Implementació d'un esquema criptogràfic per gestionar remotament de forma segura els historials mèdics dels pacients.

Estudiant: Vicent Garcia Mesas
Enginyeria en Informàtica

Consultor: Jordi Castellà Roca

11 gener 2009

Dedicatòria i agraïments.

Dedico aquest projecte a la meva parella, Josefa, per la seva paciència i suport incondicional. A més a més agraeixo al meu consultor la seva ajuda i dedicació.

Resum.

La tecnologia cada vegada arriba a límits més enllà del que ens pensem. Avui dia podem disposar de la informació que vulguem en qualsevol lloc del món amb l'ajuda d'Internet i les xarxes de comunicacions.

Les persones no estem sempre en el mateix indret atès que amb les comunicacions que tenim ens permet viatjar ràpidament d'un lloc a un altre. Doncs bé, si en un d'aquests viatges tenim un accident, seria de gran utilitat disposar del nostre historial mèdic en qualsevol indret. No obstant, atès que les dades mèdiques d'una persona són confidencials, han de viatjar xifrades per la xarxa pública.

Aquest projecte de final de carrera del segon cicle d'enginyeria informàtica presenta una implementació d'un esquema criptogràfic basat en PKI (Public Key Infrastructure) per a gestionar d'una manera segura dins una xarxa de comunicacions els historials mèdics dels pacients. Aquests historials, viatjaran xifrats per la xarxa i a més podrem assegurar que només han estat manipulats o visualitzats per usuaris amb accés, ja sigui el doctor, el pacient o el gestor de l'aplicació.

Finalment, per a efectuar el projecte, s'han utilitzat diferents eines i tecnologies de caire obert com poden ser el llenguatge de programació orientat a objectes Java, les llibreries criptogràfiques IAIK, l'estàndard XML com a model de representació de dades, JDOM per a la seva gestió, RMI per a establir les comunicacions i MySQL com a gestor de la base de dades de l'aplicació.

Índex de continguts.

Índex de figures.

1. Introducció.

1.1 Justificació del projecte.

Tenim la sort de viure una època on tecnològicament s'estan obtenint grans avenços i, el que fa uns anys podia semblar una història de ciència ficció, avui dia pot ser una realitat. Un d'aquests avenços és la comunicació de dos punts llunyans mitjançant les comunicacions distribuïdes i la xarxa.

La sanitat, un tema que sembla de gran preocupació per a les persones, pot veure's beneficiada d'aquests avenços tot i que, per a utilitzar aquestes tecnologies, és de vital importància treballar amb comunicacions segures. Un d'aquests avenços pot ser la possibilitat de gestionar dins una xarxa de comunicacions, que per definició és una zona insegura, els historials mèdics de les persones.

Si peguem una ullada a la LOPDP RD 994/1999 (Llei Orgànica de Protecció de Dades Personals), les dades que podem obtenir a partir d'un historial mèdic, es consideren de nivell alt i la transmissió d'aquests en un sistema de telecomunicacions ha de ser mitjançant sistemes que permetin xifrar el seu contingut. Per aconseguir aquesta fita, disposem d'un tipus de criptografia anomenada de clau pública que permet enviar d'un lloc a un altre d'una xarxa un contingut xifrat de manera que podem assegurar que ningú no autoritzat llegirà la informació i, a més a més, podem assegurar que el contingut ho ha estat manipulat.

1.2 Objectius.

L'objectiu principal d'aquest projecte és implementar una aplicació distribuïda segura mitjançant un esquema criptogràfic que permeti gestionar els historials mèdics dels pacients dins una xarxa de comunicacions. Els components que s'obtenen són els següents:

- **Aplicació Metge.** Permet que un metge pugui consultar i modificar l'historial d'un pacient de forma segura.
- **Aplicació Pacient.** El pacient utilitzarà aquesta aplicació per consultar les dades del seu historial.
- **Gestor central.** El gestor central és qui té el repositori amb tots els historials i en controla la seva gestió.

Per a xifrar el contingut dels historials, s'utilitza el que s'anomena una infraestructura de clau pública PKI utilitzant certificats digitals per a cada usuari que vulgui operar amb el sistema. Es treballa amb un model client servidor per a efectuar la comunicació i intercanvi de dades, on per una banda tindrem els aplicatius del metge o pacient i per l'altra el servidor que s'encarrega de gestionar les peticions d'aquests i cedir les dades emmagatzemades de manera persistent en una base de dades.

1.3 Enfocament i mètode seguit.

El projecte s'ha dividit en una sèrie de fases relacionades entre elles de manera que possibilita fer una implementació incremental. És a dir, s'implementa el codi corresponent a una fase i es prova el seu correcte funcionament, s'implementa el codi de la fase següent, es torna a provar mitjançant tests unitaris i s'integra amb la fase anterior. Aquesta metodologia la utilitzarem en totes les fases.

Les fases de que consta aquest projecte estan perfectament definides i delimitades. Són les següents:

- Instal·lació IAIK i PKI.
- Implementació dels protocols.
- Implementació de la representació de les dades (XML).
- Comunicació dels components (RMI).
- Gestió de la informació (BBDD) i registre d'usuaris.
- Interfície client.
- Interfície gestor.
- Documentació, proves finals d'integració i instal·lador de l'aplicatiu.

1.4 Planificació.

La planificació temporal de les tasques exposades en el punt anterior es pot veure en la taula següent. Cada fase s'ha considerat una prova d'avaluació continuada (PAC) obtenint una valoració per part del consultor. Això és molt positiu atès que es pot veure des d'un principi si es porta el projecte per una línia adequada. El temps indicat per a la realització de cada PAC és estimat, és a dir, s'ha fet una estimació temporal del cost per cada pac en funció de la dificultat i del temps disponible per a treballar-la.

La planificació temporal és la següent:

SETMANA	DATA INICI	DATA FI	FASE	PAC
1	17/09/08		Instal·lació IAIK i PKI	PAC1
2		29/09/08		PAC1
3	30/09/08		Implementació dels protocols	PAC2
4				PAC2
5				PAC2
6		26/10/08		PAC2
7	27/10/08		Implementació de la representació de les dades	PAC3
8				PAC3
9		16/11/08		PAC3
10	17/11/08	23/11/08	Comunicació dels components	PAC4
11	24/11/07		Gestió de la informació i registre d'usuaris	PAC5
12		7/12/08		PAC5
13	08/12/08		Interfície client	PAC6
14		21/12/08		PAC6
15	22/12/08	28/12/08	Interfície gestor	PAC7
16	29/12/08		Documentació, proves finals d'integració i instal·lador de l'aplicatiu.	PAC8
17		11/01/09		PAC8

Figura 01: Planificació temporal.

1.5 Productes obtinguts.

El projecte consta dels següents components o productes:

- **Aplicatiu del pacient.** És el programari que utilitza un pacient per accedir de manera segura al gestor del sistema. Les funcionalitats bàsiques que l'aplicatiu permet dur a terme són les següents:
  - Autenticar al pacient contra el gestor del sistema.
  - Realitzar una consulta del seu expedient.
  - Sol·licitar un servei.
  - Abandonar de forma segura el sistema.
- **Aplicatiu del metge.** És el programari que utilitza un metge per accedir de manera segura al gestor del sistema. Les funcionalitats bàsiques que l'aplicatiu permet dur a terme són les següents:
  - Autenticar al metge contra el gestor del sistema.
  - Realitzar una consulta de l'expedient d'un dels seus pacients.
  - Obtenir una llista dels seus pacients.

- Introduir dades d'una nova visita a l'expedient d'un dels seus pacients.
- Obtenir dades d'una visita d'un pacient en concret.
- Abandonar de forma segura el sistema.

- **Aplicatiu del gestor del sistema.** És el programari que gestiona el repositori d'historials mèdics de forma central. Les funcionalitats bàsiques que l'aplicatiu permet dur a terme són les següents:

  - Registrar a nous usuaris (metges o pacients).
  - Autenticar als pacients i als metges que volen accedir al repositori.
  - Acceptar les consultes dels metges i dels pacients.
  - Guardar de forma segura els historials mèdics dels pacients.
  - Verificar que les dades que s'han inserit o modificat en un historial mèdic s'ha realitzat per un usuari autoritzat.
  - Permetre que els usuaris abandonin el sistema de forma segura.

1.6 Descripció dels altres capítols de la memòria.

Estudi de les necessitats del sistema.
En aquest apartat és dóna a conèixer les característiques que té el sistema i es fa un estudi de les necessitats que ha de tenir a fi i efecte d'obtenir un producte acord amb el que es demana.

Instal·lació IAIK i PKI.
Detall i documentació de la instal·lació del programari bàsic per efectuar el projecte. El JDK de SUN, les llibreries criptogràfiques de IAIK, les polítiques de seguretat, creació de les claus i certificats dels usuaris.

Implementació dels protocols.
Es defineixen els protocols criptogràfics necessaris per a portar a terme cada acció necessària del projecte i es documenta la seva implementació.

Implementació de la representació de les dades (XML).
Es defineixen les estructures necessàries per a efectuar la transferència de dades entre els components del sistema. S'ha optat per utilitzar aquesta representació atès que s'ha imposat com una de les formes més eficients per intercanviar i emmagatzemar dades entre aplicacions i/o protocols.

Comunicació dels components (RMI).
Es defineix l'estructura necessària per a efectuar la comunicació remota del components. Per fer-ho utilitzarem RMI, que es troba incorporada a l'API

estàndard de Java. Aquesta tecnologia consta d'un servidor que publica els serveis que el client necessita i al qual accedeix de manera transparent.

Gestió de la informació (BBDD) i registre d'usuaris.
Es defineix el model de dades a utilitzar per a la persistència de les dades que utilitza l'aplicació, la creació i les regles d'accés. Per portar-ho a terme, s'utilitzarà MySQL.

Interfície client.
Desglossament de les funcionalitats que ha d'oferir la interfície client en funció del tipus de client, ja sigui un doctor o un pacient.

Interfície gestor.
Desglossament de les funcionalitats que ha d'oferir la interfície gestor per a portar a terme la seva tasca.

2. Estudi de les necessitats del sistema.

2.1. Introducció.

En aquest capítol s'intenta deixar clar les necessitats que ha de satisfer l'aplicació a implementar. Per fer-ho, prèviament s'han d'identificar els actors que interactuen amb el sistema, les accions o casos d'ús que ha de fer cada actor i els serveis que ha d'oferir l'aplicació. Tot això sota una plataforma segura utilitzant criptografia de clau pública a fi i efecte de donar-li seguretat al sistema i a les transaccions de dades entre els diferents components i actors que intervenen.

2.2. Actors implicats.

Els actors que es preveu d'inici que interactuaran amb el sistema són tres: els metges, els pacients i un gestor central del sistema. Es podria afegir altres usuaris tals com infermeres, personal d'administració, etc. però atesa la particularitat del projecte que es centra en la seguretat informàtica, s'ha considerat que els actors citats eren suficients per desenvolupar l'esquema que es vol implementar en aquest PFC, doncs, el temps disponible per realitzar el PFC és d'un quadrimestre.

A continuació s'explica les característiques de cada actor:

- **Els metges.** Aquest actor representa un col·legiat el qual interactua amb el sistema per demanar dades i consultes referents dels seus pacients i també dels historials de cada pacient disponible. A més a més aquest actor té la capacitat d'afegir una visita amb un pacient en concret al sistema. El sistema ha d'evitar que un metge pugui modificar les dades registrades sobre visites, diagnosi i tractaments ja afegides i també ha d'evitar que elimini dades d'un historial.

- **Els pacients.** Aquest actor representa un pacient donat d'alta al sistema el qual té un historial clínic i un o varis metges assignats. El sistema ha de permetre una vegada autentificat com a tal consultar el seu historial i les dades generals que es disposa d'ell. En el cas que el pacient hagi de modificar alguna dada, canvi d'adreça, telèfon, etc. seria el gestor del sistema qui té la potestat de fer-ho.

- **Gestor del sistema.** Aquest actor representa l'administrador del sistema. Serà qui s'encarrega de l'autenticació dels altres usuaris i la gestió dels diferents historials i dades que manipula el sistema tot facilitant el contingut als usuaris que així ho demanin sempre i quan tinguin accés als mateixos.

2.3. Serveis que ha d'oferir el sistema.

Els actors citats en el punt anterior interactuaran amb el sistema per a obtenir i modificar la informació. Per efectuar aquesta interacció el sistema oferirà els següents serveis: registre dels usuaris, autenticació, consultes diverses, afegir visites als historials, modificació de dades dels pacients i eliminació de dades.

A continuació s'explica cada servei:

- **Registre d'usuaris.** El registre d'usuaris és un servei que dóna el sistema a fi i efecte de registrar un usuari en concret per a que pugui interactuar, demanar o modificar dades existents al sistema. Aquest servei només el podrà utilitzar el gestor.

- **Autenticació dels usuaris.** L'autenticació dels actors és important que sigui d'una forma segura per evitar accessos no desitjats al sistema i permetre que cada usuari tingui accés als serveis que realment pot tenir. Per tant el sistema ha d'oferir el servei d'autentificació segura mitjançant un nom d'usuari i una clau prèviament donat d'alta al sistema amb el registre anteriorment citat. Per fer l'autenticació dels usuaris hi ha dues possibilitats fer-ho al principi de la connexió o fer-ho cada cop que es demani un servei. Autenticar-se un únic cop al principi de la connexió necessita mantenir un estat de connexió de manera que el gestor pugui saber a quin tipus d'usuari correspon. L'altra possibilitat és autenticar-se cada vegada que es sol·licita un servei al sistema. Tot i que la primera opció és computacionalment més eficient, s'ha optat per utilitzar la segona per evitar una càrrega de treball excessiva atès el temps limitat que es disposa per fer aquest PFC.

- **Consultes.** Les consultes són un servei important que ha d'oferir el sistema on en funció de l'actor que interactuï, es permetrà o no l'accés a les dades. Les consultes poden ser de varis tipus: consulta de les visites d'un pacient, consulta de les dades generals d'un pacient i consulta de la llista de pacients.

    - **Consulta de les visites d'un pacient**. Aquest tipus de consulta la pot demanar tant el pacient implicat com el metge que té assignat. No es preveu el fet que un metge pugui visualitzar l'historial d'un pacient que no té assignat. En el cas que el metge del pacient vulgui sol·licitar l'opinió d'un altre metge sobre l'historial, s'haurà de fer fora de l'àmbit de l'aplicació o bé mitjançant el nom d'usuari i clau del metge implicat.

    - **Consulta de les dades generals d'un pacient**. Aquest tipus de consulta la pot demanar tant el pacient implicat com el metge que té assignat. A més a més el sistema ha de poder donar accés a aquestes dades al gestor per fer modificacions o eliminacions de dades.

- **Consulta de la llista de pacients**. Un metge pot obtenir sempre que ho demani la llista dels pacients que té assignats a fi i efecte d'efectuar les gestions que té assignades tals com afegir/visualitzar visites a un historial o visualitzar les dades generals d'un pacient.

- **Afegir visites als historials.** El sistema ha d'oferir als metges la possibilitat d'afegir una visita a un historial d'un pacient. Una vegada s'ha guardat una visita al sistema, aquesta no podrà ser modificada una altra vegada pel metge. En el cas que hagi un motiu justificat per la seva modificació, hauria de ser el gestor del sistema, prèvia autorització de les parts, qui s'encarregui de canviar les dades errònies o la seva eliminació de l'expedient.

- **Modificació de dades dels pacients.** La modificació de les dades dels pacients només la pot portar a terme el gestor del sistema prèvia autorització de l'implicat. Per tant el sistema oferirà aquest servei en exclusiu al gestor. L'autorització de la modificació per part de l'implicat en principi es preveu per escrit per tant queda fora d'aquest servei el fet de registrar-ho dins la base de dades del sistema.

Per tenir una idea més acurada de les accions i actors que té el sistema, a continuació s'il·lustren els casos d'ús que disposa cada actor:

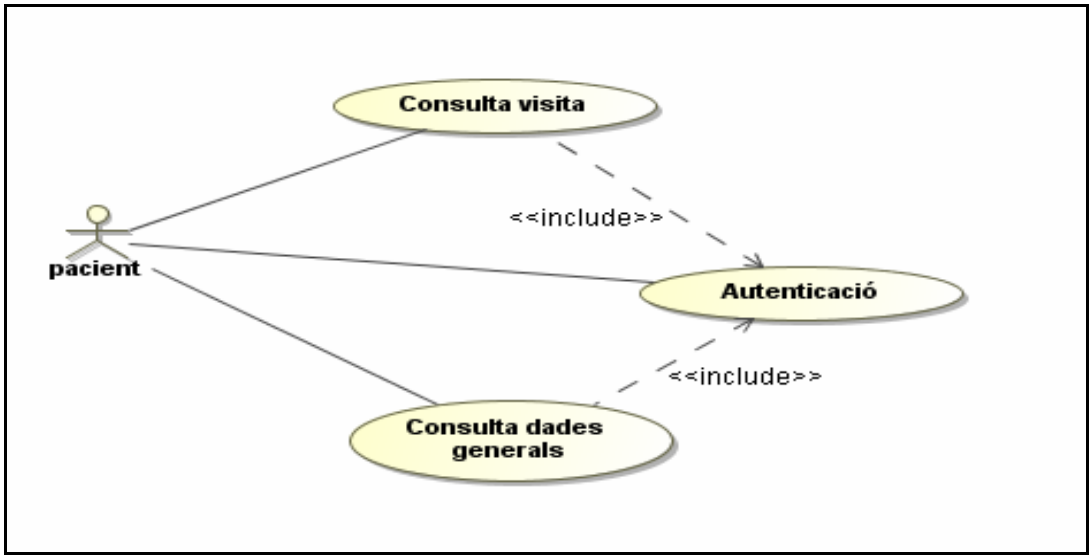

Figura 02: Cas d'ús del pacient.

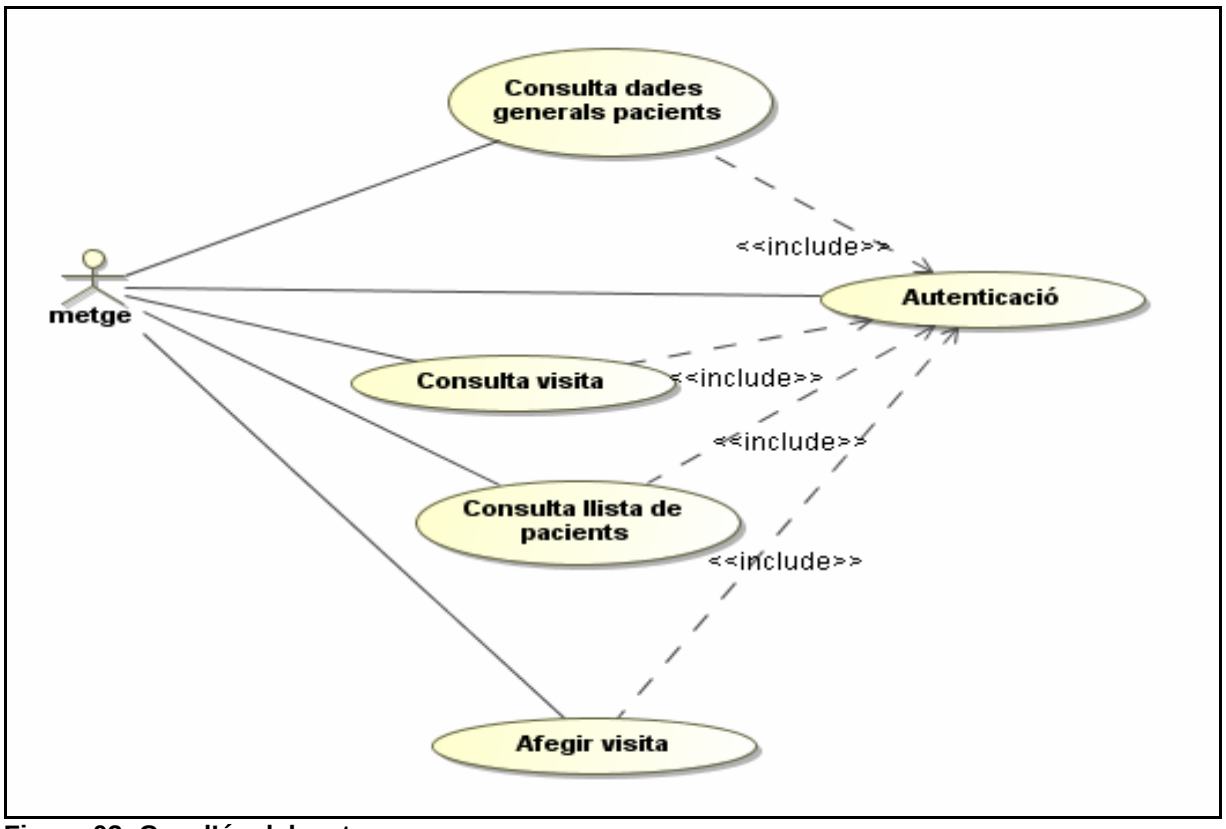

Figura 03: Cas d'ús del metge.

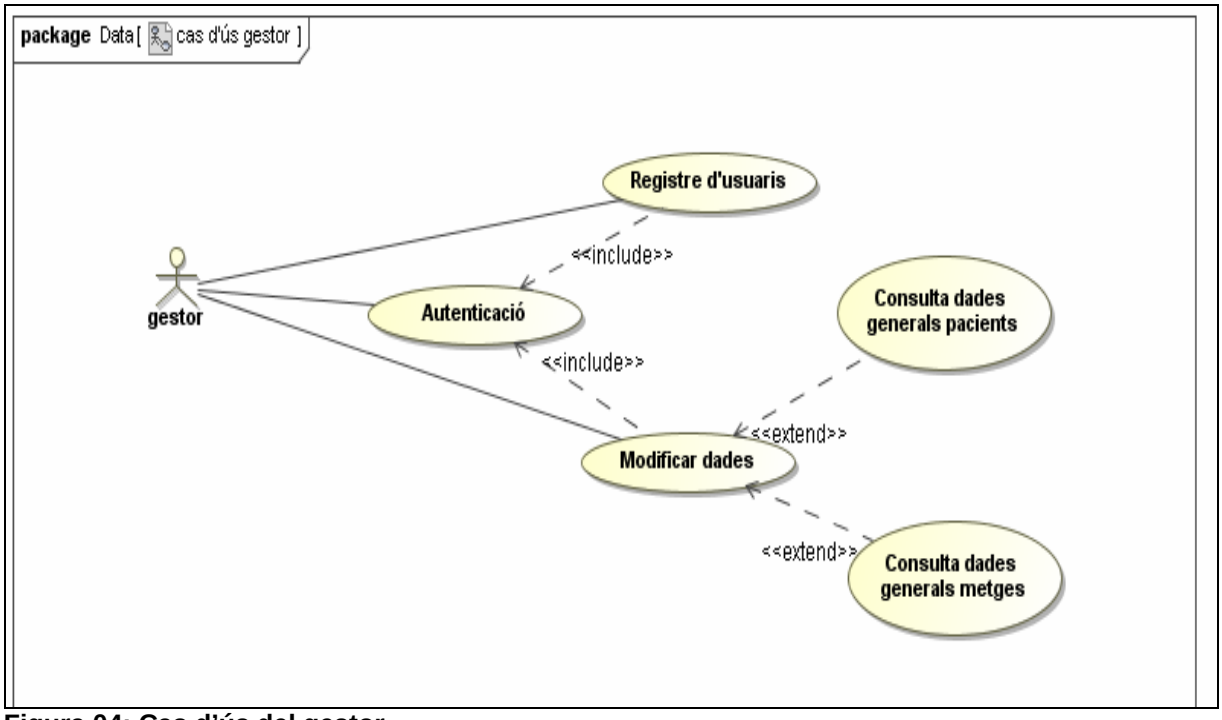

Figura 04: Cas d'ús del gestor.

2.4. Gestió i organització de la informació a tractar.

Atès que la quantitat d'informació que es pot generar es preveu molt gran, la gestió i organització de la mateixa ha d'estar perfectament definida i estructurada per a facilitar l'accés i el correcte funcionament del sistema. La peça central d'aquesta informació és l'historial mèdic. Aquest document pot contenir informació diversa i el nivell de confidencialitat d'aquestes dades és diferent. Per exemple, les dades generals d'un pacient tals com nom, grup sanguini, al·lèrgies, etc. poden ser consultades per qualsevol metge. En canvi, les dades referents a una visita, trets característics d'un pacient tals com rebre ajuda psicològica, etc. només hauria de ser accessible pel seu metge.

L'organització de les dades d'un historial doncs té tres blocs diferenciats: **Dades generals, llista de visites protegida** i **llista de metges protegida.**

- **Dades generals.** Les dades generals que consta l'historial són el nom del pacient, els seus cognoms, el número de la targeta sanitària, el seu DNI i el grup sanguini. Aquestes dades estaran guardades a la base de dades tal qual sense xifrar i només seran accessibles pel pacient implicat, el gestor del sistema i els metges que constin en la llista de metges protegida.

- **Llista de visites protegida.** Dins d'aquest bloc només s'especifica una llista de descriptors de visita prèviament xifrat i una llista d'accés també xifrada on s'especifica la clau i els metges que tenen accés a les visites especificades. Organitzar d'aquesta manera les dades evitem que es pugui relacionar una visita guardada en clar a la base de dades amb el pacient implicat. A més a més podem fer una classificació dels metges que poden visualitzar una visita en concret. El descriptor de la visita serà doncs un arxiu que conté un codi aleatori, la data, l'hora, el lloc i el metge que l'ha creat. Aquest descriptor es vincula a una visita de la base de dades.

- **Llista de metges protegida.** Aquest camp conté la llista de metges que poden tenir accés a l'historial i fins quan hi poden accedir. Per motius de seguretat i atès que només hi pot accedir el gestor del sistema, es xifra amb la clau pública del gestor.

Les visites que han realitzat els metges, tal i com s'ha comentat abans, estan emmagatzemades a la base de dades. Les dades que consta una visita són les següents: **Descriptor de visita, dades de la visita** i **signatura digital**.

- **Descriptor de visita**. Tal i com s'ha indicat abans, aquest descriptor consta d'un identificador únic i de la informació bàsica de la visita.

- **Dades de la visita**. Les dades de la visita consta de l'**anamnesi** que són les dades o informació rellevant sobre el pacient. En aquest bloc es troba els següents camps: causa de la visita, antecedents personals i antecedents familiars. El **diagnosi** que és el reconeixement del metge del problema del pacient i el **tractament** que el metge indica per afrontar la malaltia.

- **Signatura digital**. Aquesta signatura la fa el metge sobre el descriptor de la visita i les dades a fi i efecte d'evitar el repudi del metge en front a les dades creades.

A banda dels continguts ja explicats, s'ha de tenir sota persistència amb una base de dades la informació rellevant sobre els metges. Les dades que ens interessa són: el nom del metge, els seus cognoms, número de col·legiat, DNI i el seu certificat digital. A més a més també s'ha de guardar la llista dels seus pacients la qual esta xifrada amb la clau pública i signada amb la clau privada del gestor.

2.5. Requisits de seguretat.

Atesa la particularitat del projecte, hi ha parts de la informació que gestiona l'aplicació que s'ha de fer de manera segura per evitar intrusions i accessos no desitjats. A més a més s'ha d'assegurar que les dades que es faciliten no han estat alterades o modificades. Per portar a terme aquesta tasca, prèviament s'han d'identificar els quatre conceptes clau de seguretat de la informació: **confidencialitat, integritat, autenticitat** i **no-repudi**.

- La **confidencialitat**: És la propietat que assegura que només aquells que estan autoritzats tindran accés a la informació. Sovint aquesta propietat es coneix també amb el nom de privadesa.

- La **integritat:** És la propietat que assegura la no-alteració de la informació. Aquesta alteració pot ser inserció, esborrament o substitució de la informació.

- L'**autenticitat**: És la propietat que fa referència a la identificació. És el nexe d'unió entre la informació i l'emissor d'aquesta.

- El **no-repudi**: És la propietat que preserva que alguna de les parts negui algun compromís o acció pres amb anterioritat.

L'aplicació respon de la següent manera als conceptes de seguretat exposats:

La part de confidencialitat està latent en tota la gestió de l'aplicació utilitzant la infraestructura de clau pública per a decidir qui té accés a les dades. Partint del tipus d'usuari que interacciona amb el sistema, les dades es xifren amb la clau pública de l'actor de manera que només pot obtenir el contingut en clar amb la clau privada que es troba en el seu poder. Les dades que s'han de preservar són per exemple els historials mèdics, les visites dels metges, la llista de metges de cada historial i la llista de pacients de cada metge.

La part de integritat es soluciona utilitzant les signatures digitals. La idea és que l'usuari implicat signa el contingut de les dades amb la clau privada que té i només ell disposa i qualsevol altre usuari pot verificar la signatura i per tant la integritat de les dades mitjançant la clau publica i coneguda per qualsevol altre usuari que ho demani. No obstant, signar el contingut total d'unes dades sovint és una tasca lenta i computacionalment costosa i el que es fa és signar el resum equivalent d'un missatge obtingut a partir de funcions hash. En l'aplicació, la signatura s'utilitza per a signar el contingut de les visites que crea un metge per a assegurar la integritat de les mateixes, tanmateix s'utilitza la signatura digital en les transaccions i emmagatzematge de dades del gestor als diferents actors que ho demanin i a la base de dades. La llista de descriptors, la llista de metges d'un historial i la llista de pacients d'un metge han d'estar signades digitalment per assegurar la seva integritat.

La part d'autenticació és duu a terme inherentment a l'utilitzar un criptosistema de clau publica. L'accés a les dades és fa prèvia autenticació de l'actor amb la seva clau per evitar que una persona malintencionada pugui visualitzar contingut protegit. A més a més, les dades emmagatzemades han de disposar d'una prova d'autenticitat per assegurar que el contingut registrat a la base de dades és realment el mateix que es va guardar en el seu moment. Per aconseguir això s'utilitza els certificats digitals, el xifratge de les dades amb la clau privada del gestor, les signatures digitals i els sobres digitals que es creen per a preservar l'autenticitat de la informació.

Finalment, la part de no-repudi, recau sobre el fet que es necessita d'un procediment amb el qual una persona no pot negar l'autoria del contingut d'unes dades. És a dir, en el cas nostre, una vegada un metge ha creat una visita, ha fet un diagnosi i proposat un tractament, aquest no ha de poder negar l'autoria del mateix. Per portar a terme aquesta tasca, també s'utilitza la signatura digital.

3. Instal·lació IAIK i PKI.

3.1. Introducció.

A fi i efecte de proporcionar la funcionalitat que demana l'aplicació, és necessari la instal·lació d'una sèrie de components. Per a utilitzar i gestionar contingut xifrat, crear signatures i desenvolupar tot el sistema criptogràfic, és de gran ajuda utilitzar una llibreria criptogràfica. Per a desenvolupar aquest projecte, s'ha utilitzat la llibreria IAIK. Tot i que no és la única disponible en el mercat, amb aquest llibreria tenim totes les eines que es necessiten per estalviar temps en el desenvolupament i afegir totes les funcionalitats criptogràfiques que demana l'aplicació.

A més a més es fa necessari de disposar d'una infraestructura de clau publica per al correcte funcionament de tot el sistema. Això s'aconsegueix amb la instal·lació de OPENSSL que és un programari que crea i simula una PKI per gestionar certificats digitals.

3.2. Llibreria criptogràfica IAIK.

3.2.1. Definició i arquitectura.

És una llibreria criptogràfica que implementa un conjunt d'APIs i múltiples algoritmes criptogràfics junt amb mètodes per la seva gestió, tot això sota la filosofia de la programació orientada a objectes. Aquesta llibreria està realitzada per ser utilitzada dintre d'aplicacions Java i ser un proveïdor de les funcions criptogràfiques per les classes de Java que treballen amb la seguretat de la informació a partir de les versions Java 1.2 i posteriors.

Entre els múltiples algoritmes que IAIK implementa, s'inclouen els orientats a la generació de claus, simètrica i asimètriques. Codificació i descodificació de Dades, signatura i verificació d'informació, funcions hash, funcions resum, creació de certificats, etc. Entre aquests algoritmes i tècniques podem trobar RSA, SHA, DES, 3DES, MD2, MD5, certificat X509 v1, certificat X509 v2, certificat X509 v3, IDEA, etc.

3.2.2. Instal·lació de la llibreria IAIK.

La instal·lació de la llibreria que es proposa i que de fet s'ha efectuat per a desenvolupar el projecte es fa sota un entorn Windows XP. Els passos que s'han de seguir per a una correcta instal·lació de IAIK són els següents:

1. Descarregar l'última versió del JDK de SUN i instal·lar-lo. Cal vigilar que no s'utilitzi la màquina virtual de MS al compilar o executar. La versió instal·lada en el nostre cas és *jdk.1.6.0_06.*

2. Descarregar la última versió de iaik. Per fer-ho és necessari registrar-se però atès que la seva utilització és educativa no suposa cap cost. Cap la possibilitat de descarregar únicament l'arxiu *iaik_jce_full.jar*, que és la versió completa signada i que proporciona les eines necessàries per al correcte funcionament de l'aplicació. Per accedir a aquest contingut pitgeu sobre l'enllaç següent: enllaç iaik-jce.

3. Descarregar les polítiques de seguretat de java que permeten emprar qualsevol longitud de clau, *Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 6.* Per accedir a aquest contingut pitgeu sobre l'enllaç següent: enllaç a les polítiques de seguretat.

4. Copiar l'arxiu iaik_jce_full.jar als directoris:

   - C:\Archivos de Programa\Java\jdk1.6.0_06\jre\lib\ext
   - C:\Archivos de Programa\Java\jre1.6.0_06\lib\ext

5. Dins de l'arxiu jce_policy-6.zip hi ha els arxius:

   - local_policy.jar
   - US_export_policy.jar

   Copiar-los a:

   - C:\Archivos de Programa\Java\jdk1.6.0_06\jre\lib\security
   - C:\Archivos de Programa\Java\jre1.6.0_06\lib\security

6. Compilar i executar.

3.3. Infraestructura de clau púbica.

3.3.1. Definició.

La criptografia de clau pública permet l'intercanvi de missatges confidencials i íntegres de manera àgil, sempre que disposem de la clau pública de l'interlocutor amb qui ens comuniquem. El problema és com obtenir aquesta clau pública i poder estar segurs que pertany a qui ens pensem.

