SGHMS:
Sistema de gestió de historials mèdics segurs.

Jordi Casadevall i Barbacil
Enginyeria en informàtica

Jordi Castellà Roca
Consultor

Dedicatòria i agraïments

Aquest projecte culmina la fi dels meus estudis universitaris d'enginyeria en informàtica. La realització d'aquests estudis per part meva ha estat molt més llarga del normal degut a la meva implicació professional en diferents empreses abans de la finalització d'aquests estudis. Aquest fet ha provocat que durant una època no hagi dedicat el temps necessari per a tirar endavant aquests estudis. Finalment, el sistema de realització d'estudis de la UOC m'ha permès finalitzar els estudis en funció del temps disponible cada semestre, distribuït en els dies en que m'ha estat possible dedicar més temps a l'estudi de les matèries i la realització de les pràctiques. Paradoxalment, el fet d'haver allargat el temps de realització d'aquests estudis, també m'ha permès estar al dia, estudiar i conèixer i rebre formació de noves tecnologies informàtiques i computacionals mitjançant la universitat.

Tot i així, un projecte final de carrera requereix un compromís i voluntat superior al de l'estudi i realització de pràctiques de les assignatures de la carrera universitària. Reconec que la concentració durant hores seguides per poder resoldre a contrarellotge alguns problemes plantejats pel desenvolupament del projecte, m'ha provocat en vàries ocasions dificultats per distreure'm o concentrar-me en altres temes quan no és l'hora de treballar en el projecte.

Per tot això, voldria agrair i dedicar especialment aquest projecte especialment a totes les persones que m'envolten amb les quals em veig a diari, i han patit en part les conseqüències de la dedicació requerida per poder dur a terme el desenvolupament d'aquest projecte. Vosaltres sabeu qui sou.

Voldria recordar també especialment els qui em van donar la oportunitat de començar aquests estudis quan era molt més jove. Entenc que durant tants anys no era fàcil entendre que la meva intenció era finalitzar els estudis de seguida que em fos possible, però estic segur que ara estareu tan satisfets com jo de que finalment s'hagi acabat aquesta part del trajecte que estic decidit a continuar realitzant altres estudis. Com moltíssimes altres coses, us ho agrairé sempre.

Índex de continguts

Índex de figures

Capítol 1:

Capítol 3:

Capítol 4:

Capítol 5:

Capítol 6:

Capítol 7:

Capítol 8:

Capítol 1: Introducció

1.1. Justificació del PFC i context en el qual es desenvolupa: punt de partida i aportació del PFC.

Les xarxes de comunicació ens permeten realitzar algunes activitats amb independència del moment i del lloc on ens trobem. La Universitat Oberta de Catalunya és un exemple ben vàlid.

En aquest projecte ens centrarem en una d'aquestes activitats, la gestió d'historials mèdics per part de metges i pacients. Les dades de l'historial poden ajudar al metge a prendre decisions correctes en la diagnosi i tractament que ha de rebre el pacient.

L'historial mèdic d'un pacient és una informació de gran valor, i per tant s'ha de protegir. Cal garantir que només serà modificada pel personal qualificat i per tant autoritzat. Un altre punt important és la confidencialitat. Les dades mèdiques d'una persona són altament confidencials, i només poden ser accedides pel pacient o per personal mèdic.

1.2. Objectius del PFC.

Els objectius d'aquest projecte són:

- Mostrar quins són els estàndards en l'enviament d'informació de forma segura.
- Implementar un cas pràctic on s'utilitzen aquests estàndards. Aquest cas pràctic consisteix en un sistema de gestió d'historials mèdics. Aquest sistema estarà format per 3 components de programari:
    - Aplicació metge: Permet que un metge pugui consultar i modificar l'historial d'un pacient de forma segura.
    - Aplicació pacient: El pacient utilitza aquest aplicació per a consultar les dades del seu historial.
    - Gestor central: El gestor central és qui té tots els historials i en controla la seva gestió.

1.3. Enfocament i mètode seguit.

El projecte es divideix en diferents fases per tal de seguir una implementació incremental. Cada fase s'implementarà, se'n faran proves de test unitari i s'integrarà amb la fase anterior. D'aquesta manera s'aniran incorporant diferents tecnologies i funcionalitats gradualment, partint de que la feina feta fins al moment és correcta.

Les 7 fases amb les quals es repartirà el projecte, són:

1.- Instal·lació del programari bàsic i creació de la PKI

2.- Esquema criptogràfic i disseny de l'aplicació.
3.- Representació de les dades (XML)
4.- Comunicació entre components remots (RMI)
5.- Gestió i emmagatzematge de la informació (Base de dades)
6.- Interfície gràfica d'usuari dels aplicatius metge i pacient
7.- Documentació

1.4. Planificació del projecte.

La primera part del projecte, ha adquirit més pes del previst inicialment, ja que en aquesta part, s'ha dissenyat i implementat l'arquitectura de l'aplicació, deixant per més endavant només com ampliacions, la part XML (representació de dades), RMI (comunicació entre components), base de dades i interfície d'usuari.

Instal·lar IAIK PKI	openssl				
Implementació protocols	objectiu criptografia arquitectura del sistema	disseny	Implementació	test	documentació
Implementació protocols	objectiu XML	disseny	Implementació	test	documentació
Servidor RMI Client base RMI	objectiu RMI	disseny	Implementació	test	documentació
BD servidor Registre jugador	objectiu model	disseny	Implementació	test	documentació
Vista client	objectiu aplicació client	disseny	Implementació	test	documentació
Vista gestor	objectiu aplicació gestor	disseny	Implementació	test	documentació
Documentació	objectiu conclusions			test	documentació

Setmana	dilluns	dimarts	dimecres	dijous	divendres	dissabte	diumenge	
1			20	21	22	23	24	setembre
2	25	26	27	28	28	30	1	octubre
3	2	3	4	5	6	7	8	
4	9	10	11	12	13	14	15	
5	16	17	18	19	20	21	22	
6	23	24	25	26	27	28	29	
7	30	31	1	2	3	4	5	novembre
8	6	7	8	9	10	11	12	
9	13	14	15	16	17	18	19	
10	20	21	22	23	24	25	26	
11	27	28	29	30	1	2	3	desembre
12	4	5	6	7	8	9	10	
13	11	12	13	14	15	16	17	
14	18	19	20	21	22	23	24	
15	25	26	27	28	29	30	31	
16	1	2	3	4	5	6	7	gener
17	8							

	Dia festiu
	Lliurament PFC

1.5. Productes obtinguts.

- **Programari pacient.** Producte desenvolupat en Java, que disposa d'una interfície que permet als pacients consultar els seus historials mèdics.

- **Programari metge.** Producte desenvolupat en Java, que disposa d'una interfície que permet als metges consultar la llista de pacients que tenen assignats, consultar i ampliar els historials.

- **Programari gestor.** Producte desenvolupat en Java que permet gestionar el programari del servidor d'historials. En aquest projecte aquest programari només podrà engegar i parar el gestor. En un cas real, s'hauria d'incloure òbviament la possibilitat de gestionar altes, baixes i modificacions de dades de metges i pacients.

1.6. Breu descripció dels altres capítols de la memòria.

- **Capítol 2: Infraestructura de clau pública.** En aquest capítol, s'introdueixen els conceptes de signatura digital, criptografia de clau pública, i la infraestructura necessària per gestionar la criptografia de clau pública.

- **Capítol 3: Esquema criptogràfic.** Descripció i pseudocodi de l'esquema criptogràfic per cada cas d'ús.

- **Capítol 4: SGHMS: Sistema de gestió d'historials mèdics segurs.** Anàlisi de les necessitats del sistema, enumeració i descripció dels actors, guió, arquitectura del sistema i disseny en UML, de l'aplicació sense tenir en compte les funcionalitats i tecnologies descrites en els següents capítols.

- **Capítol 5: Representació de les dades (XML).** Descripció dels documents XML utilitzats per a les transferències de dades durant l'execució dels protocols criptogràfics. Ampliació del disseny i l'arquitectura del sistema en UML, tenint en compte les noves funcionalitats i tecnologies d'aquest capítol.

- **Capítol 6: Comunicació entre components remots (RMI).** Descripció de la comunicació entre els diferents components remots del sistema mitjançant la tecnologia RMI. Ampliació del disseny i l'arquitectura del sistema en UML, tenint en compte les noves funcionalitats i tecnologies d'aquest capítol.

- **Capítol 7: Gestió i emmagatzematge de la informació (Base de dades).** Model ER que definirà la base de dades. Explicació de les decisions de disseny i implementació de la base de dades en MySQL i en Java. Ampliació del disseny i l'arquitectura del sistema en UML, tenint en compte les noves funcionalitats i tecnologies d'aquest capítol.

- **Capítol 8: Interfície gràfica d'usuari.** Descriu com s'ha dissenyat la interfície dels usuaris (metges i pacients) per accedir als historials, i quin és el seu funcionament. Ampliació del disseny i l'arquitectura del sistema en UML, tenint en compte les noves funcionalitats i tecnologies d'aquest capítol.

- **Capítol 9: Conclusions.** Compliment al llarg del desenvolupament del projecte dels objectius proposats.

Capítol 2: Infraestructura de clau pública.

2.1. Introducció.

El xifratge és el procés de transformar els continguts d'un missatge en clar fent ús d'una clau secreta per tal que el missatge no pugui ser llegit, i d'aquesta manera satisfer la propietat de confidencialitat en la comunicació entre dues parts. Per poder llegir un missatge xifrat l'haurem de desxifrar per tal de transformar-lo en un missatge en clar. El xifratge i desxifratge de missatges són els fonaments sobre els quals estan construïts els sistemes segurs d'intercanvi de informació.

En un sistema segur d'intercanvi de informació hem d'assegurar que:

- Les tècniques de xifratge i claus secretes són suficientment robustes per tal que les persones no autoritzades no puguin desxifrar la informació enviada.
- Les claus només han de ser accessibles per les persones que autoritzades.

2.2. Criptografia de clau pública.

Les tècniques de xifratge s'han usat des dels inicis del llenguatge escrit. Tradicionalment (fa uns 30 anys), la clau secreta usada pel xifratge d'una informació era la mateixa per al desxifratge d'aquesta informació. Aquesta tècnica es coneix com a *clau simètrica* o criptografia de clau secreta. És una tecnologia suficientment robusta, de manera que és computacionalment intractable el problema de desxifrar sense conèixer la clau secreta. El problema que presenta el sistema de *clau simètrica* és la distribució de les claus, i la certesa de que les claus no caiguin en mans de persones no autoritzades. El problema de generar, distribuir, emmagatzemar de manera segura les claus secretes creix en progressió geomètrica respecte l'augment de l'ús del sistema d'intercanvi segur de informació.

