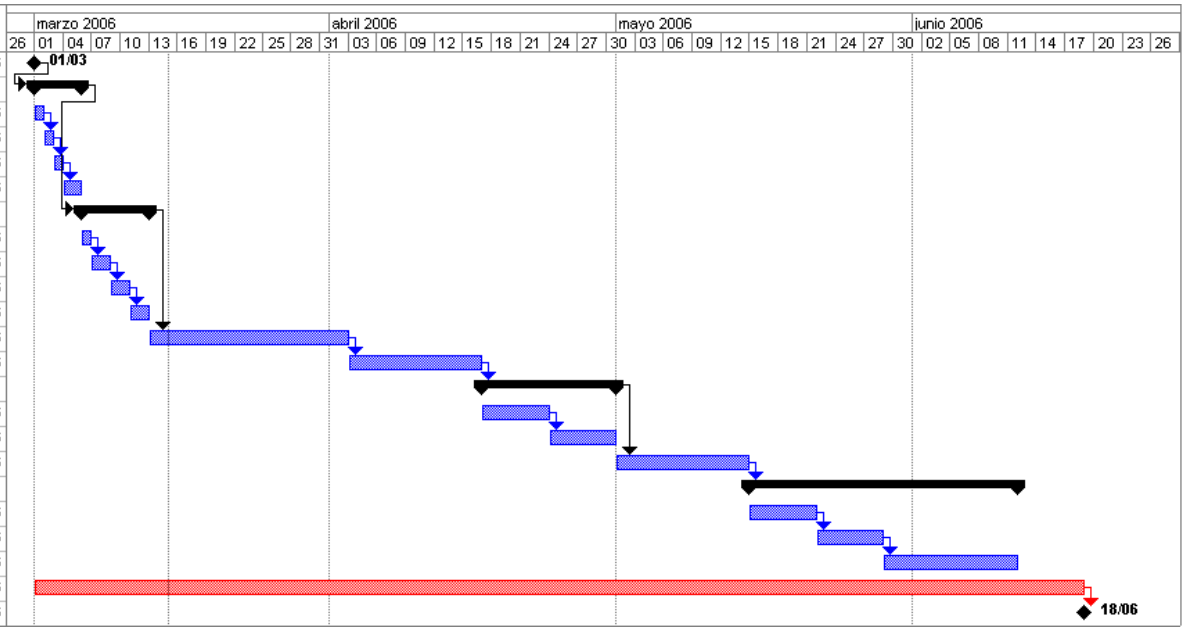
1.4. Planificació del projecte.

Es realitza una estimació del cost, en temps, que pot tenir cada fase, en funció del temps total del que disposem per realitzar aquest Projecte Fi de Carrera.

A partir d'aquesta estimació, realitzem una planificació per poder dur a terme el desenvolupament en els terminis marcats i evitar sorpreses.

La planificació obtinguda és la següent:

Id		Nombre de tarea	Duración	Comienzo	Fin
1		Inici PFC	0 días	mié 01/03/06	mié 01/03/06
2		**Instal·lar programari**	**5 días**	**mié 01/03/06**	**dom 05/03/06**
3		Instal·lar Java	1 día	mié 01/03/06	mié 01/03/06
4		Instal·lar IAIK	1 día	jue 02/03/06	jue 02/03/06
5		Instal·lar Eclipse	1 día	vie 03/03/06	vie 03/03/06
6		Instal·lar MySQL	2 días	sáb 04/03/06	dom 05/03/06
7		**Estudi necessitats del sis**	**7 días**	**lun 06/03/06**	**dom 12/03/06**
8		Actors	1 día	lun 06/03/06	lun 06/03/06
9		Accions/Serveis	2 días	mar 07/03/06	mié 08/03/06
10		Gestió de la informació	2 días	jue 09/03/06	vie 10/03/06
11		Requisits de seguretat	2 días	sáb 11/03/06	dom 12/03/06
12		Esquema Criptogràfic	21 días	lun 13/03/06	dom 02/04/06
13		XML	14 días	lun 03/04/06	dom 16/04/06
14		**RMI**	**14 días**	**lun 17/04/06**	**dom 30/04/06**
15		Servidor	7 días	lun 17/04/06	dom 23/04/06
16		Client	7 días	lun 24/04/06	dom 30/04/06
17		Base de Dades	14 días	lun 01/05/06	dom 14/05/06
18		**Interficies**	**28 días**	**lun 15/05/06**	**dom 11/06/06**
19		pacient	7 días	lun 15/05/06	dom 21/05/06
20		metge	7 días	lun 22/05/06	dom 28/05/06
21		gestor	14 días	lun 29/05/06	dom 11/06/06
22		Documentacio	110 días	mié 01/03/06	dom 18/06/06
23		Fi PFC	0 días	dom 18/06/06	dom 18/06/06



Il·lustració 1.- Planificació proposada.

1.5. Productes Obtinguts.

El sistema complert, consta dels següents productes:

Una PKI amb els certificats necessaris per que un pacient, un metge i un gestor puguin provar el sistema. Es podran afegir nous certificats a la PKI, doncs aquesta inclou l'entitat certificadora necessària.

Dues aplicacions informàtiques:

Un aplicació gràfica per l'usuari del sistema que podrà ser un pacient o un metge i que serà utilitzada per realitzar diferents peticions al servidor.

Una aplicació gràfica pel gestor que permetrà aquest controlar l'estat d'aquest aturant-lo, engegant-lo, etc.

Aquestes aplicacions invoquen crides a classes específiques que s'han desenvolupat d'acord amb les necessitats del projecte. S'obté un directori on resideixen totes les classes, incloses les aplicacions client i servidor.

El mateix directori l'obtenim per les classes compilades, algunes d'aquestes són les que executarem posteriorment.

Una base de dades MySQL que guardarà informació diversa d'utilitat pel nostre sistema, entre elles les visites realitzades als pacients, d'aquesta forma podrem aconseguir l'historia mèdica de qualsevol persona. Les sentències de creació de la base de dades es guarden en un arxiu (script) ubicat en un directori específic.

Un arxiu de definició de tipus de document XML (DTD) que es segueix en les construccions dels diferents documents XML que s'intercanvien client i servidor, i, també està ubicat en un directori específic.

1.6. Breu descripció dels altres capítols de la memòria.

1.6.1. Estudi de les necessitats del sistema.

La primera tasca que es realitza és un estudi de les necessitats del nostre sistema. Aquí s'identifiquen els actors que intervenen en el sistema, les accions o serveis que realitzen aquestos, quina informació s'ha de gestionar i finalment els requisits de seguretat a complir.

1.6.2. Disseny de l'esquema criptogràfic.

En el capítol anterior obtenim la llista de serveis que el nostre sistema ha d'oferir. Per cadascun d'aquestos es dissenya un protocol que serà utilitzat pels actors dels sistema i obtenim un esquema criptogràfic.

1.6.3. Representació de les dades: XML.

En aquest capítol es descriu el format de la informació que serà intercanviada entre els usuaris del sistema.

1.6.4. Comunicació dels components: RMI.

Els actors que intervindran en el sistema hauran d'intercanviar-se informació remotament, en aquest capítol s'explica la tècnica utilitzada.

1.6.5. Gestió de la informació: BD.

La informació rellevant es guarda de forma permanent en una Base de Dades. Aquest capítol mostra des de l'origen de les dades fins l'elecció del motor apropiat.

1.6.6. Interfícies: Usuari - Gestor.

La representació visual dels programes informàtics obtinguts, s'explica en aquest capítol, mostrant els elements que els composen.

1.6.7. Conclusions.

Es fa una reflexió, remarcant els aspectes més important del projecte fi de carrera que estem abordant.

1.6.8. Bibliografia.

Es mostra un llistat de les fonts que ha utilitzat l'autor per poder desenvolupar tant tècnica com teòricament aquest projecte.

1.6.9. Instal·lació.

Aquest capítol es fa eco de les accions que s'han d'efectuar, per poder tenir totes les eines necessàries per la implementació d'aquest projecte fi de carrera.

1.6.10. Demostració.

Per finalitzar es fa una petita demostració, que il·lustra com utilitzar els programes informàtics obtinguts.

2. Estudi de les necessitats del sistema.

2.1. Actors.

L'historial mèdic és una peça fonamental en el mon de la sanitat. Resulta molt difícil imaginar com treballarien els professionals sense aquesta informació, doncs aquesta serà utilitzada per poder seguir l'evolució d'un pacient, consultar dades d'interès com pot ser el grup sanguini, al·lèrgies, infermetats, vacunes, etc. A part d'aquest tipus de professionals, quan anem a un centre de salut podem trobar-nos amb altres i cal estudiar si existeix un vincle entre aquestos i els historials.

Diferents motius poden fer que anem a un centre de salut a demanar hora pel nostre metge de família. Ens trobem amb una persona o persones en un mostrador, aquestes que denominem 'administratius', demanen la tarja sanitària i accedeixen al programa informàtic per poder programar la visita. Aquest tipus d'actors no necessiten accedir a cap tipus d'informació sanitària del pacient com pot ser l'historial mèdic.

El dia que ens toca tornem al centre a la hora concertada i un cop arribat el nostre torn passem a una consulta on atenen dues persones, la 'infermera' i el 'metge' que formen el que es coneix com a UBA (Unitat Bàsica d'Atenció). La finalitat de la infermera és la d'ajudar al metge en les seves tasques i alliberar-lo de càrrega de treball. De forma similar a la dels auxiliars, les infermeres no han d'accedir a l'historial doncs aquesta feina és exclusiva dels facultatius, en aquest cas els metges.

Malgrat que els professionals mencionats fins el moment, són els més coneguts, el ventall és molt ampli, zeladors, ordenances, personal de manteniment, etc. Com s'ha dit anteriorment l'accés a aquesta informació és exclusiva pels facultatius.

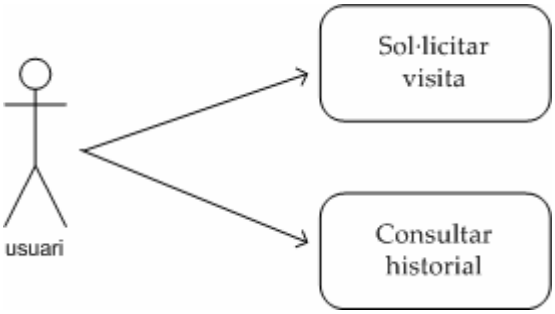
El sistema serà utilitzat pels metges i els pacients, no obstant això aquestos últims no han de ser els encarregats de gestionar-lo. Aquesta tasca la durà a terme el 'Gestor del sistema'.

Resumint, en el nostre sistema per gestionar de forma segura els historials intervindran tres actors, el pacient, el metge i el gestor del sistema.

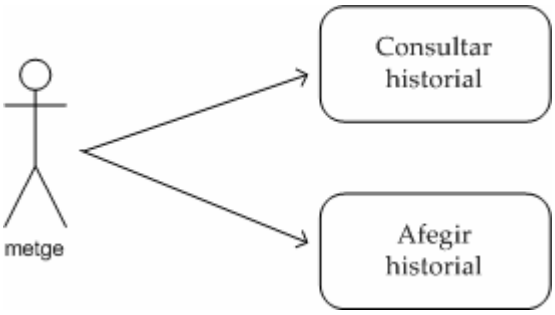
2.2. Serveis.

Els actors interactuen amb el sistema executant diferent serveis com poden ser, demanar visita, consultar dades de l'historial, afegir nova informació, etc. Si volem identificar els serveis que ha d'oferir el nostre sistema cal estudiar quines tasques podrà efectuar cada actor.

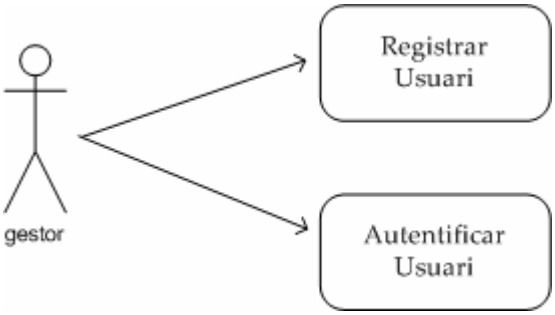
Il·lustrem les tasques de cada actor, amb els següents cassos d'ús:



Il·lustració 2.- Cas d'ús usuari



Il·lustració 3.- Cas d'ús metge



Il·lustració 4.- Cas d'ús gestor

Descripció:

- **Sol·licitar visita.**
  Quan un usuari desitja ser ates pel seu metge, procedeix a demanar visita en el seu centre de salut, be, sigui trucant per telèfon, presencialment, etc. S'identifica proporcionant informació personal, com pot ser data de naixement i sexe, tarja sanitària, nom i cognoms, etc. El resultat d'executar aquest servei és que l'usuari te visita programada per anar a un centre en un dia i hora concret per un metge determinat.

- **Consultar historial.**
  Els metges atenen els seus pacients, escolten els motius de la seva visita, poden explorar-lo, derivar-lo a un especialista, etc. Per poder prendre una decisió és possible que necessiti consultar informació de l'historial del pacient, com per exemple, saber si aquest és diabètic a l'hora de receptar un fàrmac. El resultat d'executar aquest servei és l'obtenció de informació clínica del pacient per part del metge.

- **Afegir historial.**
  El metge després d'escoltar al pacient, realitza una sèrie d'accions per determinar quina és la solució al requeriment manifestat pel pacient. Un cop pren una determinació ho fa constar en l'historial afegint aquesta informació. Després d'executar aquest servei, l'historial mèdic del pacient tindrà una nova línia amb informació.

- **Registrar usuari.**
  Com s'ha citat anteriorment, l'usuari demana cita, el metge consulta i/o afegeix informació, etc. El sistema haurà de reconeix aquestes persones per poder oferir els serveis anteriors i haurà de registrar als usuaris del sistema. El resultat d'executar aquest servei sobre un usuari o un metge, és l'addició d'informació en la base de dades, per poder identificar-lo en el nostre sistema.

- **Autentificar usuari.**
  Un cop tenim els usuaris del sistema registrats, aquestos s'han de poder autenticar, és a dir, han de poder demostrar que són qui realment diuen ser. El resultat d'executar

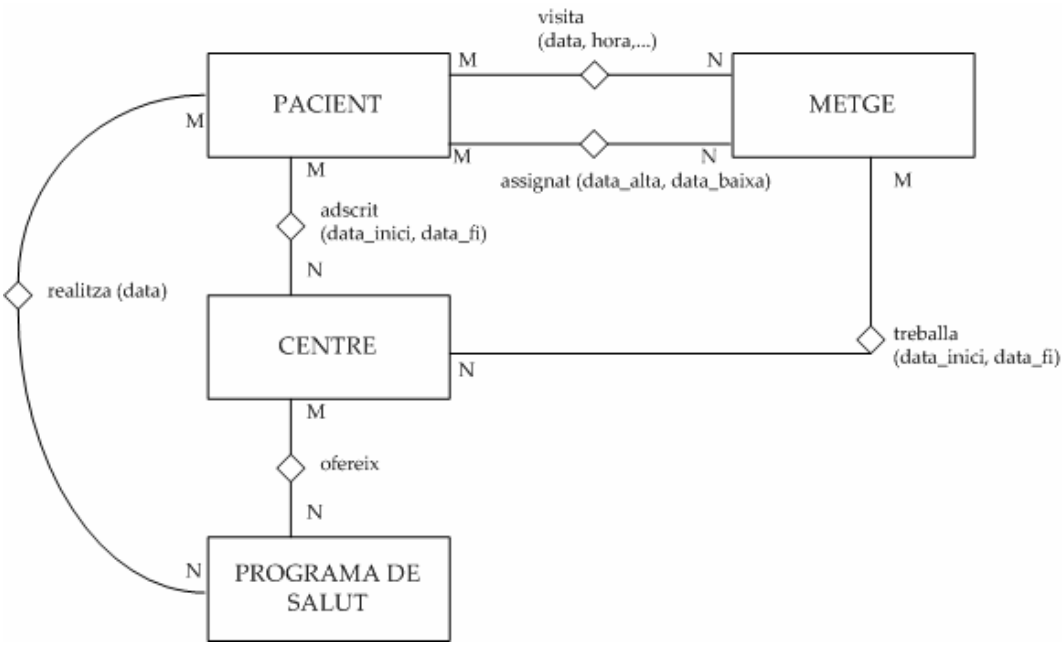
aquest servei sobre un usuari, és la demostració que aquesta persona és qui realment diu ser.