Les claus públiques són justament això: públiques; i un intrús prou hàbil podria tenir accés al directori on estan ubicades i substituir la clau pública d'algun usuari per la seva pròpia. D'aquesta manera, els interlocutors que utilitzessin la clau pensarien que s'estan comunicant amb aquell usuari quan, en realitat, ho farien amb l'intrús. És el que es coneix com *atac de l'home a mig camí*.

Diffie i Hellman (1976) van pensar que el maldecap de la distribució de claus es resoldria amb un directori segur en línia en el qual s'establís de manera unívoca el lligam entre un nom distintiu d'usuari i una clau pública. El directori públic hauria de signar totes les seves transaccions de manera que ningú no en pogués suplantar la identitat. El problema raïa en el baix rendiment que el sistema oferia en poblacions d'usuaris mitjanes o grans. L. Kohnfelder (1978) es va basar en la idea d'una autoritat central de confiança introduïda per Diffie i Hellman i va proposar crear uns registres de dades signades –els certificats– que permetrien que la distribució de claus es fes des de directoris públics que no requerissin confiança.

La signatura, realitzada per un usuari o entitat externa lleial, assegura la integritat contra una possible modificació no desitjada de les dades. La confiança en el signatari s'estén al subjecte del certificat.

Un certificat digital és una estructura de dades que conté informació del propietari de les claus criptogràfiques, la clau pública en si i una signatura digital dels dos camps anteriors que hi dóna validesa.

D'aleshores ençà, els certificats han esdevingut el principal mitjà de distribució de les claus públiques dels sistemes de clau asimètrica. De tota manera, l'ús de certificats no resol totalment el problema de la distribució de claus, sinó que la trasllada a un nivell superior. Qui ha de signar el certificat? Quins mecanismes són necessaris perquè dos usuaris que no es coneixen puguin assegurar la seva identitat en una comunicació virtual? En definitiva, quin és el model de confiança?

L'objectiu d'una infraestructura de clau pública és la gestió eficient i confiable de les claus criptogràfiques i els certificats perquè es puguin utilitzar per a funcions d'autenticació, integritat, no-repudi i confidencialitat. La infraestructura de clau pública crea un marc segur d'intercanvi de dades en un entorn típicament insegur com Internet.

El certificat és l'element central de la infraestructura de clau pública al voltant del qual es crea aquesta infraestructura de suport que abraça serveis com el registre d'usuaris, l'emissió de certificats, la seva distribució des de directoris públics, la seva renovació i revocació, la recuperació de claus, etc.

3.3.2. Arquitectura d'una PKI.

Els components d'una infraestructura de clau pública són els següents: **L'autoritat de certificació, l'autoritat de registre, els subscriptors i entitats finals, els usuaris, els repositoris, les llistes de revocació de certificats, l'autoritat de validació i l'autoritat de segellat en el temps**. Passem a comentar cadascun d'ells.

- **L'autoritat de certificació (CA): ** És la responsable d'emetre i revocar certificats. És l'entitat de confiança que dóna legitimitat a la relació d'una clau pública amb la identitat d'un usuari o servei.
- **L'autoritat de registre (RA):** És l'encarregada de verificar el lligam entre les claus públiques i la identitat dels seus titulars.
- **Els subscriptors i les entitats finals**: Són aquells que posseeixen un parell de claus (pública i privada) i un certificat associat a la clau pública.
- **Els usuaris:** Són els agents que validen signatures digitals i la seva ruta de certificació a partir de claus públiques emeses per autoritats de certificació de confiança. També poden xifrar documents per a subscriptors i entitats finals.
- **Els repositoris:** Són les estructures encarregades d'emmagatzemar la informació relativa a la infraestructura de clau pública. Els dos repositoris més importants en una infraestructura de clau pública són el repositori de certificats i el repositori de llistes de revocació de certificats.
- **Una llista de revocació de certificats (CRL):** Llista que inclou tots aquells certificats que per diversos motius són invàlids abans de la data de caducitat establerta en el mateix certificat.
- **L'autoritat de validació (VA):** És l'encarregada de comprovar la validesa dels certificats digitals.
- **L'autoritat de segellat de temps (TSA):** És l'encarregada de signar un missatge amb la finalitat de provar que existia abans d'un determinat instant de temps.

3.3.3. OpenSSL.

Per a generar una petita però funcional PKI, utilitzarem OpenSSL. OpenSSL és un projecte de programari desenvolupat pels membres de la comunitat *Open Source* per a lliure descàrrega i està basat en *SSLeay* creat per *Eric Young* i *Tim Hudso*n. Aquesta aplicació es basa en un robust paquet de ferramentes d'administració i llibreries relacionades amb la criptografia, que subministren funcions criptogràfiques a altres paquets como *OpenSSH* i navegadors web (per accessos segurs a llocs HTTPS). Aquestes ferramentes ajuden al sistema a implementar el *Secure Sockets Layer (SSL)*, així com altres protocols relacionats amb la seguretat, como el *Transport Layer Security (TLS).*

Aquest paquet de programari és important per a qualsevol que s'estigui plantejant utilitzar un cert nivell de seguritat en la seva màquina. OpenSSL també ens permet crear certificats digitals.

3.3.4. Generació dels certificats.

Per a que la nostra aplicació funcioni correctament, primer s'ha de proveir d'un certificat digital autosignat i una parella de claus per a l'autoritat de certificació CA i després d'una parella de claus i un certificat per a cadascú dels usuaris que utilitzin el sistema: gestors, metges i pacients.

Una infraestructura de clau pública pot tenir una o més autoritats de certificació. La creació d'una autoritat de certificació comença amb la generació del parell de claus (pública i privada) que s'utilitzaran per a signar i validar els certificats digitals que emeti l'autoritat de certificació. Les claus han de ser prou fortes perquè la probabilitat que un atacant les trenqui sigui extremadament difícil durant el temps de vida dels certificats que s'hi signaran. Això dependrà de la combinació de la longitud de la clau i la qualitat de l'algorisme de generació de claus.

Per a generar aquests certificats amb OpenSSL s'ha seguit els següents passos:

- **Autoritat de certificació 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:
     - La comanda per crear la parella de claus és:

openssl> genrsa -des3 -out CA.key 2048

• La longitud de la parella de claus és de **2048 bits**.
• La parella de claus està xifrada amb **Triple DES**.
• Com a paraula de pas s'ha posat **uoc0809**.
.

2. Crear un certificat autosignat:

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

• Les dades posades al certificat de la CA són les següents:
- Country Name (2 letter code) [AU]:**ES**
- State or Province Name (full name) [Some-State]:**Tarragona**
- Locality Name (eg, city) []:**Sta.Barbara**
- 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 []:**vgarciame@uoc.edu**

• La funció resum (hash) que utilitzem és la **sha1**.

• El nom del fitxer que conté el certificat és: **CA.crt**

3. Es pot verificar que el certificat és correcte amb la comanda següent:

```
openssl> verify -CAfile CA.crt CA.crt
```

- **Parella de claus i certificat per a gestor:**

1. Crear una parella de claus del criptosistema RSA:
• La comanda per crear la parella de claus és:

```
openssl> genrsa -des3 -out Gestor.key 1024
```

• La longitud de la parella de claus és de **1024bits**.
• La parella de claus està xifrada amb **Triple DES**.
• Com a paraula de pas s'ha posat també **uoc0809**.

2. Per a crear una petició de certificat s'utilitza la comanda:

```
openssl> req -new -sha1 -key Gestor.key -out Gestor.csr -config openssl.conf
```

• El fitxer *openssl.conf* es troba als annexos del projecte.

- Les dades posades al certificat del Gestor són les següents:
  - Country Name (2 letter code) [ES]:
  - State or Province Name (full name) [Tarragona]:
  - Locality Name (eg, city) [Sta.Barbara]:
  - Organization Name (eg, company) [UOC]:
  - Organizational Unit Name (eg, section) [PFC_Seguretat]: **Gestio**
  - Common Name (eg, YOUR name) []:**Gestor**
  - D.N.I. or N.S.S.[00000000A]:
  - Email Address []:**vgarciame@uoc.edu**

- La funció resum (hash) que utilitzem és la **sha1**.

- El nom del fitxer de sortida és: **Gestor.csr**

3. Es pot verificar que la petició és correcta amb la comanda següent:

```
openssl> req –in Gestor.csr –verify –text –noout
```

4. Per a emetre el certificat del Gestor s'utilitza la comanda següent:

```
openssl > x509 –req  –in  Gestor.csr –days  180 –CA  CA.crt –Cakey  CA.key
–CAcreateserial  –extfile  openssl.conf  –extensions usr_cert  –out Gestor.crt
```

   - El certificat serà vàlid **180 dies** a partir de la seva creació.
   - El fitxer amb el certificat és: **Gestor.crt**

5. Es pot verificar que el certificat és correcte amb la comanda següent:

```
Openssl> verify –CAfile  CA.crt Gestor.crt
```

6. Finalment es crea el fitxer que conté la clau privada i el certificat en format PKCS#12 amb la comanda següent:

```
openssl>pkcs12 –in  Gestor.crt –inkey  Gestor.key –name Gestor –chain  –CAfile
CA.crt –export  –out  Gestor.p12
```

- El fitxer de sortida és: **Gestor.p12**

- **Parella de claus i certificat per a metge:**

1. Crear una parella de claus del criptosistema RSA:
   • La comanda per crear la parella de claus és:

   openssl> genrsa -des3 -out Metge.key 1024

   • La longitud de la parella de claus és de **1024bits**.
   • La parella de claus està xifrada amb **Triple DES**.
   • Com a paraula de pas s'ha posat també **uoc0809**.

2. Per a crear una petició de certificat s'utilitza la comanda:

   openssl> req -new -sha1 -key Metge.key -out Metge.csr -config openssl.conf

   - El fitxer *openssl.conf* es troba als annexos del projecte.
   - Les dades posades al certificat del Metge són les següents:
     - Country Name (2 letter code) [ES]:
     - State or Province Name (full name) [Tarragona]:
     - Locality Name (eg, city) [Sta.Barbara]:
     - Organization Name (eg, company) [UOC]:
     - Organizational Unit Name (eg, section) [PFC_Seguretat]: **Metges**
     - Common Name (eg, YOUR name) []:**Joaquim**
     - D.N.I. or N.S.S.[00000000A]:**00000001B**
     - Email Address []:**vgarciame@uoc.edu**

   - La funció resum (hash) que utilitzem és la **sha1**.

   - El nom del fitxer de sortida és: **Metge.csr**

3. Es pot verificar que la petició és correcta amb la comanda següent:

   openssl> req –in Metge.csr –verify –text –noout

4. Per a emetre el certificat del Metge s'utilitza la comanda següent:

   openssl > x509 –req –in Metge.csr –days 180 –CA CA.crt –Cakey CA.key –CAcreateserial –extfile openssl.conf –extensions usr_cert –out Metge.crt

   - El certificat serà vàlid **180 dies** a partir de la seva creació.
   - El fitxer amb el certificat és: **Metge.crt**

5. Es pot verificar que el certificat és correcte amb la comanda següent:

```
Openssl> verify –CAfile  CA.crt Metge.crt
```

6. Finalment es crea el fitxer que conté la clau privada i el certificat en format PKCS#12 amb la comanda següent:

```
openssl>pkcs12 –in  Metge.crt –inkey  Metge.key –name Metge –chain  –CAfile
CA.crt –export  –out  Metge.p12
```

- El fitxer de sortida és: **Metge.p12**

- **Parella de claus i certificat per al pacient:**

1. Crear una parella de claus del criptosistema RSA:
   • La comanda per crear la parella de claus és:

```
openssl> genrsa –des3  –out  Pacient.key 1024
```

   • La longitud de la parella de claus és de **1024bits**.
   • La parella de claus està xifrada amb **Triple DES**.
   • Com a paraula de pas s'ha posat també **uoc0809**.

2. Per a crear una petició de certificat s'utilitza la comanda:

```
openssl> req -new -sha1 -key Pacient.key -out Pacient.csr –config
openssl.conf
```

   - El fitxer *openssl.conf* es troba als annexos del projecte.
   - Les dades posades al certificat del Pacient són les següents:
     - Country Name (2 letter code) [ES]:
     - State or Province Name (full name) [Tarragona]:
     - Locality Name (eg, city) [Sta.Barbara]:
     - Organization Name (eg, company) [UOC]:
     - Organizational Unit Name (eg, section) [PFC_Seguretat]: **Pacients**
     - Common Name (eg, YOUR name) []:**Felip**
     - D.N.I. or N.S.S.[00000000A]:**00000002C**
     - Email Address []:**vgarciame@uoc.edu**

   - La funció resum (hash) que utilitzem és la **sha1**.

- El nom del fitxer de sortida és: **Pacient.csr**

3. Es pot verificar que la petició és correcta amb la comanda següent:

```
openssl> req –in Pacient.csr –verify –text –noout
```

4. Per a emetre el certificat del Pacient s'utilitza la comanda següent:

```
openssl > x509 –req  –in  Pacient.csr –days  180 –CA  CA.crt –Cakey  CA.key
–CAcreateserial   –extfile  openssl.conf  –extensions usr_cert  –out Pacient.crt
```

- El certificat serà vàlid **180 dies** a partir de la seva creació.
- El fitxer amb el certificat és: **Pacient.crt**

5. Es pot verificar que el certificat és correcte amb la comanda següent:

```
Openssl> verify –CAfile  CA.crt Pacient.crt
```

6. Finalment es crea el fitxer que conté la clau privada i el certificat en format PKCS#12 amb la comanda següent:

```
openssl>pkcs12 –in  Pacient.crt –inkey  Pacient.key –name Pacient –chain  –CAfile
CA.crt –export  –out  Pacient.p12
```

- El fitxer de sortida és: **Pacient.p12**

4. Esquema criptogràfic.

4.1. Introducció.

Una vegada s'han definit els actors i la seva interacció amb el sistema, els serveis que ha d'oferir l'aplicació tot avaluant els requisits de seguretat necessaris i creat una PKI i certificats digitals per a cada usuari que interacciona, en aquest capítol s'especifica un seguit de protocols i procediments a fi i efecte de proporcionar el nivell de seguretat que demana cada servei.

Atesa l'esquematització utilitzada per explicar cada protocol i procediment, es convenient utilitzar una notació comuna que s'especifica en el punt següent.

4.2. Notació utilitzada.

Per efectuar la descripció dels protocols i els procediments dels apartats següents, s'empra la notació següent:

- N_i i N_G : Valors aleatoris utilitzats per a l'autenticació.
- Id_usuari_u : Nom de l'usuari que demana algun servei al sistema.
- *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 publica 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*, aquesta funció també rep el nom de funcions *hash*.

4.3. Protocols.

4.3.1. Proposta de protocols.

Els protocols proposats per efectuar l'esquema criptogràfic del sistema són els següents:

- **Protocol d'autentificació**. Aquest protocol s'utilitza per a efectuar l'autenticació d'un metge o un pacient al sistema.

- **Protocol de consulta de dades generals d'un pacient**. Aquest protocol s'utilitza per a demanar al gestor del sistema les dades generals que disposa el sistema d'un pacient en concret. El pot utilitzar el propietari de l'historial o un metge.

- **Protocol de consulta d'una visita d'un pacient**. Aquest protocol s'utilitza per a demanar al gestor del sistema les dades que es disposen d'una visita en concret donada d'alta anteriorment. El pot utilitzar el pacient propietari de l'historial o el metge que ha creat la visita.

- **Protocol de consulta dels pacients assignats a un metge**. Aquest protocol s'utilitza per a demanar al gestor del sistema la llista dels pacient d'un metge. El pot utilitzar només el metge i només pot visualitzar la llista dels seus pacients.

- **Protocol d'afegir una visita a un historial mèdic**. Aquest protocol s'utilitza per a inserir dades a un historial. Nomes el pot utilitzar un metge i actuant sobre un historial del qual el pacient estigui dins la llista de pacients assignats al metge.

4.3.2. Protocol d'autentificació.

Aquest protocol implementa el procés d'autenticació creat per *Needham-Schroeder*. Part d'aquest protocol s'utilitza cada vegada que un usuari demana algun servei al sistema. Cada usuari *U* s'identifica amb Id_usuari_U i disposa d'una parella de claus (P_U, S_U) amb el corresponent certificat $Cert_U$ que es troba a la base de dades del sistema. Aquest protocol pot ser utilitzat per un metge o per un pacient.

Els passos a seguir són els següents:

1. *U realitza les operacions següents:*

   I. Executar el procediment 1 amb la clau pública P_G, i obtenir $P_G[N_i, Id\_usuari_U]$;

   II. Enviar $P_G[N_i, Id\_usuari_U]$ a *G;*

2. *G realitza les operacions següents:*

   I. Executar el procediment 2 amb $P_G[N_i, Id\_usuari_U]$, i obtenir $P_U[N_i, N_G, Id\_usuari_G]$;

   II. Enviar $P_U[N_i, N_G, Id\_usuari_G]$ a *U;*

3. *U* realitza les operacions següents:

   I. Desxifrar $P_U[N_i, N_G, Id\_usuari_G]$ amb la clau privada S_U i obtenir N_i, N_G, Id_usuari_G;

   II. Xifrar N_G i N_i amb la clau pública P_G de *G*, $P_G[N_i, N_G]$;

   III. Enviar $P_G[N_i, N_G]$ a *G*;

4. *G* realitza les operacions següents:

   I. Desxifrar $P_G[N_i, N_G]$ amb la clau privada S_G, i obtenir $N_G^{'}$ i N_i ;

   II. si $N_G^{'} \stackrel{?}{=} N_G$,*G* i *U* estan autenticats bilateralment.

4.3.3. Protocol de consulta de dades generals d'un pacient.

Aquest protocol cada usuari *U* s'identifica amb Id_usuari_U i disposa d'una parella de claus (P_U, S_U) amb el corresponent certificat $Cert_U$. En el cas del gestor *G* el seu identificador d'usuari Id_usuari_G és el hash del certificat. Aquest protocol pot ser utilitzat per un metge o per un pacient. *G* verifica en cada cas el tipus d'usuari i només facilita l'historial si l'usuari hi té accés.

- L'usuari demana les seves dades generals.
- L'usuari és un metge.

Els passos a seguir són els següents:

1. *Es realitzen* els dos primers passos del procés d'autenticació mitjançant el protocol 1.

2. Una vegada autenticat, *U* realitza les operacions següents:

   I. Desxifrar $P_U[N_i, N_G, Id\_usuari_G]$ amb la clau privada S_U, i obtenir N_G, $N_i^{'}$ i Id_usuari_G;

   II. Si $N_i^{'} \stackrel{?}{=} N_i$ fer:

      a. Xifrar N_G, N_i, *Consulta_dades_generals* i *Id_usuari* amb la clau pública P_G de *G*, $P_G[N_G, N_i, Consulta\_dades\_generals, Id\_usuari]$. Consulta_dades_generals indica que es volen consultar les dades generals de l'usuari identificat amb *Id_usuari*;

      b. Enviar $P_G[N_G, N_i, Consulta\_dades\_generals, Id\_usuari_U]$ a G;

   III. Sinó retornar error;

3. *G* realitza les operacions següents:

   I. Desxifrar $P_G[N_G, N_i, Consulta\_dades\_generals, Id\_usuari]$ amb la clau privada S_G, i obtenir $N_G^{'}$, N_i ,*Consulta_dades_generals*, *Id_usuari*;

   II. Recuperar N_G de la BD. En el pas 4 del procediment 2 N_G i N_i han estat guardats a la BD;

   III. Si $N_G^{'} \stackrel{?}{=} N_G$ fer:

      a. Si ($Id\_usuari_U \stackrel{?}{=} Id\_usuari$) o ($Id\_usuari_U$ és metge) fer:

         i. Executar el procediment 3 amb *Id_usuari* i P_U, i obtenir $P_U[H]$;

         ii. Enviar $P_U[H]$ a *U*;

      b. Sinó retornar error;

   IV. Sinó retornar error;

   V. Esborrar N_G i N_i de la BD;

4. *U* realitza les operacions següents:

   I. Executar el procediment 4 amb $P_U[H]$ i obtenir H;

   II. Mostrar *H*.

4.3.4. Protocol de consulta d'una visita d'un pacient.

En aquest protocol l'usuari *U* s'identifica amb Id_usuari_U, on l'usuari pot ser un metge o un pacient. *G* verifica en cada cas el tipus d'usuari i només facilita l'historial si l'usuari hi té accés.

- L'usuari demana una de les seves visites.
- L'usuari és un metge que té accés a les visites del pacient.

Els passos a seguir són els següents:

1. *Es realitzen* els dos primers passos del procés d'autenticació mitjançant el protocol 1.

2. Una vegada autenticat, *U* realitza les operacions següents:

   I. Desxifrar $P_U[N_i, N_G, Id\_usuari_G]$ amb la clau privada S_U, i obtenir N_G, $N_i^{'}$ i Id_usuari_G;

   II. Si $N_i^{'} \stackrel{?}{=} N_i$ fer:

      a. Xifrar N_G, *Consulta_visita*, *Id_usuari i descriptor_de_visita* amb la clau pública P_G de *G*:
      $P_G[N_i, N_G, Consulta\_visita, Id\_usuari, descriptor\_de\_visita]$. *Consulta_visita* indica que es vol consultar la visita identificada per *descriptor_de_visita* de l'usuari identificat amb *Id_usuari*;

      b. Enviar $P_G[N_i, N_G, Consulta\_visita, Id\_usuari, descriptor\_de\_visita]$ a G;

   III. Sinó retornar error;

3. *G* realitza les operacions següents:

   I. Desxifrar $P_G[N_i, N_G, Consulta\_visita, Id\_usuari, descriptor\_de\_visita]$ amb la clau privada S_G, i obtenir $N_G^{'}$, N_i, *Consulta_visita*, *Id_usuari, descriptor_de_visita* ;

   II. Recuperar N_G de la BD. En el pas 4 del procediment 2 N_G i N_i han estat guardats a la BD;

   III. Si $N_G^{'} \stackrel{?}{=} N_G$ fer:

      a. Si procediment 5 [Id_usuari_U, *Id_usuari, descriptor_de_visita*] retorna que totes les verificacions són correctes fer:

i. Obtenir la visita identificada per *descriptor_de_visita* (*V*) i obtenir $P_U[V]$;

ii. Enviar $P_U[V]$ a *U*;

b. Sinó retornar error;

IV. Sinó retornar error;

V. Esborrar N_G i N_i de la BD;

4. *U* realitza les operacions següents:

I. Desxifrar $P_U[V]$ i obtenir *V*;

II. Mostrar *V*.

4.3.5. Protocol de consulta dels pacients assignats a un metge.

Amb aquest protocol s'especifica els passos a seguir per a que un metge pugui llistar els seus pacients. La necessitat d'aquest protocol ve definida pel fet que un metge vulgui buscar l'historial d'un dels seus pacients. Doncs el primer pas seria aconseguir el llistat de tots els seus pacients.

Els passos a seguir són els següents:

1. *Es realitzen* els dos primers passos del procés d'autenticació mitjançant el protocol 1.

2. Una vegada autenticat, *U* realitza les operacions següents:

I. Desxifrar $P_U[N_i, N_G, Id\_usuari_G]$ amb la clau privada S_U, i obtenir N_G, $N_i^{'}$ i Id_usuari_G;

II. Si $N_i^{'} \stackrel{?}{=} N_i$ fer:

a. Xifrar N_G *i llista_pacients* amb la clau pública P_G de *G*: $P_G[N_i, N_G, llista\_pacients, Id\_usuari_U]$. *Llista_pacients* indica que es vol un llistat dels pacients del metge identificat amb Id_usuari_U;

b. Enviar $P_G[N_i, N_G, llista\_pacients, Id\_usuari_U]$ a G;

III. Sinó retornar error;

3. *G* realitza les operacions següents:

   I. Desxifrar P_G *[*N_i *,* N_G*, llista_pacients, Id_usuari*$_U$*]* amb la clau privada S_G, i obtenir N_i , N'_G , *Id_usuari*$_U$ i *llista_pacients*;

   II. Recuperar N_G de la BD. En el pas 4 del procediment 2 N_G i N_i han estat guardats a la BD;

   III. Si $N'_G \stackrel{?}{=} N_G$ fer:

      a. Si *Id_usuari*$_U$ *és metge* fer:

         i. Obtenir P_U*[llista_pacients_protegida]*;

         ii. Enviar P_U*[llista_pacients_protegida]* a *U*;

      b. Sinó retornar error;

   IV. Sinó retornar error;

   V. Esborrar N_G i N_i de la BD;

4. *U* realitza les operacions següents:

   I. Desxifrar P_U*[llista_pacients_protegida]* i obtenir *llista_pacients_protegida*;

   II. Tractar la *llista_pacients_protegida i mostrar la llista de pacients a l'usuari*.

4.3.6. Protocol d'afegir una visita a un historial mèdic.

En aquest protocol es suposa que prèviament a la inserció de les dades el metge *M* ha consultat l'historial del pacient *P* i per tant coneix Id_usuari$_P$. Aquest protocol està pensat únicament per afegir una nova visita *V* a l'historial. El gestor un cop rep una visita *V* d'un pacient *P* verifica que ha estat signada pel metge *M* assignat al pacient. A continuació afegeix el descriptor de la visita a la llista de descriptors protegida i la visita a la Base de Dades.

Els passos a seguir són els següents:

1. *Es realitzen* els dos primers passos del procés d'autenticació mitjançant el protocol 1.

2. Una vegada autenticat, *M* realitza les operacions següents:

   I. Desxifrar $P_M[N_i, N_G, Id\_usuari_G]$ amb la clau privada S_M, i obtenir N_G, $N_i^{'}$ i Id_usuari_G;

   II. Si $N_i^{'} \stackrel{?}{=} N_i$ fer:

      a. Obtenir les dades de la visita *V* que ha de generar el metge;

      b. Signar *V* amb la clau privada S_M de *M*, $S_M[V]$;

      c. Xifrar N_i , N_G , *V, Id_usuari* i $S_M[V]$ amb la clau pública P_G de *G*: $P_G[N_i$, N_G, *afegir_visita, V, Id_usuari,* $S_M[V]$ *]*. *Afegir_visita* indica que es vol afegir *V* a l'historial del pacient identificat amb *Id_usuari*;

      d. Enviar $P_G[N_i$, N_G, *afegir_visita, V, Id_usuari,* $S_M[V]$ *]* a G;

   III. Sinó retornar error;

3. *G* realitza les operacions següents:

   I. Desxifrar $[N_i$, N_G, *afegir_visita, V, Id_usuari,* $S_M[V]$ *]* amb la clau privada S_G, i obtenir N_i , $N_G^{'}$, *afegir_visita, V, Id_usuari* i $S_M[V]$;

   II. Recuperar N_G de la BD. En el pas 4 del procediment 2 N_G i N_i han estat guardats a la BD;

   III. Si $N_G^{'} \stackrel{?}{=} N_G$ fer:

      a. Verificar que Id_usuari_M és metge;

      b. Verificar que *Id_usuari* és un pacient assignat a Id_usuari_M *;*

      c. Si les verificacions anteriors són correctes fer:

         i. Verificar la signatura digital $S_M[V]$ amb la clau pública P_M;

         ii. Obtenir el descriptor_de_la_visita de *V*;

         iii. Afegir el descriptor_de_la_visita a la llista_descriptors_de_visita;

iv. Signar amb la clau privada del Gestor P_G la llista_descriptors_de_visita;

v. Xifrar la llista_descriptors_de_visita amb una clau de sessió *K*, E_K(llista_descriptors_de_visita);

vi. Xifrar la clau de sessió *K* amb les claus públiques dels metges que estan a la llista_de_metges de l'historial de l'usuari identificat amb *Id_usuari*, obtenint una nova llista_descriptors_de_visita_protegida;

vii. Afegir *V* a la Base de Dades.

d. Sinó retornar error;

IV. Sinó retornar error;

V. Esborrar N_G i N_i de la Base de Dades;

4.4 Procediments.

4.4.1. Procediment 1:

Aquest procediment conté una part de l'autenticació del protocol de Needham-Schroeder. Aquest passos els utilitzarem en altres protocols. Serà utilitzat pels metges i pacients.

El procediment funciona d'aquesta manera:

1. S'ha de passar com a paràmetres la clau pública del gestor (P_G).
2. Obtenir un valor de forma aleatòria, N_i;
3. Xifrar N_i i Id_usuari_u amb la clau pública de *G*, $P_G[N_i, Id\_usuari_u]$;
4. Enviar $P_G[N_i, Id\_usuari_u]$ a *G*.

4.4.2. Procediment 2:

Aquest procediment conté una altra part de l'autenticació del protocol de Needham-Schroeder. Aquesta part serà executada pel Gestor.

El procediment funciona d'aquesta manera:

1. S'ha de passar com a paràmetres ($P_G[N_i,Id\_usuari_u]$).
2. Desxifrar $P_G[N_i,Id\_usuari_u]$ amb S_G, i obtenir; N_i i Id_usuari_u;
3. Obtenir el certificat de *U* a partir de Id_usuari_u. El sistema disposa d'una Base de Dades on per cada *Id_usuari* trobem el seu certificat corresponent. A partir del certificat es pot obtenir la clau pública P_U;
4. Obtenir un valor de forma aleatòria, N_G;
5. Guardar a la base de dades els valors N_i i N_G associats amb *U*;
6. Xifrar N_i, N_G, Id_usuari_G, amb la clau pública P_U de *U*, $P_U[N_i, N_G, Id\_usuari_G];$
7. Retornar $P_U[N_i, N_G, Id\_usuari_G]$.

4.4.3. Procediment 3:

L'utilitza el gestor G per trobar l'historial que se li ha demanat i xifrar-lo amb la clau de l'usuari que el vol consultar.

El procediment funciona d'aquesta manera:

1. S'ha de passar com a paràmetres (Id_usuari,P_U).
2. Buscar l'historial *H* corresponent a *Id_usuari*;
3. Xifrar *H* amb la clau pública P_U, $P_U[H];$
4. Retornar $P_U[H].$

4.4.4. Procediment 4:

Un usuari utilitza aquest procediment per tal de desxifrar un historial enviat pel gestor *G* i verificar que l'historial és correcte.

El procediment funciona d'aquesta manera:

1. Desxifrar $P_U[H]$ amb la clau privada S_U de *U*, $S_U[P_U[H]]$;

2. Verificar la signatura digital de *G* sobre *H*;

    I. Si la signatura digital de *G* sobre *H* és correcta,retornar *H*;

    II. Sinó retornar error.

4.4.5. Procediment 5:

Aquest procediment s'encarrega de verificar, en el cas que l'usuari sigui un pacient, que la visita correspongui al seu historial mitjançant els descriptors de visita. En el cas que l'usuari sigui un metge es verifica que el pacient estigui dins la seva llista de pacients , que el metge estigui a la llista de metges del pacient i que el descriptor de visita pertanyi al metge.

El procediment funciona d'aquesta manera:

1. S'ha de passar com a paràmetres (*Id_usuari*$_U$*, Id_usuari, descriptor_de_visita*).

2. Si (*Id_usuari*$_U \stackrel{?}{=}$ *Id_usuari*) fer:

    I. Verificar si *descriptor_de_visita* està dins de la *llista_de_descriptors_de_visita_xifrada* de l'usuari identificat per *Id_usuari*$_U$*;*

    II. Retornar el resultat de la verificació.

3. Si (*Id_usuari*$_U$) és un metge fer:

    I. Verificar si *Id_usuari* està dins de la *llista_de_pacients_protegida* del metge identificat per *Id_usuari*$_U$;

    II. Verificar si el metge identificat per *Id_usuari*$_U$ està a la *llista_de_metges* de l'usuari identificat per *Id_usuari*;

    III. Verificar si el *descriptor_de_visita* pertany a l'usuari *Id_usuari* emprant la llista *llista_de_visites_protegida* de l'usuari.

    IV. Retornar el resultat de la verificació.

4.5 Diagrama de classes dels protocols criptogràfics.

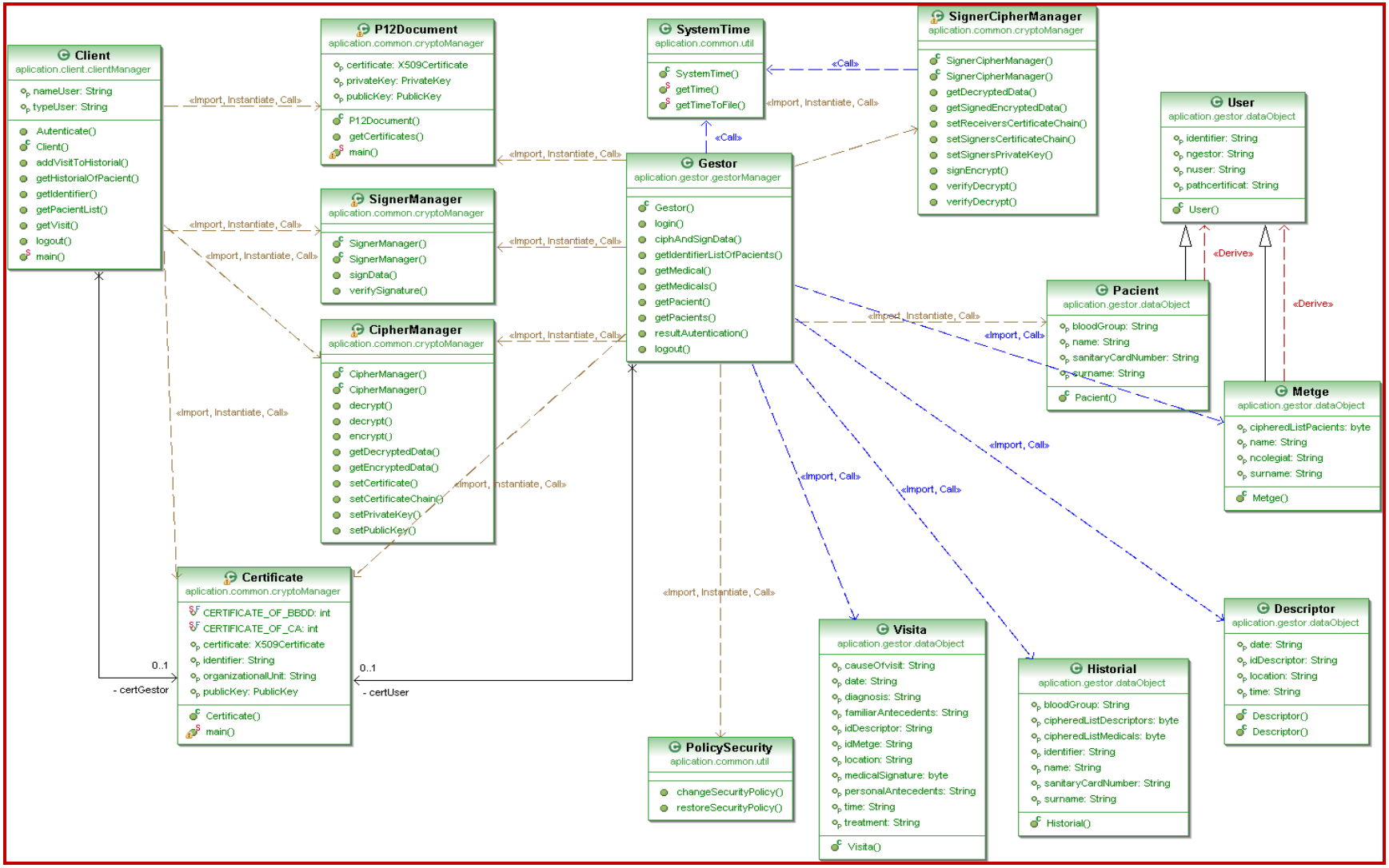

Figura 05: Diagrama de classes dels protocols criptogràfics.