Als anys 70, els experts en criptografia van introduir el concepte de *clau asimètrica* o criptografia de clau pública. Aquesta tècnica fa ús de dos claus matemàticament encadenades, vinculades entre sí. Una de les claus només s'utilitza per a xifrar els missatges, i l'altra clau per desxifrar-los. La clau que s'utilitza per a xifrar els missatges (clau pública) es pot distribuir lliurement, o es pot publicar en un directori. La clau utilitzada per desxifrar els missatges (clau privada), es guarda de forma segura donat que és una peça d'informació molt sensible.

2.3. Signatures digitals.

Les signatures digitals són signatures electròniques encadenades a la informació que es vol signar de manera que es pugui identificar inequívocament qui és el creador del missatge.

Per a crear una signatura digital, l'emissor que vol signar un missatge crea un *hash* del missatge i el xifra amb la seva clau privada. El resultat d'aquesta operació és la signatura del missatge i aquesta s'adjunta al missatge que es vol transmetre.

La clau privada, tal com s'ha explicat en l'apartat anterior, té una clau pública corresponent que el destinatari podrà utilitzar per a verificar la signatura. El destinatari utilitza la mateixa funció de *hash* per crear el *hash* del missatge. Llavors desxifra la signatura que ha rebut de l'emissor i ho compara amb el *hash* obtingut.

Una entitat en la qual podem confiar, assigna el parell de claus pública i privada a una persona particular.

Els següents factors són la base per el sistema de signatures digitals:

- Software de seguretat, que suporti les funcionalitats de la signatura digital.
- Infraestructura de seguretat, que suporti l'intercanvi de claus.
- Funcions de *hash* i algoritmes de clau pública.

2.4. Certificats.

Un certificat digital és una empremta digital virtual capaç d'autenticar absolutament la identitat d'una persona. El certificat en sí és simplement la informació de l'usuari més la signatura digital de la informació digital. Una entitat tercera part de confiança amb la qual confia una comunitat d'usuaris dels certificats (CA, autoritat de certificació) , hi adjunta la signatura digital.

2.5. Polítiques.

Una política de certificat és un conjunt de regles que indiquen l'ús del certificat.

2.6. Gestió de claus (*PKI*).

L'ús d'infraestructures de clau pública (*PKI*), permet l'intercanvi segur de signatures digitals, documents xifrats, autenticació i autorització, i altres funcions en xarxes obertes on hi han moltes entitats participants.

2.7. Autoritat de certificació (*CA*).

L'autoritat de certificació (*CA*) és l'entitat responsable d'emetre i administrar els certificats digitals. La *CA* actua com agent de confiança de la *PKI*.

Una *CA* porta a terme les següents funcions principals:

- Subministra les parelles de claus als usuaris, tot i que aquests també es poden generar el seu parell de claus.
- Certifica les claus públiques dels usuaris.
- Publica els certificats dels usuaris.
- Emet llistes de revocació de certificats (*CRL*).

La comunitat d'usuaris ha de confiar en la *CA* per la distribució, revocació, i gestió de claus i certificats. Així com els usuaris confien en la *CA* i els seus processos de negoci, també poden confiar en els certificats que emet.

La signatura de la *CA* en el certificat garanteix que qualsevol modificació en el contingut serà detectada. En tant que els certificats es fan públics i els usuaris n'extreuen la clau pública, poden estar segurs que es compleixen els següents requisits:

- Pertany a l'entitat o persona indicada en el certificat.
- Es pot utilitzar amb garanties de seguretat de la manera en que la *CA* ho certifica.

Els usuaris necessiten ser capaços de determinar el grau de seguretat o confiança que poden tenir en l'autenticitat i integritat de les claus públiques extretes dels certificats emesos per la *CA*.

La *CA* té les següents responsabilitats:

- Emetre el certificat basat en una clau pública. Idealment una entitat de confiança genera el parell de claus en una *smart card* o altre dispositiu segur equivalent.
- Garantir la unicitat del parell de claus i emetre un certificat a un usuari.
- Gestionar els certificats publicats.
- Ser part de la certificació coordinada amb altres *CA*.

2.8. Autoritat de registre (*RA*).

L'autoritat de registre (*RA*) és responsable d'emmagatzemar i verificar la informació que la *CA* necessita. Concretament, la *RA* ha de comprovar la identitat de l'usuari per tal que la *CA* li pugui assignar un certificat. La verificació de la identitat d'un usuari es fa per exemple comprovant el seu document nacional d'identitat.

La RA té dos funcions principals:

- Verificar la identitat de qui demana un certificat.
- Lliurar el certificat.

2.9. Servei de directori.

El servei de directori porta a terme principalment dos funcions:

- Publicar certificats
- Publicar llistes de revocació de certificats (*CRL*) o fer que els certificats estiguin disponibles en línia a través del protocol d'estat de certificats en línia (*OCSP*).

2.10. Aplicació de la infraestructura de clau pública en aquest projecte.

Les tecnologies esmentades anteriorment s'apliquen en aquest projecte per a fer possibles les propietats de seguretat: confidencialitat, autenticitat, integritat de dades i no repudi entre els usuaris pacients i metges, i el gestor de historials. Més endavant, en el capítol d'esquema criptogràfic, es tornarà parlar i es descriuran aquestes propietats de seguretat que al mateix temps són requisits per a l'aplicació sistema de gestió d'historials mèdics segurs.

3. SGHMS: Sistema de gestió d'historials mèdics segurs.

3.1. Introducció

En aquest capítol es descriuen les necessitats del sistema, per tant s'identifiquen els actors del sistema, les accions o serveis que realitzen, i la informació gestionada pel sistema. Cal recalcar que els serveis i accions escollits per aquest projecte són molt bàsics, i el desenvolupament està centrat principalment en els aspectes de seguretat, que són el principal objecte d'estudi en aquest projecte.

3.2. Actors

Aquest sistema serà usat bàsicament per metges, pacients i per un gestor encarregat de gestionar el sistema. Inicialment doncs partirem del següents actors:

- Metges
- Pacients
- Gestor

En una possible ampliació que no serà contemplada en aquest projecte, també podríem pensar en els actors infermers/es i assistents/es.

3.3. Accions i serveis

Els actors descrits en el punt anterior, utilitzaran el sistema per tal d'obtenir informació i modificar-la. Les possibles accions necessàries seran:

- **Registre d'usuaris.** El gestor donarà d'alta als pacients o als metges en el sistema. En aquest projecte no s'implementen aquestes funcionalitats per centrar-nos més en altres funcionalitats que es sol·licitaran remotament , seran més interessants des del punt de vista de la seguretat, i treballarem amb metges i pacients simulats creats prèviament a la base de dades.

- **Autenticació dels usuaris.** És molt important que cada usuari metge o pacient s'autentiqui de forma segura, i a més es pugui validar quin tipus de usuari és, pacient, metge o gestor.

- **Consulta d'un historial.** Els pacients utilitzaran el sistema per a consultar el seu historial, i els metges podran consultar els historials dels seus pacients. Podríem pensar en una ampliació no contemplada en aquest projecte que consistiria en que un pacient pugui consultar quan té una data de visita amb un metge o una analítica.

- **Inserció de dades a un historial.** Els metges podran afegir noves dades als historials dels seus pacients. En canvi però, per raons de seguretat no les podran modificar ni eliminar. En aquest cas, serà necessari entrar una nova visita, fent menció a les modificacions o aclariments respecte per exemple al darrer.

- **Consulta dels pacients assignats a un metge.** Els metges podran consultar quina és la llista de pacients que tenen assignats. A partir d'aquesta llista podran escollir-ne un per tal de consultar o afegir dades a l'historial.

3.4. Gestió de la informació

L'historial mèdic d'un pacient pot contenir informació molt diversa, i part d'aquesta informació podria ser considerada més confidencial que una altra. Per exemple, el grup sanguini del pacient hauria de poder ser consultat per qualsevol metge. Si el pacient ha tingut un accident i cal fer-li una transfusió aquesta dada és vital. No obstant, si el pacient rep ajuda psicològica hauria de ser confidencial, i només ho hauria de poder consultar el seu metge. En aquest projecte, simplificarem al màxim l'historial mèdic, i considerarem, que l'historial, és un conjunt de visites, on cada visita consta d'un fragment de text, una data i un número de sèrie.

En el cas dels usuaris pacients i metges, hauríem de guardar el nom, l'identificador d'usuari, el seu certificat digital i les seves dades personals. En aquest projecte, per simplificar obviarem les seves dades personals, i només ens quedarem amb el nom, l'identificador d'usuari (NIF), i el certificat digital que al mateix temps ens garantirà si l'usuari és realment un metge o un pacient.

Per tant, en aquest projecte, els pacients podran consultar els seus historials mèdics, i els metges podran consultar i afegir dades (visites) en els historials mèdics dels seus pacients. Els metges també podran consultar la llista de pacients que tenen assignats.

3.5. Esquema general del projecte.

A continuació es mostra l'esquema general de funcionament del projecte. Hi ha una clara separació dels diferents elements, es treballa en un entorn client – servidor. Les aplicacions pels usuaris metge i pacient, no tenen accés directe a la base de dades, només a través del gestor de historials.