2.3. Gestió de la informació.

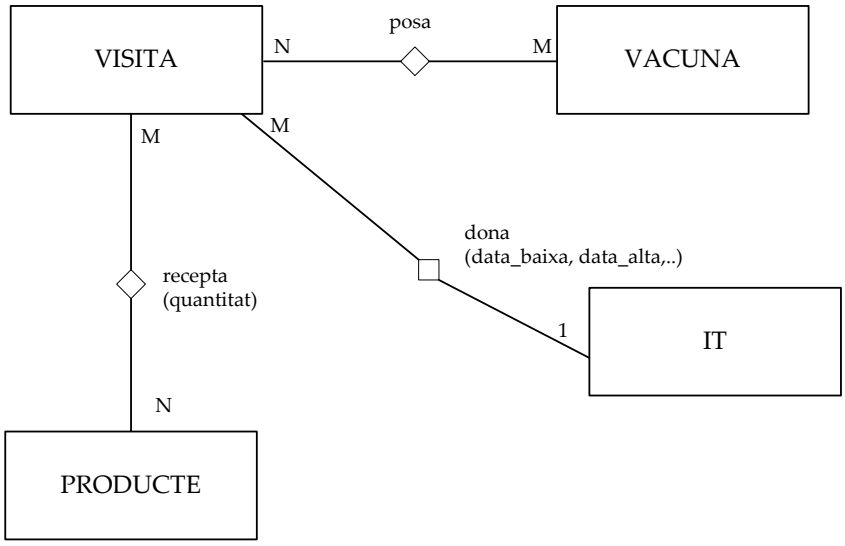
La quantitat d'informació que pot contenir un historial mèdic pot ser immensa, doncs amb aquest podem veure les visites, receptes, vacunes, ITs (Incapacitació Temporal), programes de salut que s'han aplicat, etc.

L'àmbit en el que aplicarem el nostre sistema, també pot fer variar la informació necessària. Mentre en un hospital és indispensable conèixer el grup sanguini d'un pacient, doncs pot necessitar una transfusió, en un centre d'atenció primària (CAP) no te tanta importància doncs aquesta pràctica no es du a terme. Malgrat això intentarem donar un marc comú de treball per qualsevol àmbit.

Il·lustrem l'estructuració de la informació amb el següent disseny conceptual:



Il·lustració 5.- Diagrama Entitat - Relació de Chen, primera part.



Il·lustració 6.- Diagrama Entitat - Relació de Chen, segona part.

Analitzem la informació que ha de gestionar cada entitat:

- **PACIENT**
  Aquesta entitat representa al pacient que sol·licita els serveis de salut. Aquesta guardarà les dades personals de l'usuari, tal com, nom, cognoms, adreça, etc.

- **METGE**
  Aquesta entitat representa el col·lectiu de metges que formen part del nostre sistema. La informació que s'emmagatzemarà en aquesta contindrà, a part de les dades personals, contindrà informació especifica del facultatiu, com poden ser, el número de col·legiat i especialitat.

- **CENTRE**
  En aquesta entitat es guarda informació sobre els centres de salut. S'identifica amb un codi de centre i contindrà informació descriptiva de cada centre.

- **ASSIGNAT**
  Un pacient pot ser visitat per molts metges depenent de l'especialitat d'aquestos. Un exemple quotidià és quan anem al metge de capçalera i aquest ens deriva a l'especialista, per exemple el oftalmòleg. De la mateixa forma, un metge té assignats diferents pacients. La relació *ASSIGNAT* neix, fruit de relacionat aquestes entitats (*PACIENT i METGE*) . La cardinalitat d'aquesta relació és de N:M.

- **PRODUCTE**
  L'entitat *PRODUCTE* representa els diferents productes que formen part del catàleg de productes i, per tant, un *METGE* pot receptar. Aquesta entitat emmagatzemarà els atributs que descriuen les característiques de cada *PRODUCTE*.

- **ADSCRIT**
  Es relacionen les entitats *PACIENT* i *CENTRE*, doncs un *PACIENT* està *ADSCRIT* a un *CENTRE* de salut durant un període de temps comprés entre data_inici i data_fi. Existeixen diferents motius pels que un *PACIENT* pot canviar de *CENTRE*, per canvi de residència, ho pot demanar explícitament, etc.

- **TREBALLA**
  El metge treballa en un *CENTRE* durant un període temps. Aquesta entitat contindrà aquesta informació, per això, guardarà el primer dia que va començar a treballar (data_inici) i el darrer (data_fi).

- **VISITA**
  La *VISITA* és precisament el nexe entre *PACIENT* i *METGE*. Aquestes visites les anomenem visites MEAP, doncs es guardarà el Motiu, Exploració, Avaluació i Plans a seguir. També guardarà el CENTRE on s'ha fet la visita, doncs un *PACIENT* es pot visitar en el seu propi centre i també en altres. Altra informació sobre el tipus de visita, si ha estat espontània, amb cita prèvia, programada pel metge, etc., quin servei va sol·licitar (medicina general, pediatria, odontologia, etc.) i per finalitzar el motiu.

- **PROGRAMA DE SALUT**
  Aquesta entitat representa la col·lecció de programes de salut que s'ofereixen en el nostre sistema, com per exemple, el programa antitabac. Aquesta entitat guardarà la informació que descriu completament el programa.

  Cada *CENTRE* ofereix aquells *PROGRAMES* que creu oportuns i per això sorgeix la interrelació *OFEREIX*, fruit de relacionar-les.

- **REALITZA**
  Un *PACIENT* pot realitzar diferents *PROGRAMES DE SALUT*. Per aquest motiu es relacionen aquestes dues entitats i com a fruit obtenim la interrelació *REALITZA*.

- **RECEPTA**
  Durant la *VISITA*, el *METGE* si ho troba adient recomanarà al *PACIENT* la presa d'un o varis medicaments cadascun amb una dosi, tot emplenant uns fulls que s'anomenen receptes. Aquesta situació és representada per la interrelació *RECEPTA* i és el fruit de relacionar l'entitat *VISITA* amb *PRODUCTE*.

- **VACUNA**
  Una vacuna és el procés d'iniciar o augmentar la resistència a una infermetat infecciosa. Aquesta entitat representa aquest procés i guardarà aquells camps que ho descriuen.

  Existeix un vincle entre *VACUNA* i *VISITA*, doncs en una visita poden *POSAR* diferents *VACUNES* a un *PACIENT*.

- **IT**
  Existeixen períodes durant la vida en els que un *PACIENT* està incapacitat per treballar, és el que formalment es denomina tenir una Incapacitat Temporal, com per exemple al trencar-se una cama. Aquesta entitat contindrà la llista de possibles fets que poden fer al *PACIENT* causar una baixa o incapacitat temporal.

  És possible que durant el transcurs d'una *VISITA*, el *METGE* que l'està duent a terme decideixi que el *PACIENT* està incapacitat temporalment per treballar i li doni la baixa. Es representa aquest procediment relacionant les entitats *VISITA* amb IT i obtenim la interrelació *DONA*.

2.4. Acotacions.

Com ja s'ha expressat anteriorment, la quantitat d'informació que pot contenir un historial és immensa. L'objectiu d'aquest PFC no és el aprofundir en el disseny de les bases de dades i intentar obtenir un disseny acurat del que ha de ser un historial mèdic, si no que s'ha de centrar en els aspectes criptogràfics que ens permetran transmetre aquesta informació de forma segura per una xarxa d'ordinadors.

Prenent aquest requeriment com a objectiu, és decideix que l'historial mèdic que s'implementarà només contindrà informació sobre les visites que s'han dut a terme pels usuaris i es permetrà consultar els pacients assignats a un metge.

2.5. Requisits de seguretat.

L'objectiu d'aquest PFC és centrar-nos en aquells aspectes de seguretat que permetran poder transmetre cert tipus d'informació confidencial, per una xarxa d'ordinadors. El nostre sistema ha de complir els següents requisits:

1) No ha de ser possible que cap altre individu que no sigui un metge pugui veure les dades del nostre historial mèdic.
2) Cal que "la identitat" del nostre sistema no pugui ser suplantada, és a dir, no pot ser que ningú pugui posar un servei semblant al nostre per fer-nos creure que consultem allà.
3) Tampoc no ha de ser possible que cap intrús pugui modificar les dades del nostre historial.
4) Per acabar, no pot ser que després d'afegir informació a un historial per part d'un metge, aquest es faci enrere posteriorment i no tinguem manera de provar que el metge es desdiu del que havia dit amb anterioritat.

Resumint, el nostre sistema haurà de complir quatre conceptes clau: confidencialitat, autenticació, integritat i no-repudi.

El fet que ningú a part d'un metge pugui veure les dades del nostre historial és el que es coneix com a **confidencialitat**.

D'altra banda, que ningú no pugui suplantar la identitat del nostre sistema recau en la propietat **d'autenticació**.

La **integritat** és la propietat que assegura que les dades del nostre historial no es puguin modificar per ningú que no sigui un metge sense que el sistema ho detecti.

Finalment, el fet que un metge no es pugui desdir d'haver fet una historial es coneix com a **no-repudi.**

La criptografia proporciona les eines necessàries per poder complir amb els requisits de seguretat. Amb els criptosistemes de clau simètrica es poden aconseguir les propietats de confidencialitat i integritat, mentre que amb els esquemes de clau pública es pot obtenir les de confidencialitat, integritat, autenticació i no-repudi

La criptografia de clau simètrica i la de clau pública tenen un seguit d'avantatges complementaris que fan que la combinació dels dos esquemes sigui el que s'utilitza a la pràctica. En concret, la tècnica que es coneix com a sobre digital és la que es fa servir en la majoria d'aplicacions que permeten xifrar dades.

Malgrat això, encara no complim els quatre requisits de seguretat, doncs, utilitzant criptografia de clau simètrica i pública amb sobres digitals no assegurem el no-repudi. L'utilització de **certificats digitals** ens permeten assegurar que el nostre sistema criptogràfic complirà amb els quatre requisits marcats.

3. Disseny de l'esquema criptogràfic.

A continuació es mostra el disseny dels protocols per poder dur a terme el conjunt d'accions a desenvolupar.

3.1. Notació.

- K
  Clau d'un criptosistema simètric.
- $E_K(M)$
  Xifratge simètric d'un missatge *M* amb la clau *K*.
- $D_K(C)$
  Desxifratge simètric del criptograma *C* amb la clau *K*.
- $(P_{Entitat}, S_{Entitat})$
  Parella de claus asimètriques propietat d'Entitat, on *P* correspon a la clau pública, i *S* a la privada.
- $S_{Entitat}[M]$
  Signatura digital del missatge *M* amb la clau privada *S* d'Entitat.
- $P_{Entitat}[M]$
  Xifratge del missatge *M* amb la clau asimètrica pública $P_{Entitat}$ d'Entitat.
- $H(M)$
  Sortida d'una funció resum criptogràfica del missatge *M*, aquestes funcions reben el nom de funcions hash.

3.2. Procediments.

Tot seguit detallem diferents procediments que són utilitzats pels diferents protocols:

3.2.1. Procediment 1 .

Entrada	Cap.
Sortida	$P_G[N_i, id\_usuari_\mu]$.
Crida	Usuari.
Descripció	Utilitzat per l'*usuari* durant el procés d'autenticació. Aquest procediment està basat en l'autenticació de Needham-Schroeder.

1. Obtenir un valor de forma aleatòria, N_i;
2. Xifrar N_i i Id_usuari_μ amb P_G de G, $P_G\left[N_i, Id\_usuari_\mu\right]$;
3. Enviar $P_G\left[N_i, Id\_usuari_\mu\right]$ a G;

3.2.2. Procediment 2.

Entrada	$P_G[N_i, Id\_usuari_\mu]$.
Sortida	$P_\mu[N_i, N_g, Id\_usuari_G]$.
Crida	Gestor.
Descripció	Utilitzat pel *gestor* durant el procés d'autenticació. Aquest procediment està basat en l'autenticació de Needham-Schroeder.

1. Desxifrar $\left[N_i, Id\_usuari_\mu\right]$ amb S_G i obtenir: N_i, Id_usuari_μ;
2. Obtenir de la BD el certificat de μ a partir de Id_usuari_μ;
3. Obtenir un valor de forma aleatòria, N_g;
4. Guardar en la BD $\left[N_i, N_g, Id\_usuari_\mu\right]$;
5. Xifrar N_i, N_g, Id_usuari_G amb P_μ de μ:
   $P_\mu\left[N_i, N_g, Id\_usuari_G\right]$;
6. Retornar $P_\mu\left[N_i, N_g, Id\_usuari_G\right]$;

3.2.3. Procediment 3.

Entrada	Id_usuari$_\mu$.
Sortida	$P_\mu[\hbar]$.
Crida	Gestor.
Descripció	Retorna l'historia $\hbar$ corresponent a id_usuari$_\mu$ xifrat amb P_μ.

1. Buscar a la BD l'historial $\hbar$ corresponent a Id_usuari_μ;
2. Desxifrar la part de $\hbar$ que està xifrada utilitzant la clau privada S_G de G;
3. Xifrar $\hbar$ amb la clau pública P_μ de μ, $P_\mu[\hbar]$;
4. Retornar $P_\mu[\hbar]$;

3.2.4. Procediment 4.

Entrada	$P_\mu[\hbar]$.
Sortida	$\hbar$.
Crida	Client.
Descripció	Retorna l'historia $\hbar$ de id_usuari$_\mu$ després de verificar signatures i seqüències.

1. Desxifrar $P_\mu[\hbar]$ amb la clau privada S_μ de μ, $S_\mu\left[P_\mu[\hbar]\right]$;
2. Per cada entrada de l'historial $\hbar$ que està signada fer:
   (a) Verificar la signatura digital de M;
   (b) Verificar la signatura digital de G;
   (c) Verificar la seqüència;
3. Retornar $\hbar$;

3.2.5. Procediment 5.

Entrada	Id_usuari_m.
Sortida	P_m [{Id_usuari_1 , ..., Id_usuari_n }, S_G [{Id_usuari_1 , ..., Id_usuari_n }]].
Crida	Gestor.
Descripció	Retorna els pacients assignats al metge Id_usuari_m. de forma xifrada i signada.