5. Representació de les dades: XML.

5.1 Introducció.

Als anys noranta, per donar resposta a la guerra comercial entre els navegadors web Netscape i Explorer, que provocava greus desviacions del llenguatge HTML respecte del seu propòsit original fins arribar al fet que hi havia contingut web que no era possible representar-ho en segons quin navegador, la comunitat internacional va crear el W3C (*World Wide Web Consortium*), que havia de vetllar per una estandardització real dels formats de la Xarxa. Una de les primeres tasques del W3C fou oficialitzar un versió unificada d'HTML.

Com que l'HTML no podia donar resposta a alguns dels usos que s'esperava de la Xarxa (com ara el comerç electrònic, la cerca d'informació, la personalització, etc.), l'any 1998 el W3C va publicar la versió 1.0 de l'especificació XML (***ex**tensible **m**arkup **l**anguage,* 'llenguatge d'etiquetatge extensible'). L'objectiu d'aquesta especificació és crear un SGML (*Standart Generalized Markup Language*, "llenguatge estandarditzat i generalitzat de marcatge") senzill, és a dir, un subconjunt d'SGML que n'elimini les parts més complexes i l'optimitzi per al seu ús a la Xarxa sense deixar de ser extensible (el nombre d'etiquetes permeses no és tancat sinó que és possible definir-ne de pròpies). La finalitat de tot plegat és la de crear un llenguatge prou senzill perquè la indústria consideri rendible invertir en la creació d'eines per al seu tractament.

Per tant, el llenguatge XML és, segons el Consorci Web, el format universal per a documents estructurats i dades en la web.

5.2. Usos de l'XML.

L'XML s'utilitza per a representar i transportar informació estructurada com la que es pot guardar en una base de dades. A més a més serveix per a representar informació que s'hagi d'ensenyar a persones: el llenguatge HTML es podria redefinir usant la sintaxi de l'XML i usar informació d'estil addicional per a especificar la presentació.

En tots els casos, es tracta d'un format llegible per a expressar dades estructurades. Diversos protocols usen l'XML per a codificar (seriar) les dades que s'intercanvien: per exemple, el protocol WEBDAV, que és una extensió del protocol HTTP per a tractar un servidor web com si fos un servidor de fitxers en xarxa, o el SOAP, que és un mecanisme d'invocació remota d'operacions (també conegut com *serveis web*).

No obstant, l'XML té diverses restriccions:

- És textual i les dades binàries s'han d'enviar codificades en format Base64 o bé enviar-les a part del document XML (usant un enllaç, com fa l'HTML per a les imatges).

- Pot necessitar força text, encara que després es pot comprimir emprant un compressor general com el *gzip*, el *compress*, o amb un compressor específic per a XML com el que proposa el BXML.

5.3. Característiques de l'XML.

Un document XML és text en una codificació de caràcters concreta. El text XML es pot considerar com un conjunt de **dades de caràcter** i **marques**. Les marques són text que comença pel caràcter **<** i acaba pel caràcter **>** i text que comença pel caràcter **&** i acaba pel caràcter **;**. Les dades de caràcter són tot allò que no són marques.

A més alt nivell, un document XML té dues parts, la **capçalera** i la **instància de document**:

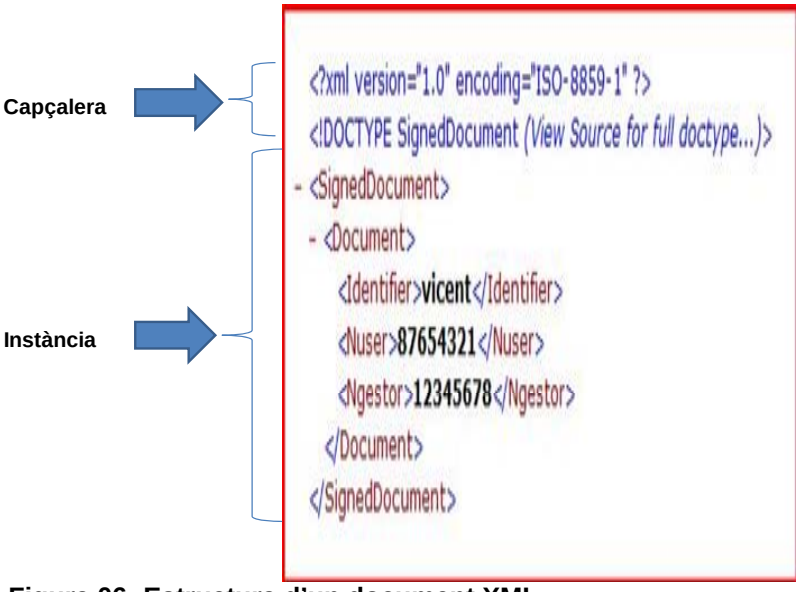

Figura 06: Estructura d'un document XML.

A la de la **capçalera** hi apareix informació per a la gestió del document, com ara el tipus de codificació emprada, la definició del tipus de document, presentació associada, etc.

La **instància de document** representa el contingut real del document i està constituïda per elements, atributs i dades. Rep el nom d'instància pel fet que és una cadena de caràcters representant un exemplar concret d'un document o d'una tipologia determinada d'un document.

Per a descriure els elements s'utilitzen **etiquetes**. Una etiqueta consta del nom del tipus de l'element i, opcionalment, d'una llista d'atributs que descriuen propietats de l'element.
Per a delimitar l'inici i el final del contingut d'un element s'utilitzen marques .

- El principi d'un element es marca de la manera següent:
  <nom_tipus_element [atributs]>

- El final d'un element es marca de la manera següent:
  </nom_tipus_element>

- Els elements buits (sense contingut) es marquen de la manera següent:
  <nom_tipus_element/>

A més a més dels ja exposats, els documents XML han de complir un seguit de normes sintàctiques:

- Els documents XML s'han de crear a partir d'una estructura jerarquitzada.

- Tot document XML ha de tenir un únic element arrel del qual ha de penjar tota la resta de document.

- Una etiqueta buida és per definició una etiqueta que no conté res, per tant és a la vegada d'inici i final.

- Els valors que tenen els atributs han d'anar obligatòriament entre cometes, podent ser simples **‘’** o dobles **“”.**

- Als documents XML es pot distingir entre majúscules o minúscules.

- Els espais en blanc dins d'un document XML són irrellevants i per tant s'eliminen o es normalitzen.

- Com tot llenguatge de marques, XML té caràcters reservats i aquestos, per a poder-los visualitzar dins al contingut reservat a les dades del document, s'han de codificar prèviament.

- El contingut dels documents XML es poden codificar en varis tipus diferents de format, per a poder visualitzar correctament paraules llatines tals com les que van accentuades, és necessari utilitzar la codificació “**ISO-8859-1**”.

5.4. Restricció de documents XML.

Un document que obeeix la sintaxi de l'XML exposada anteriorment es diu que està "**ben format**", a més a més, si el document segueix un conjunt de restriccions addicionals especificades per una comunitat determinada, es pot dir que és "**vàlid**". Les característiques del llenguatge d'una comunitat, la seva gramàtica (elements d'una llengua i les seves combinacions) es poden expressar en una secció del document, o en un document a part, de dues maneres:

- DTD o definició del tipus de document; que és l'únic format que ja hi havia a l'inici de l'XML, heretat de l'SGML; la sintaxi d'una DTD no és en XML, i no té tipus de dades per a restringir els valors.

- Esquema XML: és un format més recent basat en XML, amb tipologia de dades i capacitat per a expressar més restriccions.

Atès que en aquest projecte s'ha utilitzat un DTD a fi i efecte de verificar la correctesa del contingut d'un document XML, aquest tipus de validació s'explica més extensament:

Un **DTD** és el conjunt de normes que s'han de seguir per a crear una representació d'un document XML d'un tipus determinat. Aquestes normes s'expressen mitjançant una notació anomenada declaració de marcatge.

La declaració de marcatge és una notació descrita en l'especificació XML. És una manera d'expressar un conjunt de regles gramaticals que permeten d'indicar el següent:

- Els elements i atributs vàlids dins d'un document XML.

- Els elements que es poden utilitzar dins d'altres elements.

- Els elements i atributs opcionals.

Aquesta notació persegueix dos objectius principals. D'una banda que pugui ser creada i visualitzada d'una manera senzilla i ràpida per una persona. D'altra banda que pugui ser interpretada per una màquina a fi i efecte de permetre la validació automàtica de qualsevol document XML a partir del seu DTD.

Els tipus de dades que poden aparèixer en una DTD es redueixen a text:

- **PCDATA** (*parsed character data)*: text i possibles marques, seqüència de caràcters processats.
- **CDATA** (*character data)*: text, seqüència de caràcters excepte marca de final]]>.

5.5. Documents XML utilitzats per al projecte.

Per a l'intercanvi de la informació que es passen els diferents actors amb el sistema, utilitzarem cinc tipus diferents de documents XML. Per evitar que durant el transport de la informació les paraules accentuades es rebin erròniament, s'ha utilitzat el tipus de codificació **ISO-8859-1** , el qual permet allotjar als documents XML aquests tipus de paraules més pròpies d'idiomes llatins. A més a més com dins del document també s'han d'ubicar dades binàries tals com signatures digitals,dades xifrades,etc., serà necessari fer una transformació prèvia a base64 d'aquest contingut.

Tots els documents responen sobre una mateixa arrel anomenada ***Signed Document***. A partir d'aquesta arrel, es va generant la resta del document en funció de les dades que a d'allotjar i transportar.

5.5.1. Document XML d'autentificació.

El document utilitzat per a portar a terme l'autenticació dels diferents usuaris al sistema és el següent:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE SignedDocument (View Source for full doctype...)>
- <SignedDocument>
  - <Document>
      <Identifier>.</Identifier>
      <Nuser>.</Nuser>
      <Ngestor>.</Ngestor>
    </Document>
  </SignedDocument>
```

Figura 07: Document XML autenticació.

Tal i com es pot observar a la figura anterior, a partir de l'element arrel es deriva un altre anomenat ***Document*** on dins d'ell es troben totes les dades necessàries per a portar a terme l'autenticació. Aquestes dades són:

- **Identifier**. Identificador de l'usuari a autentificar, normalment serà el dni de la persona que es vol autenticar i que prèviament s'ha donat d'alta al sistema
- **Nuser**. Valor aleatori que genera l'usuari per a portar a terme l'autenticació explicada al protocol d'autenticació.

- **Ngestor**. Valor aleatori que genera el gestor del sistema per a portar a terme l'autenticació explicada al protocol d'autenticació.

5.5.2. Document XML per demanar un servei al sistema.

El document utilitzat per un usuari per a demanar qualsevol operació al sistema és el següent :

Figura 08: Document XML demanda de serveis.

Tal i com es pot observar a la figura anterior, a partir de l'element arrel es deriva un altre anomenat ***Document*** on dins d'ell es troben totes les dades necessàries per a portar a terme la demanda d'algun servei. Aquestes dades són:

- **Ngestor**. Valor aleatori generat pel gestor del sistema durant l'autenticació.

- **Identifier**. Aquest camp porta dos tipus de dades segons el tipus d'operació a demanar. Pot ser o bé l'identificador de l'usuari a recuperar certes dades o bé l'identificador del metge que demana un servei.

- **Operation**. Dins d'aquest camp s'indica el tipus d'operació que es vol fer. Els possibles valors són els següents: *consulta_dades_generals*, *consulta_visita*, *llista_pacients*, *afegir_visita*.

- **Dataoperation**. Atès que hi ha certes operacions les quals necessiten d'altres dades per a portar-les a terme, aquestes es posen dins d'aquest camp. Per exemple, en l'operació de *afegir_visita,* és necessari enviar a més a més el contingut de la visita a afegir.

- **DataOperationsignature**. Dins d'aquest camp es posa la signatura digital del creador de les dades que es posen al camp anterior a fi i efecte d'evitar el repudi i assegurar la integritat de les dades.

5.5.3. Document XML per enviar un historial.

El document utilitzat pel gestor del sistema per a enviar a l'usuari l'historial és el següent:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE SignedDocument (View Source for full doctype...)>
- <SignedDocument>
  - <Document>
    - <Historial>
      - <PersonalData>
          <Name>.</Name>
          <Surname>.</Surname>
          <SanitaryCardNumber>.</SanitaryCardNumber>
          <Identifier>.</Identifier>
          <BloodGroup>.</BloodGroup>
        </PersonalData>
      - <Visits>
        - <Descriptor>
            <IdDescriptor>.</IdDescriptor>
            <Date>.</Date>
            <Location>.</Location>
            <Time>.</Time>
          </Descriptor>
        - <Descriptor>
            <IdDescriptor>.</IdDescriptor>
            <Date>.</Date>
            <Location>.</Location>
            <Time>.</Time>
          </Descriptor>
        </Visits>
      </Historial>
    </Document>
    <Signature>.</Signature>
  </SignedDocument>
```

Figura 09: Document XML historial.

Tal i com es pot observar a la figura anterior, a partir de l'element arrel es deriva un altre anomenat ***Document*** on dins d'ell es troben el contingut a enviar a l'usuari de les dades d'un historial. Aquestes dades és divideixen en dos grups, les dades personals (***PersonalData***) i visites(**Visits**).

En l'apartat de dades personals tenim els següents continguts:

- **Name**. Dins d'aquest camp està el nom del propietari de l'historial.
- **Surname**. Dins d'aquest camp estan els cognoms del propietari de l'historial.
- **SanitaryCardNumber**. Número de la targeta sanitària del propietari de l'historial.
- **Identifier**. Identificador de l'usuari, normalment serà el DNI.
- **BloodGroup**. En aquest camp podem trobar el grup sanguini del propietari de l'historial.

En l'apartat de visites tenim un conjunt de descriptors amb els següents continguts cadascun:

- **IdDescriptor**. Dins d'aquest camp es troba un identificador únic relacionat amb una visita. El fet de procedir d'aquesta manera s'ha explicat a l'apartat gestió i organització de la informació a tractar.
- **Date**. Data en que es va portar o es durà a terme la visita. Les visites es visualitzaran tant les que ja s'han efectuat com les que encara estan per efectuar-se.
- **Location**. Lloc on es va portar o es durà a terme la visita. Les visites es visualitzaran tant les que ja s'han efectuat com les que encara estan per efectuar-se.
- **Time**. Hora de la visita. Les visites es visualitzaran tant les que ja s'han efectuat com les que encara estan per efectuar-se.

A més a més, al final del document podem trobar un altre apartat (**Signature**) que conté la signatura digital efectuada pel gestor del sistema per assegurar la integritat de les dades.

5.5.4. Document XML per enviar la llista de pacients.

El document utilitzat pel gestor del sistema per a enviar la llista de pacients d'un metge és el següent:

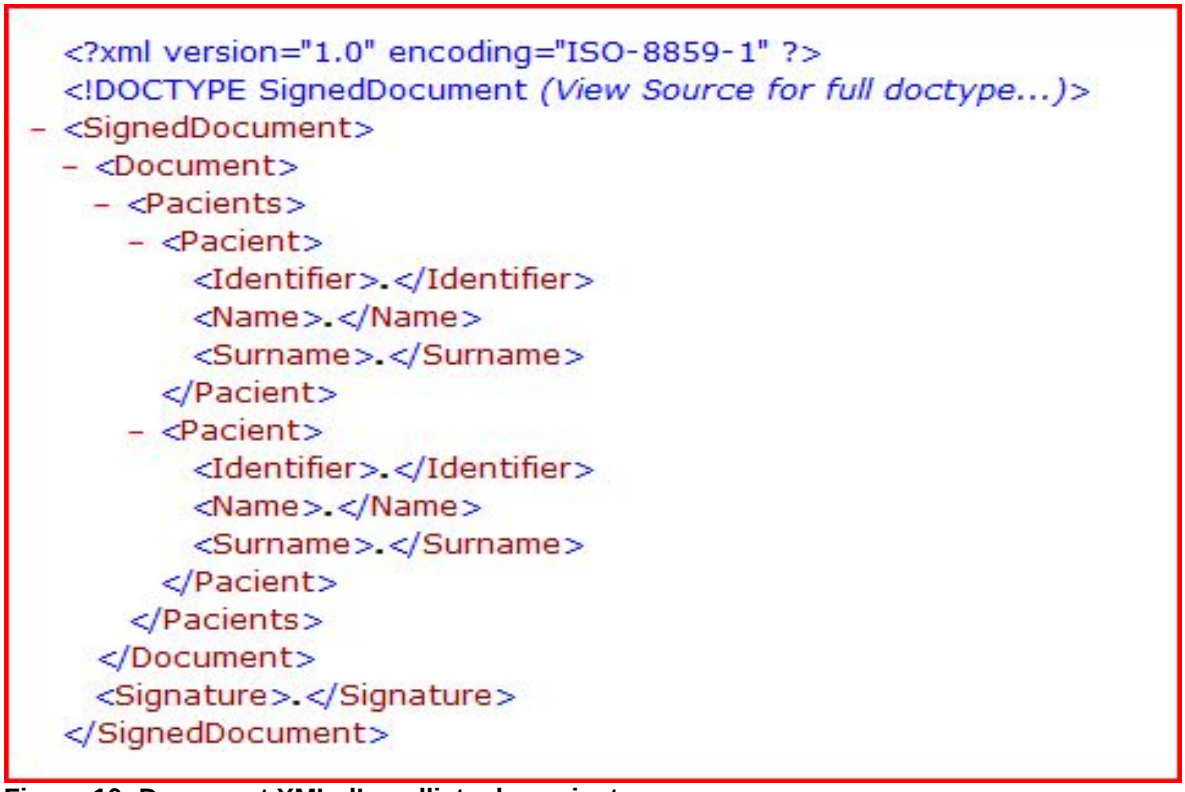

Figura 10: Document XML d'una llista de pacients.

Tal i com es pot observar a la figura anterior, a partir de l'element arrel es deriva un altre anomenat **Document** on dins d'ell hi ha una llista de pacients que corresponen al metge que ha generat la consulta. Per cada pacient es pot trobar els següents camps:

- **Identifier**. Identificador de l'usuari, normalment serà el DNI.
- **Name**. Dins d'aquest camp està el nom del pacient.
- **Surname**. Dins d'aquest camp estan els cognoms del pacient.

A més a més, al final del document podem trobar un altre apartat (**Signature**) que conté la signatura digital efectuada pel gestor del sistema per assegurar la integritat de les dades.

5.5.5. Document XML per enviar el contingut d'una visita.

El document utilitzat pel gestor del sistema per a enviar el contingut íntegre d'una visita és el següent:

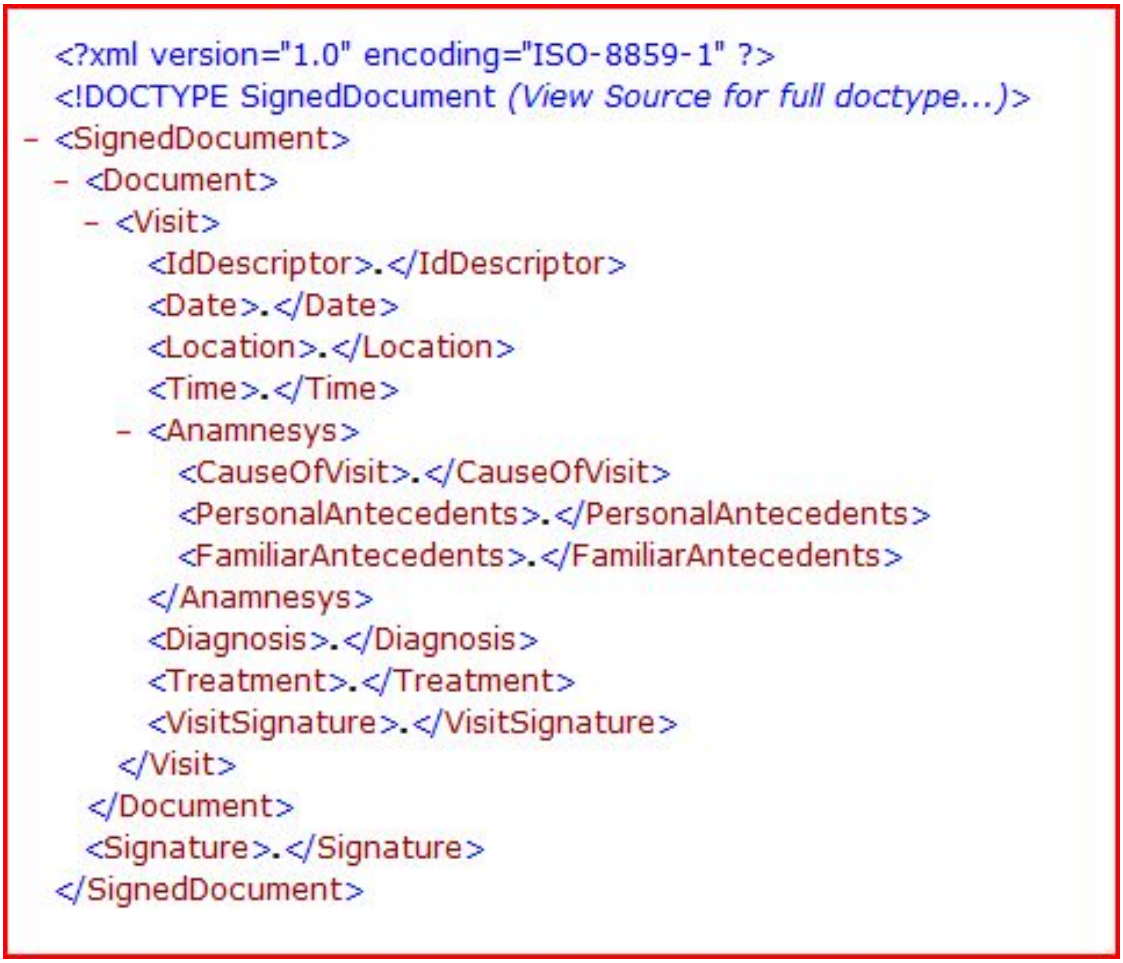

Figura 11: Document XML d'una visita.

Tal i com es pot observar a la figura anterior, a partir de l'element arrel es deriva un altre anomenat **Document** on dins d'ell hi ha el contingut a enviar a l'usuari d'una visita. Els quatre primers camps són comuns entre visita i la part de visites de l'historial i és el que es coneix com a descriptor de visita, tal i com es va explicar a l'apartat gestió i organització de la informació a tractar. Els camps són:

- **Descriptor**. Dins d'aquest camp es troba un identificador únic relacionat amb una visita.

- **Date**. Data en que es va portar o es durà a terme la visita. Les visites es visualitzaran tant les que ja s'han efectuat com les que encara estan per efectuar-se.

- **Location**. Lloc on es va portar o es durà a terme la visita. Les visites es visualitzaran tant les que ja s'han efectuat com les que encara estan per efectuar-se.

- **Time**. Hora de la visita. Les visites es visualitzaran tant les que ja s'han efectuat com les que encara estan per efectuar-se.

Seguidament es pot veure que hi ha un altre apartat, **Anamnesys,** on posarem les dades subjectives relacionades amb el pacient. Aquestes dades són:

- **CauseOfVisit**. Dins d'aquest camp podem trobar la causa per la qual ha portat al pacient a demanar al metge una visita.

- **PersonalAntecedents**. Dins d'aquests camp es poden trobar els antecedents personals que afecten al pacient.

- **FamiliarAntecedents**. Dins d'aquests camp es poden trobar els antecedents personals que afecten al pacient.

Finalment es pot trobar les dades del resultat de la visita i la signatura digital del metge sobre les dades a fi i efecte d'evitar el repudi i assegurar la integritat de les dades durant el transport. Per tant el significat dels darrers camps són:

- **Diagnosis**. Dins d'aquest camp el metge indica el diagnòstic que ha proposat després d'haver visitat al pacient.

- **Treatment**. Dins d'aquest camp el metge indica el tractament que ha de seguir el pacient a fi i efecte de millorar el seu estat o malaltia.

- **VisitSignature**. Dins d'aquest camp el metge ha de proporcionar la signatura digital feta per ell de totes les dades de que consta la visita.

A més a més al final del document podem trobar un altre apartat (**Signature**) que conté la signatura digital efectuada pel gestor del sistema per assegurar la integritat de les dades.

5.6. Document DTD per validar els XML.

Per a validar el contingut dels documents XML utilitzats al projecte i explicats en els apartats anteriors, s'ha utilitzat un únic document DTD. Això ha estat possible gràcies a que tots els documents parteixen de la mateixa arrel: ***SignedDocument***, i que els DTDs tenen la possibilitat de validar diverses estructures de dades mitjançant el símbol "|". Per tant, el document utilitzat per a la validació dels XML és el següent:

```
<?xml encoding="UTF-8"?>


<!--elements estructurals comuns -->

<!ELEMENT SignedDocument (Document|(Document, Signature))>
<!ELEMENT Document ((Identifier, Nuser, Ngestor)
|(Ngestor, Identifier, Operation, DataOperation, DataOperationSignature)
|(Historial)|(Visit)|(Pacients))>


<!--elements estructurals d'una visita -->

<!ELEMENT  Visit (IdDescriptor, Date, Location, Time, Anamnesys, Diagnosis, Treatment, VisitSignature)>
<!ELEMENT Anamnesys (CauseOfVisit, PersonalAntecedents, FamiliarAntecedents)>


<!--elements estructurals d'una llista pacients -->

<!ELEMENT Pacients (Pacient+)>
<!ELEMENT Pacient (Identifier, Name, Surname)>


<!--elements estructurals d'un historial -->

<!ELEMENT Historial (PersonalData, Visits)>
<!ELEMENT PersonalData (Name, Surname, SanitaryCardNumber, Identifier, BloodGroup)>
<!ELEMENT Visits (Descriptor+)>
<!ELEMENT Descriptor (IdDescriptor, Date, Location, Time)>

<!--elements finals -->

<!ELEMENT Identifier (#PCDATA)>
<!ELEMENT Nuser (#PCDATA)>
<!ELEMENT Ngestor (#PCDATA)>
<!ELEMENT Signature (#PCDATA)>
<!ELEMENT Operation (#PCDATA)>
<!ELEMENT DataOperation (#PCDATA)>
<!ELEMENT DataOperationSignature (#PCDATA)>
<!ELEMENT IdDescriptor (#PCDATA)>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT Location (#PCDATA)>
<!ELEMENT Time (#PCDATA)>
<!ELEMENT Diagnosis (#PCDATA)>
<!ELEMENT Treatment (#PCDATA)>
<!ELEMENT VisitSignature (#PCDATA)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Surname (#PCDATA)>
<!ELEMENT CauseOfVisit (#PCDATA)>
<!ELEMENT PersonalAntecedents (#PCDATA)>
<!ELEMENT FamiliarAntecedents (#PCDATA)>
<!ELEMENT SanitaryCardNumber (#PCDATA)>
<!ELEMENT BloodGroup (#PCDATA)>
```

Figura 12: Document DTD de validació dels XML.

5.7. Diagrama de classes de la representació de les dades.

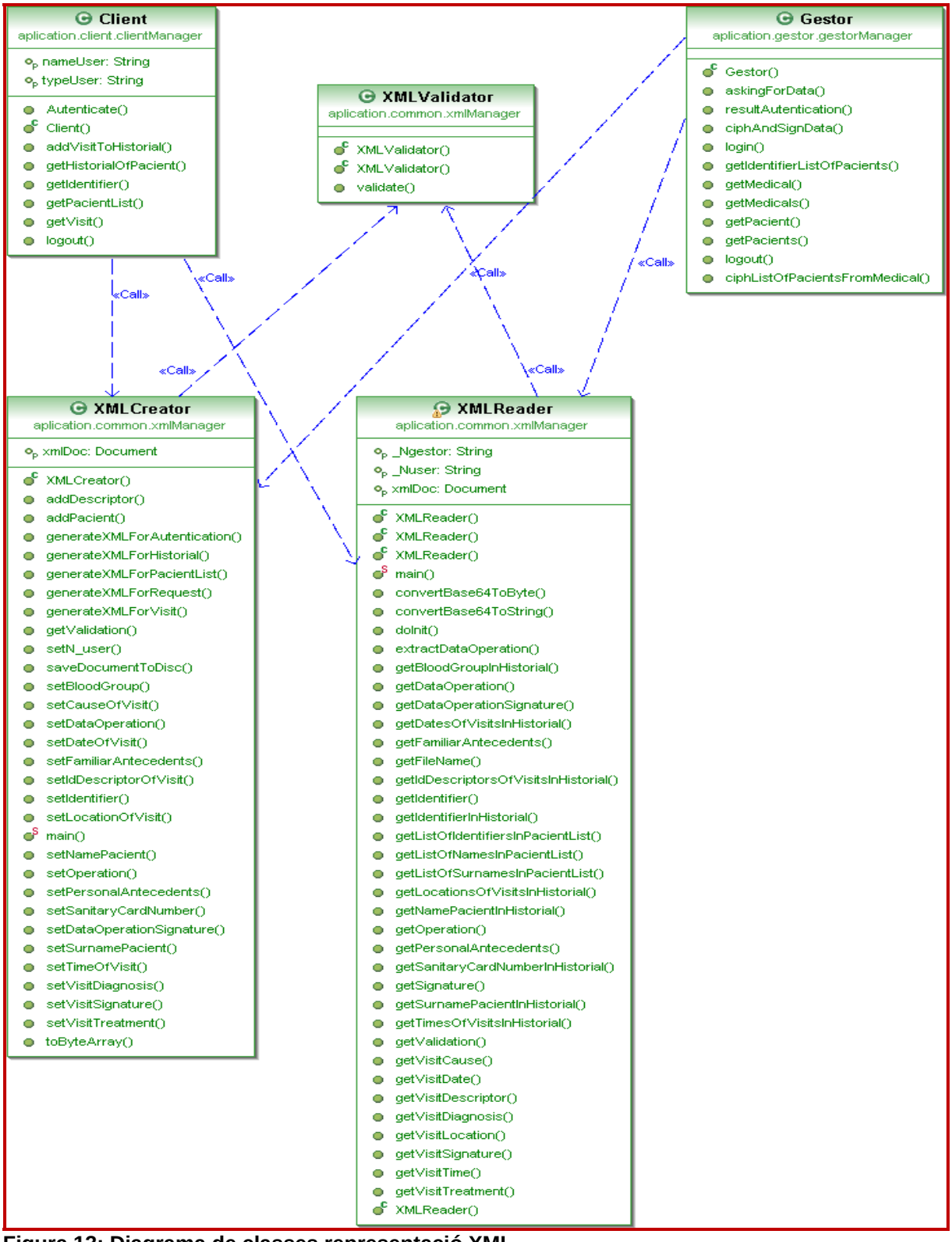

Figura 13: Diagrama de classes representació XML.

6. Comunicació dels components: RMI.

6.1. Introducció.

RMI (*Remote Method Invocation*) és una tecnologia Java que permet enviar missatges a objectes situats en altres màquines virtuals des d'una aplicació que s'executi a la mateixa o diferent màquina física (invocació local o remota). RMI forma part de Java estàndard i ve inclòs en els JDK de Sun des de la versió 1.1.

Aquesta tecnologia permet que es puguin passar arguments al procediment remot i rebre les dades que retorna (en ambdós casos podes ser tipus primitius o objectes de classes que siguin serialitzables).

Cada servei RMI es defineix mitjançant una interfície, que fixa els mètodes que es poden invocar en l'objecte remot. Aquesta interfície ha d'estar disponible tant en el client com en el servidor de dades.

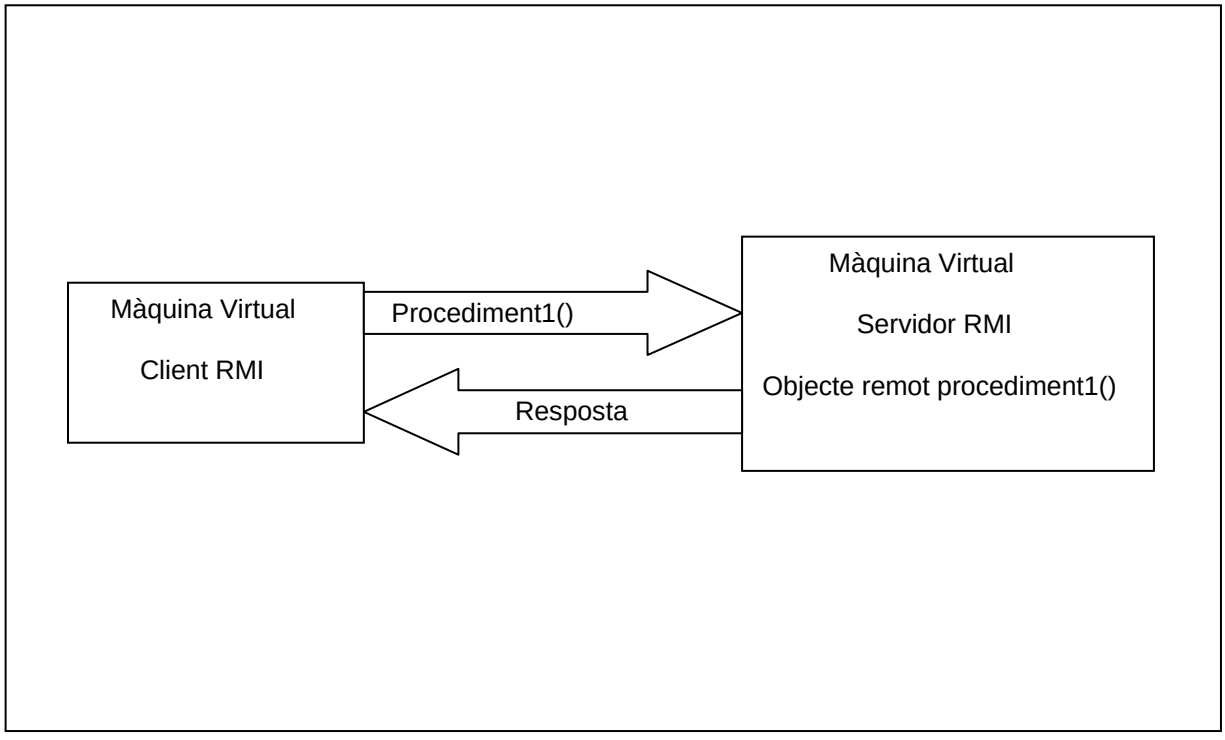

Figura 14: Comunicació RMI entre client i servidor.