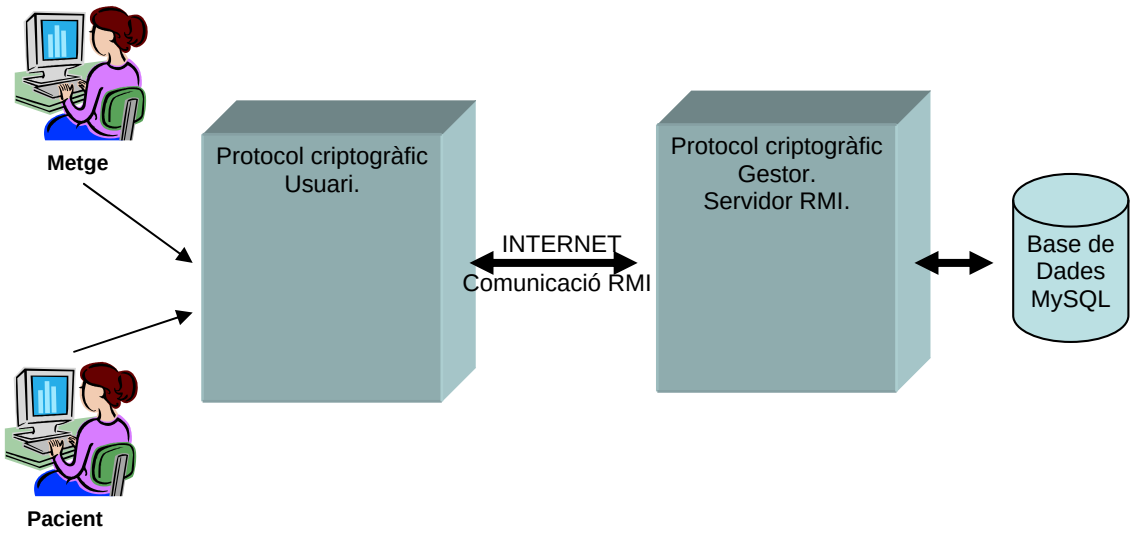

Figura 3.1.: Esquema general del projecte.

3.6. Disseny de l'aplicació: Casos d'ús en UML

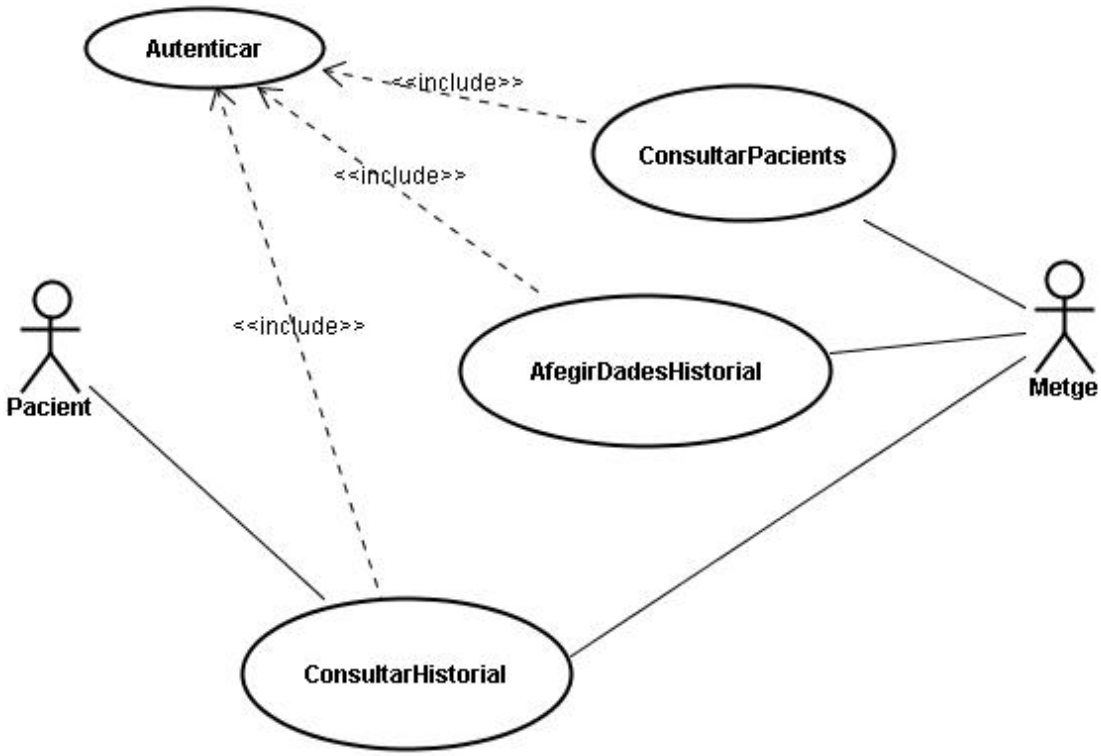

Figura 3.2.: Diagrama de casos d'ús.

Cas d'ús ConsultarHistorial: Un pacient pot consultar el seu propi historial mèdic, i un metge pot consultar l'historial d'un dels seus pacients assignats. Prèviament s'hauran d'autenticar.

Cas d'ús ConsultarPacients: Un metge pot consultar la llista de pacients assignats. Prèviament s'haurà d'autenticar.

Cas d'ús AfegirDadesHistorial: Un metge pot afegir dades a l'historial d'un dels seus pacients assignats. Prèviament s'haurà d'autenticar.

Cas d'ús Autenticar: En aquest projecte, per cadascun dels 3 casos d'ús anteriors, prèviament l'usuari s'haurà d'autenticar. Una possible ampliació no contemplada en aquest projecte, per tal de millorar l'eficiència, seria que els usuaris només s'autentiquessin una vegada al principi, i no per cada cas d'ús. Per això seria necessari mantenir l'estat de la sessió, per tal que el gestor reconegui a quin usuari pertany cada connexió.

Capítol 4: Esquema criptogràfic.

4.1. Introducció

L'esquema criptogràfic, és la part més important i raó de ser del projecte, ja que és qui garanteix els requisits de seguretat necessaris. En aquest capítol es descriuran els protocols criptogràfics de seguretat utilitzats en cada cas d'ús, i que faran ús de la signatura i xifratge de missatges emprant la infraestructura de clau pública.

4.2. Requisits de seguretat.

Com ja s'ha comentat en el capítol de infraestructura de clau pública, per tal de garantir la seguretat en l'aplicació sistema de gestió de historials mèdics segurs, necessitem que es compleixin les següents propietats:

- **Confidencialitat:** S'ha de preservar la confidencialitat de les dades de l'historial mèdic dels pacients.
- **Autenticitat:** S'ha de poder provar l'autenticitat de la informació que es guarda en el sistema.
- **Integritat:** S'ha de garantir en tot moment la integritat de la informació generada pel metge i el gestor.
- **No repudi:** No es pot negar la realització d'una acció realitzada per un usuari del sistema.

4.3. Notació emprada en els protocols.

En la descripció dels protocols s'empra la notació següent:

- (P entitat, S entitat): parella de claus asimètriques propietat d'Entitat, on P correspon a la clau pública, i S a la privada.
- S entitat[M]: Signatura digital del missatge M amb la clau privada S d'Entitat.
- E entitat(M): Xifratge del missatge M amb la clau asimètrica pública.
- P entitat d'Entitat.
- H(M): Sortida d'una funció resum criptogràfica del missatge M, aquestes funcions reben el nom de funcions hash.

4.4. Protocols per a garantir els requisits de seguretat.

A continuació, hi ha una descripció en pseudocodi del guió dels 3 protocols criptogràfics implementats:

- Protocol per la consulta d'un historial.
- Protocol per la consulta dels pacients assignats a un metge.

- Protocol per la inserció de dades a l'historial mèdic.

Alguns passos dels protocols estan agrupats en procediments anomenats "Procedure", per facilitar la comprensió del pseudocodi. En el codi real, hi han reflectits tots els passos i els procediments dels protocols.

L'autenticació dels pacients i metges es farà al principi de cadascun dels 3 protocols. Una manera de millorar l'eficiència, seria fer l'autenticació al principi i guardar la sessió, però en aquest projecte aquesta opció no està contemplada i per tant comprovarem l'autenticació cada vegada que s'executi un protocol.

Protocol per a la consulta d'un historial

En aquest protocol, cada usuari *U* s'identifica amb Id_usuariU i disposa d'una parella de claus (PU,SU) amb el corresponent certificat CertU. En el cas del gestor G, el seu identificador d'usuari Id_usuariG, és el hash del seu certificat. Aquest protocol pot ser usat 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.

Protocol 1 (Consulta d'un historial):
1. U realitza les operacions següents:

    (a) Executar el Procedure 1 amb la clau pública P_U, i obtenir $P_G[N_i$, Id usuari$_U]$;

    (b) Enviar $P_G[N_i$, Id usuari$_U]$ a G;

2. G realitza les operacions següents:

    (a) Executar el Procedure 2 amb $P_G[N_i$, Id usuari$_U]$, i obtenir $P_U[N_i,N_G$, Id usuari$_G]$;

    (b) Enviar $P_U[N_i,N_G$, Id usuari$_G]$ a U;

3. U realitza les operacions següents:

    (a) Desxifrar $P_U[N_i,N_G$, Id usuari$_G]$ amb la clau privada S_U, i obtenir N_G, $N_{0\,i}$ i Idusuari$_G$;

    (b) Si $N_G==N_i$ fer:
        i. Xifrar N_G, Consulta, Idusuari amb la clau pública P_G de G, $P_G[N_G$,Consulta, Idusuari]. Consulta indica que es vol consultar l'historial de l'usuari identificat amb Id usuari;
        ii. Enviar $P_G[N_G$,Consulta, Id usuari] a G;

    (c) Si no retornar error;

4. G realitza les operacions següents:

    (a) Desxifrar $P_G[N_G$,Consulta, Id usuari] amb la clau privada S_G, i obtenir N_{0G}, Consulta, Idusuari;

(b) Recuperar N_G de la BD. En el pas 4 del Procedure 4 N_G i N_i han estat guardats a la BD;

(c) Si $N_{G'}$==N_G fer:
i. Si (Idusuari$_U$=Id usuari) o (Id usuari$_U$ és metge i Idusuari és un pacient de Idusuari$_U$) fer:
A. Executar el Procedure 3 amb Id usuari i P_U, i obtenir P_U[H];
B. Enviar P_U[H] a U.
ii. Si no retornar error;
(d) Si no retornar error;

(e) Eliminar N_G i N_i de la BD;

5. U realitza les operacions següents:

(a) Executar el Procedure 4 amb P_U[H], i obtenir H;

(b) Mostrar H.

El procedure 1 conté part de l'autenticació del protocol de Needham-Schroeder. Aquests passos també s'utilitzaran en els altres protocols. Aquest procedure serà utilitzat per metges i pacients.