1. Cercar a la BD els pacients assignats al metge id_usuari_m, obtenint $\{id\_usuari_1,..., id\_usuari_n\}$;
2. Signar $\{id\_usuari_1,..., id\_usuari_n\}$ amb la clau privada S_G de G, $S_G\left[\{id\_usuari_1,..., id\_usuari_n\}\right]$
3. Xifrar $\{id\_usuari_1,..., id\_usuari_n\}$ i $S_G\left[\{id\_usuari_1,..., id\_usuari_n\}\right]$ amb la clau pública de Id_usuari_m P_m,
$$P_m\left[\begin{matrix}\{id\_usuari_1,..., id\_usuari_n\},\\ S_G\left[\{id\_usuari_1,..., id\_usuari_n\}\right]\end{matrix}\right];$$
4. Retornar
$$P_m\left[\begin{matrix}\{id\_usuari_1,..., id\_usuari_n\},\\ S_G\left[\{id\_usuari_1,..., id\_usuari_n\}\right]\end{matrix}\right];$$

3.2.6. Procediment 6.

Entrada	P_m [{Id_usuari$_1$, ..., Id_usuari$_n$ }, S_G [{Id_usuari$_1$, ..., Id_usuari$_n$ }]]
Sortida	{Id_usuari$_1$, ..., Id_usuari$_n$ }
Crida	Client
Descripció	Retorna els pacients assignats al metge. Id_usuari$_m$.

1. Desxifrar

$$P_m\left[\begin{matrix}\{id\_usuari_1,\ldots, id\_usuari_n\},\\ S_G\left[\{id\_usuari_1,\ldots, id\_usuari_n\}\right]\end{matrix}\right]$$

amb la clau privada S_m de m,

$$S_m\left[P_m\left[\begin{matrix}\{id\_usuari_1,\ldots, id\_usuari_n\},\\ S_G\left[\{id\_usuari_1,\ldots, id\_usuari_n\}\right]\end{matrix}\right]\right]$$

i obtenir $\{id\_usuari_1,\ldots, id\_usuari_n\}$, $S_G\left[\{id\_usuari_1,\ldots, id\_usuari_n\}\right]$;

2. Verificar la signatura digital

$S_G\left[\{id\_usuari_1,\ldots, id\_usuari_n\}\right]$

amb la clau pública P_G de G;

3. Si la verificació anterior és correcta, retornar

$\{id\_usuari_1,\ldots, id\_usuari_n\}$;

3.3. Autenticar usuari.

Quan l'usuari desitja realitzar una petició sobre el gestor, s'inicia el procés d'autenticació basat en el protocol de Needham-Schoreder i que té com a finalitat intercanviar de forma segura els identificadors.

El següent protocol autentica el client amb el gestor:

1) μ realitza les operacions següents:
   a) Executar el *Procediment 1* amb la clau pública P_μ i obtenir $P_G[N_i, Id\_usuari_\mu]$;
   b) Enviar $P_G[N_i, Id\_usuari_\mu]$ a G;

2) G realitza les operacions següents:
   a) Executar el *Procediment 2* amb $P_G[N_i, Id\_usuari_\mu]$, i obtenir $P_\mu[N_i, N_g, Id\_usuari_G]$;
   b) Enviar $P_\mu[N_i, N_g, Id\_usuari_G]$ a μ;

3) μ realitza les operacions següents:
   a) Desxifrar $P_\mu[N_i, N_g, Id\_usuari_G]$ amb la clau privada $S\mu$, i obtenir N'_i, N_g, Id_usuari_G;
   b) Si $N'_i = N_i$ fer:
      i) Xifrar N_g, Operació[1], Info[2] amb la clau pública P_G de G, $P_G[N_g, Operacio, Info]$.
      ii) Enviar $P_G[N_g, Operacio, Info]$ a G;
   c) Sinó retornar *error*;

4) G realitza les operacions següents:
   a) Desxifrar $P_G[N_g, Operacio, Info]$ amb la clau privada S_G, i obtenir N'_g, Operació, Info
   b) Recuperar N_g de la BD.
   c) Si $N'_g = N_g$, client μ i gestor G estan autenticats i G pot executar la petició sol·licitada en el paràmetre Operació.

[1] Pot ser Consulta, Llista_pacients, Inserir_visita.
[2] Aquest valor dependrà de la operació sol·licitada i podrà ser Id_usuari per l'operació Consulta, Llista_pacients o V (Dades signades de la visita) pel cas Inserir_visita.

3.4. Consulta historial.

La història clínica d'un individu és de la seva pertinença i per tant aquest té dret a consultar-la en qualsevol moment. Legalment dins els drets i obligacions de l'usuari (http://www.gencat.net/ics/usuaris/drets_deures.htm), per ficar algun exemple tenim els següents drets: Dret a accedir a les dades personals obtingudes en l'atenció sanitària, Dret de l'usuari per accedir a la documentació de la seva història clínica, és a dir, tenim dret a consultar el nostre historial mèdic.

A banda d'això, per poder realitzar la tasca professional és possible que els facultatius necessitin realitzar aquesta mateixa operació.

Aquest procés pot ser cridat tant pel pacient com pel metge, aquest darrer després de la crida haurà d'indicar l'identificador del pacient (CIP) del que desitja realitzar la consulta.

El següent protocol retorna la història clínica d'un pacient.

1) Usuari i Gestor executen el procés d'autenticació. En el pas número 4 d'aquest procés G rep $P_G[N_g$, Consulta, Id_usuari]

2) G realitza les operacions següents:
   a) Recuperar Id_usuari_μ de la BD a partir de N_g.
   b) Si (Id_usuari_μ = Id_usuari) o (Id_usuari_μ és metge i Id_usuari és pacient de Id_usuari_μ) fer:
      i) Executar el *Procediment 3* amb Id_usuari i obtenir $P_\mu[H]$.
      ii) Enviar $P_\mu[H]$ a μ
   c) Sinó retorna *error*;
   d) Esborrar N_g, N_i, Id_usuari_μ de la BD;

3) μ realitza les operacions següents:
   a) Executar el Procediment 6 amb P_μ [{Id_usuari_1 , ..., Id_usuari_n }, S_G [{Id_usuari_1 , ..., Id_usuari_n }]] i obtenir {Id_usuari_1 , ..., Id_usuari_n }.
   b) Mostrar {Id_usuari_1 , ..., Id_usuari_n }.

3.5. Consulta de pacients assignats a un metge.

Definició: Contingent és aquella part assignada o repartida a un poble o a un particular en qualsevol impost, préstec o servei.

En aquest cas cada metge té un contingent de la població assignat. Aquesta assignació es fa en funció de la quantitat de població, edat dels pacients, factors de qualitat que es volen assolir, etc. d'aquesta forma el contingent assignat a cada metge varia en funció de la especialitat, població, etc.

Amb el contingent assignat a un metge es poden fer moltes operacions per fer càlculs de qualitat, com per exemple, del contingent assignat quin % ha estat atès, etc.

El següent protocol retorna el contingent assignat a un metge.

1) Usuari i Gestor executen el procés d'autenticació. En el pas número 4 d'aquest procés G rep $P_G[N_g$, Llista_pacients, Id_usuari].

2) G realitza les operacions següents:
   a) Recuperar Id_usuari_μ de la BD a partir de N_g.
   b) Desxifrar $P_G[N_g$, Llista_pacients, Id_usuari] amb la clau privada S_G, i obtenir N'_g, Llista_pacients, Id_usuari;
   c) Si (Id_usuari_μ és metge) i (Id_usuari_μ = Id_usuari) fer:
      i) Executar el *Procediment 5* amb Id_usuari_μ i obtenir P_μ[{Id_usuari_1 , ..., Id_usuari_n }, S_G [{Id_usuari_1 , ..., Id_usuari_n }]].
      ii) Enviar a μ P_μ [{Id_usuari_1 , ..., Id_usuari_n }, S_G[{Id_usuari_1 , ..., Id_usuari_n }]].
   d) Sinó retorna *error*;
   d) Esborrar N_g, N_i, Id_usuari_μ de la BD;

3) μ realitza les operacions següents:
   a) Executar el Procediment 6 amb P_μ [{Id_usuari_1 , ..., Id_usuari_n }, S_G [{Id_usuari_1 , ..., Id_usuari_n }]] i obtenir {Id_usuari_1 , ..., Id_usuari_n }.
   b) Mostrar {Id_usuari_1 , ..., Id_usuari_n }

3.6. Afegir historial.

En l'apartat *"Consulta historial"* s'han citat alguns dels drets que tenim com a pacients i que es poden exigir en tot moment. Un altre dret molt important és el Dret a la confidencialitat de la informació. Aquest, fent un petit resum, diu que la informació relativa a les dades dels actes sanitaris s'ha de mantenir dins el dret professional estricte i el dret a la intimitat del pacient.

Aquesta operació ha d'aplicar certs mecanismes que garanteixin aquests requeriments. Com ja s'ha citat anteriorment la informació relativa a les dades dels actes sanitaris es guardarà xifrada, de forma que puguem garantir la **confidencialitat, autenticació, integritat i no-repudi.**

El següent protocol afegeix una nova línia a l'historial mèdic d'un pacient.

1) Usuari i Gestor executen el procés d'autenticació. En el pas número 4 d'aquest procés G rep $P_G[N_g, \text{Inserir\_visita}, [V, S_M[V]]]$.

2) G realitza les operacions següents:
   a) Recuperar Id_usuari_m de la BD a partir de N_g.
   b) Si (Id_usuari_m és metge) fer:
      i) Obtenir Id_usuari_p a partir de V;
      ii) Si (Id_usuari_p és un pacient de id_usuari_m) fer:
         (1) Verificar la signatura digital $S_M[V]$ amb la clau pública P_M;
         (2) Obtenir l'instant de temps actual T.
         (3) Obtenir el número de sèrie X de la última visita de l'historia H del pacient Id_usuari_p;
         (4) Incrementar en una unitat X, X+1;
         (5) Signar V, $S_M[V]$, T, X+1 amb la clau privada S_G de G, $S_G[V, S_M[V], T, X+1]$;
         (6) Xifrar V i $S_M[V]$ amb la clau pública S_G de G, $P_G[V, S_M[V]]$;
         (7) Signar Id_usuari_p i X+1 amb la clau privada S_G de G, $S_G[X+1, Id\_usuari_p]$;
         (8) Guardar en la BD: $P_G[V, S_M[V]]$, T, X+1, $S_G[V, S_M[V], T, X+1]$ i $S_G[X+1, Id\_usuari_p]$;
      iii) Sinó retorna *error*;
   c) Sinó retorna *error*;
   d) Esborrar N_g, N_i, Id_usuari_m de la BD;

3.7. Implementació.

IAIK és un conjunt de eines criptogràfiques molt complerta basades en Java, que ens permetrà entre altres, xifrar, desxifrar i signar a partir de certs certificats.

Aquesta és molt extens, implementa múltiples algoritmes criptogràfics juntament amb els seus mètodes i tot això sota la filosofia de programació orientada a objecte (POO).

Entre els múltiples algorismes que IAIK implementa, tenim els orientats a la generació de claus, simètrica i asimètrica. Codificació i descodificació de dades, funcions hash, funcions resumen, creació de certificats, etc. Entre aquestos algoritmes i tècniques podem trobar RSA, SHA, DES, triple DES, MD2, MD5, certificats X509, IDEA, etc.

4. Representació de les dades: XML.

4.1. Introducció.

Definició: XML (e**X**tensible **M**arkup **L**anguage) és un llenguatge extensible d'etiquetes desenvolupat per W3C (**W**orld **W**ide **W**eb **C**onsortium).

XML no és un llenguatge de marcatge com HTML, sinó un meta-llenguatge que ens permet definit altres llenguatges de marcatge adequats per usos determinats; és a dir, no és un llenguatge, sinó varis; no és una sintaxi, sinó varies.

Treballar amb XML proporciona els següents avantatges:

1) **Comunicació**: Si la informació es transferida en XML, qualsevol aplicació pot escriure un document de text pla amb les dades que està manejant en format XML i una altra aplicació podria rebre aquesta informació i treballar amb ella.

2) **Migració**: Si treballem amb XML seria molt senzill moure les dades entre base de dades

3) **Web**: Amb XML existeix una sola aplicació que s'encarrega de la gestió de les dades i per cada navegador podem tenir una fulla d'estil o similar per aplicar un estil apropiat.

En XML es separa el contingut de la presentació de forma total. Aquest es basa en documents de *text pla* en els que s'utilitzen *etiquetes* per delimitar els *elements* del *document*. XML defineix aquestes etiquetes en funció del tipus de dades que s'està escrivint i no de l'aparença final; amb l'afegit de permetre definir noves etiquetes i ampliar les existents.

Definició: El DTD (Document Type Definition) permet definir els tipus d'elements, atributs, entitats permeses i limitacions que ha de complir el nostre document XML.

Definició: Un document XML està ben format si segueix les especificacions del llenguatge respecte a les regles sintàctiques.

Definició: Un document XML és vàlid si a més a més d'estar ben format, segueixen una estructura i una semàntica determinada pel DTD.

Algunes regles que s'han de tenir en compte si volem crear documents XML ben formats són les següents:

1) Han de seguir una estructura estrictament jeràrquica. Una etiqueta ha d'estar correctament inclosa en una altra i els elements amb contingut han d'estar tancats correctament.
2) Els elements sense contingut han de ser: <element_sense_contingut/>
3) Només podem tenir un element arrel.
4) Els valors dels atributs han d'estar tancats entre cometes simples o dobles.
5) És sensible a majúscules minúscules.

Els documents XML que s'ajusten al seu DTD es diu que són vàlids. Un document ben format no té per que ser vàlid. Un document XML ben format senzillament respecta l'estructura i sintaxis definides en l'especificació XML, mentre que un document és vàlid si està ben format i compleix el seu DTD.

Un possible exemple per entendre la estructura d'un document XML podria ser el següent:

```
<?xml versión="1.0"?>
<!DOCTYPE MENSAJE SYSTEM "missatge.dtd">
<missatge prioritat="alta">
        <remitent>
                <nom>Iñaki Romero</nom>
                <mail>iromerol@uoc.edu</mail>
        </remitent>
        <destinatari>
                <nom>Jordi Castellà</nom>
                <mail>jcastella@uoc.edu</mail>
        </destinatari>
        <asumpte>Hola Jordi</asumpte>
        <text>¿Hola que tal? Com estàs?</text>
</missatge>
```

En aquest exemple identifiquem els següents elements:

Capçalera:

Malgrat que no és obligatori un document XML pot contenir una capçalera que descriu la versió, tipus de document i altres coses, en concret
:

1) Una declaració XML que declara el document com un document XML.
2) Una declaració de tipus de document. Enllaça aquest document amb el seu DTD.
3) Una o més comentaris e instruccions de processament.

Elements:
Aquestos poden tenir contingut (més elements, caràcters o tots dos), o be estar buits.

Atributs:
Els elements poden tenir atributs, que incorporen propietats a l'element del document.

4.2. Format dels documents.

Definició: Protocol és aquell conjunt de normes i procediments útils per la transmissió de dades, conegut entre emissor i receptor.

En la realització d'aquest PFC existeixen situacions diferents que fan necessari el intercanvi d'informació per part de client i gestor. El contingut de la informació variarà en funció de la situació en la que ens trobem, el client farà una petició al gestor i aquest contestarà. És necessari doncs que client i gestor coneguin quin ha de ser el format de les dades per cada petició. Qualsevol intercanvi d'informació es farà utilitzant documents XML, formatat i validat segons la situació.

Veiem doncs el tipus de documents que ens podrem trobar per qualsevol situació.

4.2.1. Protocol d'autenticació.

En el moment en que el client fa una petició sobre el gestor, aquestos s'han d'autenticar en el sistema. Client i gestor engeguen el protocol d'autenticació basat en l'autenticació de Needham-Schroeder (veure apartat 3.2.1 i 3.2.2) i s'intercanvien documents XML xifrats per dur-lo a terme.