Els avantatges principals d'aquest model són els següents:

- **Transparència**: unes mínimes modificacions al codi font permeten distribuir objectes entre diverses màquines. L'entorn d'execució Java (JRE) oculta i facilita la invocació, localització, activació, intercanvi de dades i objectes per la xarxa (*serialització*).

- **Eficiència**: la invocació remota és més eficient que una transferència *http* per la forma en la qual es codifiquen les dades per a la serialització, per la utilització dels canals de comunicació (usualment connexions TCP/IP), per l'eficiència i senzillesa del *stub* servidor respecte a un servidor web.

- **Seguretat**: el gestor de seguretat "security manager" i altres components poden controlar quins objectes i quins mètodes pot invocar cada màquina clienta remota, així com restringir quines accions pot dur a terme un objecte sobre la màquina en la qual corre i evitar que els errors d'un procés pugui afectar a la seguretat i integritat de la màquina.

- **Funcionalitat**: s'obté gairebé la mateixa funcionalitat que ofereix el llenguatge a la comunicació entre objectes remots que entre objectes locals. A més, es pot escollir la forma en la qual un procés servidor atén les peticions, s'activa (s'instancia), s'elimina (a falta de referències, com un objecte local).

6.2. Funcionament de RMI.

En general, una invocació remota implicarà els següents passos:

En el servidor:

- Definir la interfície del servei que serà accessible remotament. En l'RMI, per a fer que un objecte sigui remot s'ha de declarar que implementa la interfície *remote*. Atès que la invocació remota pot fallar, cada mètode de la interfície declara també l'excepció *java.rmi.RemoteException* a la secció *throws*, per poder gestionar les errades de comunicació.

- Implementar el codi del servei complint la interfície anterior i generar els *estubs* corresponents. En RMI, el compilador o generador d'estubs es denomina *rmic* i és l'encarregat de generar codi intermedi. En les últimes versions (a partir del Java 1.2), ja no és necessari generar stubs de servidor (*skeletons*) en temps de compilació. L'entorn RMI usa reflexió en temps d'execució per a fer les invocacions.

- Finalment, s'instancia l'objecte servidor i es publica o registra en un servei de noms per a fer-lo accessible als clients. En l'RMI el servei de noms es denomina *RMIRegistry*, accessible a través de l'API *java.rmi.Naming*.

En el client:

- A partir de la interfície del servei remot, generar els estubs de client. Això es pot realitzar en temps de compilació o execució. En compilació, rmic genera els stubs (per defecte en el protocol JRMP) necessaris per a la invocació remota. Tanmateix, no és necessari amb els *proxies* dinàmics (*dynamic proxies*), permet obviar l'ús d'*rmic*. Es produeix així una generació d'stubs dinàmicament en temps d'execució.

- El client pot ara localitzar el servidor mitjançant el servei de noms i invocar les operacions remotament. En aquesta fase, també es poden obtenir estubs de client dinàmicament baixant-los d'un servidor en localitzar el servei.

Finalment, cal destacar que el Java RMI permet el pas per referència (objectes "remots" els mètodes dels quals es poden invocar des de "lluny": altres màquines virtuals) o per valor (l'objecte "migra" completament per la Xarxa: resulta un objecte local i els seus mètodes s'invoquen localment).

Per a fer que un objecte pugui viatjar s'ha de seriar (*Java object serialization*). Les dades i el codi de la implementació s'han de traslladar i crear una còpia al lloc remot. Això implica que qualsevol classe Java que sigui seriable es pot passar per paràmetre entre objectes remots RMI.

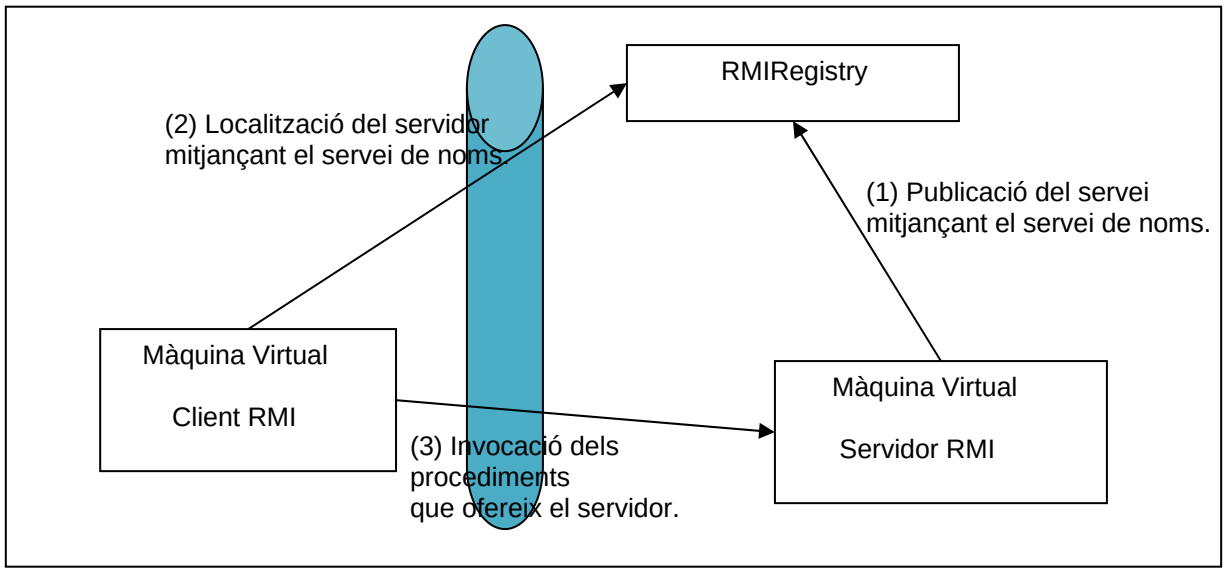

Figura 15: Esquema d'una comunicació remota mitjançant RMI.

6.3. Implementació de la comunicació RMI.

6.3.1. Implementació de la interfície remota.

Els objectes distribuïts en RMI són accessibles a través d'una interfície remota. Aquesta interfície es declara com qualsevol altra en Java amb la restricció de que ha d'heretar de *java.rmi.Remote*. En la signatura dels procediments d'aquestes interfícies només es pot utilitzar tipus bàsics, altres interfícies remotes (pas per referència) o classes serialitzables (pas per valor), i s'ha d'indicar que pot llançar la excepció *java.rmi.RemoteException* (recuperar qualsevol tipus d'error en la comunicació distribuïda).

```
public interface ServerGestor extends Remote {

    public byte[]login(byte[] dataUser) throws  RemoteException;
    public boolean resultAutentication(byte[] dataUser)throws RemoteException;
    public byte[] askingForData(byte[]dataUser) throws RemoteException;
    public boolean logout(byte[] dataUser) throws RemoteException;

}
```

6.3.2. Implementació del servidor.

Una vegada definida la interfície remota de l'objecte distribuït és necessari donar-li una implementació. La classe que implementa la interfície no té cap restricció, només que per convenció s'acostuma a nombrar com la interfície remota amb el sufix "Impl".

```
public class ServerGestorImpl implements ServerGestor {

        Gestor gestor;

        public ServerGestorImpl(){
                Security.insertProviderAt(new IAIK(), 2);
                gestor= new Gestor();
        }

        public byte[]login(byte[] dataUser){
                return gestor.login(dataUser);
        }

        public boolean resultAutentication(byte[] dataUser){
                return gestor.resultAutentication(dataUser);
        }
```

```
        public boolean logout(byte[] dataUser) throws RemoteException{
                return gestor.logout(dataUser);
        }

        public byte[] askingForData(byte[]dataUser) throws RemoteException{
                return gestor.askingForData(dataUser);
        }

}
```

6.3.3. Programació del servidor d'objectes distribuïts.

Per a fer accessibles els objectes distribuïts, és necessari implementar un programa que s'encarregui de la seva instanciació i activació. Bàsicament, les tasques d'aquesta aplicació són les següents:

- Instal·lar un *SecurityManager* a fi i efecte de permetre la descàrrega dinàmica de codi.

  ```
  System.setSecurityManager(new RMISecurityManager());
  ```

- Una vegada tenim instal·lat el mànager de seguretat, s'ha d'engegar el *RMIRegistry* per a accedir al servei de noms:

  ```
  Registry registry = LocateRegistry.createRegistry(Registry.REGISTRY_PORT);
  ```

- Instanciar els objectes distribuïts.

  ```
  ServerGestor c = new ServerGestorImpl();
  ```

- Activar els objectes distribuïts.

  ```
  UnicastRemoteObject.exportObject(c,Registry.REGISTRY_PORT);
  ```

- Registrar els objectes inicials de l'aplicació en el registre RMI.

  ```
  Naming.rebind("rmi://localhost:1099/servidorMedic", c);
  ```

6.3.4. Implementació del client.

La tasca d'un client en un sistema distribuït és obtenir les referències als objectes distribuïts. Les referències als objectes inicials de l'aplicació s'obtenen a través del registre RMI. La resta de referències es poden aconseguir a partir d'altres objectes distribuïts (pas de paràmetres o valor de retorn). Finalment a l'igual que l'aplicació servidora, és necessari registrar un mànager de seguretat per a la descàrrega de codi.

```
if (System.getSecurityManager()==null){

        System.setSecurityManager(new RMISecurityManager());
}

ServerGestor gestor = (ServerGestor)Naming.lookup("rmi://localhost:1099/servidorMedic");

gestor.login(encryptedData);
gestor.resultAutentication(encryptedData);
gestor.logout(encryptedData);
gestor.askingForData(encryptedData);
```

6.3.5. Diagrama de classes comunicació RMI.

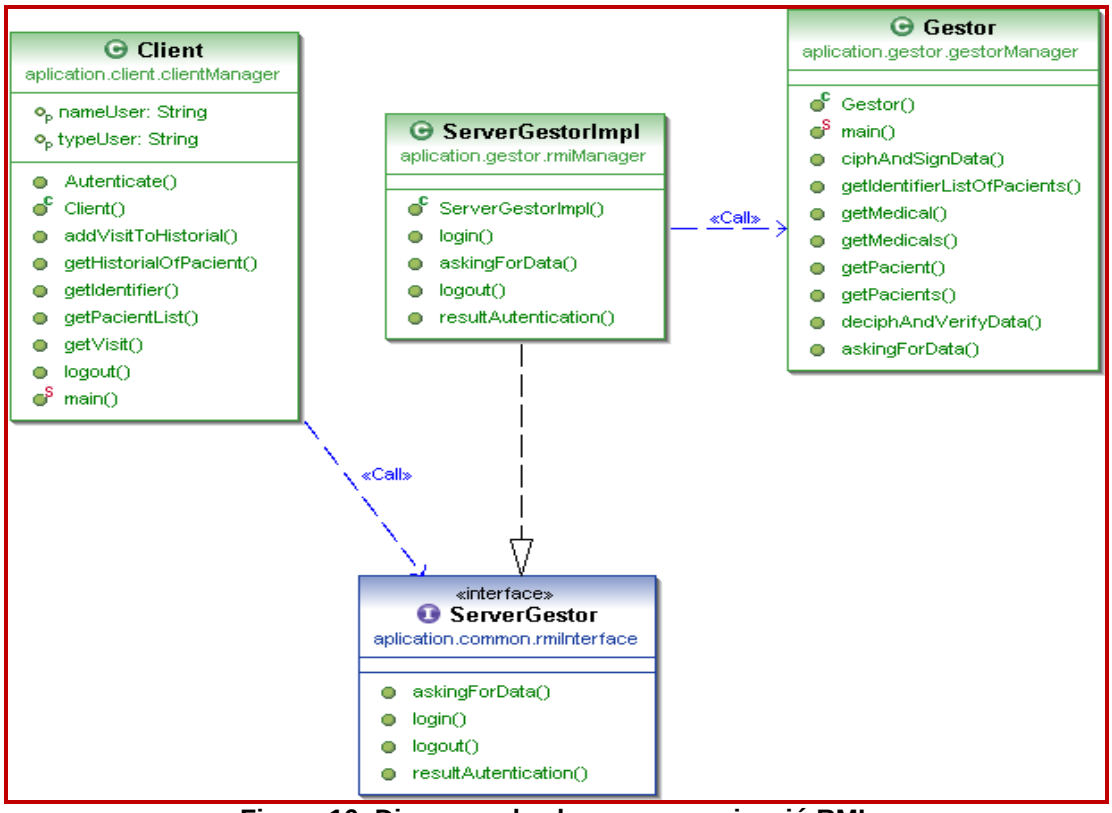

Figura 16: Diagrama de classes comunicació RMI.

6.4. Execució.

El client d'una aplicació distribuïda només necessita la interfície remota compilada pera la seva execució. L'*stub*, que és necessari per a l' accés a l'objecte distribuït, serà descarregat dinàmicament. Això no significa que no pugui disposar de l'*stub*. Si disposa d'allò, no descarregarà la versió més actual del mateix, lo qual suposa un problema de manteniment de l'aplicació. En resum, el més recomanable és la descàrrega dinàmica d'*stubs*.

Per a l'execució d'una aplicació Java distribuïda s'ha de tenir en compte els aspectes de seguretat de la màquina virtual Java. En primer lloc, s'ha de llançar un registre RMI en cada ordinador en el que es registrin objectes. En el nostre cas només s'ha de llençar el registre en la part del servidor atès que el client només demanarà els serveis indicats en la interfície.

A continuació editem un fitxer de polítiques de seguretat que permetrà a les aplicacions utilitzar i accedir als ports en els que es connecten els objectes remots. Sota aquest propòsit editem un fitxer, que anomenem "*policy.txt*", amb el següent contingut:

```
grant {
        permission java.security.AllPermission;
        permission java.net.SocketPermission "*:1024-65535","connect,accept";
};
```

Per establir la comunicació i intercanviar dades d'una manera segura entre un client i el servidor no ha estat necessari utilitzar cap mena de seguretat a nivell de xarxa com podria ser la utilització de IPsec per establir un canal segur, ni a nivell de transport com podria ser SSL o TLS atès que ja s'assegura aquesta seguretat mitjançant els protocols dissenyats per al projecte i explicats en el capítol 4.3.

7. Gestió de la informació: Base de Dades.

7.1. Introducció.

Fins aquest punt l'aplicació només treballava en temps d'execució, emmagatzemant totes les dades necessàries per al correcte funcionament en objectes i variables que es perdien una vegada executada l'aplicació.

Per tal de tenir la informació generada d'una manera persistent en el temps, cal emmagatzemar-la en una base de dades. D'aquesta manera, les dades estaran guardades i es podrà disposar d'ella quan els usuaris la demanin. La base de dades ha de guardar tota la informació dels usuaris, tant metges com pacients i a més a més la informació dels historials i les visites que generen els metges als seus pacients. Atès que existeixen relacions entre les dades que s'emmagatzemen, és necessari utilitzar una base de dades relacional a fi i efecte de poder indicar relacions entre els diferents components.

7.2. Programari gestor de base de dades. MySQL.

La utilització d'una base de dades, dóna una eina molt eficaç i segura de gestió de les dades que genera l'aplicació. Per a poder gestionar-la doncs es necessita algun programari que doni aquesta funcionalitat. En aquest projecte s'ha utilitzat MySQL Server 5.0 que és un servidor de base de dades relacionals gratuït i perfectament funcional.

7.2.1. Instal·lació de MySQL Server 5.0.

Atès que la instal·lació d'aquest programari és vital per a que funcioni correctament l'aplicació, el procediment d'instal·lació d'aquest programari s'explica pas a pas als annexos d'aquesta documentació. Anar als annexos ->

7.2.2. Instal·lació de MySQL Tools for 5.0.

Una vegada s'ha instal·lat el gestor de base de dades, és convenient utilitzar una sèrie d'eines per a fer més fàcil la tasca de gestió i manteniment. Aquestes eines es poden aconseguir instal·lant el paquet *mysql-gui-tools-5.0-r12-win32.msi* que es pot descarregar gratuïtament de la següent adreça: descàrrega de les eines.

La instal·lació d'aquestes eines és trivial i accessible des de l'accés directe del botó inici de Windows al directori MySQL. L'aplicatiu que més interessa és el que s'anomena **MySQLQueryBrowser** que permet crear la base de dades a partir de l'script que es pot trobar als annexos d'aquesta documentació. Anar als annexos->

7.3. Model entitat-relació.

El diagrama entitat-relació de la base de dades que s'utilitza al projecte és el següent:

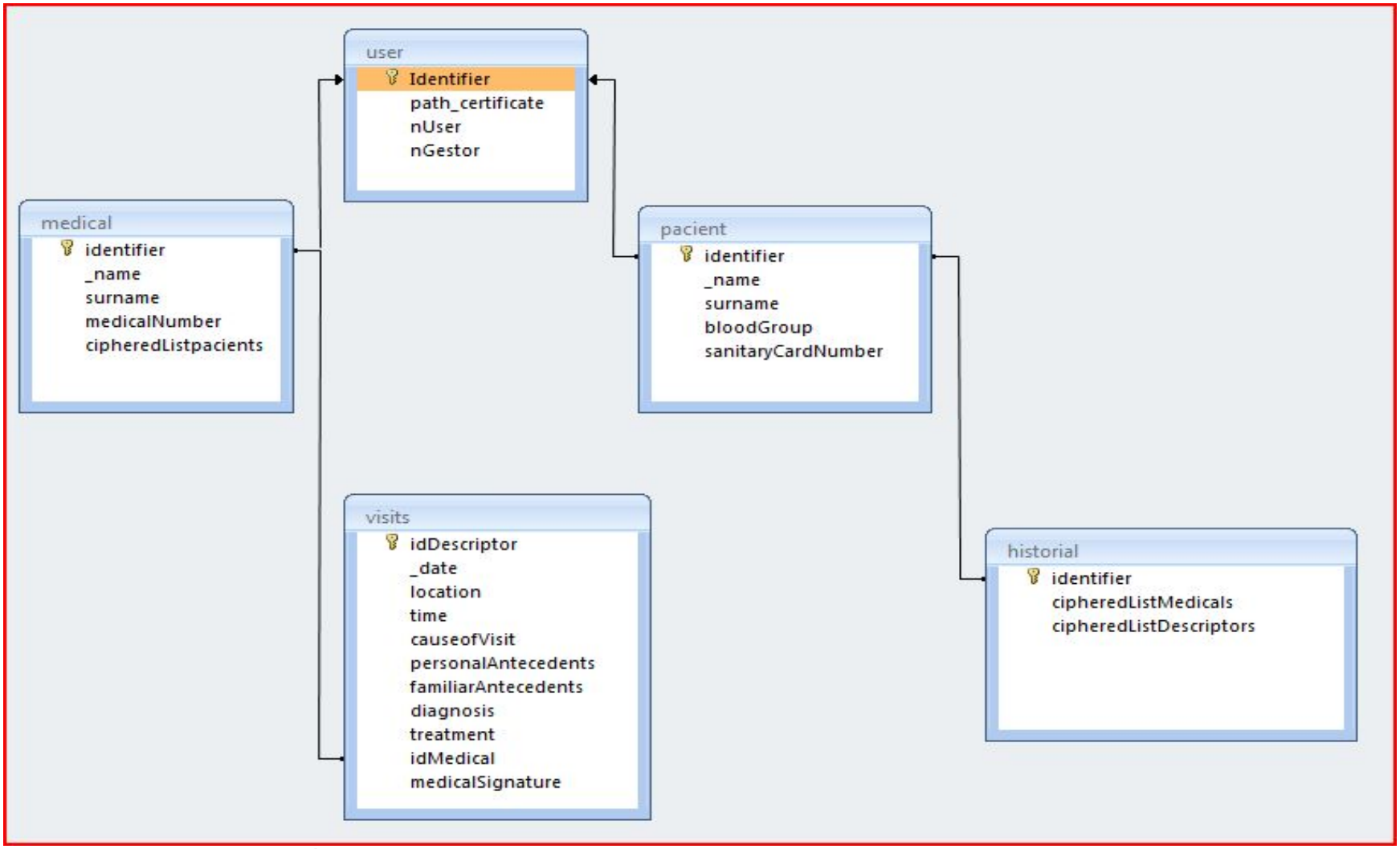

Figura 17: Model entitat-relació de la base de dades.

Les relacions que tenim al diagrama són les següents:

- **medical-user**. Entre la taula *medical* i *user* tenim una relació d'herència 0..1 a 1. Les dues taules es relacionen a partir del seu identificador.

- **medical-visits**. Entre la taula *medical* i *visits* tenim una relació 1 a 0...* . És a dir cada metge pot tenir des de cap a moltes visites generades, en canvi una visita només pot ser generada per un únic metge.

- **pacient-user**. Entre la taula *pacient* i *user* tenim una relació d'herència 0..1 a 1. Les dues taules es relacionen a partir del seu identificador.

- **pacient-historial**. Entre la taula pacient i historial tenim una relació 1 a 1. És a dir a cada pacient només li correspon un historial i a l'inrevés, per cada historial només li correspon un pacient.

- A més a més tenim tres relacions amagades mitjançant el xifratge entre *pacient* i *medical*, entre *medical* i *historial* i entre *historial* i *visits*. Aquestes relacions es xifren i signen pel gestor a fi i efecte d'evitar que si un usuari malintencionat accedís a la base de dades no pogués saber en cap cas la relació entre els metges i els pacients i tampoc entre les visites generades que es guarden en clar i a quin pacient correspon.

7.3.1.Taula user.

Aquesta taula ens guarda les dades necessàries per a iniciar i mantenir una comunicació segura entre el sistema i l'usuari remot. Els camps que té són els següents:

- **identifier**. Identificador de l'usuari que inicia la sessió. Es correspon amb el seu DNI i ha de ser el mateix que consta al camp *dnQualifier* del seu certificat. És la clau primària de la taula i és del tipus *varchar* amb 9 caràcters.

- **path-certificate**. És el nom del certificat de l'usuari que s'ha de buscar al directori de l'autoritat de certificació (CA). Aquest directori es troba a l'arrel del projecte i s'anomena **CAPFC**. Els certificats es troben al directori **certs**. A partir del certificat, el sistema pot saber a partir del camp Organizational Unit Name (OU) si l'usuari és un metge o un pacient. És del tipus *varchar* amb 100 caràcters.

- **nUser**. És el número aleatori que ha generat l'usuari per a comunicar-se amb el sistema. Aquest valor s'esborra una vegada el sistema a facilitat les dades de la consulta o s'ha autentificat correctament l'usuari. És del tipus *varchar* amb 50 caràcters.

- **nGestor**. És el número aleatori que ha generat el gestor per a comunicar-se amb l'usuari. Aquest valor s'esborra una vegada el sistema a facilitat les dades de la consulta o s'ha autentificat correctament l'usuari. És del tipus *varchar* amb 50 caràcters.

7.3.2.Taula pacient.

Aquesta taula ens guarda les dades necessàries d'un usuari que és pacient. Els camps que té són els següents:

- **identifier**. Identificador del pacient. Es correspon amb el seu DNI, és la clau primària de la taula i és del tipus *varchar* amb 9 caràcters.
- **name**. Nom del pacient. És del tipus *varchar* amb 25 caràcters.
- **surname**. Cognoms del pacient. És del tipus *varchar* amb 50 caràcters.
- **bloodGroup**. Grup sanguini del pacient. És del tipus *varchar* amb 5 caràcters.
- **sanitaryCardNumber**. Número de la seguretat social del pacient. És del tipus *varchar* amb 15 caràcters.

7.3.3.Taula medical.

Aquesta taula ens guarda les dades necessàries d'un usuari que és metge. Els camps que té són els següents:

- **identifier**. Identificador del metge. Es correspon amb el seu DNI, és la clau primària de la taula i és del tipus *varchar* amb 9 caràcters.
- **name**. Nom del metge. És del tipus *varchar* amb 25 caràcters.
- **surname**. Cognoms del metge. És del tipus *varchar* amb 50 caràcters.
- **medicalNumber**. Número de col·legiat del metge. És del tipus *varchar* amb 12 caràcters.
- **cipheredListPacients**. Llista d'identificadors dels pacients assignats al metge que està xifrada i signada pel gestor a fi i efecte d'evitar que un intrús pugui saber el seu contingut i modificar-lo. És del tipus blob (*Binary Large OBjects*), objectes binaris grans, que correspon a una cadena de bytes d'una longitud màxima de 65.535 bytes.

7.3.4.Taula historial.

Aquesta taula ens guarda les dades necessàries d'un historial per relacionar els metges que té assignats un pacient i les visites que té donades d'alta, tots dos camps signats i xifrats pel gestor del sistema. Els camps que té són els següents:

- **identifier**. Identificador del pacient propietari del historial. Es correspon amb el seu DNI, és la clau primària de la taula i és del tipus *varchar* amb 9 caràcters.

- **cipheredListMedicals**. Llista d'identificadors dels metges assignats a l'historial del pacient que està xifrada i signada pel gestor a fi i efecte d'evitar que un intrús pugui saber el seu contingut i modificar-lo. És del tipus blob (*Binary Large OBjects*), objectes binaris grans, que correspon a una cadena de bytes d'una longitud màxima de 65.535 bytes.

- **cipheredListDescriptors**. Llista d'identificadors dels metges assignats a l'historial del pacient que està xifrada i signada pel gestor a fi i efecte d'evitar que un intrús pugui saber el seu contingut i modificar-lo. És del tipus blob (*Binary Large OBjects*), objectes binaris grans, que correspon a una cadena de bytes d'una longitud màxima de 65.535 bytes.

7.3.5.Taula visits.

Aquesta taula ens guarda les dades necessàries d'una visita que s'ha inserit al sistema per un metge en concret. Els camps que té són els següents:

- **idDescriptor**. Identificador únic de la visita . És la clau primària de la taula i és del tipus *int unsigned*, enter sense signe, amb 10 valors.

- **date**. La data en que es va efectuar la visita. És del tipus *varchar* amb 11 caràcters i el format és dd/mm/aaaa.

- **location**. El lloc on es va efectuar la visita. És del tipus *varchar* amb 50 caràcters.

- **time**. L'hora en que es va efectuar la visita. És del tipus *varchar* amb 5 caràcters i el format és hh:mm.

- **causeOfVisit**. La causa de la visita. És del tipus *varchar* amb 300 caràcters.

- **personalAntecedents**. Els antecedents personals del pacient que s'efectua la visita. És del tipus *varchar* amb 300 caràcters.

- **familiarAntecedents**. Els antecedents familiars del pacient que s'efectua la visita. És del tipus *varchar* amb 300 caràcters.

- **diagnosis**. El diagnòstic del metges sobre la visita. És del tipus *varchar* amb 300 caràcters.

- **treatment**. El tractament del metge per pal·liar la causa de la visita. És del tipus *varchar* amb 300 caràcters.

- **idMedical**. L'identificador del metge que ha efectuat i creat la visita. És del tipus *varchar* amb 9 caràcters.

- **medicalSignature**. La signatura digital del metge sobre les dades de la visita que es troba emmagatzemada a la base de dades. Aquest camp té la finalitat d'evitar un canvi no autoritzat de les dades que consta la visita i a més a més el no repudi per part del metge. És del tipus blob (*Binary Large OBjects*), objectes binaris grans, que correspon a una cadena de bytes d'una longitud màxima de 65.535 bytes.

7.4. Implementació.

Per fer la implementació d'aquesta part ha estat necessari afegir una nova classe que s'encarregui de gestionar les peticions que fa el sistema a la base de dades. Aquesta classe només la té la part servidora atès que el client no pot demanar directament cap dada sinó que les ha de demanar al gestor i aquest s'encarrega de facilitar les dades si escau.

Aquesta classe s'anomena ***EngineDB.class*** i es pot trobar dins al paquet ***integrationManager***. El correcte funcionament de l'aplicació fa necessari de tenir una llibreria nova que s'encarregui de gestionar la connexió amb el servidor MySQL. Aquesta llibreria anomenada *mysql-connector-java-5.1.6-bin.jar* es pot descarregar de la següent adreça: descàrrega de la llibreria. No obstant aquesta llibreria es troba allotjada al directori lib del projecte i l'aplicació pot funcionar correctament sense necessitat d'instal·lar-la.

El funcionament de la classe és el següent:

- Cada vegada que es necessita obtenir dades s'ha d'establir una connexió amb la base de dades. Aquest procés és necessari i a la vegada eficient pel fet que si obrim una connexió quan instanciem la classe i no la tanquem fins que eliminem la instància pot donar el cas que el servidor MySQL ens rebutgi certes connexions pel fet que en tenim massa d'obertes. La connexió es fa de la següent manera:

```
private void connectToDataBase() {
        try{
                Class.forName("org.gjt.mm.mysql.Driver");
                Connection connection = (Connection) DriverManager.getConnection
               ("jdbc:mysql://localhost/databasepfc", userDB, passwordDB);
        }catch ( ClassNotFoundException e){
                System.out.println(SystemTime.getTime()+"No es troba el driver de connexió
                amb la base de dades\n s'ha de verificar si es troba la llibreria <mysql-
                connector-java-5.1.6-bin.jar>");
        }catch ( SQLException e){
                System.out.println(SystemTime.getTime()+"No es pot connectar amb la base
                de dades");
        }

}
```

- Una vegada s'han obtingut les dades s'ha de tancar la connexió i es fa de la següent manera:

```
private  void closeConnection(){
        try{
                connection.close();
        }catch (SQLException e){
                System.out.println(SystemTime.getTime()+"No es pot tancar la  connexió amb
                la base de dades");
        }
}
```

Finalment, per a provar el correcte funcionament de l'aplicació fins aquest punt, és necessari crear varis usuaris metges i pacients amb els seus corresponents certificats. Per a crear els certificats, als annexos d'aquesta documentació es pot obtenir una sèrie d'arxius per lots que els generen tot indicant per consola les claus d'accés i els identificadors i altres camps dels certificats tal i com s'especifica al capítol *3.3.4. Generació dels certificats*. Anar als annexos ->

Per a tots els usuaris que s'han creat i fins i tot el gestor, a fi i efecte de fer més fàcil l'execució de l'aplicació, s'ha utilitzat la mateixa clau per als arxius p12. Aquesta clau és "**uoc0809**".

7.4.1. Diagrama de classes de la gestió de la informació.

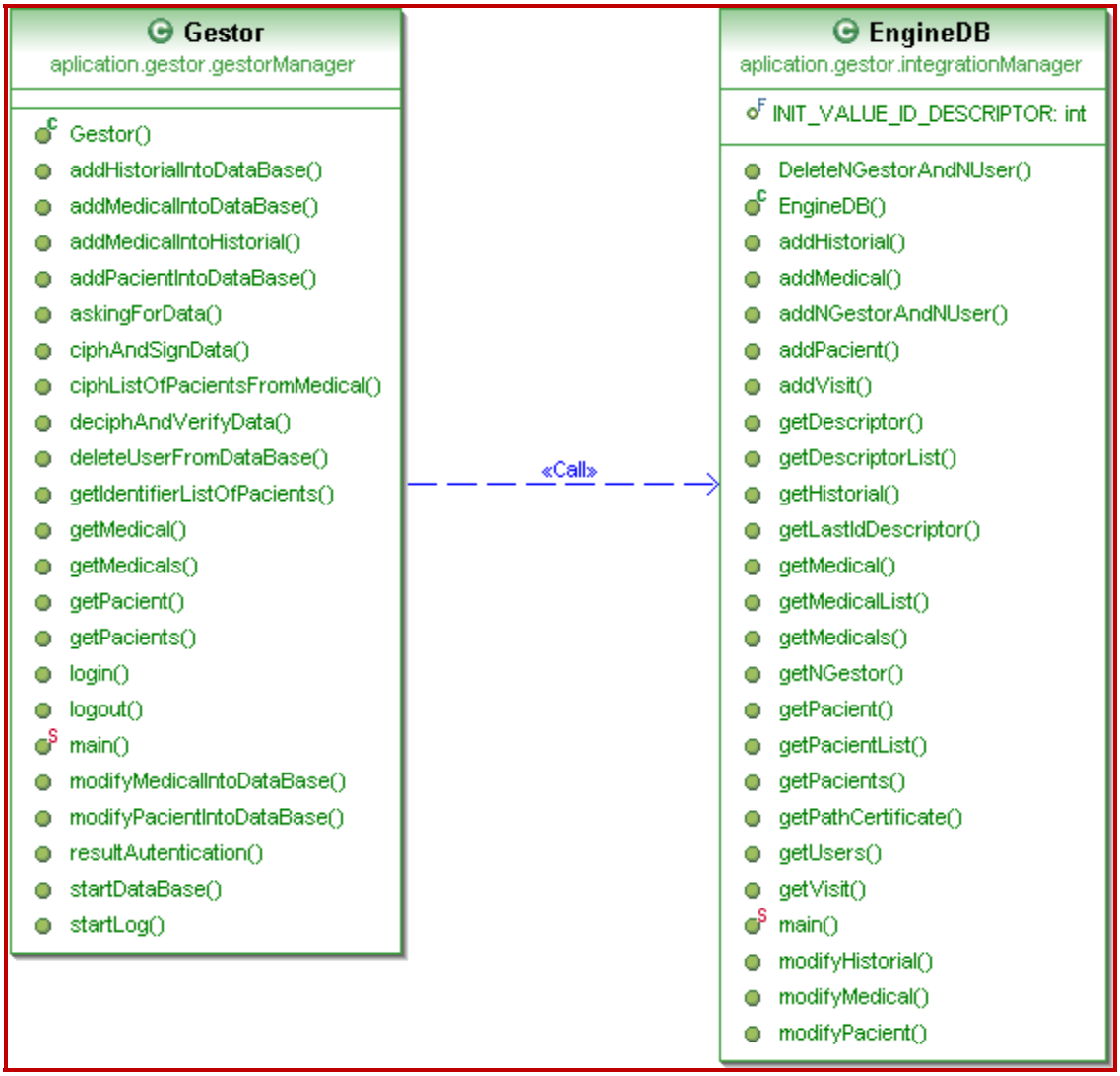

Figura 18: Diagrama de classes gestió de la informació.