Procedure 1

1. Obtenir un valor de forma aleatòria, N_i;

2. Xifrar N_i i Id usuari$_u$ amb la clau pública de G, P_G[N_i, Id usuari$_U$];

3. Enviar P_G[N_i, Id usuari$_U$] a G.

El procedure 2 conté una altra part de l'autenticació del protocol de Needham-Schroeder. Aquesta part serà executada pel Gestor

Procedure 2 (P_G(N_i, Id usuari$_U$))

1. Desxifrar P_G[N_i, Id usuari$_U$] amb S_G, i obtenir; N_i i Id$_u$suari$_U$;

2. Obtenir el certificat de U a partir de Id usuari$_U$. A partir del certificat es pot obtenir la clau pública P_U;

3. Obtenir un valor de forma aleatòria, N_G;

4. Guardar a la BD els valors N_i i N_G associats amb U;

5. Xifrar N_i,N_G, Id usuari$_G$, amb la clau pública P_U de U, P_U[N_i,N_G, Id usuari$_G$];

6. Retornar P_U[N_i,N_G, Id usuari$_G$].

El gestor G utilitza el Procedure 3 per trobar l'historial que se li ha demanat i xifrar-lo amb la clau de l'usuari que el vol consultar.

Procedure 3 (id usuari,P_U)

1. Buscar l'historial H corresponent a id usuari;

2. Desxifrar la part de H que est`a xifrada utilitzant la clau privada S_G de G;

3. Xifrar H amb la clau pública P_U, $P_U[H]$;

4. Retornar $P_U[H]$.

Un usuari (Metge o pacient) utilitza el Procedure 4 per tal de desxifrar un historial enviat pel gestor G i verificar que l'historial és correcte.

Procedure 4

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

2. Per cada entrada de l'historia H que està signada fer:

(a) Verificar la signatura digital de M;
(b) Verificar la signatura digital de G;
(c) Verificar la seqüència;

3. Retornar H.

Protocol per a la consulta dels pacients assignats a un metge.

Una operació típica d'un metge es buscar l'historial d'un dels seus pacients. Amb el Protocol 2 un metge pot obtenir el llistat del seus pacients. En el llistat només s'envien el identificadors d'usuari, i el nom de cada pacient. Aquesta és la informació mínima per mostrar els noms i recuperar un historial.

Protocol 2

1. U realitza les operacions següents:

    (a) Executar el Procedure 1 amb la clau pública P_U, i obtenir $P_G[N_i$, Id usuari$_U]$;

    (b) Enviar $P_G[N_i$, Id usuari$_U]$ a G;

2. G realitza les operacions següents:

    (a) Executar el Procedure 2 amb $P_G[N_i$, Id usuari$_U]$, i obtenir $P_U[N_i,N_G$, Id usuari$_G]$;

    (b) Enviar $P_U[N_i,N_G$, Id usuari$_G]$ a U;

3. U realitza les operacions següents:

(a) Desxifrar $P_U[N_i, N_G$, Id usuari$_G]$ amb la clau privada S_U, i obtenir N_G, $N_{0\,i}$ i Id usuari$_G$;

(b) Si $N_{0\,?}=N_i$ fer:

i. Xifrar N_G i Llista pacients amb la clau pública P_G de G, $P_G[N_G$, Llista pacients]. Llista pacients indica que es vol un llistat dels identificadors d'usuari corresponents als pacients del metge identificat amb Id usuari$_U$;

ii. Enviar $P_G[N_G$, Llista pacients] a G;

(c) Si no retornar error;

4. G realitza les operacions següents:

(a) Desxifrar $P_G[N_G$, Llista pacients] amb la clau privada S_G, i obtenir N_{0G} i Llista pacients;

(b) Recuperar N_G de la BD. En el pas 4 del Procedure 4 N_G i N_i han estat guardats a la BD;

(c) Si $N_{0G?}=N_G$ fer:

(El camp Organizational Unit Name (eg, section) del certificat de Id usuari$_U$ indica si l'usuari és Metge o Pacient, i amb la BD podem saber els pacients assignats a un metge)

i. Si Id usuari$_u$ és metge fer:

A. Executar el Procedure 5 amb Id usuari$_U$ i P_U, i obtenir $P_U[\{$Id usuari$_1, \ldots,$ Id usuari$_n\}, S_G[\{$Id usuari$_1, \ldots,$ Id usuari$_n\}]]$;

B. Enviar a U
$P_U[\{$Id usuari$_1, \ldots,$ Id usuari$_n\}, S_G[\{$Id usuari$_1, \ldots,$ Id usuari$_n\}]]$;

ii. Sino retornar error;

(d) Sino retornar error;

(e) Eliminar N_G i N_i de la BD;

5. U realitza les operacions següents:

(a) Executar el Procedure 6 amb
$P_U[\{$Id usuari$_1, \ldots,$ Id usuari$_n\}, S_G[\{$Id usuari$_1, \ldots,$ Id usuari$_n\}]]$,
i obtenir {Id usuari$_1, \ldots,$ Id usuari$_n$};

(b) Mostrar {Id usuari$_1, \ldots,$ Id usuari$_n$}.
Amb el Procedure 5 el gestor G obté el llistat dels pacients assignats al metge Id usuari.

Procedure 5 (Id usuari, P_U)

1. Cercar a la BD els pacients assignats al metge Id usuari, obtenint {Id usuari$_1$, . . . , Id usuari$_n$};

2. Signar {Id usuari$_1$, . . . , Id usuari$_n$} amb la clau privada S_G de G, S_G[{Id usuari$_1$, . . . , Id usuari$_n$}];

3. Xifrar {Id usuari$_1$, . . . , Id usuari$_n$} i S_G[{Id usuari$_1$, . . . , Id usuari$_n$}] amb la clau pública de Id usuari, P_U, P_U[{Id usuari$_1$, . . . , Id usuari$_n$}, S_G[{Id usuari$_1$, . . . , Id usuari$_n$}]];

4. Retornar P_U[{Id usuari$_1$, . . . , Id usuari$_n$}, S_G[{Id usuari$_1$, . . . , Id usuari$_n$}]].

El Metge utilitza el Procedure 6 per obtenir la llista dels seus pacients i verificar que ha estat generada pel Gestor G.

Procedure 6 (P_U({Id usuari$_1$, . . . , Id usuari$_n$}, S_G({Id usuari$_1$, . . . , Id usuari$_n$})))

1. Desxifrar P_U[S_G[{Id usuari$_1$, . . . , Id usuari$_n$}]] amb la clau privada S_U de U, S_U[P_U[S_G[{Id usuari$_1$, . . . , Id usuari$_n$}]]] i obtenir S_G[{Id usuari$_1$, . . . , Id usuari$_n$}] ;

2. Verificar la signatura digital S_G[{Id usuari$_1$, . . . , Id usuari$_n$}] amb la clau pública P_G de G;

3. Si la verificació anterior és correcta retornar {Id usuari$_1$, . . . , Id usuari$_n$}.

Protocol per a la inserció de dades a l'historial mèdic

Aquest protocol està pensat per afegir una visita V a l'historial H d'un pacient P. El gestor comprova que la visita V ha estat signada pel metge M assignat al pacient. A continuació s'afegeix la visita a l'historial H. Per garantir que l'ordre de visites no es modifica, s'afegeix a cada visita una marca temporal T i un número de sèrie X. Amb aquestes dades podem saber l'ordre i l'instant de la visita. La visita V, el temps T i el número de sèrie són signats pel gestor G. Si un atacant elimina un registre al mig de l'historial, es detectarà perquè hi haurà un salt en el número de sèrie. La marca temporal T no protegeix l'historial contra l'eliminació duna visita. L'atacant no podrà refer la seqüència sense la clau privada del gestor G. Suposem que aquesta clau està ben protegida.

A continuació, el gestor G xifra les dades de la visita amb la seva clau pública i ho guarda a la base de dades. Si un atacant accedeix a la base de dades no pot veure les dades confidencials. Finalment afegeix a l'historial una signatura digital de quin és l'últim número de sèrie X de l'historial H. Si un atacant elimina l'última visita, es detectarà perquè hi haurà un salt entre l'última visita i el número de sèrie X signat.

Protocol 3

1. M realitza les operacions següents:

    (a) Executar el Procedure 1 amb la clau pública P_M, i obtenir $P_G[N_i$, Id usuari$_M]$;

    (b) Enviar $P_G[N_i$, Id usuari$_M]$ a G;

2. G realitza les operacions següents:

    (a) Executar el Procedure 2 amb $P_G[N_i$, Id usuari$_M]$, i obtenir $P_U[N_i,N_G$, Id usuari$_G]$;

    (b) Enviar $P_M[N_i,N_G$, Id usuari$_G]$ a M;

3. M realitza les operacions següents:

    (a) Desxifrar $P_M[N_i,N_G$, Id usuari$_G]$ amb la clau privada S_M, i obtenir N_G, $N_{0\,i}$ i Id usuari$_G$;

    (b) Si $N_{0\,i?}=N_i$ fer:

        i. Obtenir les dades de la visita V. La visita hauria d'incloure com a mínim Id usuari$_P$;

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

        iii. Xifrar N_G, V i $S_M[V]$ amb la clau pública P_G de G, $P_G[N_G$, Inserir visita, V, $S_M[V]]$. Inserir visita indica que es vol afegir V a l'historial del pacient P;

        iv. Enviar $P_G[N_G$, Inserir visita, V, $S_M[V]]$ a G;

    (c) Sino retornar error.

4. G realitza les operacions següents:

    (a) Desxifrar $P_G[N_G$, Inserir visita, V, $S_M[V]]$ amb la clau privada S_G, i obtenir N_{0G}, Inserir visita,V i $S_M[V]$;

    (b) Recuperar N_G de la BD. En el pas 4 del Procedure 4 N_G i N_i han estat guardats a la BD.

    (c) Si $N_{pG?}=N_G$ fer:

        i. Obtenir Id usuari$_P$ a partir de V;

        (El camp Organizational Unit Name (eg, section) del certificat de Id usuari$_M$ indica si l'usuari és Metge o Pacient)

        ii. Verificar que Id usuari$_M$ és metge

(Amb la BD podem saber els pacients assignats a un metge)

iii. Verificar que Id $usuari_P$ és un pacient assignat a Id $usuari_M$

iv. Si les verificacions anteriors són correctes fer:

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

B. Obtenir l'instant de temps actual T;

C. Obtenir el número de sèrie X de l'última visita de l'historial H del pacient Id $usuari_P$;

D. Incrementar en una unitat X, X + 1;

E. Signar V, $S_M[V]$, T, X + 1, amb la clau privada S_G de G, $S_G[V, S_M[V], T,X + 1]$;

F. Xifrar V i $S_M[V]$ amb la clau pública S_G de G, $P_G[V, S_M[V]]$;

G. Signar Id $usuari_P$ i X+1 amb la clau privada S_G de G, $S_G[X + 1, Id\ usuari_P]$;

H. Guardar a la BD $P_G[V, S_M[V]]$, X + 1, T, $S_G[V, S_M[V], T,X + 1]$ i $S_G[X + 1, Id\ usuari_P]$;

v. Si no retornar error.