L'estructura d'aquest document és la següent:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE HCE SYSTEM "hce.dtd">
<HCE>
        <Needham_Schroeder>
                <param1>-</param1>
                <param2></param2>
                <param3></param3>
        </Needham_Schroeder>
</HCE>
```

El valor de cada element dependrà de la fase del protocol d'autenticació en la que ens trobem.

Inicialment el client vol sol·licitar un servei, engega el protocol d'autenticació i envia un document XML com l'anterior de forma xifrada, el gestor el tracta i respon amb un altre document xifrat similar a l'anterior. Quan l'usuari sol·licitarà l'execució d'un servei al gestor, aquest enviarà part de la informació utilitzada en l'autenticació de forma que el gestor la utilitzarà per validar si aquest usuari és el que s'ha autenticat en el sistema amb anterioritat.

El valor dels diferents elements és diferent en cada fase del protocol;

En una primera fase el client estableix els següents elements del document

Param1	N_i
Param2	Id_usuari_μ
Param3	No l'utilitza

En la segona fase el gestor guarda aquesta informació en la BD i respon amb un altre document, on el valors dels elements són els següents:

Param1	N_i
Param2	N_g
Param3	Id_usuari$_G$

En una darrera fase el client envia la petició d'execució d'un servei al gestor indicant quin ha de ser aquest, els valors dels elements del document en aquest cas són:

Param1	N_i
Param2	Operació
Param3	Info

Veiem doncs que l'element "Param2" conté el servei a executar. Els possibles valors d'aquest són els següents (Veure apartat 3.3):

Consulta	Consultar l'historia clínica de l'usuari.
Llista_pacients	Consulta la llista de pacients d'un metge.
Inserir_visita	Inserir una visita en l'historia clínica d'un usuari per part d'un metge.

4.2.2. Consulta historial.

El client, en aquest cas un pacient o un metge, sol·licita al gestor l'obtenció d'una història mèdica indicat el Codi Identificació Personal (CIP) del pacient. El gestor fa les verificacions de seguretat oportunes i realitza una consulta en la BD, obté les dades sol·licitades i les retorna en un document XML signat que conté la següent estructura:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE HCE SYSTEM "hce.dtd">
<HCE>
        <Historial>
                <Actuacio>
                        <CIP><CIP/>
                        <NIF><NIF/>
                        <Pg><Pg/>
                        <T><T/>
                        <X1><X1/>
                        <Sg><Sg/>
                        <Sg2><Sg2/>
                <Actuacio/>
                .       .       .       .       .       .
                <Actuacio>
                        <CIP><CIP/>
                        <NIF><NIF/>
                        <Pg><Pg/>
                        <T><T/>
                        <X1><X1/>
                        <Sg><Sg/>
                        <Sg2><Sg2/>
                <Actuacio/>
        </Historial>
        <Signature></Signature>
</HCE>
```

L'element Pg es construeix a partir d'un altre document XML que conté informació especifica de la visita. En aquest cas aquest document XML es signat pel metge, i l'estructura és la següent:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE HCE SYSTEM "hce.dtd">
<HCE>
        <Visita>
                <CIP><CIP/>
                <Data><Data/>
                <Hora><Hora/>
                <M><M/>
                <E><E/>
                <A><A/>
                <P><P/>
        </Visita>
        <Signature></Signature>
</HCE>
```

4.2.3. Consulta de pacients assignats a un metge.

Quan el client, en aquest cas un metge, sol·licita aquest servei espera obtenir una llista corresponent als seus pacients. Per fer això el gestor realitza una consulta en la BD, obté els pacients assignats al metge i els retorna en un document XML signat que conté la següent estructura:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE HCE SYSTEM "hce.dtd">
<HCE>
        <Pacients>
                <Pacient>
                        <CIP></CIP>
                        <Nom></Nom>
                        <Cognoms></Cognoms >
                </Pacient>
                .       .       .       .       .       .
                <Pacient>
                        <CIP></CIP>
                        <Nom></Nom>
                        <Cognoms></Cognoms >
                </Pacient>
        </Pacients>
        <Signature></Signature>
</HCE>
```

4.2.4. Afegir historial.

En els apartats anteriors ja s'ha vist el format de dos tipus de documents XML: el document que guarda la informació de la visita i el document que conté l'historial del pacient, que de fet és una col·lecció de visites.

Quan el pacient realitza una visita, el metge introdueix al sistema la informació rellevant. Es composa el document XML, es signa per part del metge i s'envia de forma xifrada al gestor.

Tornem a recordar l'estructura del document XML que conté la informació sobre la visita:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE HCE SYSTEM "hce.dtd">
<HCE>
        <Visita>
                <CIP><CIP/>
                <Data><Data/>
                <Hora><Hora/>
                <M><M/>
                <E><E/>
                <A><A/>
                <P><P/>
        </Visita>
        <Signature></Signature>
</HCE>
```

El gestor rep un document XML amb aquest format, verifica la signatura i si tot es correcte realitza els càlculs i signatures necessaris per guardar la informació en la Base de Dades de forma segura.

Per realitzar les signatures, el gestor composa un altre document XML que segueix la següent estructura:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE SEGURETAT SYSTEM "seguretat.dtd">
<Seguretat>
        <Info>
                <Param1><Param1/>
                <Param2><Param3/>
                <Param3><Param3/>
        </Info>
        <Signature></Signature>
</Seguretat>
```

El valor dels elements "Param" dependrà de la signatura que estem realitzant:

1) S_G[V, S_M[V], T, X+1]

a) Param1: Conté V, S_M[V], és a dir, conté una estructura XML[3] de tipus visita.
b) Param2 Conté el valor T.
c) Param3: Conté el valor X+1.

2) S_G[X+1, Id_usuari_P]

a) Param1: En aquest cas es nul (no té valor).
b) Param2: Conté el valor X+1.
c) Param3: Conté el valor Id_usuari_P.

L'element "<Signature>" conté la signatura dels elements continguts a "<Info>" i aquest valor és el que es guarda en la base de dades.

4.2.5. DTD.

El DTD, tal i com s'ha explicat anteriorment, ens permet descriure la sintaxi dels nostres documents XML: estructura, elements, atributs i valors permesos.

El contingut d'aquest fitxer és el següent:

[3] Conté el subarbre de l'element <Visita>

```
<?xml version="1.0"?>
<!ELEMENT HCE      (Needham_Schroeder|
                   (Historial,Signature)|
                   (Visita,Signature)|
                   (Pacients,Signature))>

<!-- Protocol autenticar -->
<!ELEMENT Needham_Schroeder (param1,param2,param3)>
<!ELEMENT param1 (#PCDATA)>
<!ELEMENT param2 (#PCDATA)>
<!ELEMENT param3 (#PCDATA)>

<!-- Protocol Consular Historial -->
<!ELEMENT Historial (Visita+)>

<!-- Protocol Afegir Visita -->
<!ELEMENT Visita (CIP,NIF,Pg,T,X1,Sg,Sg2)>

<!ELEMENT CIP       (#PCDATA)>
<!ELEMENT NIF       (#PCDATA)>
<!ELEMENT Pg        (#PCDATA)>
<!ELEMENT T         (#PCDATA)>
<!ELEMENT X1        (#PCDATA)>
<!ELEMENT Sg        (#PCDATA)>
<!ELEMENT Sg2       (#PCDATA)>

<!-- Signature -->
<!ELEMENT Signature (#PCDATA)>


<!-- Protocol Consultar Llista Pacients -->
<!ELEMENT Pacients (Pacient+)>

<!-- Pacient -->
<!ELEMENT Pacient (CIP,Nom,Cognoms)>

<!ELEMENT CIP       (#PCDATA)>
<!ELEMENT Nom       (#PCDATA)>
<!ELEMENT Cognoms   (#PCDATA)>
```

4.3. Implementació.

En la fase d'implementació s'ha utilitzat la llibreria JDOM (**J**ava **D**ocument **O**bject **M**odel), llibreria de lliure distribució que conté el codi font per manipular dades XML, optimitzat pel llenguatge de programació Java. Aquesta es pot descarregar des de la pàgina web http://www.jdom.org/downloads/index.html. En l'apartat Apèndix A. Instal·lació., s'explica el procés amb més detall.

S'implementen les classes necessàries per poder crear cadascun dels documents XML citats anteriorment.

La relació entre les classes i el document XML creat és la següent:

XMLAutenticar	Document XML amb estructura <Needham_Schroeder> Conté informació d'autenticació entre client i gestor.
XMLHistorial	Document XML amb estructura <Historial> i <Signature> Conté una col·lecció de visites.
XMLPacients	Document XML amb estructura <Pacients> i <Signature> Conté una col·lecció de pacients assignats a un metge.
XMLSeguretat	Document XML amb estructura <Seguretat> i <Signature> Conté informació utilitzada en el procés d'inserció d'una visita.
XMLVisita	Document XML amb estructura <Visita> Conté informació relativa a la visita.

5. Comunicació dels components: RMI.

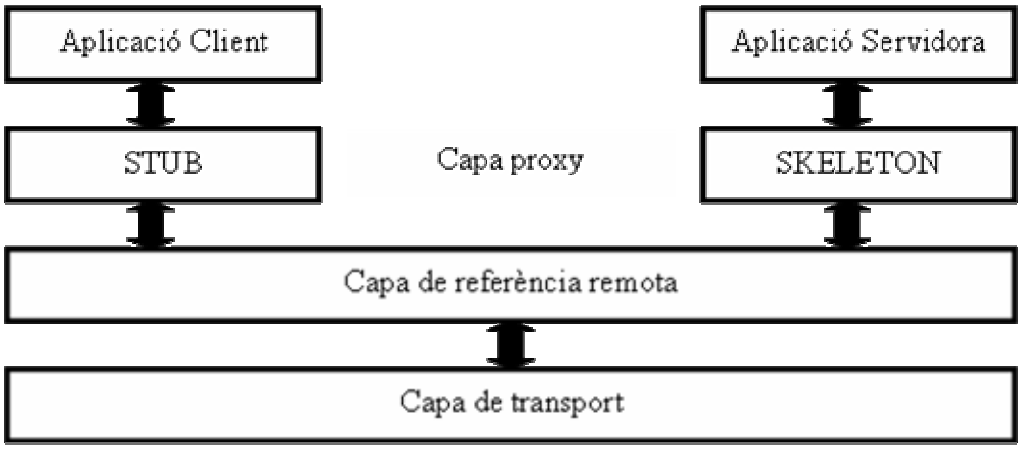
En els capítols anteriors, es parla contínuament del fet que client i gestor "s'intercanvien informació" per poder efectuar les operacions.

Hem vist que aquesta informació s'envia en forma de document XML que varia depenent de la situació. Fins aquí hem vist "el què" i falta descriure "el cóm", és a dir, hem vist quin tipus d'informació s'envia, però no hem vist com.

RMI (**JAVA R**emote **M**ethod **I**nvocation) és el mecanisme ofert en el llenguatge JAVA que permet fer crides a mètodes d'objectes remots situats en la mateixa o diferents màquines virtuals de Java. RMI segueix el concepte de programació distribuïda i forma part del **JDK** (**J**ava **D**evelopment **K**it), és a dir, no s'ha de fer cap instal·lació addicional per poder treballar amb aquestes llibreries..

En una aplicació RMI podem distingir entre aplicacions clients i servidores. L'aplicació servidora és l'encarregada de crear objectes remots, fer accessibles via RMI els mètodes per a que puguin ser cridats pels clients.

La arquitectura RMI es pot veure com un model de quatre capes:



Il·lustració 7.- Arquitectura RMI

La primera capa és la d'aplicació i correspon a la implementació real de les aplicacions client i servidor. Qualsevol aplicació que desitja que els seus mètodes estiguin disponibles remotament han de declarar aquestos mètodes en una interfície que implementa java.rmi.Remote.

La capa 2 és la capa proxy, o capa stub-skeleton. Aquesta capa és la que interactua directament amb l'aplicació. Totes les crides a objectes remots i accions amb paràmetres i retorn d'objectes tenen lloc en aquesta capa.

La capa 3 és la de referència remota i és la responsable de la gestió de la part semàntica de les invocacions remotes.

La capa 4 és la de transport i és la responsable de realitzar les connexions necessàries i gestió del transport de les dades d'una màquina a una altra.

Tota aplicació RMI normalment es descompon en dues parts:

1) Un **servidor**, que crea alguns objectes remots, crea referències per fer-los accessibles i espera a que el client els invoqui.
2) Un **client**, que obté una referència a objectes remots en el servidor i els invoca.

Treballar amb aquesta tecnologia té els següents avantatges:

1) Client - Servidor
2) Simple
3) Segur
4) Baixa sobrecàrrega
5) Eficient, potent, flexible
6) Basat en Java

5.1. Implementació.

5.1.1. Servidor.

El servidor RMI consisteix en definir un objecte remot que va serà utilitzat pels clients. Primer es defineix una interfície i l'objecte remot serà una classe que implementarà aquesta interfície.

En aquest PFC, la classe *RemoteServerGE.java* defineix l'interfície de la següent forma:

```
public interface RemoteServerGE extends Remote {

        //Llista de mètodes Remots
        byte [] initCommunication (byte[] Pg) throws RemoteException;
        byte [] performAction (byte[] Pg) throws RemoteException;
}
```

La classe gestor és l'encarregada d'implementar aquesta interfície, definint les accions que s'han de realitzar en cada mètode.

```
public class Gestor extends UnicastRemoteObject implements RemoteServerGE {
. . . . . .
```

Aquesta classe s'ha d'encarregar de fer-se visible pel client, utilitzant la classe Namig amb el mètode rebind(...)

```
private final String _host = "rmi://localhost:2001/RemoteServerGE";
. . . . . .
Naming.rebind(_host, this);
```

Abans de llançar l'aplicació servidora s'han de crear els objectes "Stub" i "Skeleton" per mantenir una referència a l'objecte remot. Això ho podem aconseguir amb la comanda rmic:

```
javac Gestor.java
rmic -d . Gestor
```

També s'ha de llençar el registre de RMI, en aquest cas:

Windows: start rmiregistry 2001 **Linux:** rmiregistry 2001 &

En aquest cas 2001 és el port pel que connectarem i que és el mateix que especifiquem en el mètode rebind(...) citat anteriorment.

Un cop hem creat els fitxers "stub", "skeleton" i hem arrancat el registre RMI, estem en disposició de llançar l'aplicació servidora.

5.1.2. Client.

El client accedirà a l'objecte remot (Server) que acabem de veure. Aquest senzillament haurà de cercar l'objecte remot en el registre RMI de la màquina remota. Per poder-lo fer utilitzem la classe Namig i el mètode lookup (...)

```
server = (RemoteServerGE)Naming.lookup("rmi://localhost:2001/RemoteServerGE");
```

Després de cercar l'objecte remot, estem en disposició de cridar als mètodes que aquest fa visibles:

```
. . . . . .
byte pU[] = server.initCommunication(Pg);
. . . . . .
byte PuH[] = server.performAction(Pg2);
. . . . . .
```

6. Gestió de la informació: BD.

Definició: Una Base de Dades és un conjunt d'informació emmagatzemada en memòria auxiliar que permet accedir directament.

Al guardar la informació en una base de dades aquesta perdura en el temps i es pot utilitzar posteriorment. La informació es guarda de forma estructurada i serà accessible en temps real.