8. Visualització de dades. Interfícies gràfiques.

8.1. Introducció.

Una vegada tenim tots els components de l'aplicació implementats per a oferir la funcionalitat demanada, és necessari un sistema d'interacció home-màquina per tal de que els usuaris puguin demanar als sistema les tasques que volen. A tal efecte, s'ha dissenyat una interfície per a cada tipus d'usuari que s'ha previst en el capítol 2: Estudi de les necessitats del sistema.

Un disseny de la interfície gràfica de fàcil ús, simple i robust és molt important per a assegurar l'èxit de l'aplicació, atès que serà la part del sistema que estarà en contacte directe amb els usuaris que la utilitzaran i per tant d'ella depèn que l'aplicació sigui ben rebuda o no.

Es pot diferenciar dos blocs, el primer bloc són les interfícies dissenyades per als clients, pacients i metges, que és l'encarregada d'obtenir les dades necessàries que introdueixen els usuaris per tal de demanar els serveis d'autentificació, consulta, creació de visites si escau i sortida segura del sistema al servidor. El segon bloc són les interfícies dissenyades per al gestor del sistema que s'encarrega d'obtenir les dades necessàries per a posar en funcionament el gestor de la base de dades, el servidor i creació/modificació dels usuaris que poden accedir al sistema.

En els següents punts s'indica amb detall el disseny i interacció que ofereix cada interfície en funció de l'usuari a que està destinat.

8.2. Interfícies usuaris. Pacient.

Tal i com s'especifica en el capítol 2, l'aplicatiu pacient ha d'oferir les següents funcionalitats:

- Autentificar al pacient contra el gestor del sistema.
- Realitzar una consulta del seu expedient.
- Sol·licitar un servei.
- Abandonar de forma segura el sistema.

8.2.1. Autentificació.

Per accedir a l'aplicació, la primera tasca que ha de fer un pacient és l'autentificació davant el sistema. Per fer-ho, necessita un nom d'usuari, un arxiu p12 que contingui el seu certificat i el parell de claus, pública i privada, i una clau secreta per a

proporcionar seguretat a l'arxiu p12. A més a més, és possible que el servidor canviï l'adreça IP de connexió o el port per on funcionarà. Per tant la primera finestra que veu un pacient quan entra és la d'autentificació la qual demana aquestes dades. Es pot veure que a la part inferior hi ha dos botons, el de **Sortir** permet tancar l'aplicació i el d'**Acceptar** permet enviar les dades introduïdes al servidor per a procedir a l'autentificació :

Figura 19: Finestra d'autentificació del pacient.

A fi i efecte d'agilitzar les proves d'execució, per defecte la finestra ja dóna com a opció d'adreça IP "*localhost*" i com a port "*1099*". No obstant es pot canviar aquestes dades per provar el seu funcionament en diferents màquines a través d'una xarxa. A més a més, per facilitar la cerca de l'arxiu p12 hi ha un botó que permet buscar i seleccionar-lo en el directori per defecte que té l'aplicació per guardar aquests fitxers:

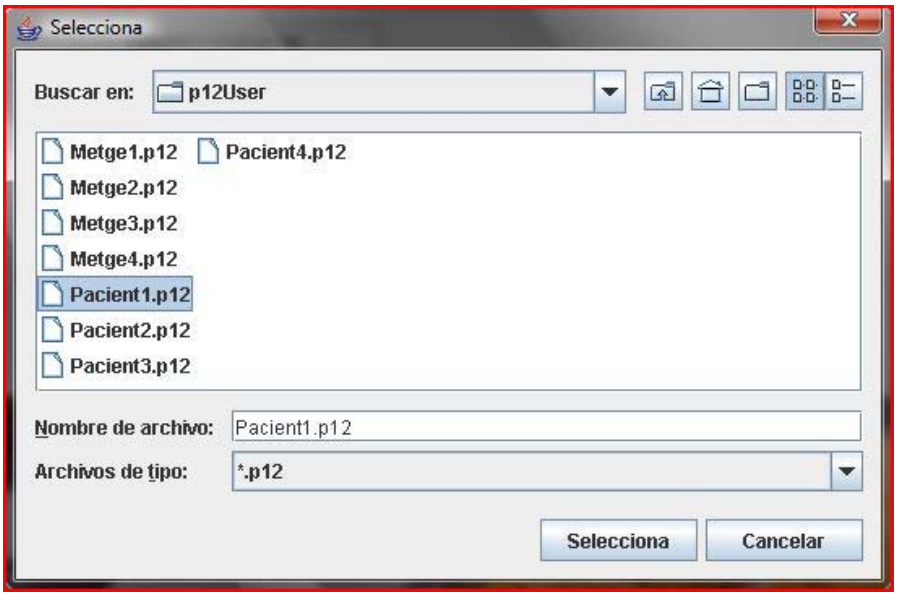

Figura 20: Finestra de selecció de l'arxiu p12 del pacient.

8.2.2. Interfície principal.

Una vegada s'ha autentificat correctament el pacient, accedeix a la seva finestra principal que l'indica el tipus d'usuari que és i el seu nom, recuperat tot del certificat que es troba a l'arxiu p12 seleccionat. Aquesta interfície permet fer dues accions diferents: **Consultar l'historial propi** o **Tancar la sessió**. Per executar-les només cal seleccionar l'opció i prémer el botó **Acceptar**. Per causes de robustesa i facilitat d'ús per a l'usuari, aquesta finestra no permet visualitzar una visita directament atès que és molt difícil que un usuari pugui recordar un número i s'ha pensat que seria més efectiu no posar aquesta opció directament sinó que la visualització de les dades d'una certa visita es fa a partir dels descriptors que apareixen a l'historial.

Figura 21: Finestra principal del pacient.

8.2.3. Consultar el seu historial.

Tal i com s'ha especificat anteriorment, per consultar l'historial s'ha de seleccionar l'opció "**Consulti el seu Historial**" i prémer el botó **Acceptar** a la finestra principal de l'aplicació. En aquesta finestra, es pot observar que es visualitzen les dades personals del pacient i una llista amb els descriptors de visita vinculats amb l'historial que es troben al sistema. Aquestes dades només són de lectura, tal i com s'ha especificat en el capítol 2, si un usuari vol modificar alguna dada personal, ho ha de comunicar al gestor del sistema i aquest serà qui efectuarà el canvi. Des d'aquesta finestra hi ha la possibilitat de visualitzar qualsevol visita que el pacient tingui a la base de dades. Per fer-ho només cal seleccionar-la de la taula i prémer el botó **Veure Visita**.

A més a més hi ha un parell de botons més, el que es diu **Pàgina Anterior** que serveix per tornar a la interfície principal del pacient i un altre anomenat **Tancar Sessió** que serveix per sortir d'una manera segura del sistema tornant a la pàgina d'autentificació.

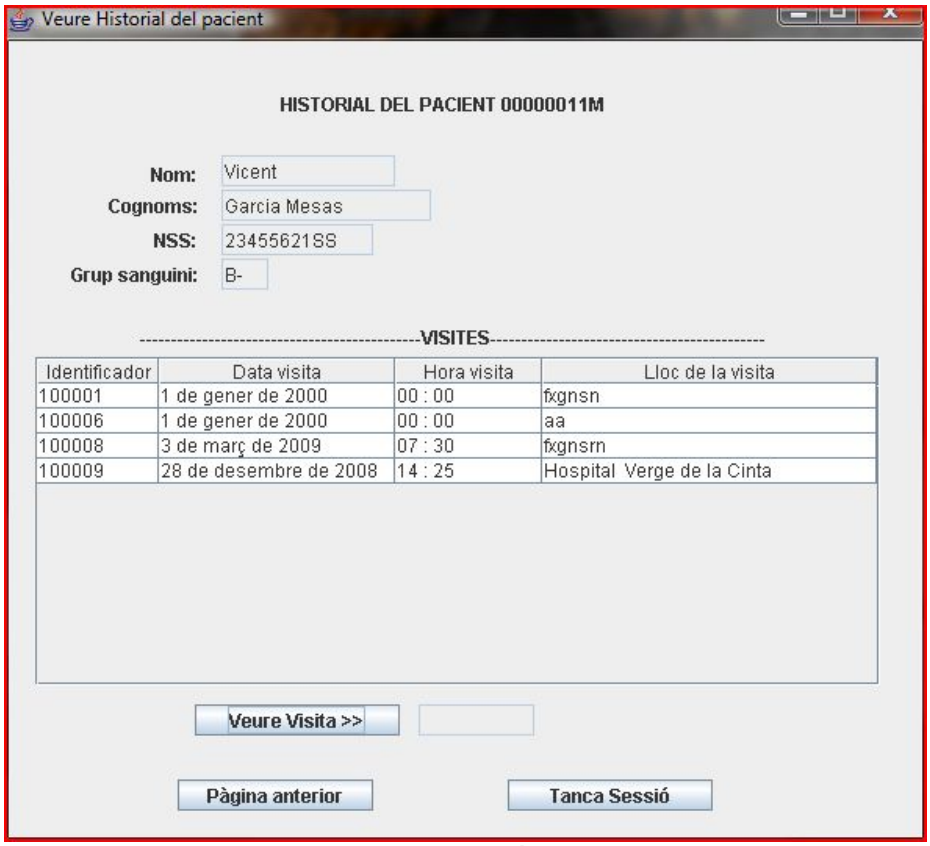

Figura 22: Finestra de visualització de l'historial del pacient.

8.2.4. Consultar una visita.

Aquesta finestra permet visualitzar totes les dades que consta una visita seleccionada a partir del descriptor de visita que hi havia a la finestra anterior. Atès que només és de lectura, només hi ha un botó anomenat **Acceptar** que permet tancar la finestra quan l'usuari pacient hagi acabat de veure les dades i tornar a la finestra de l'historial.

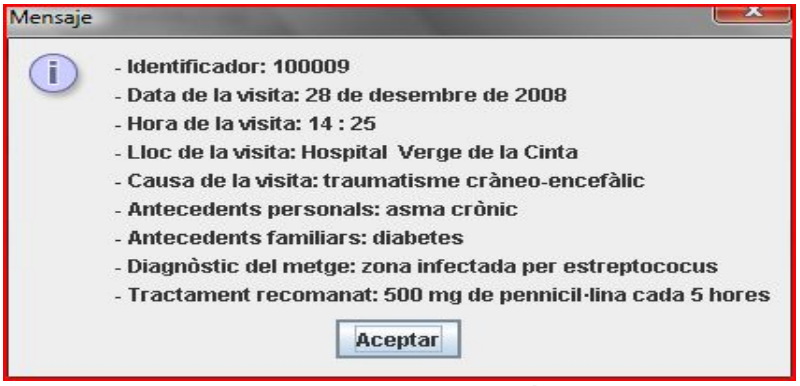

Figura 23: Finestra de visualització d'una visita.

8.2.5. Sortir del sistema.

El pacient té vàries maneres de sortir adequadament del sistema. La primera d'elles es troba a la finestra principal seleccionant l'opció **Sortir del sistema** i prement el botó **Acceptar**. Una altra manera es troba a la finestra de l'historial prement el botó **Tancar Sessió**. A més a més si es tanquen les finestres de la interfície principal o de visualització de l'historial també s'executa l'acció de tancar la sessió. Qualsevol de les maneres de tancar la sessió mostra un missatge especificant el resultat del tancament i en el cas que s'hagi efectuat correctament, et torna a la pàgina d'autentificació. En el cas que no es pugui tancar la sessió, l'aplicació es tanca directament.

Figura 24: Missatge de tancament correcte de la sessió.

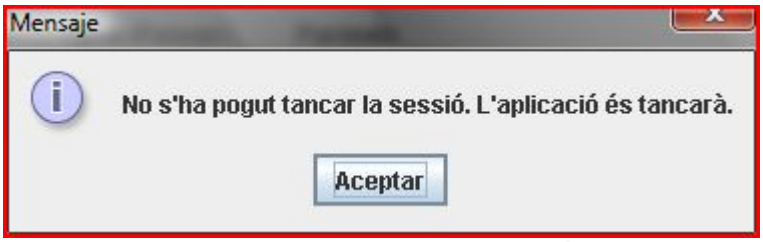

Figura 25: Missatge de tancament de sessió amb errades.

8.3. Interfícies usuaris. Metge.

Tal i com s'especifica en el capítol 2, l'aplicatiu metge ha d'oferir les següents funcionalitats:

- Autenticar al metge contra el gestor del sistema.
- Realitzar una consulta de l'expedient d'un dels seus pacients.
- Obtenir una llista dels seus pacients.
- Introduir dades d'una nova visita a l'expedient d'un dels seus pacients.
- Obtenir dades d'una visita d'un pacient en concret.
- Abandonar de forma segura el sistema.

8.3.1. Autentificació.

Tal i com passava amb el pacient, el metge quan accedeix a l'aplicació, la primera finestra que veu és la l'autentificació davant el sistema. El disseny i funcionament és igual que el que s'ha explicat per al pacient. El metge ha d'especificar el seu identificador, clau secreta del seu arxiu p12, l'arxiu en si, l'adreça IP del servidor i el port. Per procedir a l'autentificació ha de prémer el botó **Acceptar**.

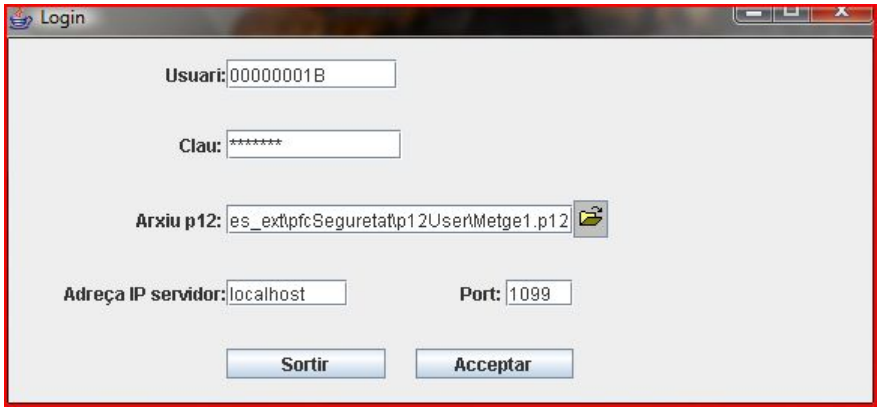

Figura 26: Finestra d'autentificació del metge.

8.3.2. Interfície principal.

Una vegada s'ha autentificat correctament el metge, accedeix a la seva finestra principal que l'indica el tipus d'usuari que és i el seu nom, recuperat tot del certificat que es troba a l'arxiu p12 seleccionat. Aquesta interfície permet fer dues accions diferents: **Obtenir la seva llista de pacients** o **Tancar la sessió**. Per executar-les només cal seleccionar l'opció i prémer el botó **Acceptar**.

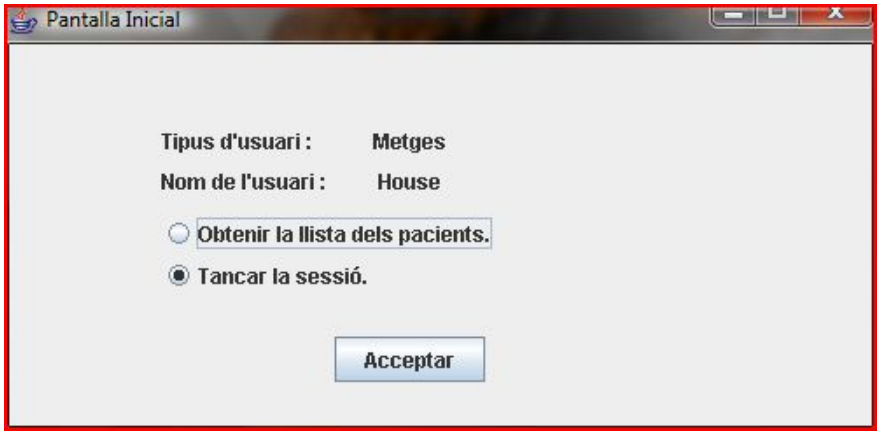

Figura 27: Finestra principal del metge.

Tal i com s'ha explicat amb el pacient, per causes de robustesa i facilitat d'ús per a l'usuari, aquesta finestra no permet ni seleccionar un pacient en concret per visualitzar el seu historial ni afegir una visita a un pacient atès que és molt difícil que un metge pugui recordar els pacients que té assignats i s'ha pensat que seria més efectiu no posar aquestes opcions directament sinó que es fan a partir de la llista de pacients que té assignats el metge.

8.3.3. Consultar els pacients assignats.

En aquesta finestra, es pot observar que es visualitza una llista amb els pacients que té assignats el metge. Les dades de cada pacient que es mostra és el seu identificador, nom i cognoms. Dades necessàries per a que el metge pugui saber de quin pacient es tracta. Dins d'aquesta finestra podem visualitzar qualsevol historial o afegir una visita als pacients que té assignats. Per fer-ho només s'ha de seleccionar el pacient i prémer **Veure Historial** o **Afegir Visita** segons l'acció que el metge vulgui efectuar. A més a més a la part inferior hi ha un parell de botons, un anomenat **Pàgina Principal** que serveix per a tornar a la interfície principal i un altre, **Tanca sessió** que permet sortir d'una manera segura de l'aplicació tornant a la pàgina d'autentificació.

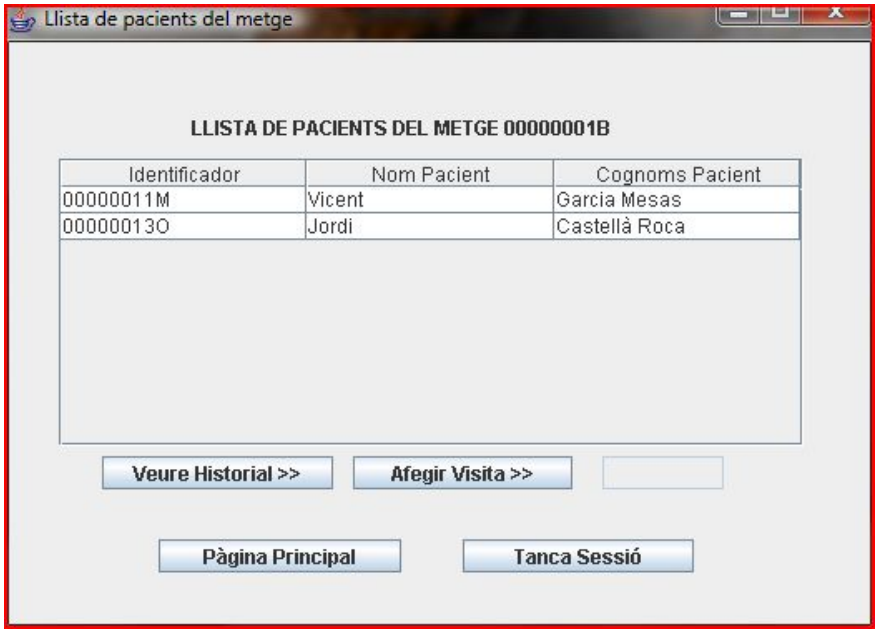

Figura 28: Finestra de llistar els pacients d'un metge.

8.3.4. Afegir una visita.

Aquesta finestra permet a un metge afegir una altra visita al pacient seleccionat. Per fer-la més fàcil d'utilitzar i evitar dades incorrectes, s'ha dissenyat llistes de selecció per als camps data i hora de manera que el metge només ha d'indicar el dia, mes, any, hora i minut de les llistes. Els camps que són necessaris s'han marcat amb un asterisc. Si el metge deixa algun d'aquests camps sense omplir, l'aplicació mostra un missatge d'informació indicant el camp que falta. Per donar d'alta la visita al sistema una vegada s'ha omplert els camps necessaris, s'ha de prémer el botó **Afegeix Visita**. El sistema mostra un missatge de confirmació amb les dades que es guardaran a la base de dades. En el cas que el metge no vulgui donar d'alta la visita, pot prémer el botó **Cancel·lar** o bé tancar la finestra.

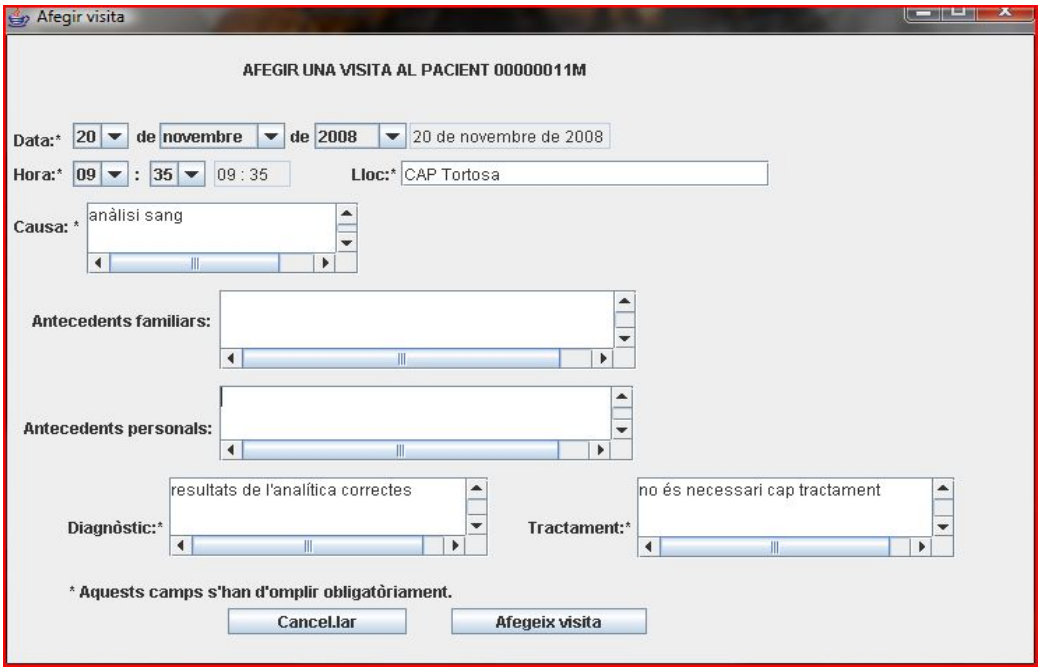

Figura 29: Finestra de donar d'alta una nova visita.

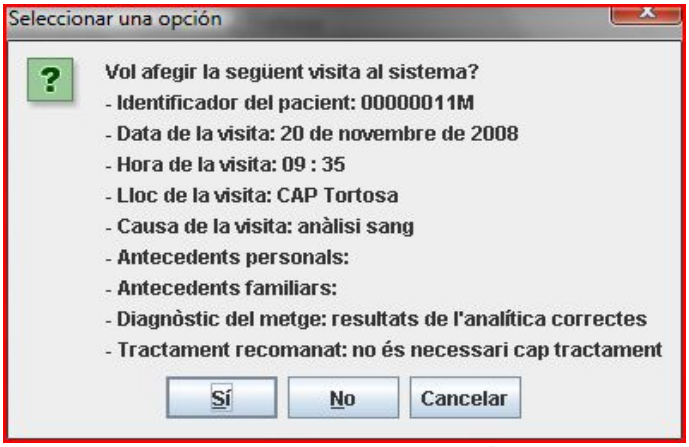

Figura 30: Finestra de confirmació de dades.

Atès que els camps lloc, causa, antecedents familiars, antecedents personals, diagnòstic i tractament tenen mides predefinides a la base de dades, si alguna d'aquestes dades sobrepassa aquest límit, el sistema mostra un missatge d'informació indicant quin camp s'ha sobrepassat i el límit que té per a que el metge pugui canviar les dades errònies.

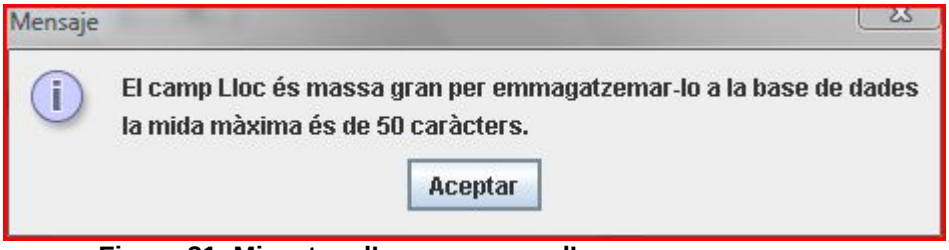

Figura 31: Missatge d'error a causa d'un camp massa gran.

8.3.5. Consultar l'historial d'un pacient assignat.

Aquesta finestra permet a un metge veure l'historial del pacient seleccionat a la finestra de llistar els seus pacients. Com es pot observar el disseny i funcionament és idèntic a la finestra anàloga del pacient per veure el seu historial. L'únic que canvia és el resultat de prémer el botó **Pàgina Anterior** que en aquest cas l'usuari metge tornaria a la seva llista de pacients.

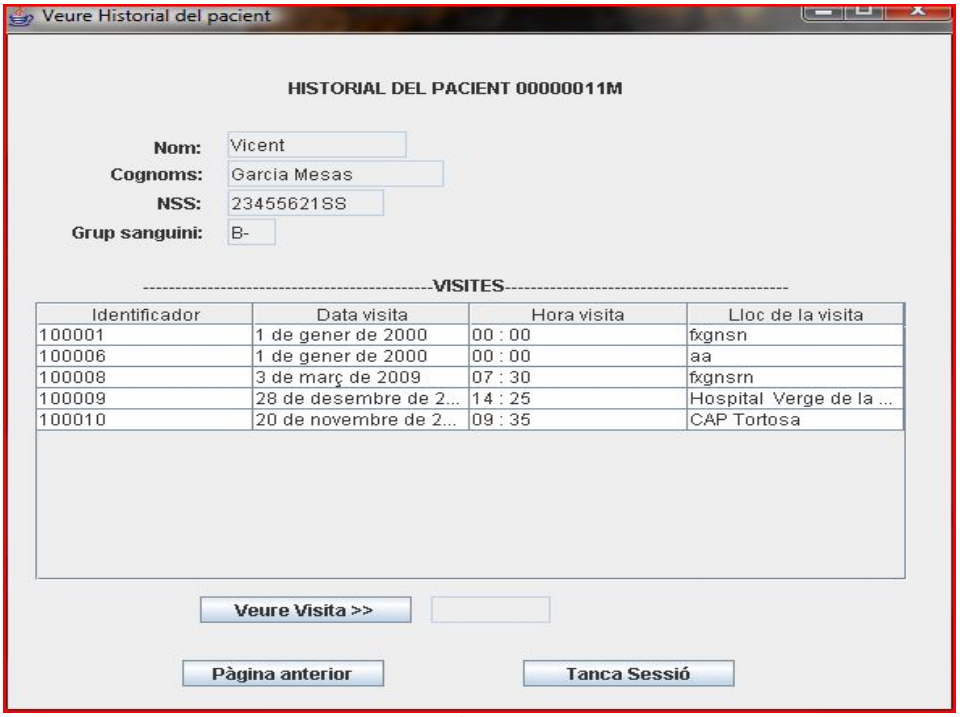

Figura 32: Finestra de visualització de l'historial d'un pacient del metge.

8.3.6. Consultar una visita d'un pacient assignat.

En aquesta finestra el metge pot visualitzar les dades complertes d'una visita seleccionada de l'historial d'un dels seus pacients. El disseny i funcionament és idèntic a la finestra anàloga del pacient per veure una de les seves visites.

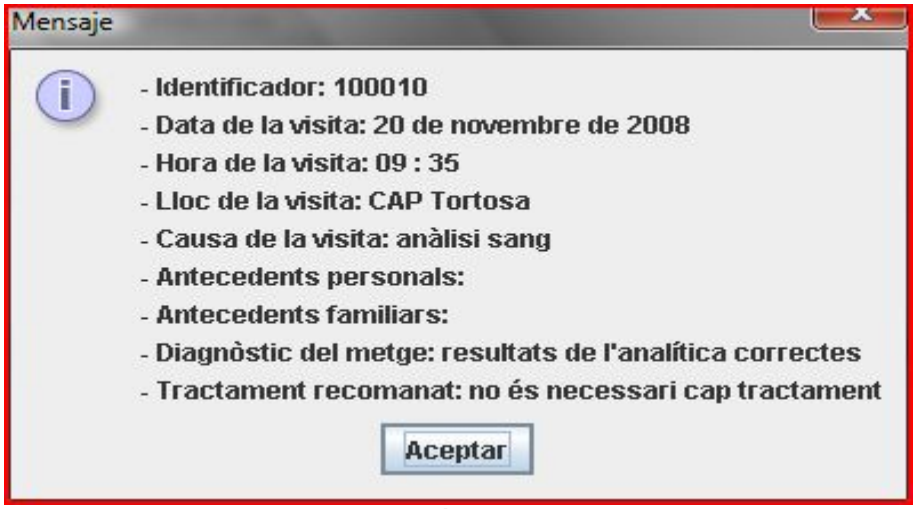

Figura 33: Finestra de visualització d'una visita d'un pacient del metge.

8.3.7. Sortir del sistema.

El metge té vàries maneres de sortir adequadament del sistema. La primera d'elles es troba a la finestra principal seleccionant l'opció **Sortir del sistema** i prement el botó **Acceptar**. Una altra manera es troba a la finestra de llista els pacients que té assignats prement el botó **Tanca Sessió**. Una altra manera es troba a la finestra de visualització de l'historial d'un dels seus pacients prement també el botó **Tanca Sessió**. A més a més si es tanquen les finestres de la interfície principal, llistar els pacients o de visualització de l'historial d'un pacient també s'executa l'acció de tancar la sessió. Qualsevol de les maneres de tancar la sessió mostra un missatge especificant el resultat del tancament i en el cas que s'hagi efectuat correctament, et torna a la pàgina d'autentificació. En el cas que no es pugui tancar la sessió, l'aplicació es tanca directament.

Figura 34: Missatge de tancament correcte de la sessió.

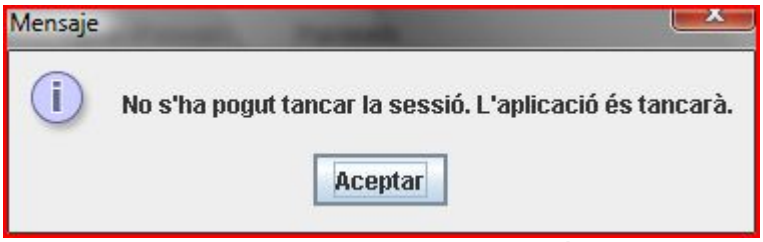

Figura 35: Missatge de tancament de sessió amb errades.

8.4. Interfície gestor.

Tal i com s'especifica en el capítol 2, l'aplicatiu gestor ha d'oferir les següents funcionalitats:

- Registrar a nous usuaris (metges o pacients).
- Autenticar als pacients i als metges que volen accedir al repositori.
- Acceptar les consultes dels metges i dels pacients.
- Guardar de forma segura els historials mèdics dels pacients.
- Verificar que les dades que s'han inserit o modificat en un historial mèdic s'ha realitzat per un usuari autoritzat.
- Permetre que els usuaris abandonin el sistema de forma segura.

8.4.1. Autentificació.

L'entrada a l'aplicació del gestor també requereix que s'identifiqui com a tal. No obstant per fer-ho només necessita un certificat on el camp OU sigui "Gestio" i un parell de claus , pública i privada, per efectuar la comunicació segura i signar i xifrar el contingut de la base de dades necessari. El gestor ha d'especificar la seva clau secreta de l'arxiu p12, i la drecera on es troba aquest arxiu. Per facilitar la tasca de cerca s'ha facilitat un botó que busca aquest arxiu al directori que té l'aplicació per guardar els arxius p12 de gestió. Per procedir a l'autentificació s'han d'omplir les dades i prémer el botó **Acceptar**. El botó **Sortir** permet tancar directament l'aplicació.

Figura 36: Finestra d'autentificació del gestor.

En el cas que no s'introdueixin dades o que aquestes no siguin correctes per engegar el sistema, l'aplicació mostra un missatge d'error indicant la causa. A tall d'exemple es mostra el missatge quan s'introdueix un arxiu p12 amb un certificat que no és d'un gestor:

Figura 37: Missatge d'error certificat de gestió erroni.

8.4.2. Interfície principal.

Una vegada s'ha autentificat correctament l'usuari gestor, es mostra la interfície principal de gestió de l'aplicació. Aquesta finestra mostra en una primera instància un parell de caixes de text per indicar un nom d'usuari que tingui accés a la base de dades i la seva corresponent clau. Un botó **Engegar BBDD** que permet enviar les dades de l'usuari per poder iniciar la comunicació amb la base de dades, un botó **Visualitza un log** que permet veure un log emmagatzemat al directori de l'aplicació i un botó **Tanca Aplicació** que permet tancar l'aplicació. Tots els demés components, estan o bé no visibles o bé deshabilitats. Per tant la primera tasca que ha de fer el gestor del sistema serà introduir les dades de connexió amb la base de dades i engegar-la. En el cas d'exemple, s'ha utilitzat "*root*" com a usuari i "*vgm*" com a clau.