(d) Si no retornar error.

4.5. Disseny de l'aplicació: Diagrama de classes en UML

A continuació, es mostra el diagrama de classes en aquest punt, amb el protocol criptogràfic ja desenvolupat.

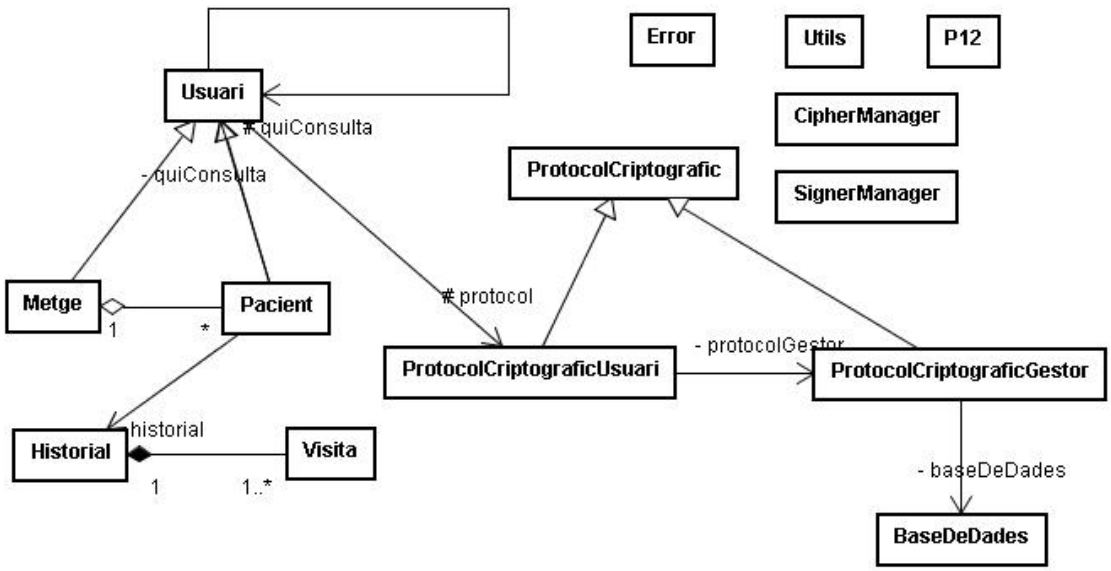

Figura 4.1.: Diagrama de classes en UML: base

Les classe *usuari*, i per tant també les classes *metge* i *pacient*, representen 2 conceptes: L'entitat sobre la qual se'n volen consultar dades, i també qui fa la consulta. Per això, la classe *pacient* té l'atribut *quiConsulta* de la classe *usuari*, que és qui fa la consulta. Per tant, pel cas de consulta d'expedient per part d'un metge, s'haurà d'instanciar la classe *pacient*, que oferirà la operació de consultar expedient, però l'objecte tindrà l'atribut *quiConsulta,* del tipus usuari, en aquest cas metge.

En aquest punt ja s'han separat les classes que contenen les operacions del protocol criptogràfic, així ja queda preparat per quan s'hi afegeixi la tecnologia RMI de comunicació remota entre les classes *ProtocolCriptograficUsuari* i *ProtocolCriptograficGestor.*

En aquest punt, la classe base de dades és només un senzill simulador per a poder provar el protocol criptogràfic i el desenvolupament de l'aplicació en aquest punt.

A continuació, les classes, amb els seus atributs, operacions, i una breu descripció.

Classe *Usuari*:

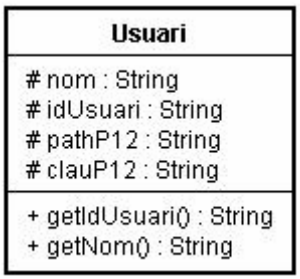

Figura 4.2.: Classe usuari

Classe de generalització de metge i pacient, amb els atributs i operacions comuns.

Classe *Metge*:

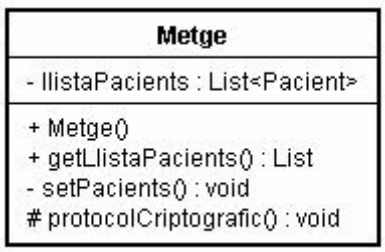

Figura 4.3.: Classe Metge

El metge té una llista de pacients associada, que permet ser consultada amb el mètode: “getLlistaPacients()”.

Classe *Pacient*:

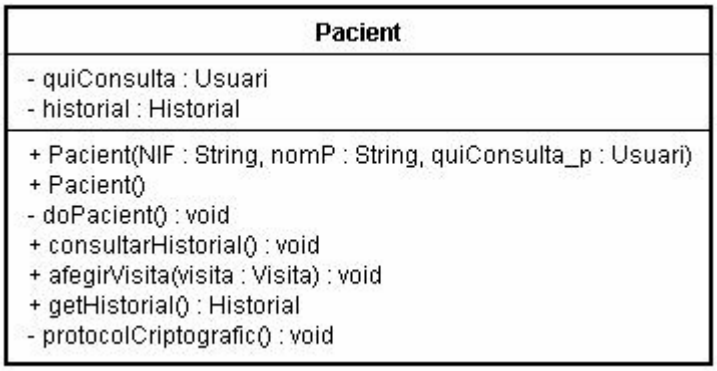

Figura 4.4.: Classe Pacient

Aplicant el patró de disseny “Expert”, a pacient li pertoca saber quin és el seu historial, mitjançant la operació consultarHistorial(). Però tal com hem explicat anteriorment, necessita tenir associat un usuari que serà qui farà la consulta, per poder fer correctament l’autenticació.

Classe *Historial*:

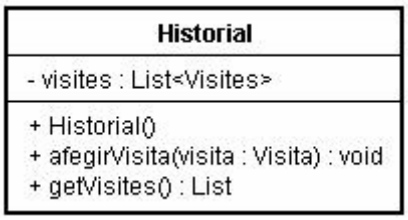

Figura 4.5.: Classe Historial

L’historial té associada una llista de visites. Té una operació per afegir visita, i una altra operació per obtenir totes les visites.

Classe *Visita*:

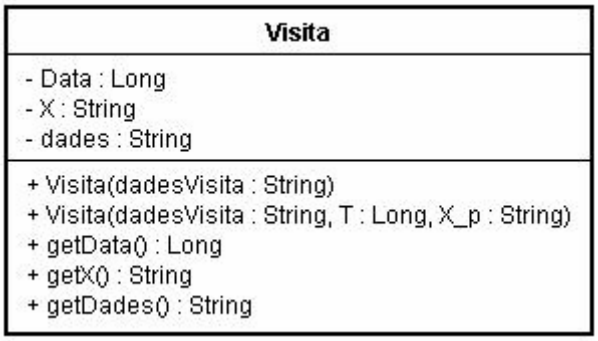

Figura 4.6.: Classe Visita

Una visita conté un camp de text anomenat “dades”, un número de sèrie de visita respecte la resta de visites de l’historial, i una data de visita.

Classe *ProtocolCriptografic*:

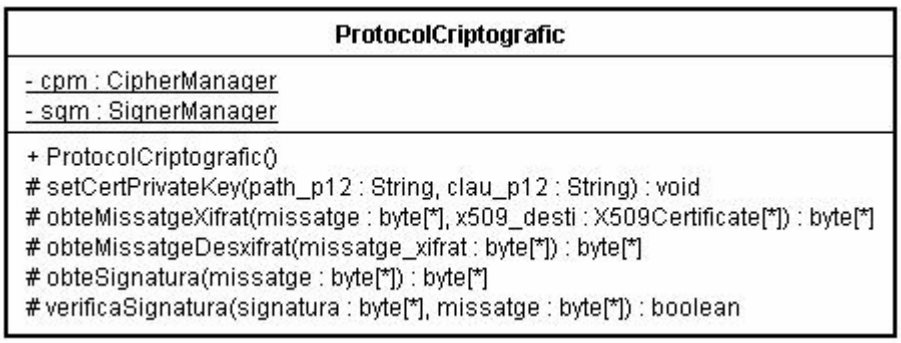

Figura 4.7.: Classe ProtocolCriptografic

Aquesta classe és abstracta, no s’instancia directament, i conté atributs i operacions comunes per les 2 parts del protocol criptogràfic: usuari i gestor. Té els atributs *CipherManager i SignerManager,* classes proporcionades pel consultor que serveixen per a xifrar missatges i signar-los respectivament.

Classe *ProtocolCriptograficUsuari*:

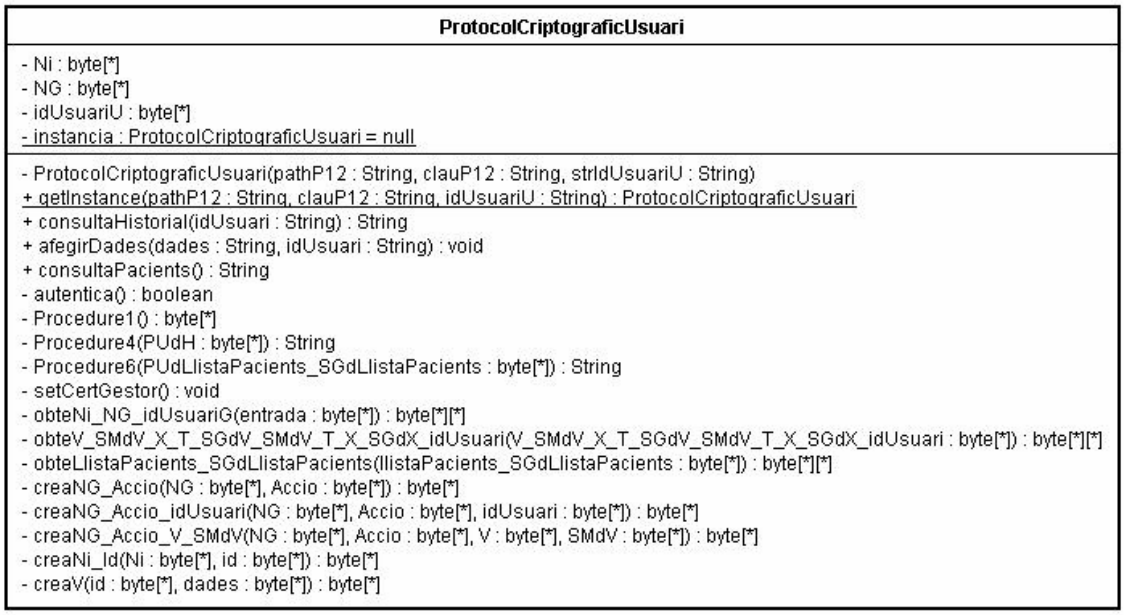

Figura 4.8: Classe ProtocolCriptograficUsuari

En aquesta classe s'ha aplicat el patró de disseny "Singleton", ja que només en voldrem una sola instància. És la classe on es troben totes les operacions del protocol criptogràfic a la part de l'usuari, la part local.