Al guardar la informació en una base de dades, permetrem entre d'altres, a un pacient, consultar la seva història mèdica en tot moment, encara que hagin passat dies i dies des de la darrera visita.

Aquest fet també ens permetrà millorar la qualitat del servei, doncs es pot realitzar una explotació de les dades per fer càlculs de qualitat, com per exemple: número de visites per metge, número de pacients atesos per metge, etc.

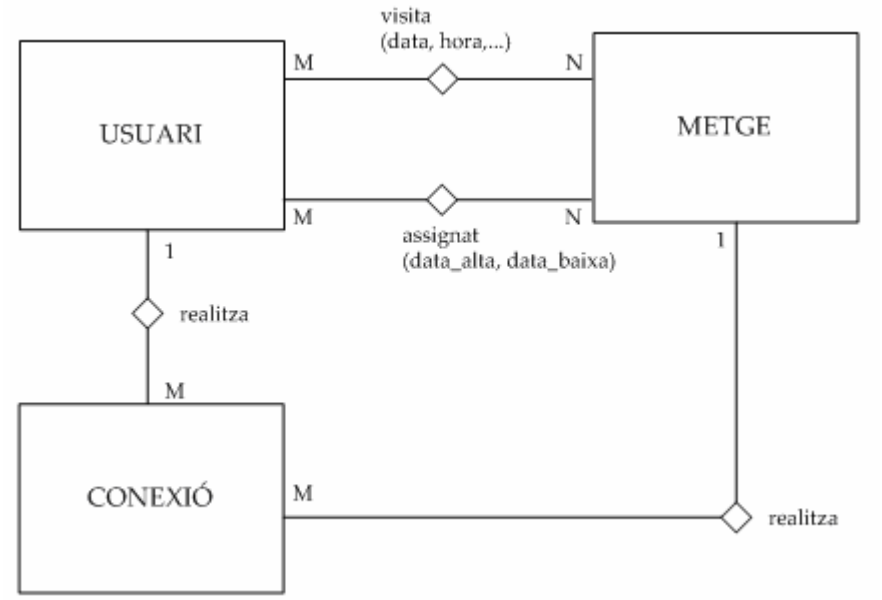
6.1. Implementació.

6.1.1. Model Entitat - Relació.

6.1.2. Disseny conceptual.

En l'apartat 2.3 fruit de l'anàlisi que es fa sobre el sistema s'obté un disseny conceptual que s'ajusta a les necessitats de la nostra aplicació. El disseny obtingut contempla moltes situacions que poder fer inassolible l'execució d'aquest PFC, per això en l'apartat següent (2.4) s'acoten els elements a tractar amb la finalitat de centrar-nos en els aspectes criptogràfics del projecte i no en el disseny de la base de dades.

El següent disseny conceptual s'obté a partir d'acotar els elements que formen part del nostre sistema, analitzats en els diferents apartats vistos fins el moment: i s'ajusta als requeriments de seguretat.

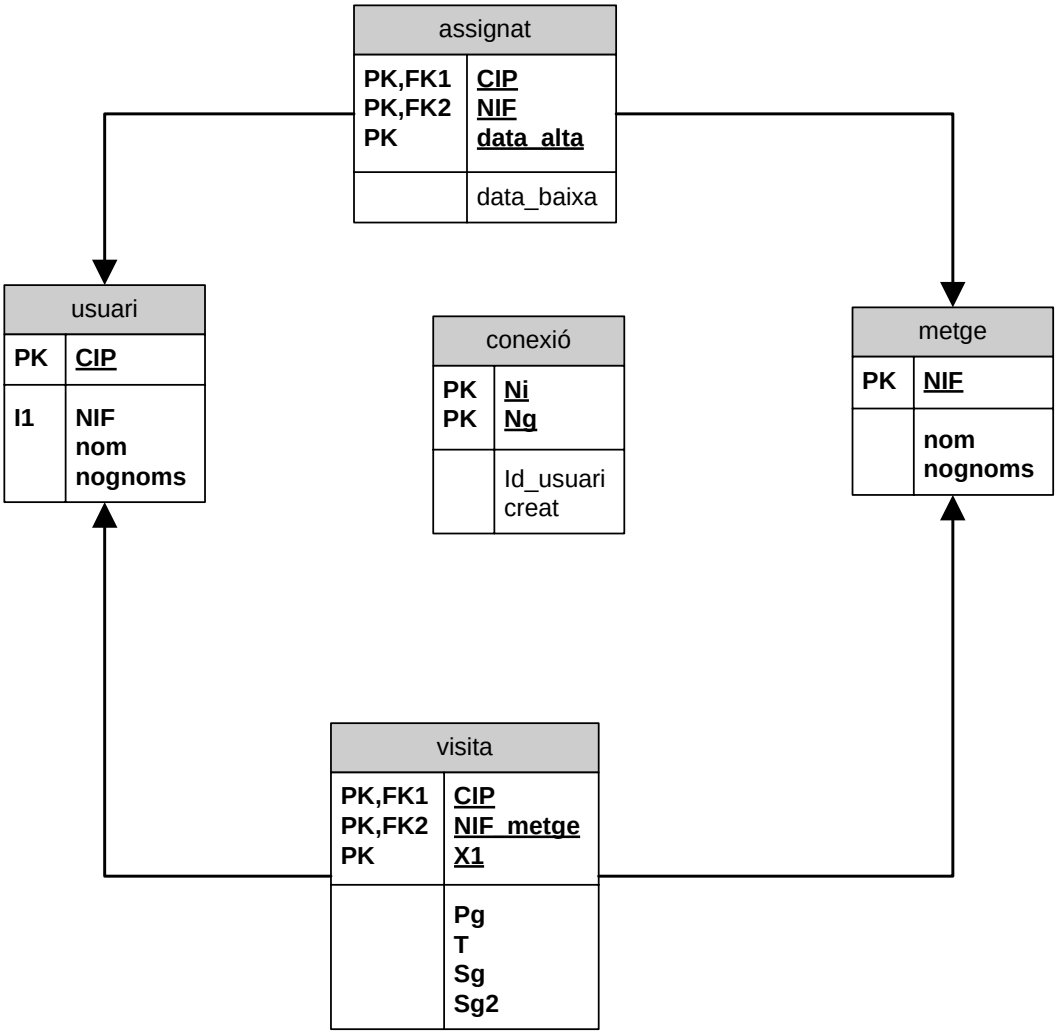


Il·lustració 8.- Disseny lògic. Diagrama Entitat - Relació de Chen.

6.1.3. Disseny Lògic.

Un cop tenim el disseny conceptual, expressat en forma de diagrama d'entitat - relació de Chen, podem efectuar la transformació al model relacional.

El següent disseny lògic s'obté al transformar el diagrama entitat - relació en el model relacional:



Il·lustració 9.- Disseny Lògic. Model relacional

6.1.4. Motor.

Existeix un ampli ventall de bases de dades en el mercat, per la implementació d'aquest PFC s'ha triat el motor MySQL doncs es tracta d'un producte de lliure distribució i està desenvolupat sota la filosofia de "codi obert".

MySQL permet definir diferents tipus de taules. En aquest cas es fa servir el "InnoDB Storage Engine". InnoDB dota a MySQL amb un entorn transaccional segur amb possibilitat de confirmar (commit), cancel·lar (roolback) i mecanismes de recuperació davant falles. InnoDB també suporta clàusules de tipus "FOREIGN KEY"

A més, a partir de la versió 5, ofereix la possibilitat de treballar amb procediments emmagatzemats, disparadors, vistes, etc.

Definició: Un procediment emmagatzemat (stored procedure) és un programa (o procediment) el qual està emmagatzemat físicament a la base de dades.

Utilitzar procediments emmagatzemats té moltes avantatges, algunes d'elles:

1) Descarrega el trànsit entre usuari i servidor, doncs la petició s'executa directament en el motor de la base de dades, incrementant el rendiment de la xarxa.
2) Independència de dades, doncs es separa la lògica de l'aplicació.
3) Disseny modular.
4) Compartició, diferents aplicacions poden compartir procediments, amb la disminució del codi resultant.
5) El manteniment dels programes és més senzill.
6) Un canvi en el procediment es reflexa automàticament en les aplicacions.

En aquest projecte, totes les operacions (alta, baixa, consulta) s'han implementat amb procediments emmagatzemats, beneficiant-nos de les seves avantatges.

Definició: Un script és un programa escrit per un llenguatge interpretat.

S'ha creat un script de creació de la base de dades, aquest, crea la base de dades, un usuari amb permís per accedir i totes les taules amb les restriccions necessàries. El citat script el podem veure a continuació:

```
CREATE DATABASE IF NOT EXISTS iromerol_pfc DEFAULT CHARACTER SET latin1 COLLATE latin1_bin;

USE iromerol_pfc;

GRANT ALL PRIVILEGES ON *.* TO 'user'@'localhost' IDENTIFIED BY 'user' WITH GRANT OPTION;


DROP TABLE IF EXISTS Conexio;

CREATE TABLE Conexio (
        Ni varchar(30) NOT NULL,
        Ng varchar(30) NOT NUll,
        userId varchar(14) NOT NULL,
        creat datetime NOT NULL,
        primary key (Ni, Ng, userId)
) TYPE=INNODB;


DROP TABLE IF EXISTS Assignat;
DROP TABLE IF EXISTS Visita;
DROP TABLE IF EXISTS Pacient;
DROP TABLE IF EXISTS Metge;
DROP TABLE IF EXISTS pki;


CREATE TABLE Pacient (
        CIP varchar(14) NOT NULL,
        NIF varchar(9) NOT NUll,
        nom varchar(14) NOT NULL,
        cognoms varchar(50) NOT NULL,
        primary key (CIP)
) TYPE=INNODB;
```

```
CREATE TABLE Metge (
        NIF varchar(9) NOT NUll,
        nom varchar(14) NOT NULL,
        cognoms varchar(50) NOT NULL,
        primary key (NIF)
) TYPE=INNODB;


CREATE TABLE Assignat (
        CIP varchar(14) NOT NULL,
        NIF varchar(9) NOT NUll,
        data_alta date NOT NULL,
        data_baixa date,
        primary key (CIP, NIF, data_alta),
        foreign key(CIP) references Pacient(CIP),
        foreign key(NIF) references Metge(NIF)
) TYPE=INNODB;


CREATE TABLE Visita (
        CIP varchar(14) NOT NULL,
        NIF varchar(9) NOT NUll,
        Pg longblob NOT NULL,
        T long NOT NULL,
        X1 int NOT NULL,
        Sg longblob NOT NUll,
        Sg2 longblob NOT NUll,
        primary key (CIP, NIF, X1),
        foreign key(CIP) references Pacient(CIP),
        foreign key(NIF) references Metge(NIF)
) TYPE=INNODB;


CREATE TABLE pki (
        id varchar(14) NOT NULL,
        cert longblob NOT NULL,
        primary key (id)
) TYPE=INNODB;
```

Per un altre costat disposem d'un script de creació dels procediments que es pot veure a continuació:

```
DROP PROCEDURE IF EXISTS addConexio;

DELIMITER $$

CREATE PROCEDURE addConexio (IN _Ni varchar(30), IN _Ng varchar(30), IN
        _userId varchar(14))
BEGIN
        START TRANSACTION;
        DELETE FROM Conexio WHERE DATE_ADD(creat, INTERVAL 30 SECOND) <
        NOW();
        INSERT INTO Conexio (Ni, Ng, userId, creat) VALUES (_Ni, _Ng, _userId,
        NOW());
        COMMIT;
END;
$$

DELIMITER ;

DROP PROCEDURE IF EXISTS delConexio;

DELIMITER $$

CREATE PROCEDURE delConexio (IN _Ng varchar(30))
BEGIN
        START TRANSACTION;
        DELETE FROM Conexio WHERE Ng = _Ng;
        COMMIT;
END;
$$

DELIMITER ;
```

```
DROP PROCEDURE IF EXISTS getConexio;

DELIMITER $$

CREATE PROCEDURE getConexio (IN _Ng varchar(30))
BEGIN
        SELECT userId, Ng FROM Conexio WHERE Ng = _Ng AND DATE_ADD(creat,
        INTERVAL 30 SECOND) >= NOW();
END;
$$

DELIMITER ;
```

```
DROP PROCEDURE IF EXISTS getPacients;

DELIMITER $$

CREATE PROCEDURE getPacients (IN _NIF_Metge varchar(9))
BEGIN
	SELECT u.CIP, u.nom, u.cognoms FROM Assignat a, Pacient u WHERE a.NIF =
	_NIF_METGE AND a.CIP = u.CIP AND a.data_alta <= NOW() AND (data_baixa
	>= NOW() OR data_baixa IS NULL);
END;
$$

DELIMITER ;


DROP PROCEDURE IF EXISTS getPacient;

DELIMITER $$

CREATE PROCEDURE getPacient (IN _CIP varchar (14), IN _NIF_Metge varchar(9))
BEGIN
	SELECT CIP FROM Assignat WHERE CIP = _CIP AND NIF = _NIF_METGE
	AND data_alta <= NOW() AND (data_baixa >= NOW() OR data_baixa IS
	NULL);
END;
$$

DELIMITER ;

DROP PROCEDURE IF EXISTS getCertificat;

DELIMITER $$

CREATE PROCEDURE getCertificat (IN _id varchar(14))
BEGIN
	SELECT cert FROM pki WHERE id = _id;
END;
$$

DELIMITER ;
```

```
DELIMITER $$

CREATE PROCEDURE addVisita (IN _CIP varchar(14), IN _NIF varchar(9), IN _Pg
        longblob, IN _T long, IN _X1 int, IN _Sg longblob, IN _Sg2 longblob)
BEGIN
        START TRANSACTION;
        INSERT INTO Visita (CIP, NIF, Pg, T, X1, Sg, Sg2) VALUES (_CIP, _NIF, _Pg,
        _T, _X1, _Sg, _Sg2);
        COMMIT;
END;
$$

DELIMITER ;


DROP PROCEDURE IF EXISTS getVisites;


DELIMITER $$

CREATE PROCEDURE getVisites (IN _CIP varchar(14))
BEGIN
        SELECT * FROM Visita WHERE CIP = _CIP ORDER BY X1;
END;
$$

DELIMITER ;
```

En el script de creació de taules podem observar que les signatures les guardem en un BLOB (Binary Large Object), que ens permetrà guardar dades binàries grans. Una camp d'aquest tipus pot tenir un màxim de 4GB (2^{32}) bytes.

La següent taula descriu cadascun dels elements dels scripts anteriors:

- **Taules:**

Conexio	Conté informació de les connexions entre client i servidor. S'utilitza en el procés d'autenticació.
Pacient	Conté informació que descriu el pacient.
Metge	Conté informació que descriu el metge
Assignat	Conté la relació dels pacients assignats als metges.
Visita	Conté informació que descriu una visita.

- **Procediments:**

Procediments

addConexio	Afegeix informació d'una nova connexió.
delConexio	Esborra una connexió existent.
getConexio	Obtenir informació d'una connexió existent.
getPacients	Obtenir els pacients assignats a un metge.
getPacient	Obtenir el pacient assignat a un metge.
addVisita	Afegir informació d'una nova visita.
getVisites	Obtenir les visites d'un pacient, és a dir, l'historia mèdica.

6.1.5. Java.

Java, el llenguatge utilitzat pel desenvolupament de les aplicacions, necessitarà realitzar operacions sobre la base de dades. Per a poder-lo fer, és necessari utilitzar una llibreria (connector) que prèviament s'ha de descarregar i instal·lar. En l'Apèndix A. s'explica aquest procés amb més detall.