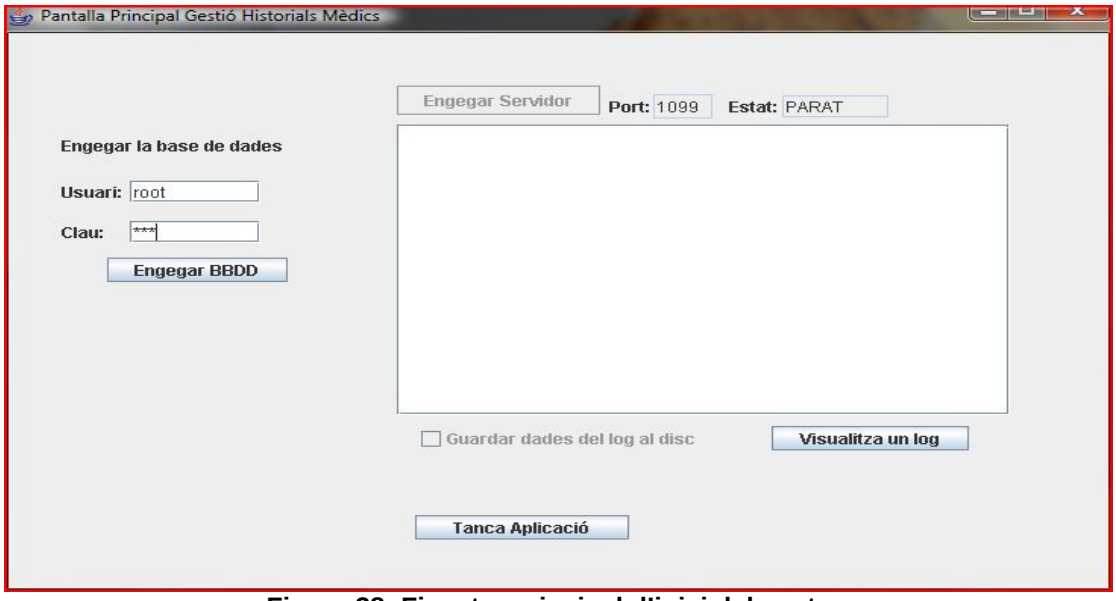

Figura 38: Finestra principal d'inici del gestor.

Una vegada funciona correctament la comunicació amb la base de dades, es visualitza totes les tasques de manteniment que pot fer el gestor i s'habilita l'encesa del servidor que permet la comunicació amb els usuaris remots. A més a més hi ha una casella de selecció, **Guardar dades del log al disc**, que permet guardar el registre de comunicació al directori *logs* de l'aplicació per poder-lo recuperar quan es vulgui mitjançant el botó **Visualitza un log**. Atès que les dades que es registren poden arribar a comprometre la seguretat del sistema, abans de guardar-se, es xifren amb la clau privada del gestor i es signen amb la seva clau pública.

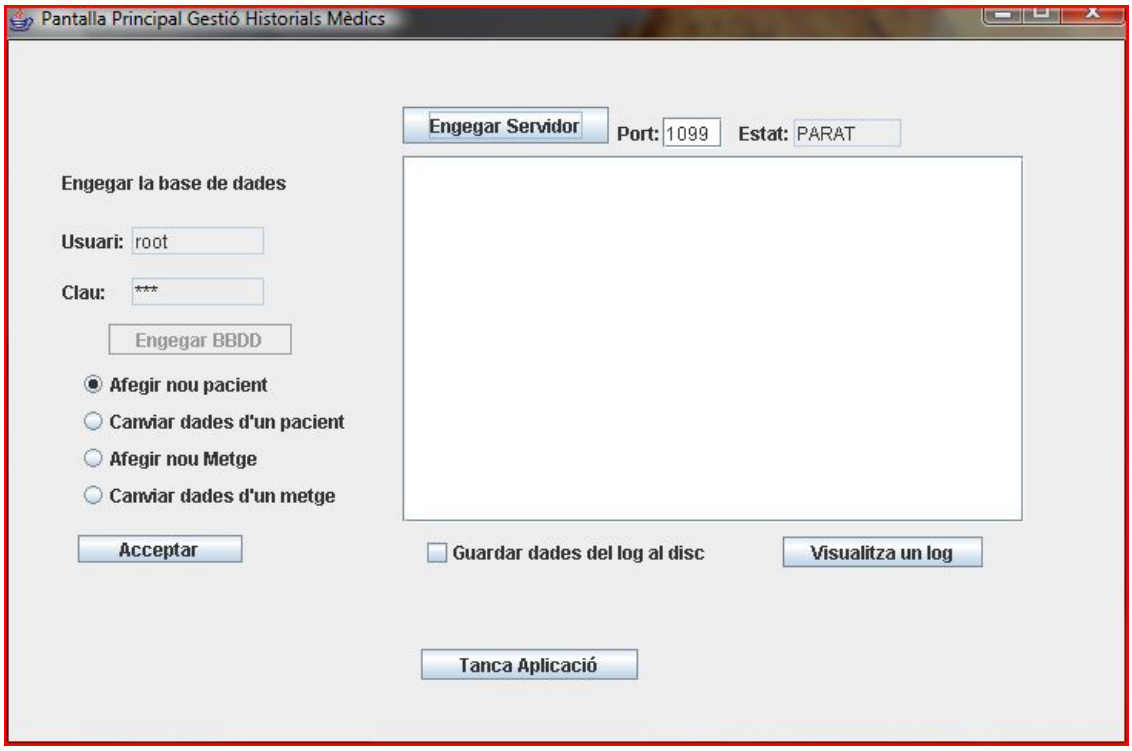

Figura 39: Finestra principal del gestor amb la base de dades funcionant.

8.4.3. Servidor de dades.

Per poder habilitar la comunicació amb els usuaris remots, s'ha d'engegar el servidor. La manera de fer-ho és molt simple, primer s'ha de seleccionar el port de comunicació. Per defecte s'utilitza el port que la comunicació RMI té, que és el 1099. No obstant aquest port es pot canviar sempre i quan estigui dins el rang de ports disponibles del sistema. Una vegada s'indica el port de desplegament, només s'ha de prémer el botó **Engegar Servidor**. A la finestra de log es pot visualitzar el resultat del desplegament.

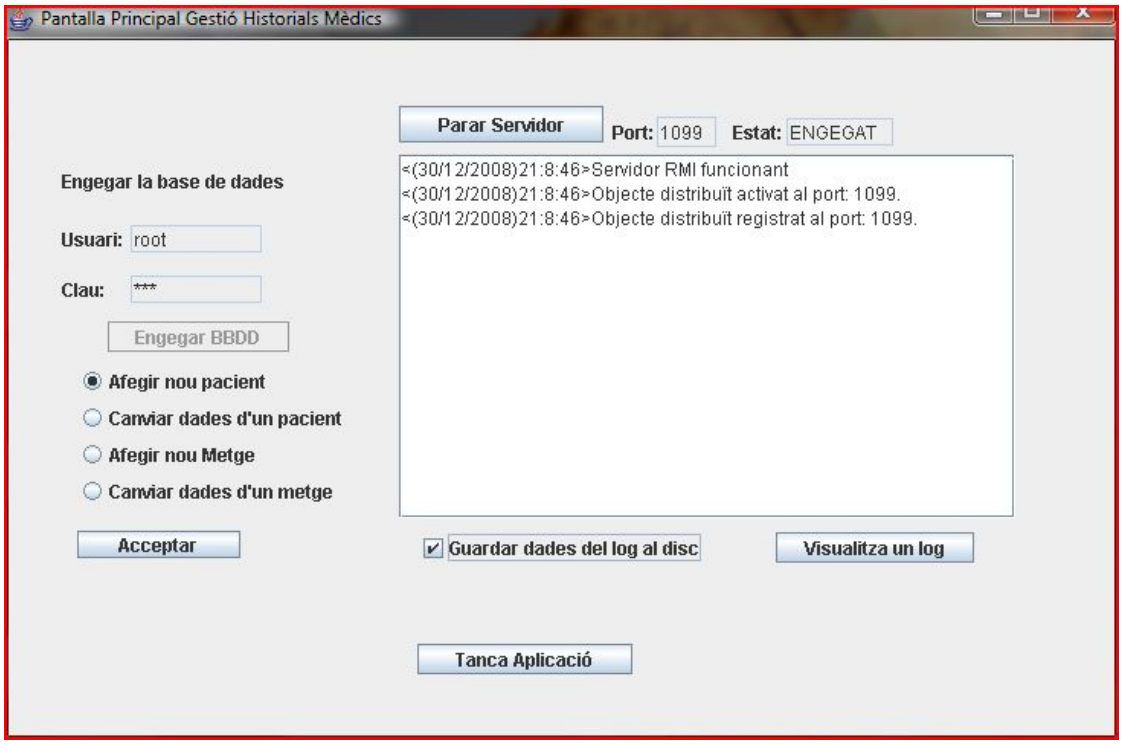

Figura 40: Finestra principal del gestor amb el servidor funcionant.

8.4.4. Servei de registre de trànsit.

El servei de registre de trànsit indica totes les gestions que està fent el sistema per facilitar les dades que se li demanen. A més a més tal i com s'ha indicat, sempre i quan es guardi aquest registre al disc, es pot recuperar prement el botó Visualitza un log. El resultat de l'acció d'aquest botó és el següent:

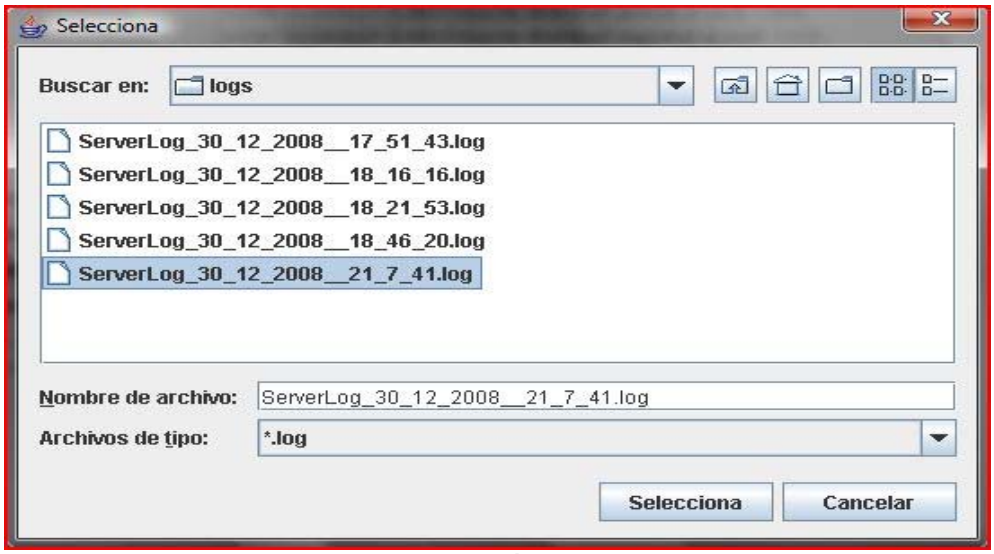

Figura 41: Finestra de selecció de l'arxiu log a visualitzar.

Una vegada s'ha seleccionat l'arxiu a visualitzar, s'obre una altra finestra amb el contingut de l'arxiu log prèviament desxifrat i analitzat la signatura del gestor per assegurar la integritat de les dades.

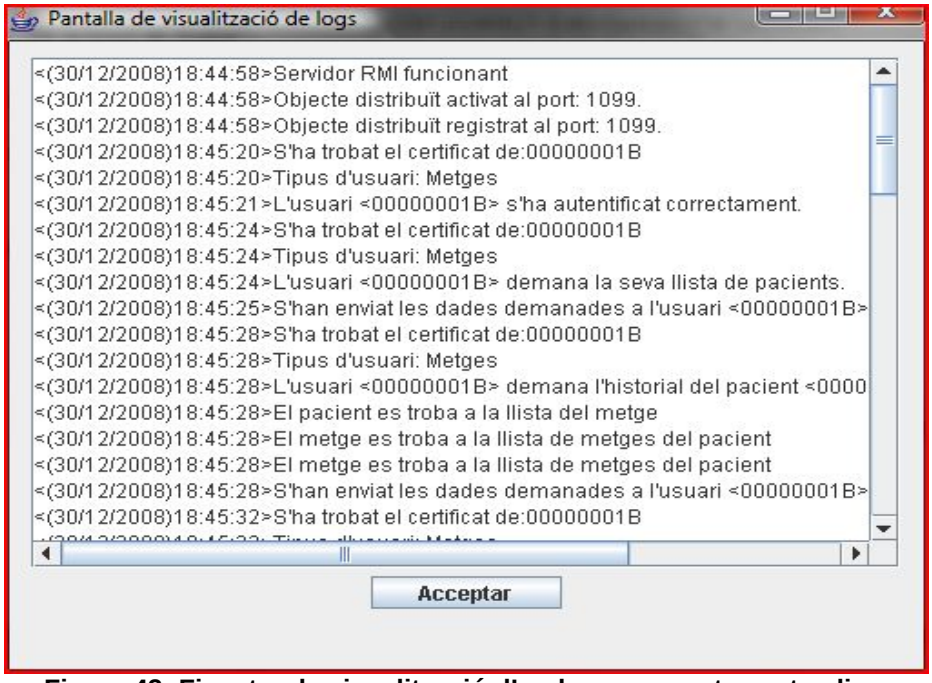

Figura 42: Finestra de visualització d'un log emmagatzemat a disc.

8.4.5. Afegir nous usuaris.

Segons el capítol 2 d'aquesta documentació, una altra de les tasques que se li ofereixen al gestor és la d'afegir o modificar usuaris al sistema. Per portar-la a terme, a la finestra principal de l'aplicació una vegada engegada la connexió amb la base de dades es pot veure quatre opcions:

- Afegir nou pacient.
- Canviar dades d'un pacient.
- Afegir nou metge.
- Canviar dades d'un metge.

Per afegir un nou pacient, el gestor ha de seleccionar la primera opció i prémer el botó **Acceptar** a la finestra principal de l'aplicació. Una vegada fet això es mostra una nova finestra on es pot donar d'alta el pacient. Per facilitar la tasca del gestor i evitar errades inconsistents, el certificat del pacient només es pot seleccionar a partir del botó de cerca. Una vegada seleccionat, el sistema comprova que realment sigui el certificat d'un pacient i mostra l'identificador que hi consta dins. Aquestes dades no es poden modificar doncs són les dues dades que identifiquen unívocament a un

usuari pacient. La resta de dades tal com el nom del pacient, els cognoms, el número de la seguretat social i el grup sanguini, si que s'introdueixen manualment però amb la restricció de longitud de camp que té la base de dades. Si es volgués donar d'alta un usuari amb una longitud de camp més gran del degut, el sistema mostra un missatge d'informació. A més a més si es vol donar d'alta a un usuari que ja ho està, el sistema mostra un missatge d'informació. La finestra disposa de dos botons, un **Donar d'alta** per afegir les dades al sistema i un altre **Cancel·lar** per ignorar les dades i tancar la finestra.

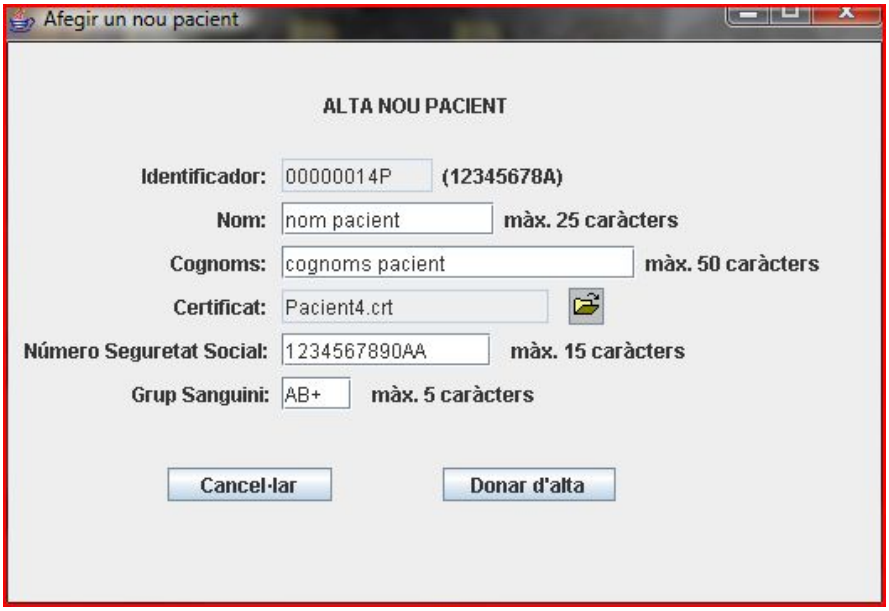

Figura 43: Finestra per donar d'alta un nou pacient.

Per afegir un nou metge, el gestor ha de seleccionar la tercera opció i prémer el botó **Acceptar** a la finestra principal de l'aplicació. Una vegada fet això es mostra una nova finestra on es pot donar d'alta el metge. Per facilitar la tasca del gestor i evitar errades inconsistents, el certificat del metge només es pot seleccionar a partir del botó de cerca. Una vegada seleccionat, el sistema comprova que realment sigui el certificat d'un metge i mostra l'identificador que hi consta dins. Aquestes dades no es poden modificar doncs són les dues dades que identifiquen unívocament a un usuari metge. La resta de dades tal com el nom del metge, els cognoms i el número de col·legiat, si que s'introdueixen manualment però amb la restricció de longitud de camp que té la base de dades. Atès que quan es dóna d'alta un metge s'ha d'especificar la seva llista de pacients, a la finestra es mostra un llistat de tots els pacients que estan a la base de dades on es pot seleccionar els pacients que s'assigna al nou metge. Si es vol donar d'alta a un metge que ja ho està, el sistema mostra un missatge d'informació. La finestra disposa de dos botons, un **Donar d'alta** per afegir les dades al sistema i un altre **Cancel·lar** per ignorar les dades i tancar la finestra. El procediment de donar d'alta a un metge va lligat a més a més a afegir a la llista de metges de l'historial de cada pacient assignat, l'identificador del nou metge. D'aquesta manera el sistema pot saber si un metge en concret té accés a un historial

o no a partir de la llista de pacients del metge i la llista de metges de l'historial del pacient.

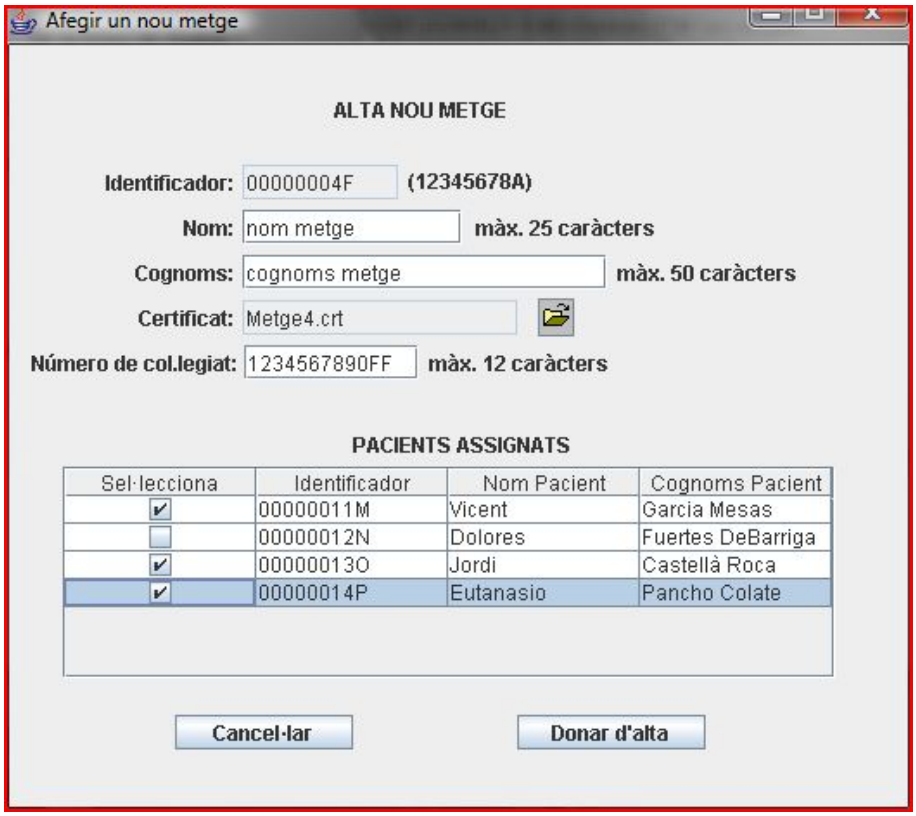

Figura 44: Finestra per donar d'alta un nou metge.

8.4.6.Modificar dades dels usuaris.

Per modificar les dades d'un pacient prèviament donat d'alta, el gestor ha de seleccionar la segona opció i prémer el botó **Acceptar** a la finestra principal de l'aplicació. Una vegada fet això sobre una nova finestra que llista tots els pacients que estan donats d'alta al sistema. el gestor ha de seleccionar el pacient a modificar i prémer el botó **Acceptar**. Si no es vol modificar les dades de cap pacient, només cal prémer el botó **Cancel·lar**.

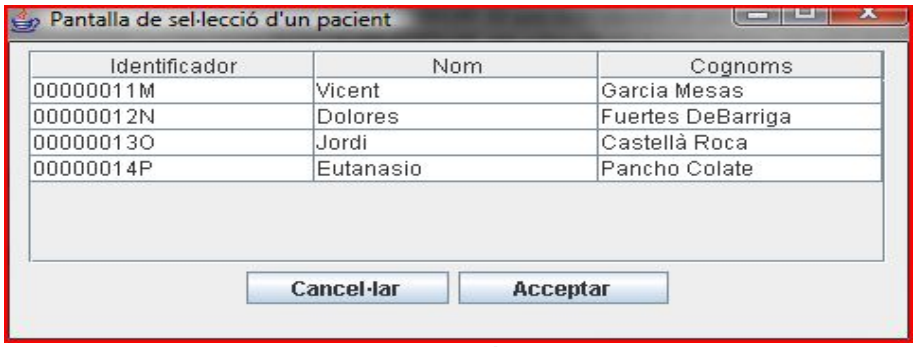

Figura 45: Finestra de selecció del pacient a modificar.

Una vegada seleccionat el pacient a modificar, s'obre una altra finestra amb les dades del pacient que estan emmagatzemades a la base de dades. Per evitar errades no es pot canviar ni l'identificador ni el certificat atès que aquestes dades són les que l'identifiquen unívocament. Per tant les dades modificables són el nom, cognoms, número de la seguretat social i grup sanguini. Per efectuar la modificació doncs, el gestor canvia les dades necessàries tenint sempre en compte la restricció de mida de cada camp i pitja el botó **Modificar**. Si no es vol efectuar cap canvi, només cal prémer el botó **Cancel·lar** que tanca la finestra sense actualitzar les dades.

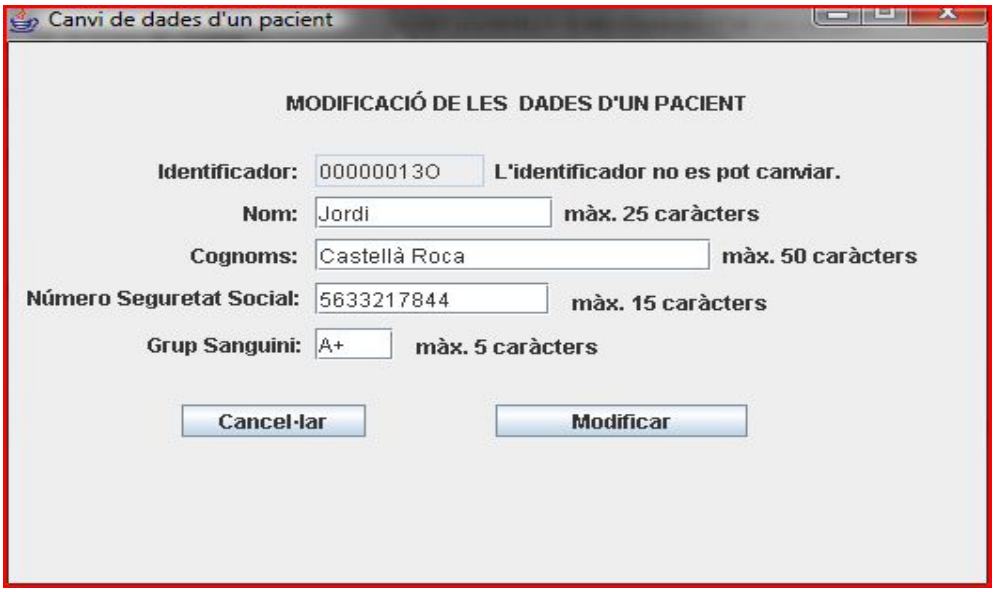

Figura 46: Finestra de modificació de les dades d'un pacient.

Per modificar les dades d'un metge prèviament donat d'alta, el gestor ha de seleccionar la quarta opció i prémer el botó **Acceptar** a la finestra principal de l'aplicació. Una vegada fet això sobre una nova finestra que llista tots els metges que estan donats d'alta al sistema. el gestor ha de seleccionar el metge a modificar i prémer el botó **Acceptar**. Si no es vol modificar les dades de cap metge, només cal prémer el botó **Cancel·lar**.

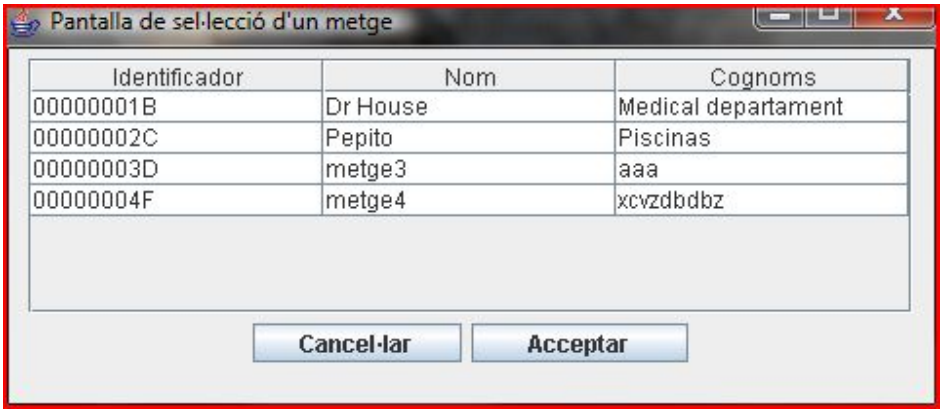

Figura 47: Finestra de selecció del metge a modificar.

Una vegada seleccionat el metge a modificar, s'obre una altra finestra amb les dades del metge que estan emmagatzemades a la base de dades. Per evitar errades no es pot canviar ni l'identificador ni el certificat atès que aquestes dades són les que l'identifiquen unívocament. Per tant les dades modificables són el nom, cognoms i número de col·legiat. A més a més es visualitza una llista de tots els pacients que estan registrats a la base de dades i seleccionats aquells que ja té assignats. D'aquesta llista es pot selecciona o desseleccionar qualsevol pacient per generar la nova llista de pacients assignats al metge.

Per efectuar la modificació doncs, el gestor canvia les dades necessàries tenint sempre en compte la restricció de mida de cada camp i pitja el botó **Modificar**. Si no es vol efectuar cap canvi, només cal prémer el botó **Cancel·lar** que tanca la finestra sense actualitzar les dades.

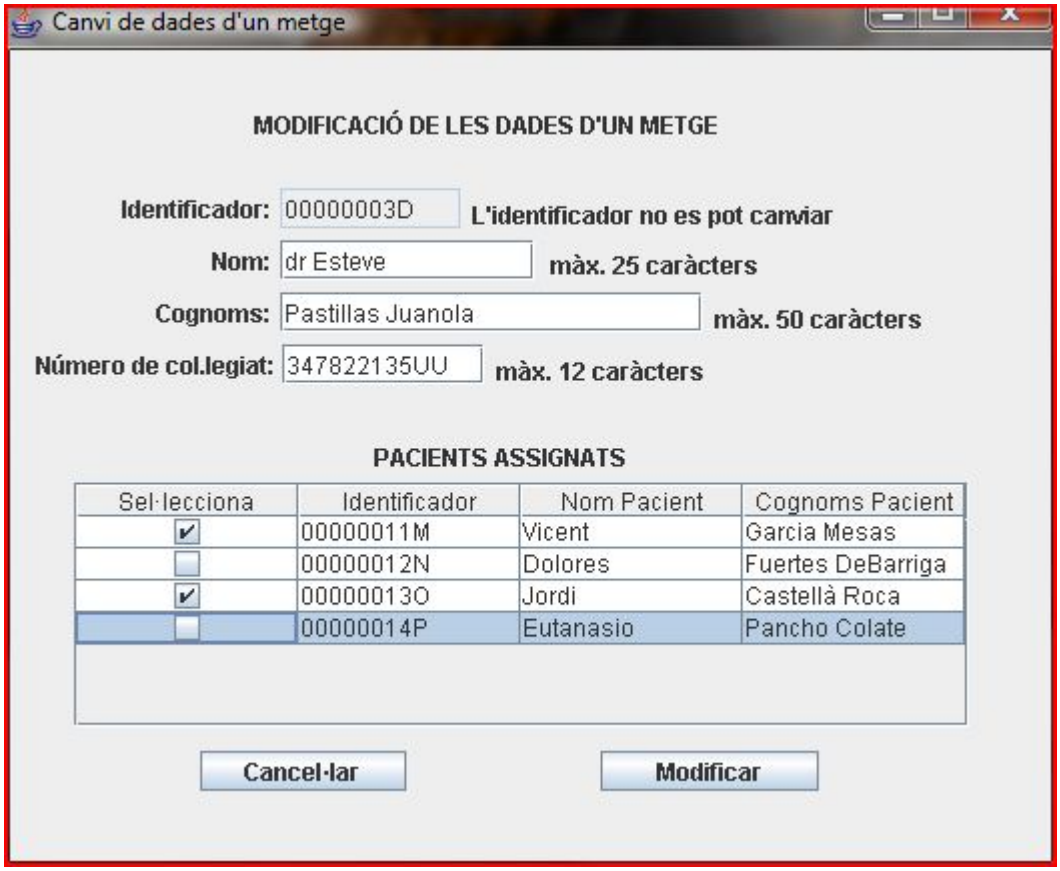

Figura 48: Finestra de modificació de les dades d'un metge.

8.5. Implementació.

Per fer la implementació d'aquesta part, ha estat necessari afegir una nova classe per cada interfície dissenyada tant per als usuaris com per al gestor. Aquestes classes a banda de dissenyar l'aspecte visual de cada interfície s'encarrega de gestionar els esdeveniments que demana cada usuari tot instanciant a les classes *manager* en cada cas. A més a més s'ha afegit una nova classe per a usuari (*RunClient*) i per a gestor (*RunGestor*) que és l'entrada a l'aplicació en cada cas. Aquesta nova classe s'encarrega de visualitzar en primer terme la interfície d'autentificació de l'usuari o del gestor segons sigui el cas.

Per fer la implementació de les interfícies gràfiques s'han utilitzat les llibreries *AWT (Abstract Window Toolkit*) i *swing* que porta integrades la JRE de java per defecte. A més a més i atès que s'ha utilitzat l'IDE de desenvolupament *Eclipse* per efectuar el projecte, ha estat necessari instal·lar un plugin de disseny gràfic d'interfícies java. Aquest plugin s'anomena *jigloo* i es pot descarregar gratuïtament de la plana web de cloudGarden. Descàrrega del plugin->

Per provar el funcionament de l'aplicació, s'ha creat un arxiu per lots per al client anomenat *runClient.bat* i per al gestor anomenat *runGestor.bat* al directori arrel de l'aplicació.

8.5.1. Diagrama de classes part client.

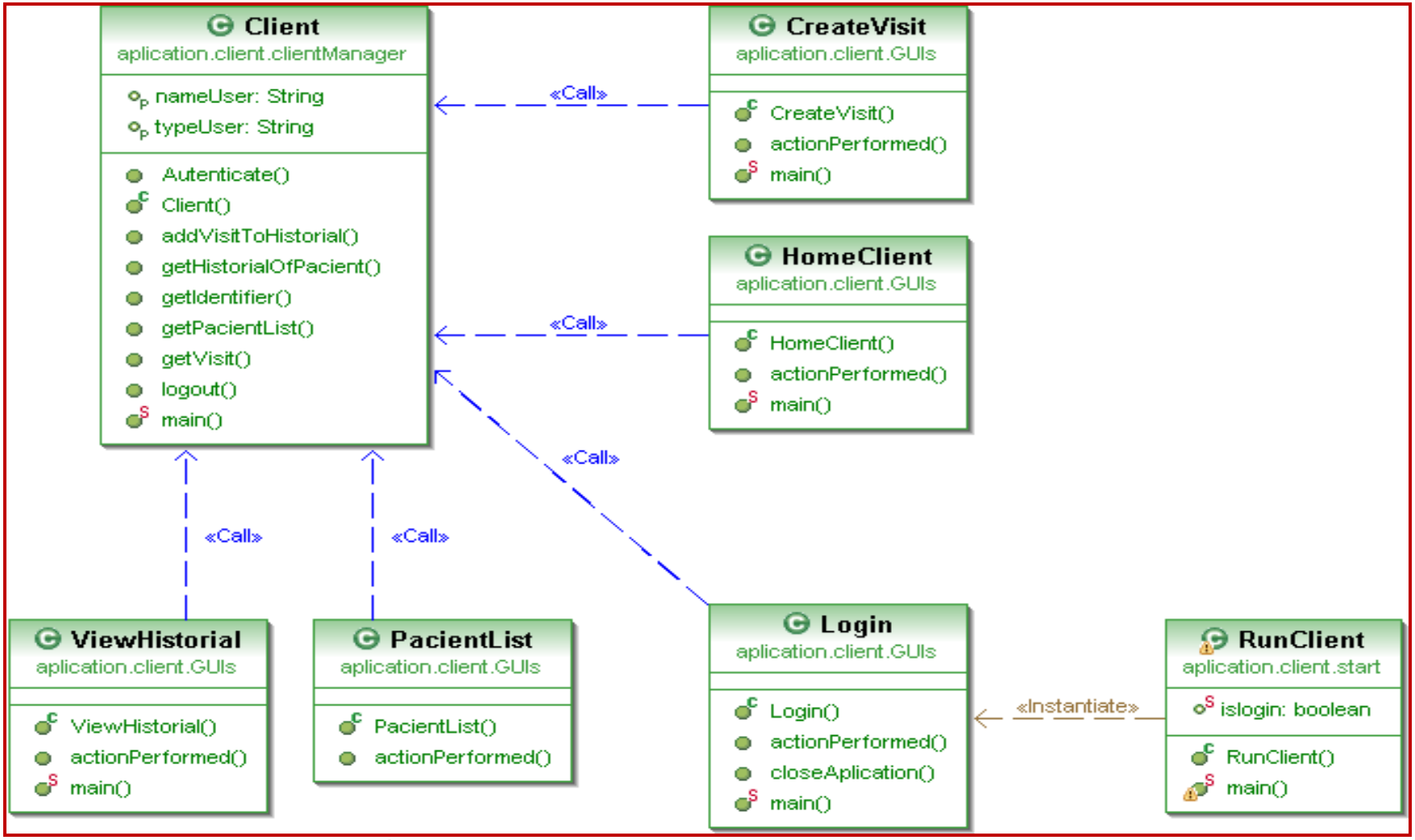

Figura 49: Diagrama de classes interfície client.