Classe *ProtocolCriptograficGestor*:

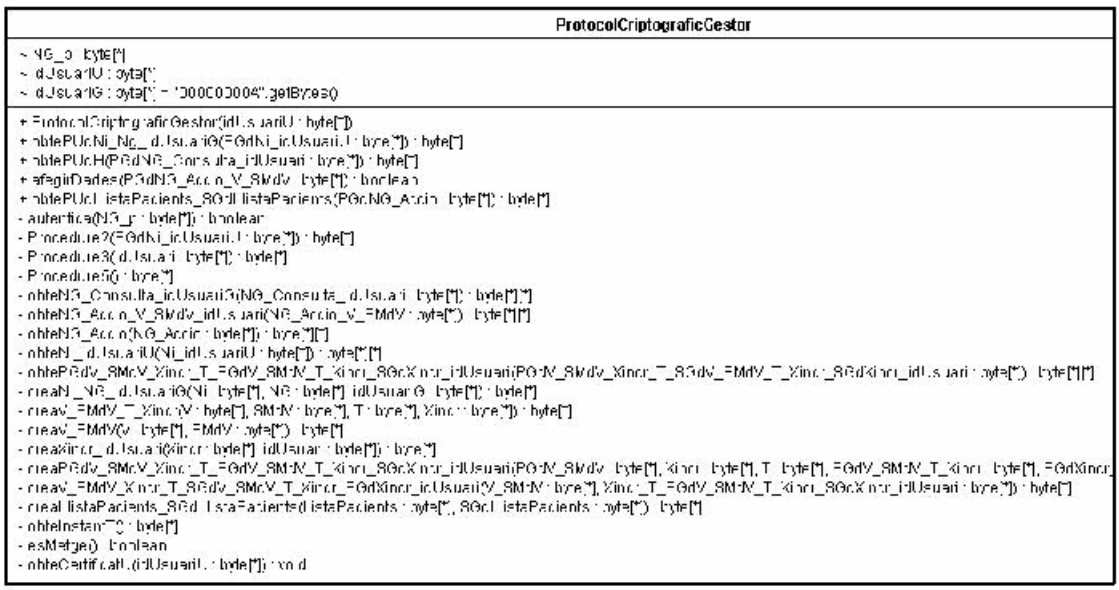

Figura 4.9: Classe ProtocolCriptograficGestor

Aquesta classe s'instanciarà cada vegada que hi hagi una connexió al servidor de historials. Conté la part del gestor del protocol criptogràfic, i és qui accedirà a la classe de base de dades.

Altres classes:

La classe *Utils*, conté operacions amb cadenes de bytes, nombres aleatoris i altres operacions relacionades amb criptografia.

La classe *Error*, és la classe encarregada de gestionar els errors que es produeixin per exemple quan una signatura digital no és pot validar.

Les classes *P12, SignerManager i CipherManager*, són classes per facilitar les operacions concretes d'extreure certificats, signar missatges i xifrar missatges respectivament.

Capítol 5: Representació de dades. XML.

5.1. Introducció

L'*XML* (eXtensible Markup Language), és una forma de representar dades que deriva del *SGML* (Standart Generalized Markup Language), i que s'ha imposat des de la seva aparició degut a l'eficiència per intercanviar i emmagatzemar dades entre aplicacions i protocols. L'*XML* es pot utilitzar per guardar i estructurar dades, per enviar-les entre aplicacions a través de la xarxa o per intercanviar informació entre organitzacions. És un avantatge el fet de estar representat en text pla, per tant no dependre de cap plataforma ni llenguatge, i tenir un ús lliure.

En aquest projecte, s'ha utilitzat l'*XML* per a representar historials mèdics, llistats de pacients, i també per a representar les dades que s'intercanvien i es transfereixen en els protocols. Per tant, s'han substituït les rutines de concatenació de cadenes de bytes utilitzades fins al moment, per crides a les diferents classes de representació de dades *XML*: Historials, Llistats de pacients i dades en general.

Per a crear documents *XML* i extreure'n dades, s'ha utilitzat la llibreria pública *JDOM*.

5.2. Estructura dels documents *XML*.

L'estructura del document *XML* per a representar historials mèdics, és la següent:

```
<?xml version="1.0" encoding="UTF-8"?>
<Historial>
<Visita>
<V_SMdV/>
<X/>
<T/>
<SGdV_SMdV_T_X/>
<SGdX_idUsuari/>
</Visita>
</Historial>
```

- <Historial> és l'arrel del document, on es guarda tota la informació de l'historial mèdic.
- <Visita> representa les dades de cada visita. Per tant, n'hi haurà una per cada visita de l'historial.
- <V_SMdV> representa les dades concretes de la visita juntament amb la signatura del metge de les mateixes dades.
- <X> és el número de sèrie de la visita dins l'historial mèdic.
- <T> és l'instant de temps (data i hora) de la visita.

- <SGdV_SMdV_T_X> representa la signatura del gestor de les dades: <V_SMdV>, <T> i <X> comentades anteriorment.
- <SGdX_idUsuari> representa la signatura del gestor de les dades: X, i identificador d'usuari (pacient)

L'estructura del document *XML* per a representar llistats de pacients, és la següent:

```
<?xml version="1.0" encoding="UTF-8"?>
<LlistaPacients>
<Pacient>
<Nom/>
<Id/>
</Pacient>
</LlistaPacients>
```

- <LlistaPacients> és l'arrel del document, on es guarda tota la informació del llistat de pacients.
- <Pacient> representa la informació per a cada pacient.
- <Nom> és on es guarda el nom del pacient.
- <Id> és on es guarda l'identificador del pacient.

L'estructura del document *XML* per a representar llistats de pacients, és la següent:

```
<?xml version="1.0" encoding="UTF-8"?>
<Contenidor>
<Dada/>
<Dada/>
</Contenidor>
```

- <Contenidor> és l'arrel del document, on es guarda tota la informació que es vol transferir.
- <Dada/> és on es guarda la informació de cada dada, n'hi hauran tantes com dades es vulguin enviar en el contenidor.

5.3. Disseny de l'aplicació: Diagrama de classes en *UML*.

Les següents classes encarregades de manegar les dades en *XML* s'afegeixen al diagrama de classes:

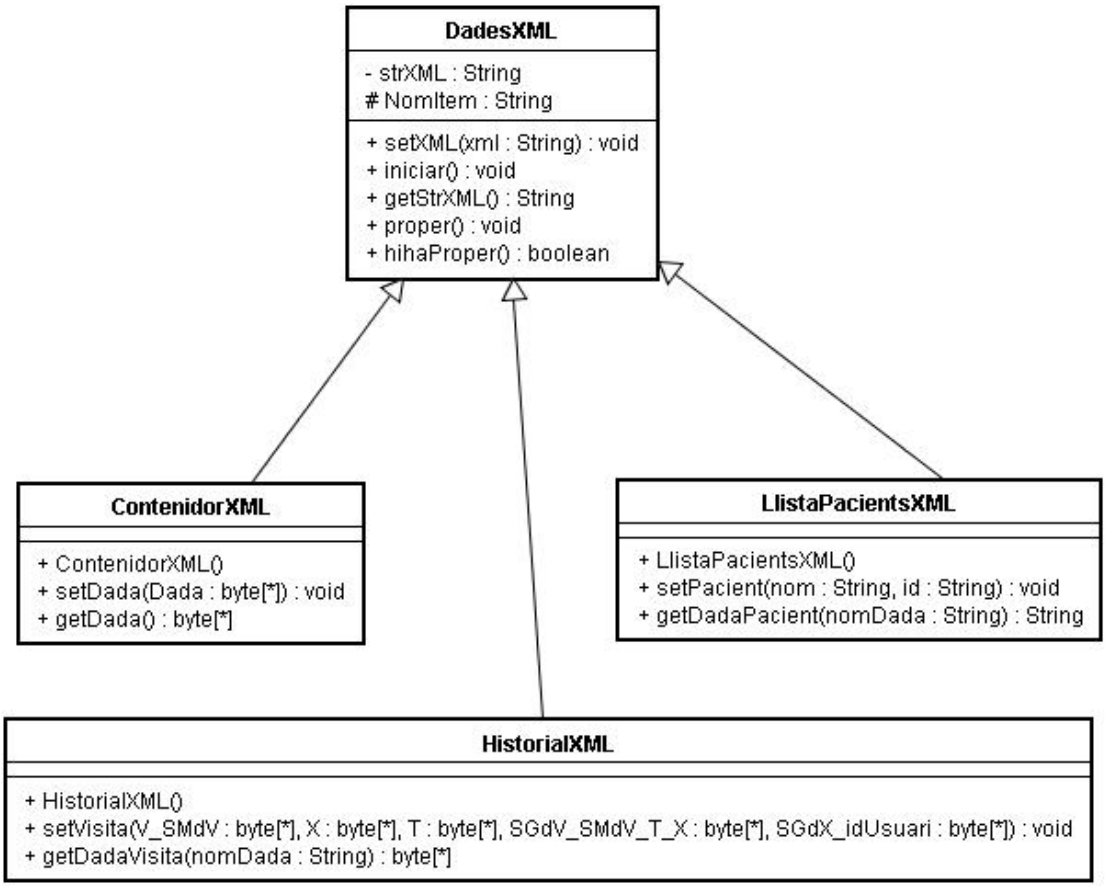

Figura 5.1: Diagrama de classes en UML: *XML*

Les classes *ProtocolCriptograficUsuari* i *ProtocolCriptograficGestor*, s'han modificat degut al canvi en la gestió de dades en *XML*. S'han eliminat les funcions específiques de tractament de dades concatenant arrays de byte, ja que ara aquestes operacions es fan mitjançant les classes de gestió de dades en *XML*. En aquest punt, són tal com es mostren a continuació:

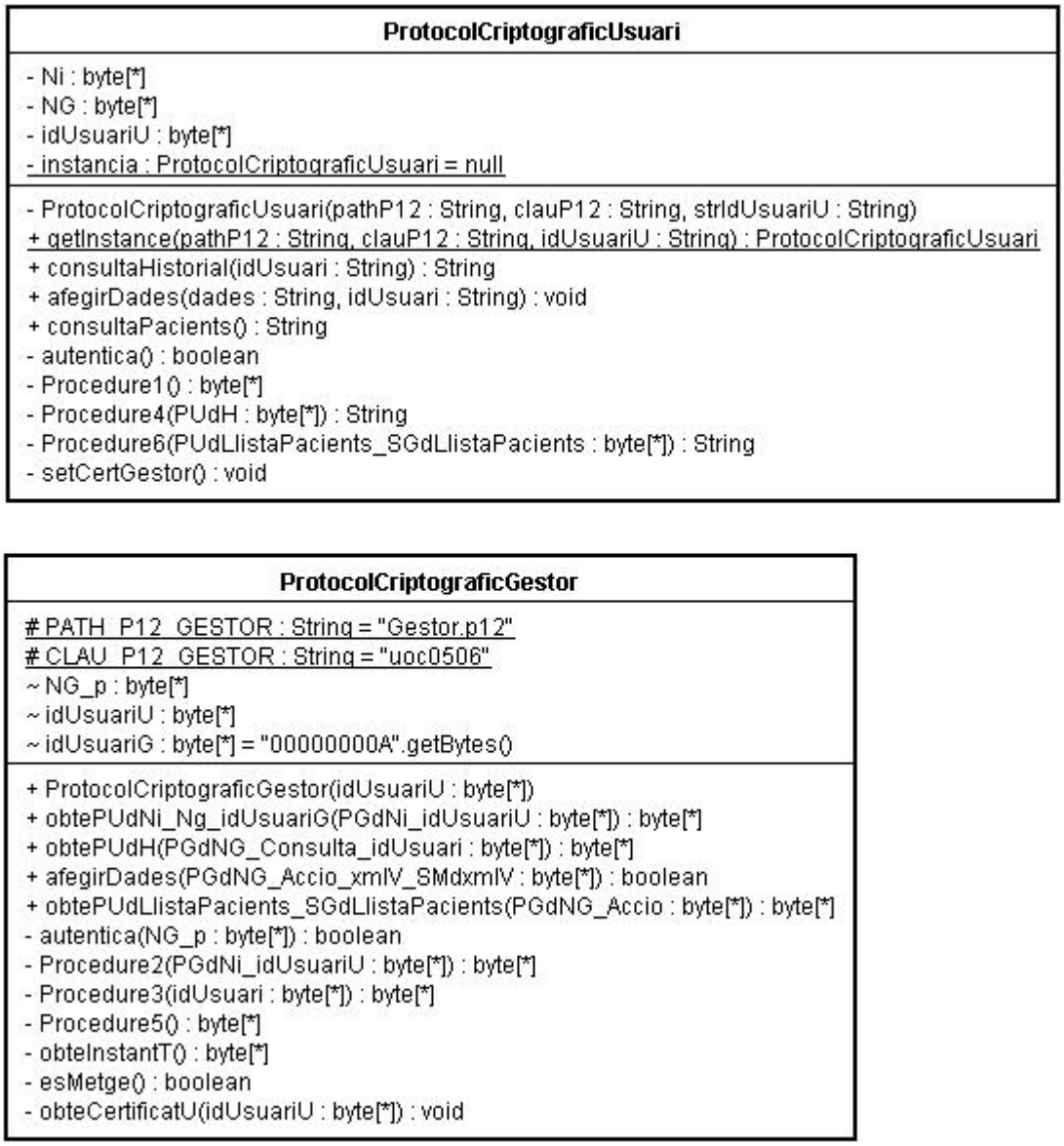

Figura 5.2: Classes ProtocolCriptograficUsuari i ProtocolCriptograficGestor

5.4. Funcionament i implementació de la representació de dades mitjançant *XML*.

El funcionament de les 3 estructures *XML* utilitzades, es basa en generar documents *XML* , que es guarden en variables tipus *String*, i també en recuperar les dades d'un document *XML* d'una variable tipus *String*.

En el cas dels historials i els llistats de pacients, l'estructura del document XML està dissenyada especialment per cada un d'aquests casos. En el cas del contenidor genèric, només es van emmagatzemant dades una darrera l'altre amb el mateix ordre que posteriorment es recuperaran mitjançant iteradors de visites en el cas d'historials, iteradors de pacients en el cas de llistat de pacients, i iteradors de dades genèriques en el cas dels contenidors genèrics.

Capítol 6: Comunicació entre components remots. *RMI.*

6.1. Introducció.

Fins aquest punt, l'aplicació SGHMS, s'executava en una sola màquina, i la comunicació entre els usuaris (metges i pacients) i el gestor, es portava a terme directament quan des de la classe ProtocolCriptograficUsuari s'instanciava directament la classe ProtocolCriptograficGestor, i es feien crides als seus mètodes. Però l'objectiu d'aquest projecte, és que els pacients, usuaris i el gestor estiguin en llocs i per tant màquines diferents, separades geogràficament i comunicades per xarxa, en aquest cas Internet. Per aconseguir separar les aplicacions d'usuari (metge i pacient) de les del gestor, utilitzarem la tecnologia RMI, descrita en el següent apartat.

6.2. *RMI*

RMI (Remote Method Invocation), és una tecnologia incorporada a l'API estàndard de Java, que permet crides a mètodes executats en una màquina virtual diferent respecte des de la que es fa la crida. L'avantatge principal, és que les diferents màquines virtuals es poden executar en dispositius físics diferents.

Per tant, tindrem una part client i una part servidor. A la part del servidor, es creen objectes remots que es fan públics mitjançant una interfície per tal que la part client pugui invocar els mètodes de l'objecte remot. El servidor fa ús d'un registre d'objectes per associar-los un nom. Quan un client vol executar un mètode d'un objecte remot, prèviament ha de cercar el seu nom al registre i posteriorment fer la crida al mètode corresponent.

6.3. Disseny de l'aplicació: Diagrama de classes en *UML*.

A continuació, es mostra una ampliació del diagrama de classes, on queda modificada la relació entre les classes *ProtocolCriptograficUsuari* i *ProtocolCriptograficGestor*, degut a que entre aquestes 2 classes, hi ha la interfície i classe corresponent a la comunicació entre components mitjançant la tecnologia *RMI*.

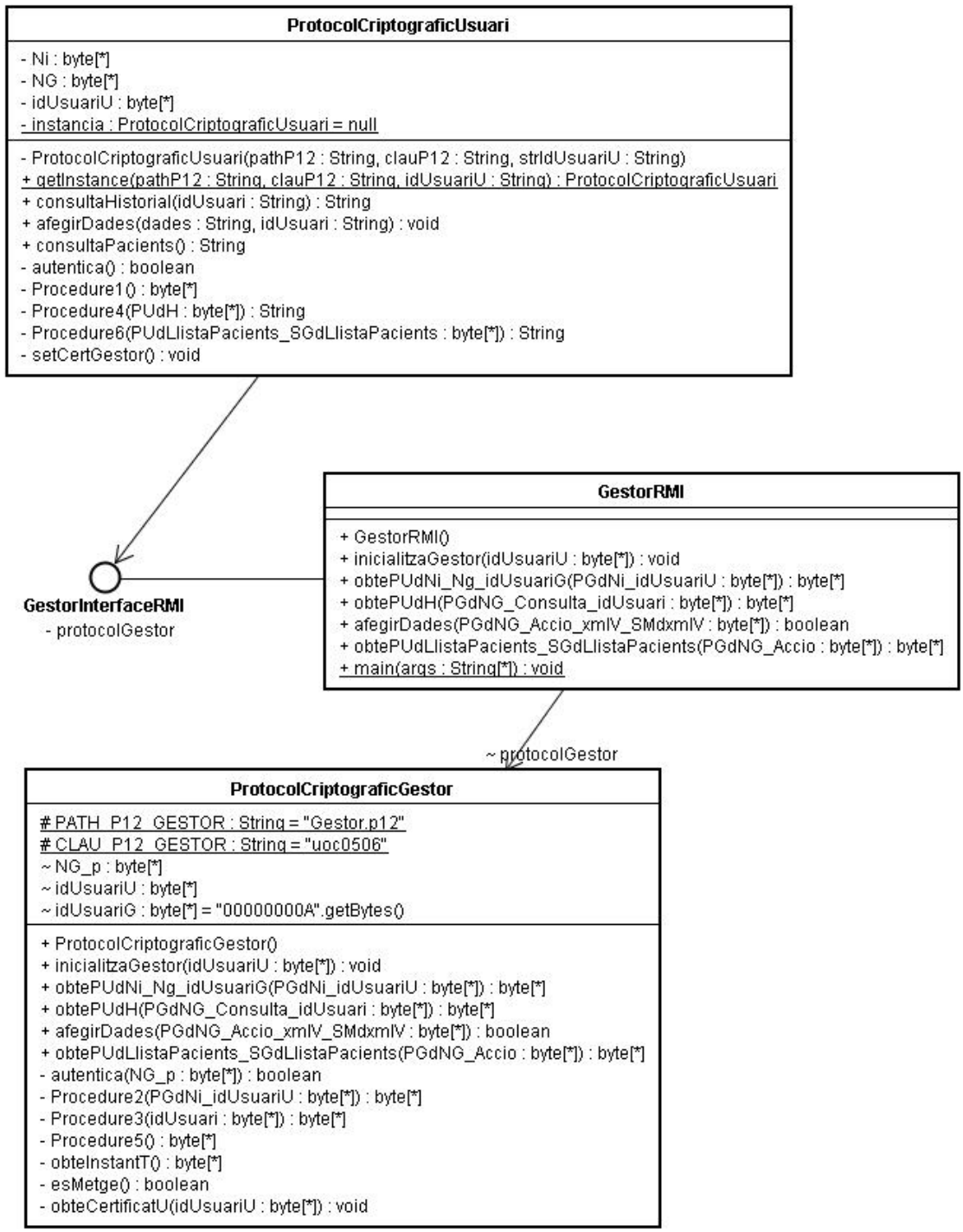

Figura 6.1.: Diagrama de classes en UML: *RMI*

6.4. Funcionament i implementació de la comunicació entre components mitjançant *RMI*.

No s'ha hagut de recórrer a cap programari addicional degut a que el sistema RMI està inclòs dins l'*API* de *Java*. S'han creat dues noves classes, *GestorRMI*, on hi ha la implementació dels mètodes que cridats des de la

classe *ProtocolCriptograficUsuari*. De tota manera, aquests mètodes es limiten a fer la crida al mètode corresponent de la classe ProtocolCriptograficGestor i tornar els mateixos resultats retornats per aquests mètodes. S'ha creat un nou mètode *inicialitzaGestor*, per incloure-hi les instruccions que fins aquest punt estaven en el constructor de la classe ProtocolCriptograficGestor. D'aquesta manera, es podrà fer una crida des de la classe *ProtocolCriptograficUsuari*, que a partir d'ara no és qui instancia la classe *ProtocolCriptograficGestor* perquè d'això s'encarregarà la classe *gestorRMI*. Tal com requereix *RMI*, s'ha creat una interfície *GestorInterfaceRMI*, on s'enumeren els mètodes que s'han d'executar remotament des de la classe *ProtocolCriptograficUsuari*.