El gestor és l'encarregat d'accedir a la base de dades i només ell ho podrà fer doncs és l'únic que coneix els paràmetres de connexió. Seguint amb el model de programació orientada a objecte, es crea una nova classe que serà l'encarregada de fer les connexions amb la base de dades i executar els procediments. Aquesta classe serà utilitzada pel gestor per poder respondre a les peticions del client.

La classe en gestió s'anomena BDManager.java, una breu descripció dels mètodes que aquesta classe ofereix:

addConexio	Afegeix informació d'una nova connexió.
	Crida al procediment: addConexio

delConexio	Esborra una connexió existent.
	Crida al procediment: delConexio

getNgConexio	Obtenir l'atribut Ng d'una connexió existent.
	Crida al procediment: getConexio

getUserIdConexio	Obtenir l'identificador de l'usuari d'una connexió existent.
	Crida al procediment: getConexio

estaAssignat	Retorna cert si el pacient està assignat al metge en l'actualitat.
	Crida al procediment: getPacient

getLlistaPacients	Obté la llista de pacients assignats a un metge en l'actualitat.
	Crida al procediment: getPacients

getHistorial	Obté l'història mèdica d'un pacient.
	Crida al procediment: getVisites

getNumVisita	Obté el darrer número de seqüència de les visites d'un pacient.
	Crida al procediment: getVisites

addVisita	Afegeix una nova visita en la Base de Dades.
	Crida al procediment: addVisita

7. Interfície.

Definició: Una interfície és la part gràfica d'una aplicació informàtica que facilita la interacció de l'usuari amb l'ordinador a través de la utilització d'elements com imatges, icones, finestres, botons, etc.

Es defineixen dues interfícies d'usuari, una per facilitar la feina al gestor del sistema i una altra pel usuari final, que en aquest cas pot ser un pacient o un metge. L'usuari, és a dir, el pacient o el metge, executa les peticions al servidor (gestor) a través de l'interfície d'usuari accedint a les diferents opcions que aquesta presenta. En el cas de l'interfície d'usuari aquesta executarà unes accions o unes altres depenent del tipus. Per ficar un exemple, l'interfície permetrà executar l'operació "obtenir llista de pacients" a un metge, mentre que no li permetrà al pacient.

La interfície d'usuari implementa les finestres necessàries per autenticar-se en el sistema, introduïr o mostrar informació, executar peticions i mostrar l'estat actual en el que es troba l'aplicació.

La interfície del gestor implementa la finestra necessària per poder-se autenticar en el sistema, si l'usuari pertany al grup de gestors es permetrà engegar-lo o aturar-lo i mostra l'estat actual en el que es troba l'aplicació.

7.1. Implementació.

La implementació de les interfícies s'ha dut a terme utilitzant les llibreries Swing que porta incorporades el JDK de Java. Cal remarcar que la utilització d'un editor com "Eclipse" de lliure distribució, ha estat fonamental per agilitzar aquesta tasca, doncs porta un mòdul per treballar amb components visuals.

7.2. Interfície de gestor.

Anem a veure els aspectes més rellevants d'aquesta interfície.

7.2.1. Pantalla principal.

La següent imatge il·lustra aquesta pantalla:

Il·lustració 10.- Interfície gestor: pantalla principal.

Aquesta interfície es caracteritza pel fet que té les opcions situades a la dreta en forma de botons i disposa d'una finestra on es mostra l'estat en el que es troba l'aplicació.

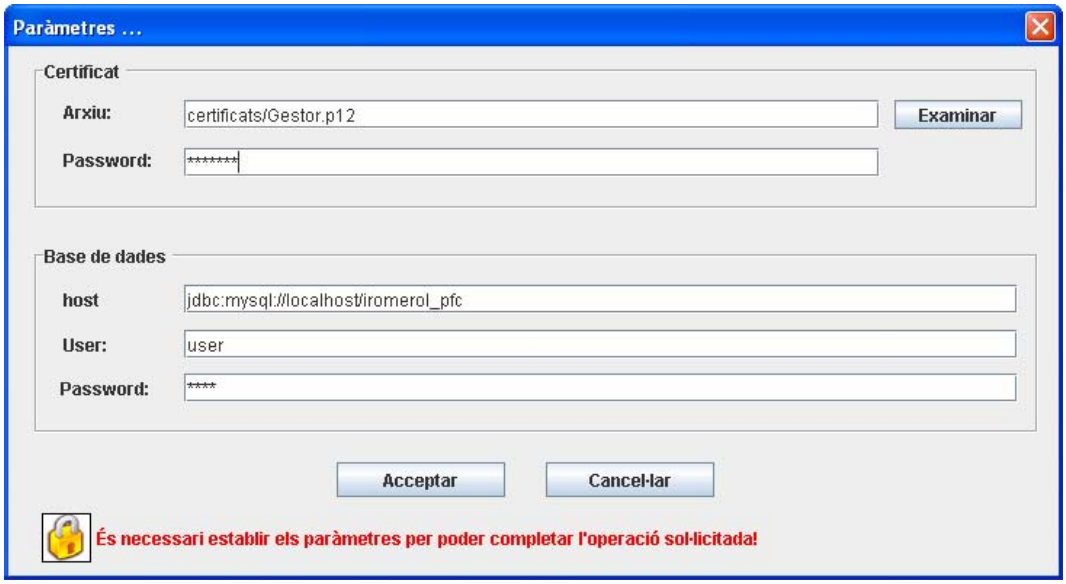
Si el servidor es troba aturat, es pot executar l'operació "engegar" i de forma similar si es troba engegat es podrà executar l'operació "aturar".

Per realitzar cada operació l'usuari s'ha de validar en el sistema. Un cop fet si aquest pertany al grup de gestió, l'operació es durà a terme, altrament es mostrarà un missatge d'error en la finestra d'estat.

7.2.2. Paràmetres de connexió.

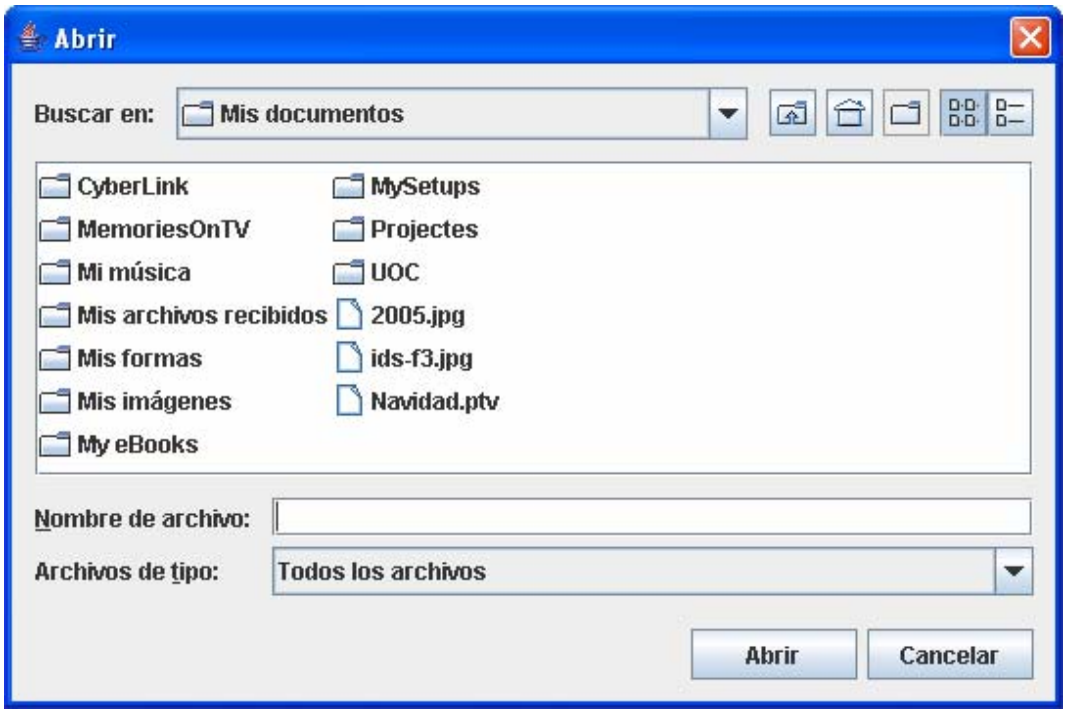
Aquesta finestra és la que s'utilitza el gestor per validar-se en el sistema. En aquesta finestra introduïm els valors rellevants del certificat i paràmetres de connexió a la Base de Dades.

Per facilitar la tasca a l'usuari, es mantenen les dades ja establertes excepte les contrasenyes, de forma que en crides successives només farà falta introduir les contrasenyes necessàries.



Il·lustració 11.- Interfície gestor: pantalla de paràmetres.

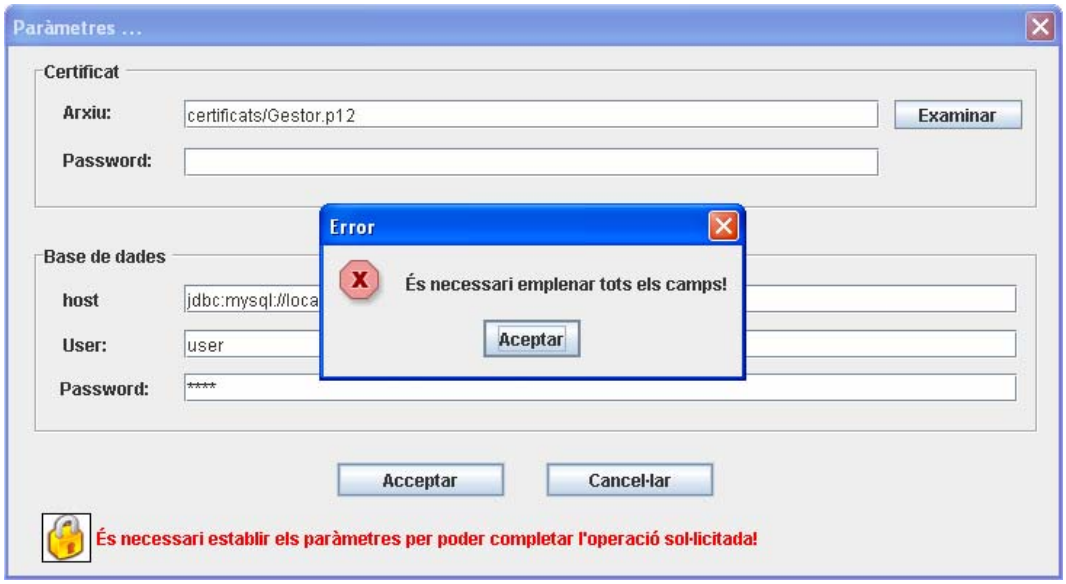
Aquesta pantalla presenta un botó d'examinar per facilitar la feina a l'usuari, d'aquesta forma podem indicar on es troba ubicat el certificat del gestor.



Il·lustració 12.- Interfície gestor: pantalla examinar.

En cas que l'usuari hagi acabat i triï l'opció acceptar, l'interfície abans verifica que hi són totes les dades necessàries, si no és així, ho indica a l'usuari.

La següent imatge il·lustra aquesta situació:



Il·lustració 13.- Interfície gestor: falta algun camp en pantalla paràmetres.

7.3. Interfície Client.

Com s'ha comentat anteriorment aquesta interfície serà utilitzada tant pel pacient com pel metge. Les finestres d'un i altre són pràcticament idèntiques. A continuació es presenten els aspectes més importants d'aquesta interfície.

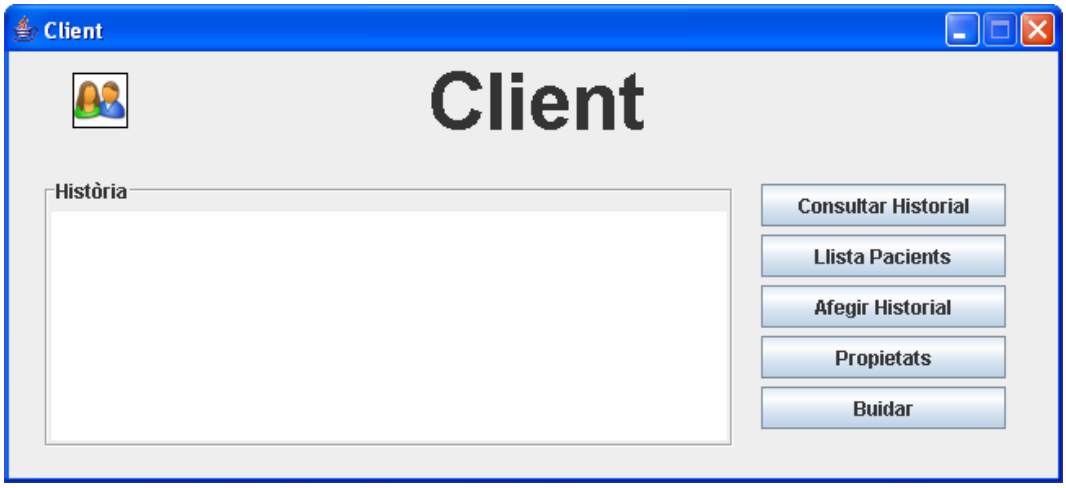
7.3.1. Pantalla principal.

Igual que en el cas del gestor, aquesta interfície segueix la mateixa filosofia tant en disseny com en comportament.

Pel que fa al disseny tenim la finestra d'estat situada a la part esquerra, mentre que les opcions estan disponibles en la part dreta.

Pel que fa al comportament, cada acció necessita la validació de l'usuari. Si l'usuari que demana l'acció té privilegis per a fer-ho, s'executa, en altre cas el sistema mostra un missatge d'error en la finestra d'estat.

La següent imatge mostra com és aquesta interfície:



Il·lustració 14.- Interfície client: pantalla principal.

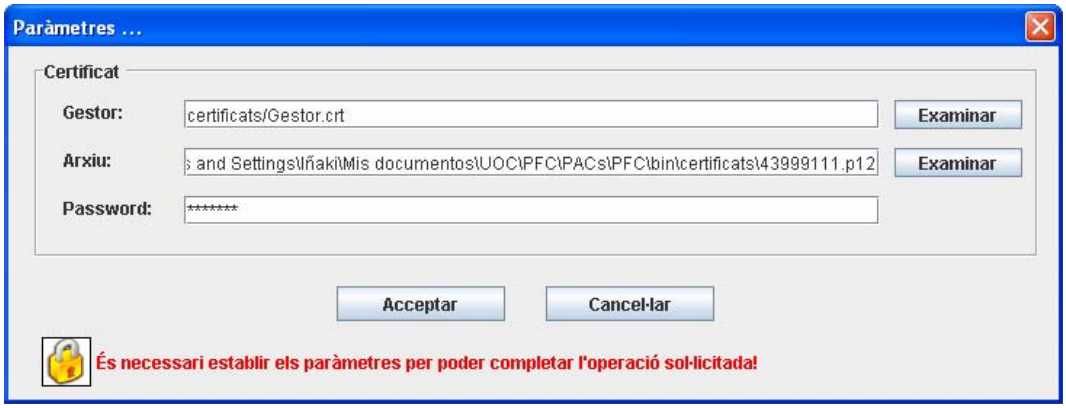
Si l'usuari que executa l'acció no té permís per a fer-ho, es mostra el següent missatge:

Il·lustració 15.- Interfície client: Error en pantalla principal.