8.5.2. Diagrama de classes part gestor.

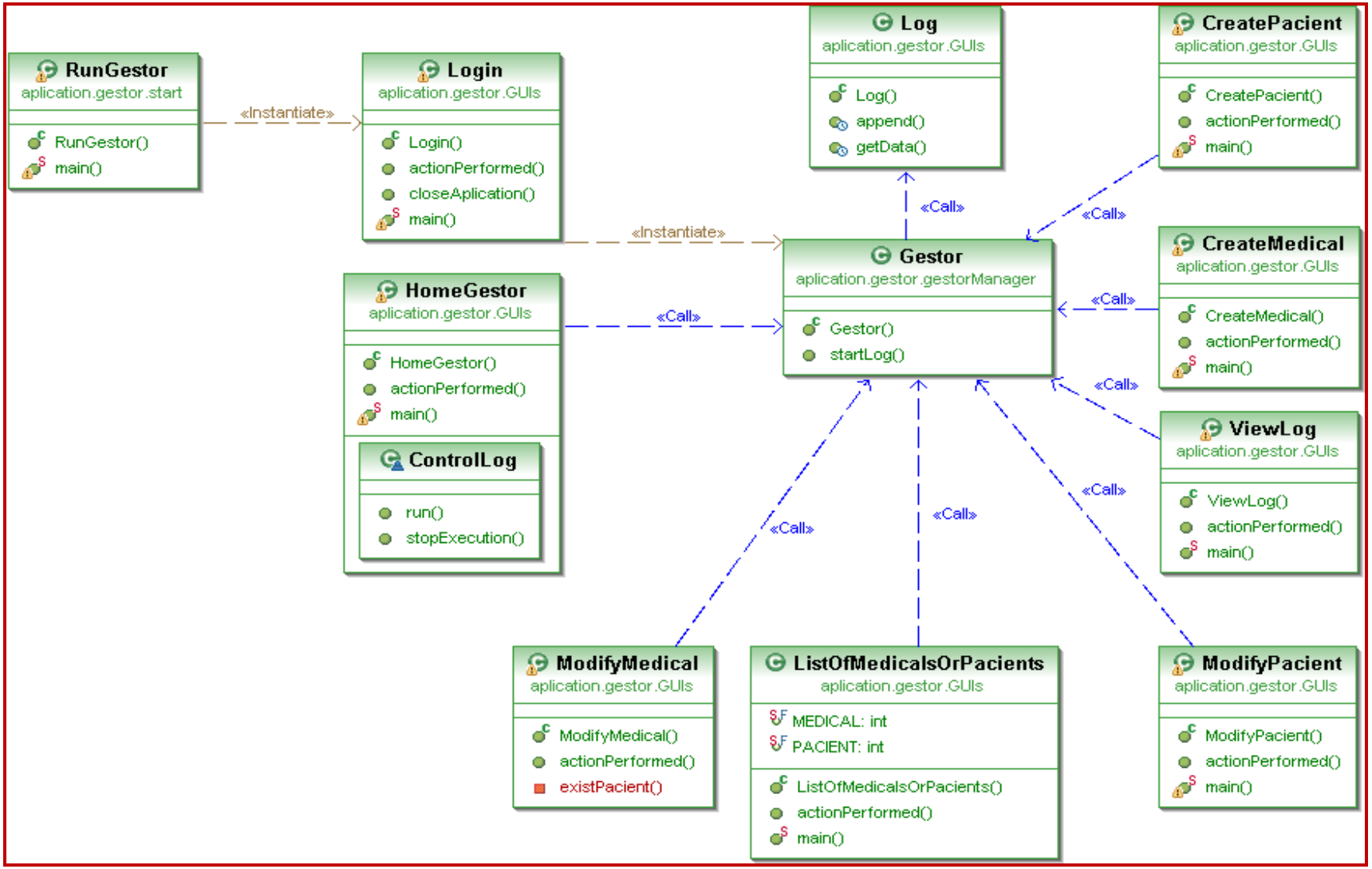

Figura 50: Diagrama de classes interfície gestor.

9. Conclusions i línies de treball futures.

9.1. Conclusions.

Una vegada finalitzat el disseny i desenvolupament d'aquest projecte, es posa de manifest les fites o objectius aconseguits. La idea bàsica objecte era crear un sistema capaç de gestionar d'una manera segura una sèrie de documents clínics en un sistema distribuït. Per veure l'estat d'assoliment del mateix, es comenta part per part les fites assolides i el que s'ha aconseguit.

- S'ha dissenyat un sistema criptogràfic per proveir de seguretat tota gestió de dades a fi i efecte d'assegurar tots els aspectes de seguretat que es requereixen. Aquests aspectes són autenticitat, confidencialitat, integritat i no-repudi. S'ha utilitzat un sistema de clau pública per efectuar-ho, atès que és el sistema que s'adequava millor per portar a terme el sistema criptogràfic. Una vegada desenvolupat el projecte, es pot dir que compleix tots els requisits de seguretat demanats, obtenint un producte segur per a la gestió de dades confidencials.

- S'ha desenvolupat un sistema de representació de dades que permet l'intercanvi estructurat i actual. L'ús xml per a la representació de dades assegura una forma eficient i clara d'emmagatzemar i transportar dades d'una aplicació a una altra.

- S'ha utilitzat RMI (Remote method Invocation) per establir la comunicació entre els diferents clients (metges o pacients) amb el gestor central del sistema de manera que qualsevol client pot obtenir les dades que sol·liciti, independentment de la seva ubicació. Aquest sistema, tot i que no és el més eficient actualment, és una manera senzilla i pràctica d'enviar dades per un sistema distribuït com pot ser Internet o qualsevol xarxa de comunicacions.

- S'ha utilitzat una base de dades relacional per a poder emmagatzemar d'una manera eficient les dades que utilitza el sistema. Per portar a terme aquesta part del projecte, s'ha optat pel gestor de Base de Dades MySQL, que ofereix un bon rendiment pel volum de dades que es gestiona en aquest PFC i a més a més està disponible en les principals plataformes. Per la complexitat i els objectius fixats en el projecte, és una de les millors opcions que hi ha disponibles en el mercat. Les dades emmagatzemades compleixen els requeriments de seguretat proposats, de manera que si qualsevol persona no autoritzada accedís a la base de dades, no podria vincular la relació existent entre pacients, metges i visites, atès que aquesta informació es troba xifrada i signada pel gestor del sistema.

- Finalment, s'ha proveït d'interfícies gràfiques tant als clients com al gestor central per a facilitar la tasca de visualització, gestió i demanda de dades, és a dir, les funcionalitats demanades a l'aplicació. Per portar-ho a terme s'ha

utilitzat les llibreries gràfiques que incorpora l'SDK de SUN, que són SWT i swing. A més a més s'ha dissenyat un sistema de log que permet emmagatzemar totes les tasques que fa el sistema quan està en funcionament. D'aquesta manera quan es detectin problemes de comunicació o accessos no permesos, es pot tenir constància de tot el que ha fet el sistema.

Com a resultat d'aquest projecte doncs, s'ha obtingut un producte plenament funcional que gestiona de manera segura els historials dels pacients en un entorn distribuït.

Finalment com a conclusió personal, s'ha de dir que el disseny i desenvolupament d'aquest projecte m'ha permès aprofundir en una gran varietat de tecnologies tals com sistema de clau pública, representació xml, comunicacions remotes, gestió de la informació i interacció humana amb els ordinadors; tecnologies vistes en assignatures durant el transcurs de l'enginyeria. Tal i com va exposar el consultor Jordi Castellà, aquest projecte és prou complert, doncs per desenvolupar-lo és necessari assolir o posar de manifest una sèrie de coneixements molt amplis i diversos i no només en aspectes de seguretat, que era el tema principal del projecte. Aquest projecte no s'hagués portat a terme tal i com s'ha efectuat sense l'ajuda inestimable del meu consultor Jordi, no només pels dubtes que m'ha resolt d'una manera ràpida i eficient, sinó pels ànims que m'ha donat missatge rere missatge i m'ha donat forces per acabar-lo exitosament. Sincerament, gràcies.

9.2. Línies de treball futures.

Tot i la perfecta funcionalitat que té l'aplicació, no és pot dir que estigui acabada doncs encara es pot desenvolupar alguns aspectes que afegirien millores o ampliacions substantives al mateix. A continuació es proposen algunes d'elles:

- En la part criptogràfica seria convenient un mètode d'obtenir i gestionar els certificats dels actors del sistema d'una manera més segura, doncs ara per ara s'utilitza una petita PKI local creada a tal efecte i pot ser objecte d'atacs per a falsificar els certificats i tenir accés al sistema un usuari malintencionat. Per millorar aquest aspecte es podria utilitzar una CA externa i segura que s'encarregui de l'autenticitat dels certificats i en la seva revocació en cas contrari. A tal d'exemple es podrien nomenar VeriSign o Thawte .
  A més a més, les claus privades dels usuaris s'obtenen a partir d'arxius p12 que han d'estar emmagatzemats localment en cada ordinador on s'executi l'aplicació, una altra font d'atacs. Per millorar-ho es podria utilitzar una *smart-card* criptogràfica.

- En la part de representació de dades, es podria utilitzar xml per a l'intercanvi de dades també entre la base de dades i el gestor del sistema atès que ara

per ara les dades es guarden en components de dades, xifrats o en clar, però sense utilitzar xml. D'aquesta manera tota la comunicació entre els diferents components es faria en xml i es podria beneficiar de les millores que es detallen en l'apartat 5.

- En la part de comunicació remota es podria substituir RMI per Web Services, tecnologia més actual i que permet integrar-se més fàcilment a les tecnologies que s'utilitzen actualment. A més a més per a proveir de més seguretat a la comunicació remota, es podria utilitzar SSL per a establir un canal segur de comunicació.

- En la part de la gestió de la informació, les millores són bastants. Una possible millora doncs seria proveir de seguretat i integritat a les dades personals dels metges i pacients que ara per ara estan en clar. Inclús les dades de la visita, tot i que es troben signades pel metge que les ha creat, és podrien guardar xifrades pel gestor a fi i efecte d'evitar que ningú no autoritzat les pugui veure. A més a més es podria afegir molta més informació a la base de dades, des d'imatges de radiografies fins a contingut multimèdia de seguiment de certes operacions.

- Finalment a la part de la interfície gràfica, es podria millorar substancialment el disseny i navegació entre pantalles. Atès el poc temps que hi havia per desenvolupar el projecte, el disseny de les interfícies ha estat el més simple possible sense que això impliqui en un minvament de funcionalitats. És a dir, és podrien dissenyar interfícies més amigables de manera que l'usuari li sigui més atractiu interaccionar amb el sistema. A més a més és podria millorar la gestió dels esdeveniments afegint combinacions de tecles per a fer una certa funcionalitat o equiparar la tecla "enter" del teclat amb el botó "acceptar" de les pantalles, si escau. A banda de tot això i per evitar que els usuaris s'hagin d'instal·lar cap programari especial, una millora considerable seria migrar les interfícies per a que es puguin visualitzar en un navegador web. D'aquesta manera aprofitaríem plenament la funcionalitat que ofereix xml atès que hi ha eines que permeten crear planes web a partir del contingut xml directament.

10. Glossari.

- **Anamnesi**: És la reunió de dades subjectives, relatives a un pacient, que es compon d'antecedents familiars i personals, experiències i, en particular, records, que s'utilitzen per analitzar la seva situació clínica.

- **API**: *Application Programming Interface.* És un conjunt de funcions residents en biblioteques que permeten a una aplicació funcioni sota un determinat sistema operatiu.

- **AWT**. És un kit de ferramentes de gràfiques, interfícies d'usuari i sistema de finestres independents de la plataforma original de Java. AWT forma part de les *Java Foundation Classes* (JFC).

- **Certificat digital:** Document digital que vincula una determinada clau pública a un usuari.

- **DTD.** *Document Type Definition*. Format heretat de l'SGML que permet d'expressar la sintaxi vàlida d'un document XML usant un llenguatge especial.

- **Esquema XML.** Vocabulari XML per a descriure les normes d'estructura, tipus de dades i valors, que s'apliquen a les dades estructurades que pugui contenir un document XML determinat.

- **Funcions hash.** És una funció que fa correspondre a un missatge *m* de mida variable una representació *H*(*m*) de mida fixa. Típicament, *H*(*m*) té de 128 a 160 bits i s'anomena *el valor hash del missatge*.

- **Historial mèdic:** Conjunt de dades relatives als antecedents mèdics d'una persona.

- **IAIK**: Llibreria criptogràfica per a Java.

- **IDE**. És un entorn de desenvolupament integrat, en anglès, *Integrated Development Environment* ('*IDE*'). És un programa composat per un conjunt de ferramentes per a un programador.

- **IPsec**: *Internet Protocol Security*. Conjunt de protocols que treballen en la capa 3 , capa de xarxa del model OSI, la finalitat dels quals és assegurar les comunicacions sobre el protocol IP, autentificant i xifrant cada paquet en un flux de dades. A més a més IPsec inclou protocols per a l'establiment de claus de xifrat.

- **Java**: Una tecnologia desenvolupada per Sun Microsystems per aplicacions de programari independent de la plataforma.

- **MySQL**: Gestor de bases de dades relacionals SQL de lliure distribució.

- **Protocol de Needham-Schroeder***.* Protocol d'intercanvi de claus que evita l'inconvenient de la distribució de claus. El fet innovador és que dos usuaris poden pactar una clau secreta compartida sobre un canal insegur.

- **PKI**: *Public Key Infrastructure.* Conjunt de maquinari, programari, persones, polítiques i procediments necessaris per a crear i gestionar certificats digitals basats en criptografia de clau pública.

- **RMI**: *Remote Method Invocation.* Mecanisme que ofereix Java per a invocar un mètode d'una manera remota.

- **SGML**: *standard generalized markup language*. Llenguatge de marques publicat el 1986 per a expressar en un mateix document les dades i la seva estructura lògica. L'HTML i l'XML són subconjunts de l'SGML definits pel consorci web.

- **Signatura digital**. És l'equivalent electrònic de la signatura convencional.

- **Skeleton**: Objecte en una comunicació remota que permet als servidors rebre invocacions remotes a mètodes implementats en objectes locals.

- **Sobre digital:** Tècnica de xifratge híbrida que aconsegueix treure partit dels avantatges dels criptosistemes simètrics i els criptosistemes de clau pública, tot utilitzant ambdues tècniques.

- **SSL/TLS**: *Secure Socket Layer/Transport Layer Security*. Protocols criptogràfics que proporcionen comunicacions segures per una xarxa. Proporcionen autentificació i privacitat de la informació entre els extrems d'una xarxa de comunicacions.

- **Stub**: Objecte en una comunicació remota que permet als clients invocar mètodes d'objectes remots. Quan l'aplicació client invoca un objecte definit en l'stub, aquest s'encarrega de fer arribar la invocació al seu destí.

- **Swing**. És una biblioteca gràfica per Java que forma part de les *Java Foundation Classes* (JFC). Inclou *widgets* per interfície gràfica d'usuari tals com caixes de text, botons, desplegables i taules.

- **text en clar:** Text, missatge o document que no ha estat xifrat i, per tant, és comprensible per a tothom que l'inspeccioni.

- **text xifrat:** Text, missatge o document un cop que ha estat xifrat i, per tant, incomprensible per a tothom que l'inspeccioni directament. El text xifrat té per defecte la propietat de confidencialitat.

- **XML:** *eXtensible Markup Language.* Especificació que defineix una sintaxi i unes regles sobre l'ús d'etiquetes per a estructurar la informació.

11. Bibliografia.

Alavedra, O.; Domingo, J.; Herrera, J.; Rius, M.A.; Sebé,F.; Soriano, M. (2002) *Comerç electrònic.* Barcelona:UOC

Ceballos, F.J. (2000) *Java2 Curso de programación*. Madrid: RA-MA Editorial

Cuenca, M.J.; Marco, M.J.; Nicolau, F. (2005) *Competència Comunicativa per a professionals de la informàtica*. Barcelona: UOC

Domingo, J.; Herrera, J. ; Rifà, H. (2004) *Criptografia*. Barcelona: UOC

Marquès, J.M.; Navarro, L. (2004) *Arquitectura de sistemes distribuïts.* Barcelona:UOC.

Megías,D.; Minguillón, J.; (2003) *Compiladors II*. Barcelona: UOC.

Zukowski, J. (1999) *Programación en Java2*. Madrid: Anaya Multimedia

URLs:
Portal wikipèdia:
http://es.wikipedia.org/wiki/Wikipedia:Portada

Plana web de l'aplicació OpenSSL:
http://www.openssl.org

Plana web de SUN:
http://java.sun.com

Plana web de IAIK:
http://jce.iaik.tugraz.at

Plana web de JDOM:
http://www.jdom.org/

Plana web on s'explica el funcionament i l'arquitectura RMI:
http://java.sun.com/javase/technologies/core/basic/rmi/whitepaper/index.jsp

Plana web de MySQL:
http://www.mysql.com/

Plana web de Jigloo:
http://www.cloudgarden.com/jigloo/

12. Annexos.

12.1. Configuració de OpenSSL.

El contingut de l'arxiu de configuració *openssl.cnf* es detalla a continuació:

```
#####################################################################
[ req ]
default_bits = 1024
default_keyfile = privkey.pem
distinguished_name = req_distinguished_name
attributes = req_attributes
x509_extensions = usr_cert         # The extentions to add

# Passwords for private keys if not present they will be prompted for
# input_password = secret
# output_password = secret

# This sets a mask for permitted string types. There are several options.
# default: PrintableString, T61String, BMPString.
# pkix : PrintableString, BMPString.
# utf8only: only UTF8Strings.
# nombstr : PrintableString, T61String (no BMPStrings or UTF8Strings).
# MASK:XXXX a literal mask value.
# WARNING: current versions of Netscape crash on BMPStrings or UTF8Strings
# so use this option with caution!
string_mask = nombstr

# req_extensions = v3_req # The extensions to add to a certificate request

[ req_distinguished_name ]
countryName = Country Name (2 letter code)
countryName_default = ES
countryName_min = 2
countryName_max = 2

stateOrProvinceName = State or Province Name (full name)
stateOrProvinceName_default = Tarragona

localityName = Locality Name (eg, city)
localityName_default = Sta.Barbara

0.organizationName = Organization Name (eg, company)
0.organizationName_default = UOC
```

```
organizationalUnitName = Organizational Unit Name (eg, section)
organizationalUnitName_default = PFC Seguretat

commonName = Common Name (eg, YOUR name)
commonName_max = 64

dnQualifier = D.N.I or N.S.S.
dnQualifier_default = 00000000-A

emailAddress = Email Address
emailAddress_max = 40

[ req_attributes ]
challengePassword = A challenge password
challengePassword_min = 4
challengePassword_max = 20

unstructuredName = An optional company name

[ usr_cert ]

# These extensions are added when 'ca' signs a request.
# This goes against PKIX guidelines but some CAs do it and some software
# requires this to avoid interpreting an end user certificate as a CA.

basicConstraints=CA:FALSE

# Here are some examples of the usage of nsCertType. If it is omitted
# the certificate can be used for anything *except* object signing.

# This is OK for an SSL server.
# nsCertType = server

# For an object signing certificate this would be used.
# nsCertType = objsign

# For normal client use this is typical
nsCertType = client, email

# Copy extensions

# and for everything including object signing:
# nsCertType = client, email, objsign

# This is typical in keyUsage for a client certificate.
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
```

```
# This will be displayed in Netscape’s comment listbox.
nsComment = "PFC Seguretat"

# PKIX recommendations harmless if included in all certificates.
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer:always

# This stuff is for subjectAltName and issuerAltname.
# Import the email address.
subjectAltName=email:copy

# Copy subject details
issuerAltName=issuer:copy

#nsCaRevocationUrl = http://www.domain.dom/ca-crl.pem
#nsBaseUrl
#nsRevocationUrl
#nsRenewalUrl
#nsCaPolicyUrl
#nsSslServerName
###############################################################################
```

12.2. Instal·lació de MySQL.

Per instal·lar MySQL en l'ordinador on s'executarà el servidor de dades, s'han de seguir els següents passos:

- Descarregar el fitxer ***mysql-5.0.67-win32.zip*** del servidor de MySQL, concretament s'ha d'anar a la següent adreça: *descàrrega de mySQL*, i es descomprimeix.

- Una vegada descomprimit l'arxiu, s'ha d'executar el fitxer setup.exe resultant i seguir el passos següents:

a) A la pantalla d'inici s'ha de prémer _Next > :

Figura 51: MySQL. Pantalla d'inici d'instal·lació.

b) A la següent pantalla es selecciona el tipus Typical i es prem el botó _Next > :

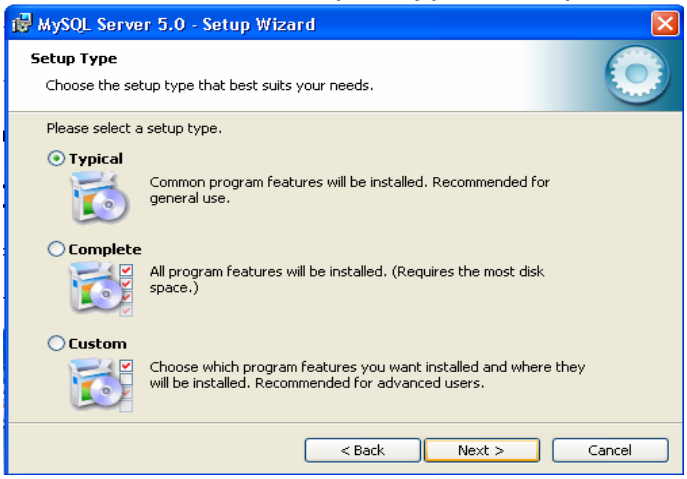

Figura 52: MySQL. Pantalla de selecció del tipus d'instal·lació.

c) A la següent pantalla s'ha de prémer _Install :

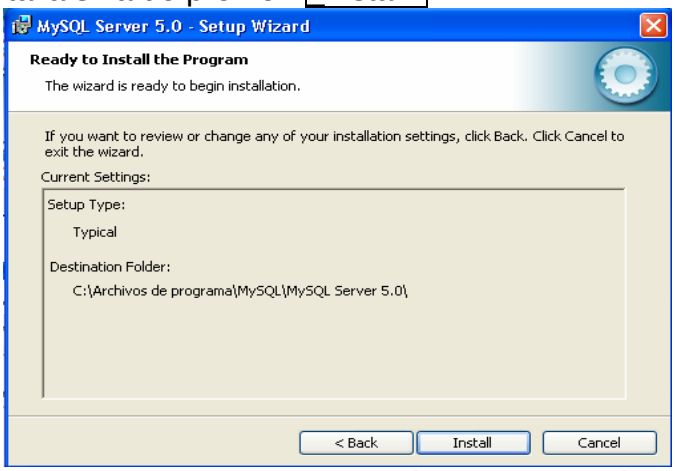

Figura 53: MySQL. Pantalla de resum del què i on s'instal·larà el programari.

d)

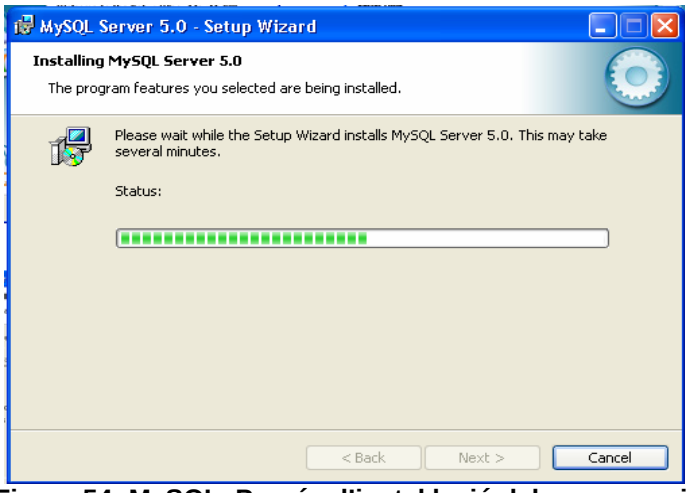

Figura 54: MySQL. Procés d'instal·lació del programari.

e) En la següent pantalla es necessari prémer el botó _Next > :

Figura 55: MySQL. Pantalla de presentació del programari instal·lat.

f) En la següent pantalla s'ha de seleccionar *configure the MySQL Server now* i prémer el botó _Finish_ :

Figura 56: MySQL. Pantalla de selecció de configuració.

g) En la següent pantalla es necessari prémer el botó _Next >_ :

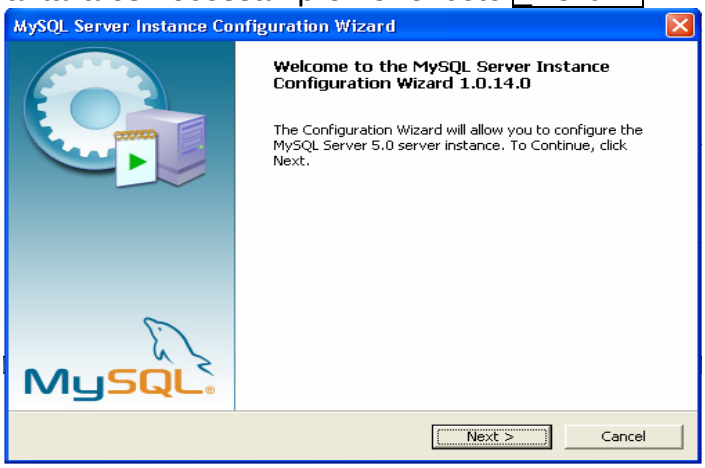

Figura 57: MySQL. Pantalla d'inici de configuració.

h) En la següent pantalla s'ha de seleccionar *Standard Configuration* i prémer el botó _Next >_ :

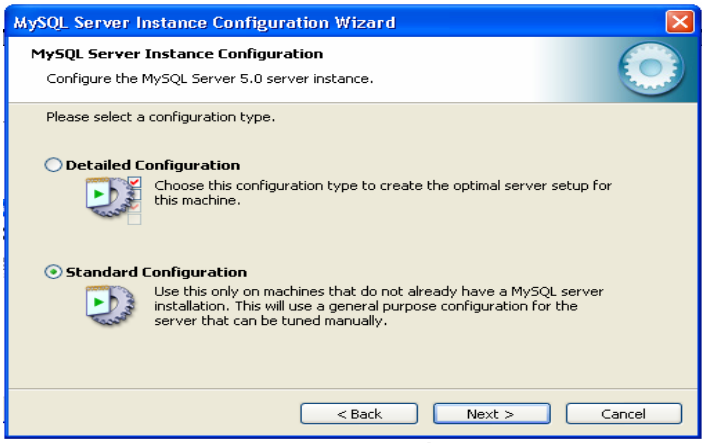

Figura 58: MySQL. Pantalla de selecció del tipus de configuració.

i) En la següent pantalla s'ha de seleccionar *Install As Windows Service*, el *Service Name* ha de ser **MySQL501**, seleccionar *Launch the MySQL Server Automatically* i prémer el botó _Next > :

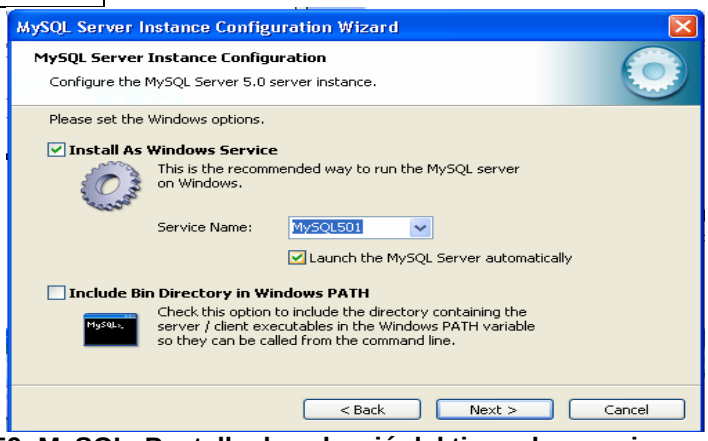

Figura 59: MySQL. Pantalla de selecció del tipus de servei a configurar.

j) La següent pantalla permet canviar la clau del root. S'ha de seleccionar *Modify Security Settings* i posar la nova clau del root. Per provar aquest projecte la clau és "**vgm**". Més endavant l'usuari i clau es podrà seleccionar de la interfície del gestor.

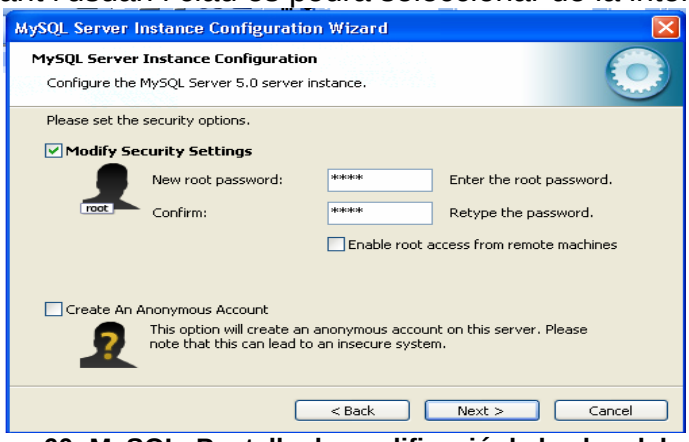

Figura 60: MySQL. Pantalla de modificació de la clau del root.

k)En la següent pantalla s'ha de prémer el botó _Execute :

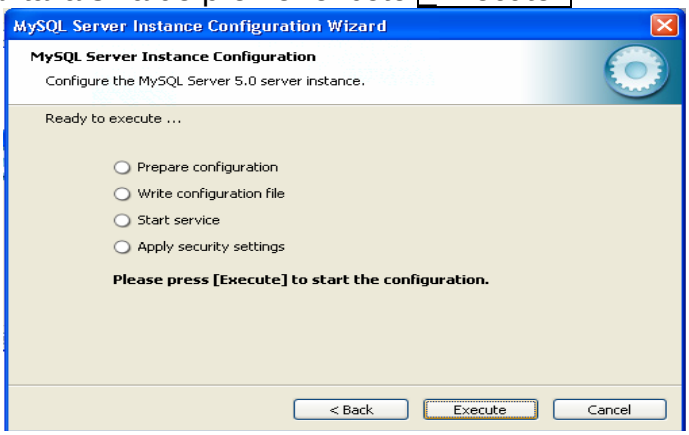

Figura 61: MySQL. Pantalla d'inici d'enregistrament de la configuració demanada.

l)Una vegada configurat el servidor i usuari, s'ha de prémer el botó _Finish:

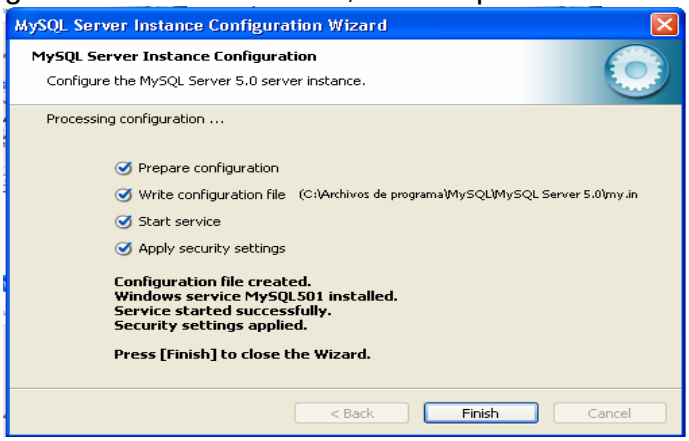

Figura 62: MySQL. Pantalla final de configuració realitzada.