En aquest cas un pacient ha intentat executar l'opció llista de pacients, que només la poden executar els usuaris que pertanyen al grup de metges.

7.3.2. Paràmetres de connexió.

A l'igual que l'interfície anterior, aquesta també és pràcticament idèntica que la de paràmetres de connexió del gestor. En aquest cas només s'ha d'introduir la informació necessària per validar-se en el sistema:

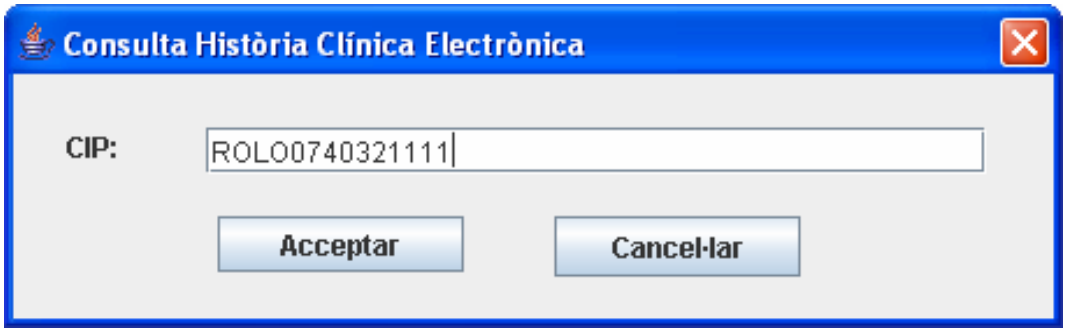


Il·lustració 16.- Interfície client: pantalla paràmetres de connexió.

Al igual que en el cas anterior podem examinar l'ordinador per indicar la ubicació del certificat. També de la mateixa manera, el sistema verifica que hi figuren totes les dades abans d'acceptar.

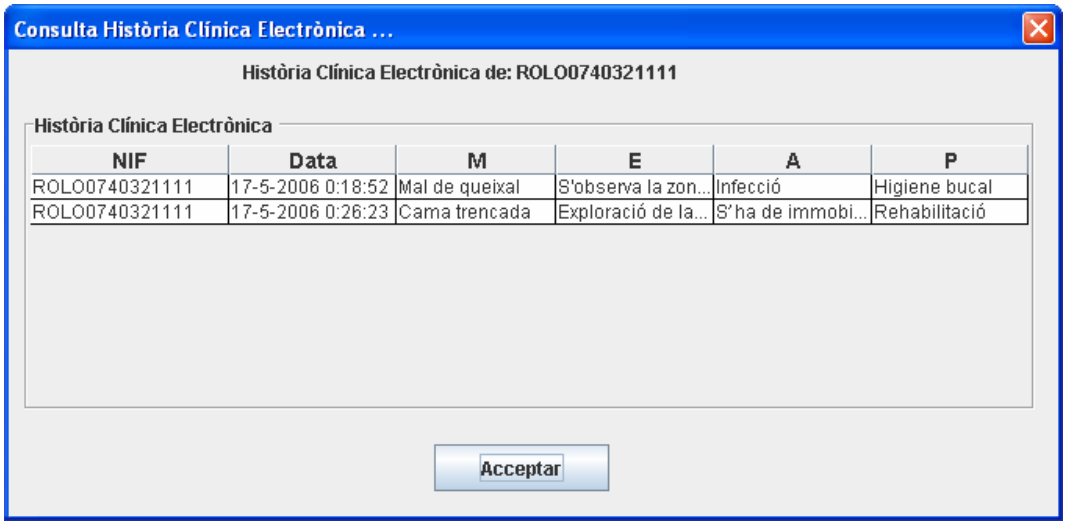
7.3.3. Consulta historial.

Aquesta operació la pot executar qualsevol dels dos usuaris. Si es crida per part del pacient i tot és correcte, es mostra l'historial directament, mentre que si es crida per part d'un metge, surt una finestra intermèdia que serveix per indicar el pacient a consultar, en aquest cas la finestra es mostra a continuació:



Il·lustració 17.- Interfície client: pantalla introduir CIP pacient.

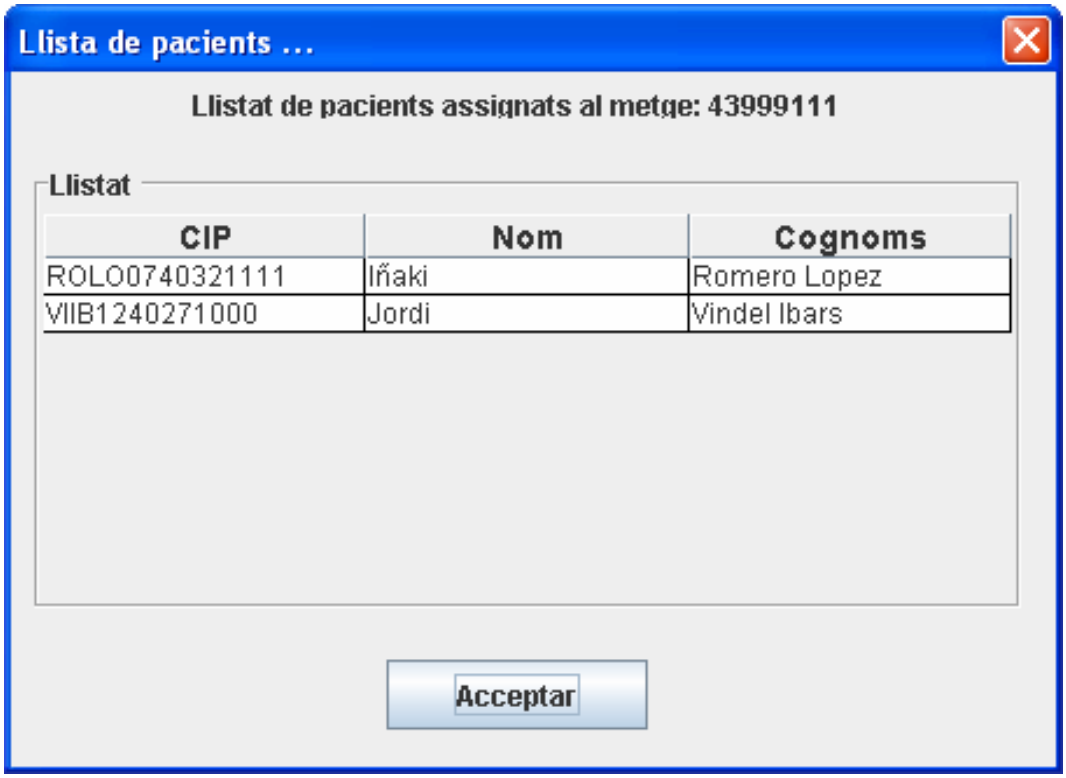
En aquest cas, el gestor verificarà si el pacient està assignat al metge autenticat, si no és així mostrarà un missatge d'error i en cas contrari lliurarà l'historia mèdica del pacient, que es mostra en l'interfície d'aquest en una pantalla com aquesta:



Il·lustració 18.- Interfície client: pantalla consulta historial.

7.3.4. Consulta pacients assignats a un metge.

Com s'ha citat anteriorment, aquesta operació només la pot realitzar un usuari que pertanyi al grup de metges. Si es dona aquest fet, es mostra una pantalla amb els pacients assignats al metge i que podria ser com aquesta:

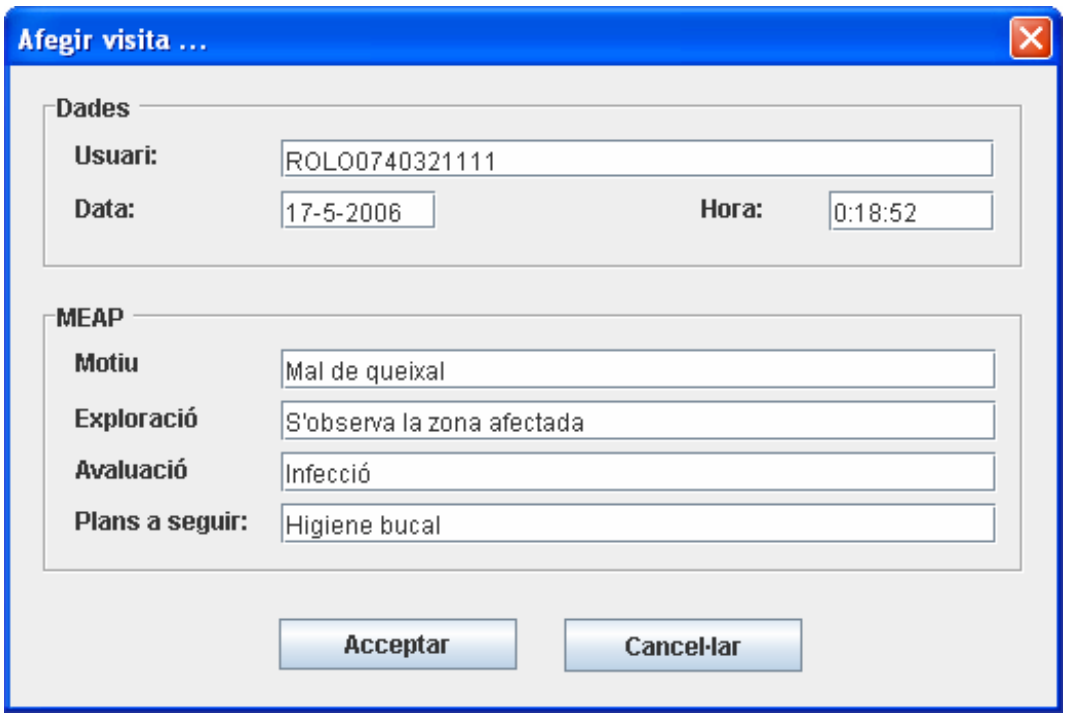


Il·lustració 19.- Interfície client: pantalla llista de pacients.

Al demanar aquesta opció, client envia la petició al gestor e informa en l'àrea d'estat del resultat obtingut.

7.3.5. Afegir visita.

Al igual que en el cas anterior, només un usuari del grup metges pot realitzar aquesta operació. Un cop s'ha validat l'usuari en el sistema, accedim a la pantalla d'introducció de dades de la visita, aquesta es pot veure a continuació:

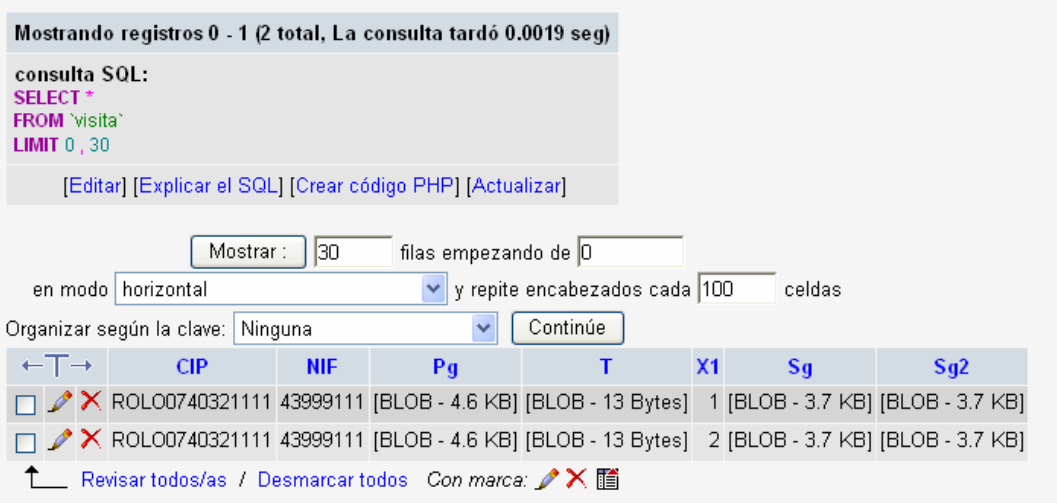


Il·lustració 20.- Interfície client: pantalla afegir visita.

Quan es mostra la pantalla, inicialment surten tots els camps en blanc llevat de la Data i Hora que ens la dona el sistema i que es pot editar. El sistema verifica que hi són totes les dades necessàries abans d'acceptar l'operació, que es podria cancel·lar triant el botó que mostra aquesta opció.

Un cop acceptat, s'envia la petició al gestor amb la informació de la visita i finalment es mostra un missatge en l'àrea d'estat que indica si l'operació s'ha efectuat amb èxit o si ha hagut algun error.

Si l'operació es dur a terme de forma satisfactòria, la informació rellevant a la visita es guarda en la base de dades de forma xifrada amb les signatures corresponents. Una exploració de la taula visites de la base de dades dona el següent resultat:



Il·lustració 21.- Interfície client: explorar taula visita de la base de dades.

Veiem com la informació està guardada en la base de dades de forma totalment segura.

8. Conclusions.

Aquest treball s'ha centrat en els aspectes criptogràfics necessaris per desenvolupar un aplicatiu accessible remotament, amb gestió de bases de dades i tot de forma segura.

S'ha utilitzat una PKI, definint com ha de ser l'esquema criptogràfic de manera que aquest compleixi amb les següents propietats:

- **Integritat:** La informació guardada en la base de dades no es podrà modificar o si es produeix una modificació el sistema no detectarà, doncs la validació de les signatures serà errònia.

- **Confidencialitat:** La informació sensible s'ha de mantenir de forma confidencial, de manera que si algú aconseguís accedir a la base de dades no entendria res, doncs la informació es guarda de forma xifrada i només la persona amb el certificat apropiat aconseguirà desxifrar el contingut.

- **Autenticitat:** El sistema pot demostrar quin usuari està realitzant la petició i quin servidor l'està servint, gràcies als certificats digitals.

- **No-repudi:** Un metge no es podrà desdir posteriorment, del que ha pogut dir anteriorment en una visita.

S'ha implementat una aplicació segons el model de client - servidor, utilitzant Java RMI per poder comunicar-se entre elles.

S'ha adoptat l'estàndard XML i es defineix el format de les dades que s'intercanvien client i servidor.

Segons el model de POO, s'ha desenvolupat una nova classe encarregada exclusivament de la gestió de la base de dades, establint la connexió i fent crides a procediments emmagatzemats.

Per finalitzar s'han implementat les interfícies gràfiques que facilitaran la feina tant a client com a servidor, definint les pantalles necessàries per poder realitzar les tasques requerides

9. Bibliografia.

1) Ajuda E-CAP.
[Pdf; Març-2006; Intranet del Institut Català de la Salut]

2) Aprenda Java.
[Pdf; Abril-2006; http://www.aprendergratis.com/]

3) Java Swing.
[Pdf; Maig-2006; http:www.jorgesanchez.net]

4) Swing Second Edition.
[Ebook; Maig-2006; Matthew Robinson i Pavel Vorobiev; Editorial Manning]

5) ICS (Institut Català de la Salut).
[Internet; Juny-2006; http://www.gencat.net/ics/]

6) Drets i deures de l'usuari de la sanitat pública.
[Internet; Juny-2006; http://www.gencat.net/ics/usuaris/drets_deures.htm]

7) Wikipedia, l'enciclopèdia lliure.
[Internet; Juny-2006; http://es.wikipedia.org/wiki/Portada]

8) SUN Microsystems.
[Internet; Juny-2006; http://java.sun.com/]

9) Java en castellano.
[Internet; Juny-2006; http://www.programacion.com/java/]

10) Adictos al trabajo.
[Internet; Juny-2006;
http://www.adictosaltrabajo.com/java/java.php?pagina=seccionjava]

10. Apèndix A. Instal·lació.

El present projecte ha estat provat en diferents plataformes, tant en plataformes Microsoft com en plataformes Linux.