12.3. Script de creació de la base de dades.

L'script utilitzat per a crear la base de dades de l'aplicació és el següent:

```
-- phpMyAdmin SQL Dump
-- version 2.8.0.3
-- http://www.phpmyadmin.net
--
-- Servidor: localhost
-- Tiempo de generaci n: 30-11-2008 a las 20:51:06
-- Versi n del servidor: 5.0.67
-- Versi n de PHP: 5.1.2
--
-- Base de datos: `databasepfc`
--
CREATE DATABASE `dataBasePFC` DEFAULT CHARACTER SET utf8 COLLATE utf8_bin;
USE `dataBasePFC`;

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `user`
--

CREATE TABLE `user` (
  `identifier` varchar(9) collate utf8_bin NOT NULL,
  `path_certificate` varchar(100) collate utf8_bin NOT NULL,
  `nUser` varchar(50) collate utf8_bin default NULL,
  `nGestor` varchar(50) collate utf8_bin default NULL,
  PRIMARY KEY  (`identifier`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;

-- --------------------------------------------------------


--
-- Estructura de tabla para la tabla `medical`
--

CREATE TABLE `medical` (
  `identifier` varchar(9) collate utf8_bin NOT NULL,
  `name` varchar(25) collate utf8_bin NOT NULL,
  `surname` varchar(50) collate utf8_bin NOT NULL,
  `medicalNumber` varchar(12) collate utf8_bin NOT NULL,
  `cipheredListPacients` blob NOT NULL,
  PRIMARY KEY  (`identifier`),
  KEY `FK_identifierM` (`identifier`),
  CONSTRAINT `FK_identifierM` FOREIGN KEY (`identifier`) REFERENCES
`user`(`identifier`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;



-- --------------------------------------------------------
```

```
--
-- Estructura de tabla para la tabla `pacient`
--

CREATE TABLE `pacient` (
  `identifier` varchar(9) collate utf8_bin NOT NULL,
  `name` varchar(25) collate utf8_bin NOT NULL,
  `surname` varchar(50) collate utf8_bin NOT NULL,
  `bloodGroup` varchar(5) collate utf8_bin NOT NULL,
  `sanitaryCardNumber` varchar(15) collate utf8_bin NOT NULL,
  PRIMARY KEY  (`identifier`),
  KEY `FK_identifierP` (`identifier`),
  CONSTRAINT `FK_identifierP` FOREIGN KEY (`identifier`) REFERENCES
`user`(`identifier`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;


-- --------------------------------------------------------
--
-- Estructura de tabla para la tabla `historial`
--

CREATE TABLE `historial` (
  `identifier` varchar(9) collate utf8_bin NOT NULL,
  `cipheredListMedicals` blob default NULL,
  `cipheredListDescriptors` blob default NULL,
  PRIMARY KEY  (`identifier`),
  KEY `FK_identifierH` (`identifier`),
  CONSTRAINT `FK_identifierH` FOREIGN KEY (`identifier`) REFERENCES
`pacient`(`identifier`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;

-- --------------------------------------------------------


--
-- Estructura de tabla para la tabla `visits`
--

CREATE TABLE `visits` (
  `idDescriptor` int(10) unsigned NOT NULL auto_increment,
  `date` varchar(50) collate utf8_bin  NOT NULL,
  `location` varchar(50) collate utf8_bin  NOT NULL,
  `time` varchar(10)  NOT NULL,
  `causeOfVisit` varchar(300) collate utf8_bin  NOT NULL,
  `personalAntecendents` varchar(500) collate utf8_bin default NULL,
  `familiarAntecendents` varchar(500) collate utf8_bin default NULL,
  `diagnosis` varchar(300) collate utf8_bin  NOT NULL,
  `treatment` varchar(300) collate utf8_bin  NOT NULL,
  `idMedical` varchar(9) collate utf8_bin  NOT NULL,
  `medicalSignature` blob,
  PRIMARY KEY  (`idDescriptor`),
  KEY `FK_idMedical`(`idMedical`),
  CONSTRAINT `FK_idMedical` FOREIGN KEY (`idMedical`) REFERENCES `medical`
(`identifier`)
) ENGINE=InnoDB AUTO_INCREMENT=100001 DEFAULT CHARSET=utf8
COLLATE=utf8_bin;
```

12.4. Arxius per lots de creació dels certificats.

Els arxius per lots utilitzats per a crear els certificats dels usuaris de l'aplicació són els següents:

12.4.1. Generació de les claus amb OpenSSL.

```
@echo off
echo Script per a generar les claus dels actors del PFC.
echo Les claus les desarà a la carpeta certificats creada al disc C:
echo.
echo ================================
echo =                                =
echo =      Generacio de les claus    =
echo =                                =
echo ================================
echo.
echo  -1r pas: creem la carpeta certificats.
echo.
echo.
md c:\certificats
echo.
echo  -2n pas: generem les claus de la CA.
echo.
echo.
C:\OpenSSL\bin\openssl genrsa -des3 -out C:\certificats\CA.key 2048
echo.
echo.
echo pitja qualsevol tecla per generar les claus del gestor.
pause>nul
echo.
echo -3r pas: generem les claus del gestor.
echo.
echo.
C:\OpenSSL\bin\openssl genrsa -des3 -out C:\certificats\Gestor.key 1024
echo.
echo.
echo pitja qualsevol tecla per generar les claus del metge.
pause>nul
echo.
echo -4rt pas: generem les claus dels metges.
echo.
echo.
C:\OpenSSL\bin\openssl genrsa -des3 -out C:\certificats\Metge1.key 1024
echo.
```

```
echo.
C:\OpenSSL\bin\openssl genrsa -des3 -out C:\certificats\Metge2.key 1024
echo.
echo.
C:\OpenSSL\bin\openssl genrsa -des3 -out C:\certificats\Metge3.key 1024
echo.
echo.
C:\OpenSSL\bin\openssl genrsa -des3 -out C:\certificats\Metge4.key 1024
echo.
echo.
echo pitja qualsevol tecla per generar les claus del pacient.
pause>nul
echo.
echo -5e pas: generem les claus dels pacients.
echo.
echo.
C:\OpenSSL\bin\openssl genrsa -des3 -out C:\certificats\Pacient1.key 1024
echo.
echo.
C:\OpenSSL\bin\openssl genrsa -des3 -out C:\certificats\Pacient2.key 1024
echo.
echo.
C:\OpenSSL\bin\openssl genrsa -des3 -out C:\certificats\Pacient3.key 1024
echo.
echo.
C:\OpenSSL\bin\openssl genrsa -des3 -out C:\certificats\Pacient4.key 1024
echo.
echo.
echo pitja qualsevol tecla per sortir.
pause>nul
```

12.4.2. Generació del certificat autosignant de la CA amb OpenSSL.

```
@echo off
echo Script per a generar un certificat autosignat per la CA.
echo El certificat el desarà a la carpeta certificats creada al disc C:
echo.
echo ================================
echo =                              =
echo =     Generacio certificat CA  =
echo =                              =
echo ================================
echo.
echo.
echo  -1r pas: generem el certificat autosignat de la CA.
```

```
echo.
echo.
C:\OpenSSL\bin\openssl req -new -sha1 -x509 -key C:\certificats\CA.key -out C:\certificats\CA.crt -days 360
echo.
echo.
echo pitja qualsevol tecla per verificar el certificat.
pause>nul
echo.
echo -2n pas: Una vegada generat, verifiquem que tot ha anat bé.
echo.
echo.
C:\OpenSSL\bin\openssl verify -CAfile C:\certificats\CA.crt C:\certificats\CA.crt
echo pitja qualsevol tecla per sortir.
pause>nul
```

12.4.3. Generació dels certificats del gestor i usuaris amb OpenSSL.

```
@echo off
echo Script per a generar els certificats dels actors del PFC.
echo Els certificats els desarà a la carpeta certificats creada al disc C:
echo.
echo ================================
echo =                                =
echo =   Generació dels certificats   =
echo =    dels actors del sistema     =
echo =                                =
echo ================================
echo.
echo.
echo  -1r pas: generem el certificat del Gestor.
echo.
echo        Creem una petició de certificat per al gestor:
echo.
C:\OpenSSL\bin\openssl req -new -sha1 -key C:\certificats\Gestor.key -out C:\certificats\Gestor.csr -config openssl.conf
echo.
echo.
echo        Verifiquem la petició:
echo.
C:\OpenSSL\bin\openssl req -in C:\certificats\Gestor.csr -verify -text -noout
echo.
echo.
echo        Emetem el certificat per al gestor:
echo.
```

```
C:\OpenSSL\bin\openssl x509 -req -in C:\certificats\Gestor.csr -days 180 -CA C:\certificats\CA.crt -CAkey C:\certificats\CA.key -CAcreateserial -extfile openssl.conf -extensions usr_cert -out C:\certificats\Gestor.crt
echo.
echo.
echo        Verifiquem el certificat del gestor:
echo.
C:\OpenSSL\bin\openssl verify -CAfile C:\certificats\CA.crt C:\certificats\Gestor.crt
echo.
echo.
echo pitja qualsevol tecla per generar el certificat del metge.
pause>nul
echo.
echo -2n pas: generem el certificat del Metge1.
echo.
echo        Creem una petició de certificat per al metge1:
echo.
C:\OpenSSL\bin\openssl req -new -sha1 -key C:\certificats\Metge1.key -out C:\certificats\Metge1.csr -config openssl.conf
echo.
echo.
echo        Verifiquem la petició:
echo.
C:\OpenSSL\bin\openssl req -in C:\certificats\Metge1.csr -verify -text -noout
echo.
echo.
echo        Emetem el certificat per al metge1:
echo.
C:\OpenSSL\bin\openssl x509 -req -in C:\certificats\Metge1.csr -days 180 -CA C:\certificats\CA.crt -CAkey C:\certificats\CA.key -CAcreateserial -extfile openssl.conf -extensions usr_cert -out C:\certificats\Metge1.crt
echo.
echo.
echo        Verifiquem el certificat del metge1:
echo.
C:\OpenSSL\bin\openssl verify -CAfile C:\certificats\CA.crt C:\certificats\Metge1.crt
echo.
echo.
echo -2n pas: generem el certificat del Metge2.
echo.
echo        Creem una petició de certificat per al metge2:
echo.
C:\OpenSSL\bin\openssl req -new -sha1 -key C:\certificats\Metge2.key -out C:\certificats\Metge2.csr -config openssl.conf
echo.
echo.
echo        Verifiquem la petició:
```

```
echo.
C:\OpenSSL\bin\openssl req -in C:\certificats\Metge2.csr -verify -text -noout
echo.
echo.
echo        Emetem el certificat per al metge2:
echo.
C:\OpenSSL\bin\openssl x509 -req -in C:\certificats\Metge2.csr -days 180 -CA C:\certificats\CA.crt -CAkey C:\certificats\CA.key -CAcreateserial -extfile openssl.conf -extensions usr_cert -out C:\certificats\Metge2.crt
echo.
echo.
echo        Verifiquem el certificat del metge2:
echo.
C:\OpenSSL\bin\openssl verify -CAfile C:\certificats\CA.crt C:\certificats\Metge2.crt
echo.
echo.
echo -2n pas: generem el certificat del Metge3.
echo.
echo        Creem una petició de certificat per al metge3:
echo.
C:\OpenSSL\bin\openssl req -new -sha1 -key C:\certificats\Metge3.key -out C:\certificats\Metge3.csr -config openssl.conf
echo.
echo.
echo        Verifiquem la petició:
echo.
C:\OpenSSL\bin\openssl req -in C:\certificats\Metge3.csr -verify -text -noout
echo.
echo.
echo        Emetem el certificat per al metge3:
echo.
C:\OpenSSL\bin\openssl x509 -req -in C:\certificats\Metge3.csr -days 180 -CA C:\certificats\CA.crt -CAkey C:\certificats\CA.key -CAcreateserial -extfile openssl.conf -extensions usr_cert -out C:\certificats\Metge3.crt
echo.
echo.
echo        Verifiquem el certificat del metge3:
echo.
C:\OpenSSL\bin\openssl verify -CAfile C:\certificats\CA.crt C:\certificats\Metge3.crt
echo.
echo.
echo -2n pas: generem el certificat del Metge4.
echo.
echo        Creem una petició de certificat per al metge4:
echo.
C:\OpenSSL\bin\openssl req -new -sha1 -key C:\certificats\Metge4.key -out C:\certificats\Metge4.csr -config openssl.conf
```

```
echo.
echo.
echo        Verifiquem la petició:
echo.
C:\OpenSSL\bin\openssl req -in C:\certificats\Metge4.csr -verify -text -noout
echo.
echo.
echo        Emetem el certificat per al metge4:
echo.
C:\OpenSSL\bin\openssl x509 -req -in C:\certificats\Metge4.csr -days 180 -CA
C:\certificats\CA.crt -CAkey C:\certificats\CA.key -CAcreateserial -extfile
openssl.conf -extensions usr_cert -out C:\certificats\Metge4.crt
echo.
echo.
echo        Verifiquem el certificat del metge4:
echo.
C:\OpenSSL\bin\openssl verify -CAfile C:\certificats\CA.crt C:\certificats\Metge4.crt
echo.
echo.
echo pitja qualsevol tecla per generar el certificat del pacient.
pause>nul
echo.
echo -3r pas: generem el certificat del Pacient1.
echo.
echo        Creem una petició de certificat per al pacient1:
echo.
C:\OpenSSL\bin\openssl req -new -sha1 -key C:\certificats\Pacient1.key -out
C:\certificats\Pacient1.csr -config openssl.conf
echo.
echo.
echo        Verifiquem la petició:
echo.
C:\OpenSSL\bin\openssl req -in C:\certificats\Pacient1.csr -verify -text -noout
echo.
echo.
echo        Emetem el certificat per al pacient1:
echo.
C:\OpenSSL\bin\openssl x509 -req -in C:\certificats\Pacient1.csr -days 180 -CA
C:\certificats\CA.crt -CAkey C:\certificats\CA.key -CAcreateserial -extfile
openssl.conf -extensions usr_cert -out C:\certificats\Pacient1.crt
echo.
echo.
echo        Verifiquem el certificat del pacient1:
echo.
C:\OpenSSL\bin\openssl verify -CAfile C:\certificats\CA.crt C:\certificats\Pacient1.crt
echo.
echo.
```

```
echo -3r pas: generem el certificat del Pacient2.
echo.
echo        Creem una petició de certificat per al pacient2:
echo.
C:\OpenSSL\bin\openssl req -new -sha1 -key C:\certificats\Pacient2.key -out C:\certificats\Pacient2.csr -config openssl.conf
echo.
echo.
echo        Verifiquem la petició:
echo.
C:\OpenSSL\bin\openssl req -in C:\certificats\Pacient2.csr -verify -text -noout
echo.
echo.
echo        Emetem el certificat per al pacient2:
echo.
C:\OpenSSL\bin\openssl x509 -req -in C:\certificats\Pacient2.csr -days 180 -CA C:\certificats\CA.crt -CAkey C:\certificats\CA.key -CAcreateserial -extfile openssl.conf  -extensions usr_cert  -out C:\certificats\Pacient2.crt
echo.
echo.
echo        Verifiquem el certificat del pacient2:
echo.
C:\OpenSSL\bin\openssl verify -CAfile  C:\certificats\CA.crt C:\certificats\Pacient2.crt
echo.
echo.
echo -3r pas: generem el certificat del Pacient3.
echo.
echo        Creem una petició de certificat per al pacient3:
echo.
C:\OpenSSL\bin\openssl req -new -sha1 -key C:\certificats\Pacient3.key -out C:\certificats\Pacient3.csr -config openssl.conf
echo.
echo.
echo        Verifiquem la petició:
echo.
C:\OpenSSL\bin\openssl req -in C:\certificats\Pacient3.csr -verify -text -noout
echo.
echo.
echo        Emetem el certificat per al pacient3:
echo.
C:\OpenSSL\bin\openssl x509 -req -in C:\certificats\Pacient3.csr -days 180 -CA C:\certificats\CA.crt -CAkey C:\certificats\CA.key -CAcreateserial -extfile openssl.conf  -extensions usr_cert  -out C:\certificats\Pacient3.crt
echo.
echo.
echo        Verifiquem el certificat del pacient:
echo.
```

```
C:\OpenSSL\bin\openssl verify -CAfile  C:\certificats\CA.crt C:\certificats\Pacient3.crt
echo.
echo.
echo -3r pas: generem el certificat del Pacient4.
echo.
echo        Creem una petició de certificat per al pacient4:
echo.
C:\OpenSSL\bin\openssl req -new -sha1 -key C:\certificats\Pacient4.key -out C:\certificats\Pacient4.csr -config openssl.conf
echo.
echo.
echo        Verifiquem la petició:
echo.
C:\OpenSSL\bin\openssl req -in C:\certificats\Pacient4.csr -verify -text -noout
echo.
echo.
echo        Emetem el certificat per al pacient4:
echo.
C:\OpenSSL\bin\openssl x509 -req -in C:\certificats\Pacient4.csr -days 180 -CA C:\certificats\CA.crt -CAkey C:\certificats\CA.key -CAcreateserial -extfile openssl.conf  -extensions usr_cert  -out C:\certificats\Pacient4.crt
echo.
echo.
echo        Verifiquem el certificat del pacient4:
echo.
C:\OpenSSL\bin\openssl verify -CAfile  C:\certificats\CA.crt C:\certificats\Pacient4.crt
echo.
echo.
echo pitja qualsevol tecla per sortir.
pause>nul
```

12.4.4. Generació dels PKCS12 del gestor i usuaris amb OpenSSL.

```
@echo off
echo Script per a generar els PKCS12 dels actors del PFC.
echo els arxius els desarà a la carpeta certificats creada al disc C:
echo.
echo ================================
echo =                              =
echo =     Generació dels PKCS#12   =
echo =                              =
echo ================================
echo.
echo  -1r pas: creem el PKCS#12 del Gestor.
echo.
```

```
echo.

C:\OpenSSL\bin\openssl pkcs12 -in C:\certificats\Gestor.crt -inkey C:\certificats\Gestor.key -name Gestor -chain -CAfile C:\certificats\CA.crt -export -out C:\certificats\Gestor.p12

echo.
echo.
echo pitja qualsevol tecla per generar-ho per als Metges.
pause>nul
echo  -2n pas: creem el PKCS#12 del Metge1.
echo.
echo.
C:\OpenSSL\bin\openssl pkcs12 -in C:\certificats\Metge1.crt -inkey C:\certificats\Metge1.key -name Metge1 -chain -CAfile C:\certificats\CA.crt -export -out C:\certificats\Metge1.p12

echo.
echo.
echo  -2n pas: creem el PKCS#12 del Metge2.
echo.
echo.
C:\OpenSSL\bin\openssl pkcs12 -in C:\certificats\Metge2.crt -inkey C:\certificats\Metge2.key -name Metge2 -chain -CAfile C:\certificats\CA.crt -export -out C:\certificats\Metge2.p12

echo.
echo.
echo  -2n pas: creem el PKCS#12 del Metge3.
echo.
echo.
C:\OpenSSL\bin\openssl pkcs12 -in C:\certificats\Metge3.crt -inkey C:\certificats\Metge3.key -name Metge3 -chain -CAfile C:\certificats\CA.crt -export -out C:\certificats\Metge3.p12

echo.
echo.
echo  -2n pas: creem el PKCS#12 del Metge4.
echo.
echo.
C:\OpenSSL\bin\openssl pkcs12 -in C:\certificats\Metge4.crt -inkey C:\certificats\Metge4.key -name Metge4 -chain -CAfile C:\certificats\CA.crt -export -out C:\certificats\Metge4.p12

echo.
echo.
echo pitja qualsevol tecla per generar-ho per als pacients.
```

```
pause>nul
echo -3r pas: creem el PKCS#12 del Pacient1.
echo.
echo.
C:\OpenSSL\bin\openssl pkcs12 -in C:\certificats\Pacient1.crt -inkey C:\certificats\Pacient1.key -name Pacient1 -chain -CAfile C:\certificats\CA.crt -export -out C:\certificats\Pacient1.p12

echo.
echo.
echo -3r pas: creem el PKCS#12 del Pacient2.
echo.
echo.
C:\OpenSSL\bin\openssl pkcs12 -in C:\certificats\Pacient2.crt -inkey C:\certificats\Pacient2.key -name Pacient2 -chain -CAfile C:\certificats\CA.crt -export -out C:\certificats\Pacient2.p12

echo.
echo.
echo -3r pas: creem el PKCS#12 del Pacient3.
echo.
echo.
C:\OpenSSL\bin\openssl pkcs12 -in C:\certificats\Pacient3.crt -inkey C:\certificats\Pacient3.key -name Pacient3 -chain -CAfile C:\certificats\CA.crt -export -out C:\certificats\Pacient3.p12

echo.
echo.
echo -3r pas: creem el PKCS#12 del Pacient4.
echo.
echo.
C:\OpenSSL\bin\openssl pkcs12 -in C:\certificats\Pacient4.crt -inkey C:\certificats\Pacient4.key -name Pacient4 -chain -CAfile C:\certificats\CA.crt -export -out C:\certificats\Pacient4.p12

echo.
echo.
echo pitja qualsevol tecla per sortir.
pause>nul
```

12.5. Instal·lació de l'aplicació.

Per facilitar la instal·lació de l'aplicació a les màquines dels clients i el gestor, es facilita un instal·lador per a cada part, és a dir, un instal·lador que s'encarrega d'emmagatzemar al disc dur del client, pacient o metge, tots els arxius necessaris per al correcte funcionament de l'aplicació i un altre que s'encarrega d'emmagatzemar al disc dur del gestor tots els arxius necessaris per a posar en marxa el servidor.

Atesa la implementació del projecte, només és necessari de disposar una màquina virtual java amb un JRE de SUN. Totes les llibreries afegides (iaik, jdom,etc.), arxius de polítiques de seguretat, certificats i arxius p12 dels usuaris han estat incorporats dins el programari. Les proves de compatibilitat s'han efectuat amb les versions 1.5 i 1.6. amb qualsevol altra versió no es pot assegurar el correcte funcionament, no obstant s'ha procurat evitar l'ús de mètodes "*deprecated*" per assegurar una compatibilitat futura amb pròximes versions de la JRE durant el cicle de vida lògic d'una aplicació estàndard. No obstant en la part del gestor, tal i com s'indica en la documentació, és necessari la instal·lació d'un gestor de base de dades i la creació de les taules del sistema. a més a més de disposar d'un compte amb privilegis totals sobre MySQL per a efectuar les crides i emmagatzematge de la informació.

A continuació és detalla el procés d'instal·lació de cada part.

12.5.1. Instal·lació del gestor.

Per a instal·lar l'aplicació gestora al disc dur, s'ha d'executar l'arxiu *setupGestor.bat* facilitat al directori "*instal·lar aplicació*".

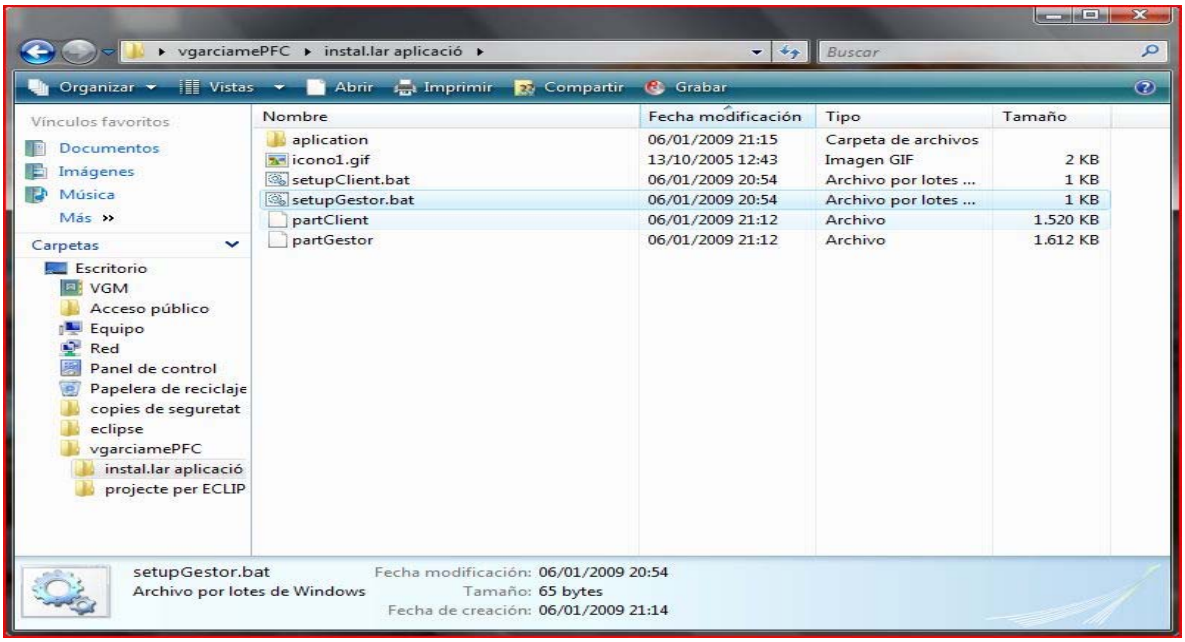

Figura 63: Contingut del directori *instal·lar aplicació*.

Una vegada executat, l'aplicació ens mostra el següent missatge:

Figura 64: Missatge de presentació de l'instal·lador.

En el cas que es vulgui instal·lar, s'ha de prémer el botó "Sí" i apareix el següent missatge:

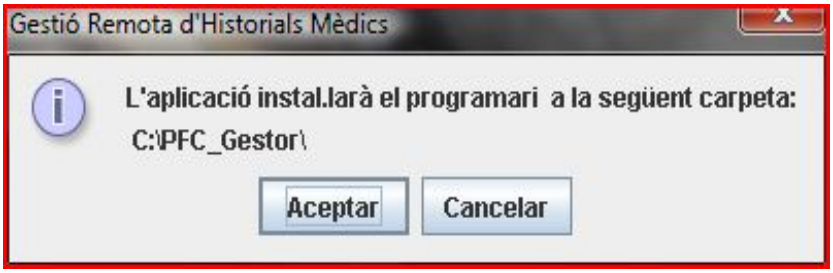

Figura 65: Missatge d'ubicació de l'aplicació.

Atesa la particularitat de que l'aplicació té accessos directes i per facilitar que aquestos apuntin correctament als arxius d'execució de l'aplicació, es fa necessari que el programari s'instal·li en un lloc predefinit. Per defecte doncs l'aplicació s'instal·la al directori ***C:\PFC_Gestor***.

Si el programari s'ha instal·lat correctament, apareix el següent missatge:

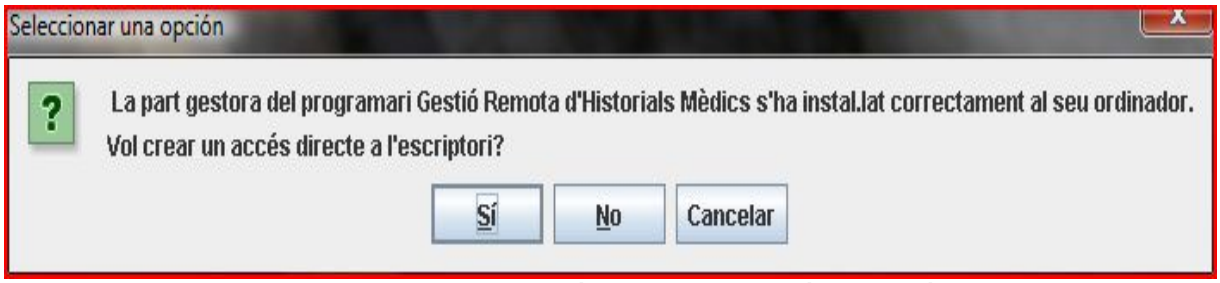

Figura 66: Missatge d'instal·lació correcta i creació de l'accés directe.

Una vegada instal·lat, l'instal·lador demana si es vol crear un accés directe a l'escriptori. En el cas que es vulgui s'ha de prémer "Sí ", en qualsevol altre cas, prémer "No"o "Cancel·lar".

La instal·lació del programari s'ha efectuat amb un Windows XP i amb un Vista. En el primer cas ha estat un èxit però amb Vista, la creació dels accessos directes a l'escriptori dóna errades a causa del sistema d'arxius que té. En el cas que no es puguin crear apareix el següent missatge:

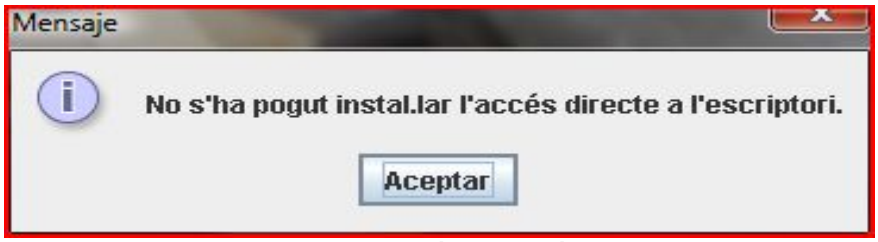

Figura 67: Missatge de creació de l'accés directe incorrecte.

En el cas que no es puguin crear els accessos directes a l'escriptori, es pot copiar directament aquell que es troba al directori arrel de l'aplicació:

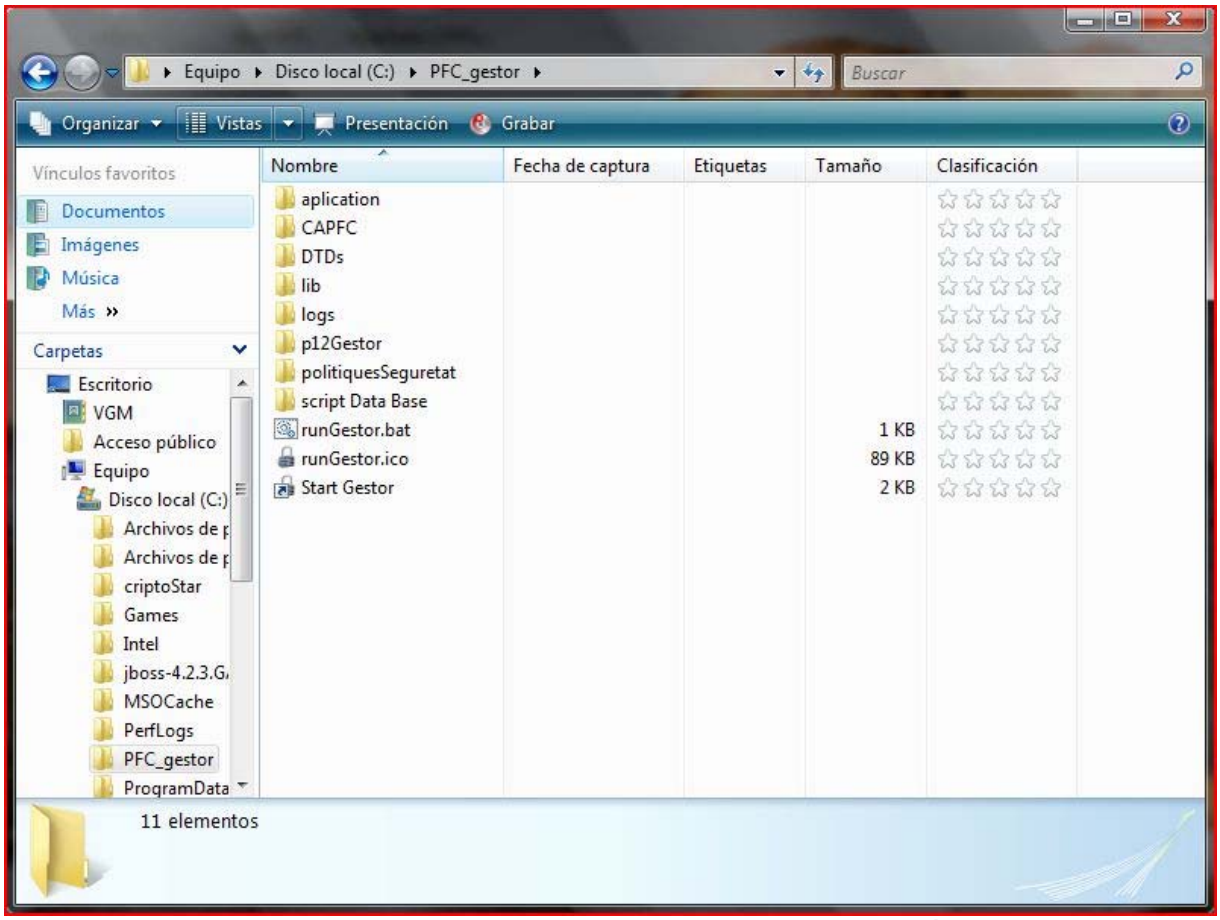

Figura 68: Contingut del directori on s'ha instal·lat l'aplicació gestora.

Finalment es pot veure tots els arxius que s'han copiat al disc mitjançant la consola d'instal·lació:

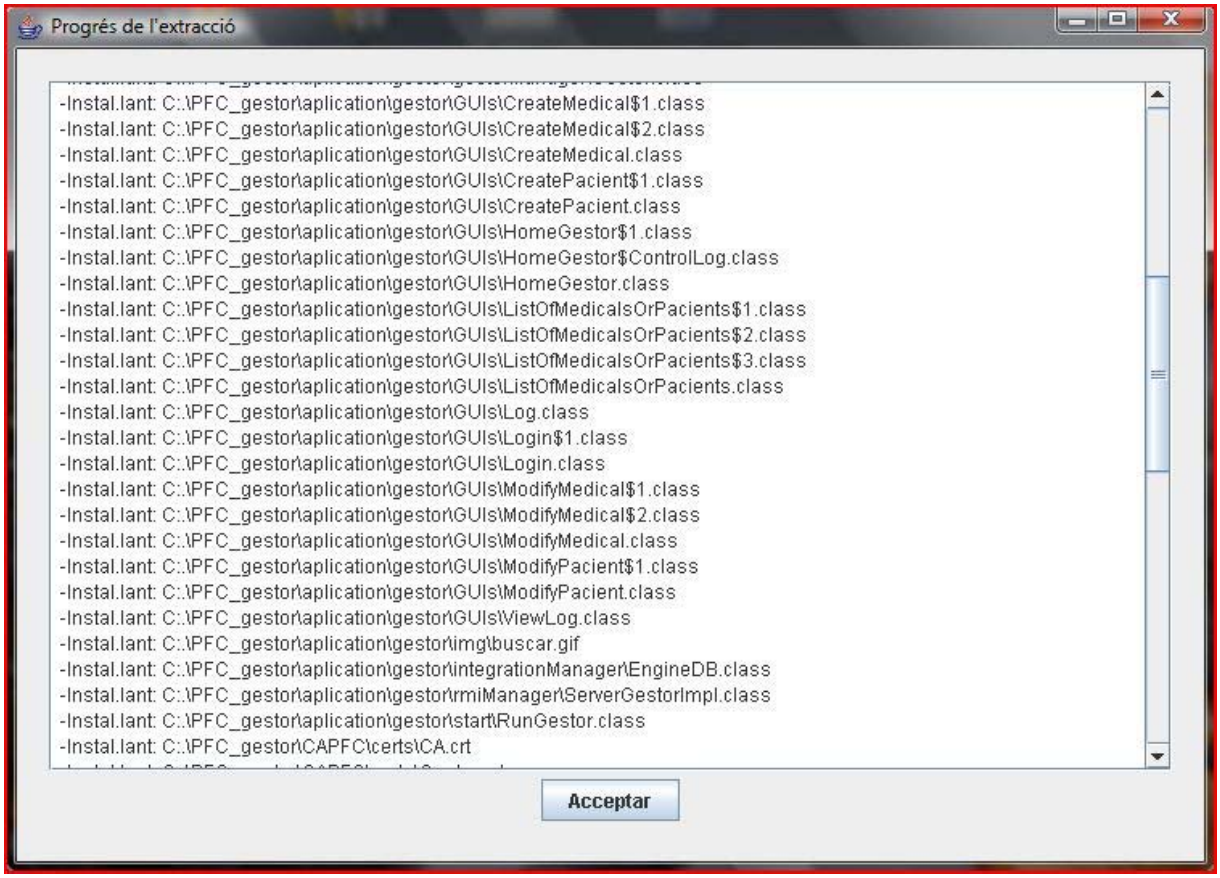

Figura 69: Consola del procés d'instal·lació del programari.

Una vegada instal·lat el programari gestor, s'ha de crear la base de dades utilitzant l'script que es facilita al directori arrel de l'aplicació, dins la carpeta *script Data Base*.

12.5.2. Instal·lació del client.

El procés d'instal·lació de la part client és exactament el mateix que s'ha indicat per al gestor, l'única diferència és que s'ha d'executar l'arxiu *setupClient.bat*.