A continuació es desglossen les diferents fases d'instal·lació sobre aquestos entorns:

10.1. IAIK.

Els passos per instal·lar IAIK són els següents:

1) Descarregar la darrera versió del JDK de SUN i instal·lar-lo.

2) Descarregar la darrera versió d'IAIK. Cal registrar-se malgrat que no suposa cap cost. Descarreguem l'arxiu *iaik_jec_full.jar*.

3) Descarregar les polítiques de seguretat de Java que permeten operar amb qualsevol longitud de clau. Java Cryptografy Extension (JCE) Unlimited Strength Jurisdiction Policy Files 5.0 RC.

4) (**Windows**) Copiar l'arxiu iaik_jce_full.jar als directoris:

   a) `c:\Archivos de Programa\Java\jdk1.5.0\jre\lib\ext`

   b) `c:\Archivos de Programa\Java\jre1.5.0\lib\ext`

5) (**Linux**) Copiar l'arxiu iaik_jce_full.jar al directori:

   c) `$JAVA_HOME/jre/lib/ext`

6) (**Windows**) Dins de l'arxiu jce_policy-1.5.0-beta2.zip hi ha els arxius: local_policy.jar i US_export_policy.jar, copiar-los a:

   d) `c:\Archivos de Programa\Java\jdk1.5.0\jre\lib\security`

   e) `c:\Archivos de Programa\Java\jre1.5.0\lib\security`

7) (**Linux**) Dins de l'arxiu jce_policy-1.5.0-beta2.zip hi els arxius: local_policy.jar i US_export_policy.jar, copiar-los a:

```
f) $JAVA_HOME/jre/lib/security
```

10.2. PKI.

En plataformes Linux no és necessari instal·lar cap programa addicional, doncs els nuclis incorporen el conjunt de sentències openssl per poder crear certificats digitals.

En Windows no disposem d'aquesta eina, però es pot descarregar de la pàgina web: http://www.openssl.org. Un cop descarregat només s'ha de configurar el path, indicant on es troba el directori amb la comanda openssl.

Tot seguit es descriu els passos a seguir per crear la PKI:

10.2.1. Crear una parella de claus i un certificat de CA.

En aquesta primera fase crearem una parella de claus del criptosistema RSA, que correspondran a la CA que utilitzarem al PFC. Els passos són els següents:

1) Crear una parella de claus del criptosistema RSA:

```
a) > openssl genrsa -des3 -out CA.key 2048
```

- La longitud de la parella de claus es de 2048 bits.
- La parella de claus estar`a xifrada amb Triple DES.
- Password = uoc0506.

2) Crear un certificat autosignat:

```
a) > openssl req -new -sha1 -x509 -key CA.key -out CA.crt -
   days 360
```

- Country Name (2 letter code) [AU]:ES
- State or Province Name (full name) [Some-State]:Barcelona
- Locality Name (eg, city) []:Barcelona
- Organization Name (eg, company) [Internet Widgits Pty Ltd]:UOC
- Organizational Unit Name (eg, section) []:PFC Seguretat
- Common Name (eg, YOUR name) []:CA PFC Seguretat
- Email Address []:iromerol@uoc.edu

- La funció resum (hash) que utilitzem és la sha1.
- El nom del fitxer que conté el certificat és: CA.crt

3) Verificar que el certificat és correcte:

```
a) > openssl verify -CAfile CA.crt CA.pem
```

10.2.2. Crear una parella de claus i emetre un certificat.

Aquest procés és quasi idèntic per la generació de certificats de gestor, pacient i metge. A continuació és descriu el procés remarcant les diferències entre ells:

Primer crearem una parella de claus RSA de 1024 bits, generarem una petició de certificat i emetrem un certificat a partir de la petició anterior. Finalment guardarem el certificat i la clau privada en un fitxer amb el format Personal Information Exchange Syntax Standard (PKCS#12).

1) Crear una parella de claus del criptosistema RSA:

```
a) > openssl genrsa -des3 -out UserId.key 1024
```

- La longitud del parell de claus és de 1024 bits.
- La parella de claus esta xifrada amb Triple DES (-des3).
- Password = uoc0506.

El nom del fitxer UserId.key, és el següent depenent del certificat a crear:

Gestor	NIF del gestor.
Pacient	CIP del pacient.
Metge	NIF del metge.

2) Crear una petició de certificat:

```
g) > openssl req -new -sha1 -key UserId.key -out UserId.csr
   -config openssl.conf
```

- Country Name (2 letter code) [ES]: País de la persona.
- State or Province Name (full name) [Barcelona]: Província de la persona.
- Locality Name (eg, city) [Barcelona]: Localitat de la persona.
- Organization Name (eg, company) [UOC]: UOC.
- Organizational Unit Name (eg, section) [PFC Seguretat]: Depèn[4].
- Common Name (eg, YOUR name) []: Nom de la persona.

[4] Gestió, Pacients o Metges segons si es tracta d'un gestor, un pacient o un metge respectivament.

- D.N.I or N.S.S. [00000000-A]: Identificador de la persona (UserId)[5].
- Email Address []: Adreça electrònica de la persona..

- El fitxer openssl.conf conté la configuració per defecte.
- La funció resum (hash) és la sha1.
- El nom del fitxer de sortida és: UserId.csr

3) Verificar que la petició és correcta:

```
h) > openssl req -in UserId.csr -verify -text -noout
```

4) Emetre el certificat a partir de la petició de certificat:

```
i) > openssl x509 -req -in UserId.csr -days 180 -CA CA.crt -
   CAkey CA.key -CAcreateserial -extfile openssl.conf -
   extensions usr cert -out UserId.crt
```

- El certificat serà vàlid uns 180 dies.
- El fitxer amb el certificat és UserId.crt.

5) Verificar el certificat:

```
j) > openssl verify -CAfile CA.crt UserId.crt
```

6) Crear el fitxer que conté la clau privada i el certificat en format PKCS#12.

```
k) > openssl pkcs12 -in UserId.crt -inkey UserId.key -name
   Gestor -chain -CAfile CA.crt -export -out UserId.p12
```

- El fitxer de sortida és: UserId.p12.

4) Opcionalment podem canviar el format del certificat a DER:

```
a) > openssl X509 -in UserId.crt -outform DER -out
   userid_DER.crt
```

10.3. XML

Els passos per instal·lar la llibreria **JDOM** són els següents:

1) Descarregar des de la pàgina web http://www.jdom.org/downloads/index.html.
2) Descomprimir l'arxiu descarregat.
3) En la carpeta build de l'arxiu descomprimit està ubicat la llibreria *jdom.jar*. Aquesta s'ha d'incloure en el CLASSPATH.

[5] NIF si es tracta d'un gestor o un metge, CIP si és un pacient.

10.4. Base de dades.

El procés d'instal·lació de la base de dades MySQL, és molt senzill. En l'àrea de descarrega de la pàgina web de MySQL: http://dev.mysql.com/downloads/, podem descarregar el motor de base de dades i també podem descarregar el connector necessari per poder efectuar crides des de Java, a sentències de manipulació de la base de dades MySQL.

A l'hora d'efectuar la descàrrega tenim un apartat per cada sistema operatiu.

Un cop hem descarregat el connector de MySQL per Java, aquest s'ha d'incloure en el CLASSPATH.

10.5. Editor.

Encara que no és obligatori, en el desenvolupament d'aquest PFC s'ha utilitzat un editor, que ha facilitat molt i molt la feina. Dels possibles s'ha triat l'editor de Java Eclipse, doncs es tracta d'un producte de lliure distribució.

La seva instal·lació també resulta molt senzilla. Des de l'àrea de descarrega de la pàgina web d'eclipse podem descarregar el programa: http://www.eclipse.org/downloads/.

Un cop descarregat, només s'ha de descomprimir i ja es pot utilitzar, no és necessari realitzar cap operació addicional.

11. Apèndix B. Demostració.

A continuació es mostra una petita demostració del producte obtingut en el desenvolupament d'aquest PFC.

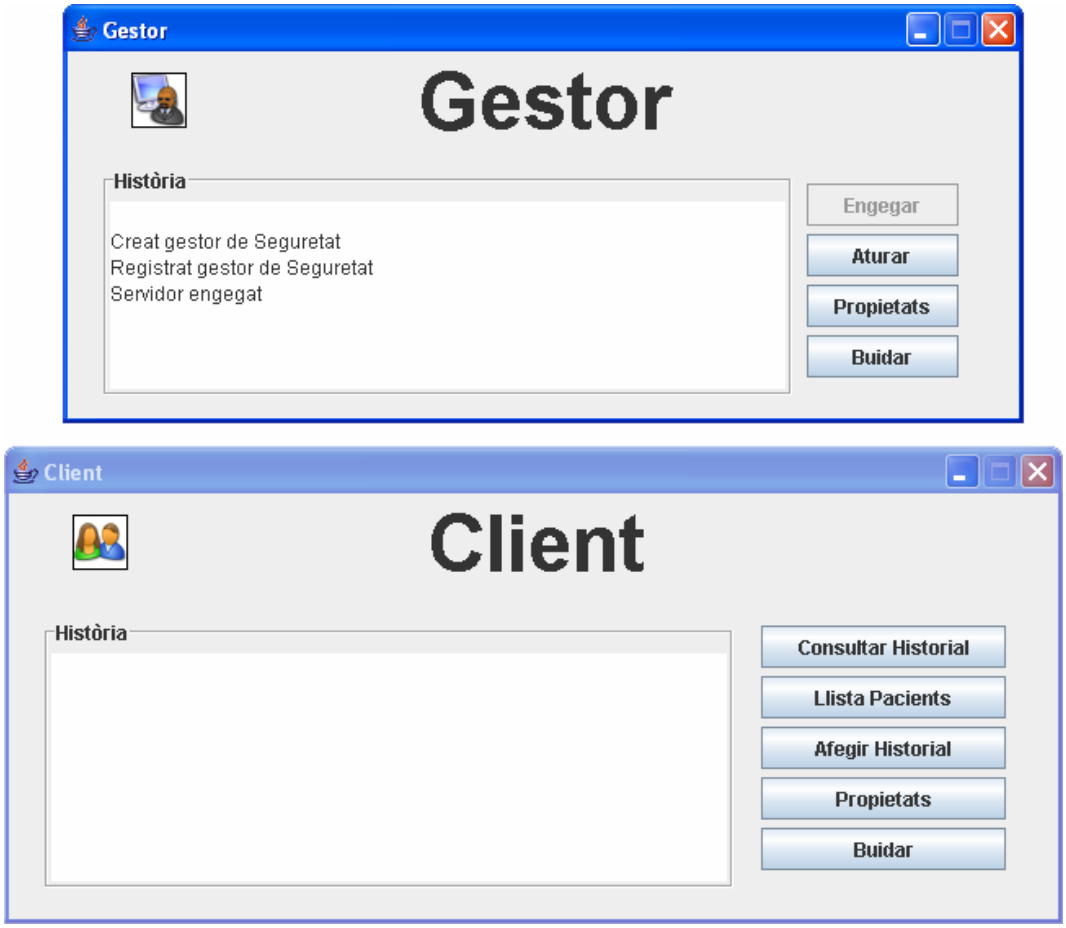
1) Primer verifiquem que el servidor de base de dades està actiu.
   a) `Windows: Taulell de control -> Eines administratives -> Serveis`
   b) `Linux: > /etc/init.d/mysqld start`

2) Si no s'ha executat l'script de creació de la base de dades ara és el moment:
   a) `> mysql -u root -p`
   b) `> source creacio_script.txt`
   c) `> quit`

3) Arrancar el servidor RMI
   d) `Windows: > start rmiregistry`
   e) `Linux: > rmiregistry &`

4) Arrancar l'aplicació client i servidora.
   f) `java hce.Gestor`
   g) `java hce.Client`

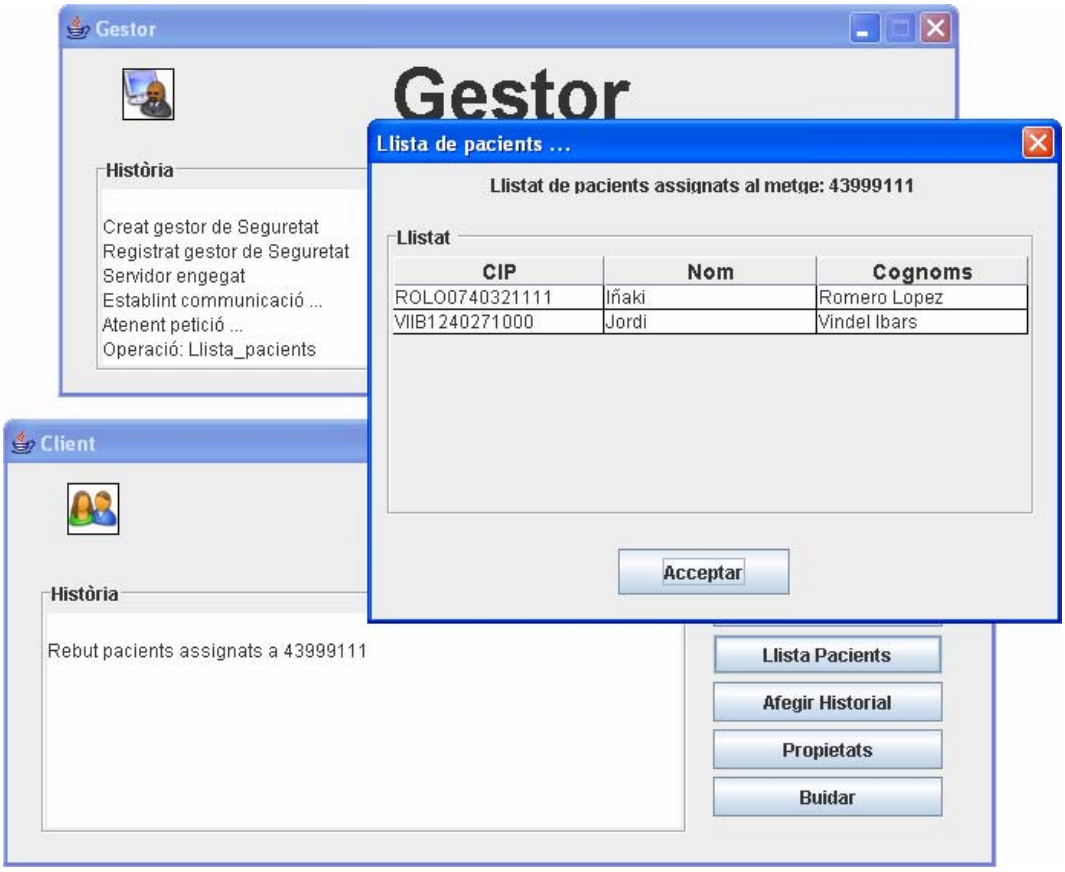
5) Engenar el servidor.



Il·lustració 22.- Demostració: Interfície client i gestor

6) Executar l'operació desitjada, en aquest cas obtenir llista de pacients d'un metge:

El resultat obtingut és el següent:



Il·lustració 23.- Demostració: resultat de l'execució.

7) Per finalitzar podem tancar els aplicatius. En el cas del gestor, aquest demanarà al usuari que s'autentiqui per verificar si aquest té permís per realitzar l'operació sol·licitada.