Així doncs, en el moment que creem una instància de la classe *ProtocolCriptograficUsuari*, cerquem l'objecte remot que porta el nom de la interfície esmentada anteriorment *GestorInterfaceRMI*, i a partir d'aquest podem fer crides als mètodes que s'executaran a la màquina virtual on s'executa el servidor RMI implementat en la classe *gestorRMI*:

```
private GestorInterfaceRMI protocolGestor;
byte[ ] PUdH

try {

protocolGestor =GestorInterfaceRMI)Naming.lookup("rmi://localhost:2001/GestorInterfaceRMI");
protocolGestor.inicialitzaGestor(idUsuariU);
PUdH=protocolGestor.obtePUdH(PGdNG_Accio_idUsuari);


} catch(Exception e) {

 e.printStackTrace();

}
```

D'aquesta manera la classe *ProtocolCriptograficUsuari* invoca els mètodes de la classe *ProtocolCriptograficGestor* tal com fins ara, amb la diferència de que els mètodes d'aquesta última classe no s'executaran en la mateixa màquina sinó en una altra màquina remota gràcies a la tecnologia *RMI*.

Per a engegar el servidor *GestorRMI* i que quedi preparat per a rebre crides dels usuaris en un port *TCP*, en aquest cas el 2001, s'ha creat un script anomenat *ServidorGestor.bat* que inclou les ordres necessàries en *RMI*.

```
rmic GestorRMI
start rmiregistry 2001
start java GestorRMI
```

A partir d'aquest moment es pot executar l'aplicació client (pacient o metge) que farà tots els passos necessaris, fins que des de la classe *ProtocolCriptograficUsuari* s'invoquin remotament els mètodes de *ProtocolCriptograficGestor* a través de *GestorRMI* i la interfície *GestorInterfaceRMI*.

Capítol 7: Gestió de la informació. Base de dades.

7.1. Introducció.

Fins aquest moment, totes les operacions s'han fet amb un únic metge i un únic pacient. Les dades afegides a les visites, es carregaven a la memòria i es llegien en una mateixa execució. Per tal de poder treballar amb varis metges i pacients i que la informació quedi guardada independentment de la execució del programa, ens cal que la informació relativa i derivada de metges, pacients i visites es guardi en una base de dades.

La base de dades per tant, ens permetrà que la informació sigui persistent en el temps. També ens permetrà fer auditories de les dades guardades, i mitjançant tècniques de *mineria de dades i data ware housing*, podríem extreure informació relativa al comportament dels metges i pacients respecte aquesta aplicació. Una possible ampliació consistiria en guardar també la informació d'accés a l'aplicació per part de metges i pacients, per a poder treure conclusions a partir d'auditories, o per a controlar-ne l'accés mateix.

7.2. Sistema gestor de bases de dades: MySQL.

El MySQL és un sistema de bases de dades relacional de codi obert, suportat per vàries plataformes com Linux, Windows, MAC. Ha esdevingut un dels gestors de bases de dades més populars de codi obert gràcies a la seva potènica, alta fiabilitat i facilitat d'ús en comparació amb altres sistemes. Actualment és usat mundialment per més de 10.000 instal·lacions, des d'aplicacions basades en web, fins a sistema gestor de bases de dades de grans empreses.

Al voltant del MySQL s'ofereixen vàries aplicacions d'administració, suport i eines per gestionar les dades, i afinar aquest sistema gestor de bases de dades.

7.3. Model relacional.

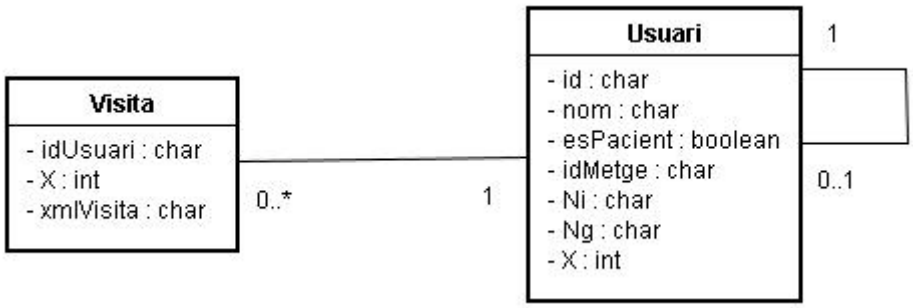

Figura 7.1: Diagrama de relacional de la base de dades.

El model relacional doncs per aquest projecte serà:

Usuari(id, nom, esPacient, idMetge, Ni, Ng, X)

Visita(idUsuari, X, xmlVisita)

Descripció dels camps:

Taula usuari

- **id:** Identificador de l'usuari (metge o pacient). És un camp tipus *varchar,* amb mida de 9 bytes que representa el NIF (8 números i una lletra) de cada usuari.

- **nom:** Nom de l'usuari (metge o pacient). És un camp tipus *varchar*, amb mida de 45 bytes.

- **esPacient:** Indicador del tipus d'usuari. És un camp tipus *integer***,** que pren valor=1 si es tracta d'un pacient, i valor=0 si es tracta d'un metge.

- **idMetge:** Clau forana relacionada amb la taula *usuari* que indica quin és el metge assignat al pacient. Camp tipus *varchar* amb mida de 9 bytes que representa l'identificador de l'usuari tipus metge assignat a el pacient. Per tant aquest camp només s'utilitza en cas que l'usuari representat pel registre sigui un pacient.

- **Ni:** Valor aleatori calculat per l'aplicació d'usuari (client) i utilitzat temporalment pel protocol criptogràfic. Camp de tipus *text***.**

- **Ng:** Valor aleatori calculat per l'aplicació del gestor (servidor) i utilitzat temporalment pel protocol criptogràfic. Camp de tipus *text***.**

- **X:** Número de sèrie en curs que correspon a l'última visita de l'historial mèdic d'un pacient. Camp de tipus *integer*. Aquest camp només tindrà s'utilitza en cas que l'usuari representat pel registre sigui un pacient.

Taula visita

- **idUsuari:** Clau forana relacionada amb la taula *usuari*, que representa l'identificador de l'usuari pacient a qui correspon la visita del seu historial mèdic. És un camp tipus *char* que representa el NIF (8 números i una lletra) de cada usuari.

- **X:** Número de série de la visita d'un historial mèdic d'un pacient. Camp de tipus *integer*.

- **xmlVisita:** Representació en XML de les dades que es guarden en una visita. L'estructura de dades en XML, s'ha explicat en el capítol 5: representació de dades. Camp de tipus *text*.

7.4. Implementació.

Per l'accés a la base de dades des de la part servidor de l'aplicació, el gestor del sistema, hem utilitzat el connector *JDBC*, que permet accedir a bases de dades MySQL des d'aplicacions programades en Java.

Tots els accessos a la base de dades, es fan des de la classe *BaseDeDades*, que fins ara es limitava a guardar les visites de l'historial en memòria, i a partir d'ara accedeix a la base de dades per llegir o escriure les dades necessàries. Els tipus de dades utilitzats són bàsics: cadenes de caràcter, text i enteres. Ja que les dades de les visites, inclòs la data, signatures i dades xifrades, estan en xml en format text, codificat en *Base64*.

A continuació es mostren fragments d'exemple de codi de la classe *BaseDeDades*, on s'accedeix a la base de dades MySQL del projecte.

Mètode constructor *BaseDeDades()*:

```
...
try {
// Carregar driver de MySQL
Class.forName("com.mysql.jdbc.Driver");
// Connectar-se a la base de dades "jdbc:mysql://host:port/database"
conn=DriverManager.getConnection("jdbc:mysql://localhost/sghms","root",
"tropemit");
 } catch (SQLException e)  {
        ...
 }
 }  catch (Exception e) {
System.out.println(e);
 }
```

Mètode guardarVisita*()*:

```
try {
Statement stmt = conn.createStatement();
// Executar consulta
strSQL="SELECT X FROM usuari WHERE id="'+strIdUsuari+"'";
ResultSet rs = stmt.executeQuery(strSQL);
if (rs.next()) {
int intX=rs.getInt("X");
intX++;
strSQL="UPDATE usuari SET X="+intX+" WHERE id="'+strIdUsuari+"'";
System.out.println(strSQL);
stmt.executeUpdate(strSQL);
strSQL="INSERT INTO visita(idUsuari,X,xmlVisita)
VALUES("'+strIdUsuari+"',"+intX+",'"+xmlVisita+"')";
                                System.out.println(strSQL);
                                stmt.executeUpdate(strSQL);
}

} catch (SQLException e) {
...
}
} catch (Exception e) {
...
}
```

La realimentació adquirida en aquest punt que l'aplicació ja funciona correctament amb una interfície de comandes, ha fet que es prengués la decisió de fer petits canvis d'organització en els mètodes i en els paràmetres passats en les crides. Per tant, les següent classes queden finalment com es mostren a continuació:

Classes de la part client de l'aplicació:

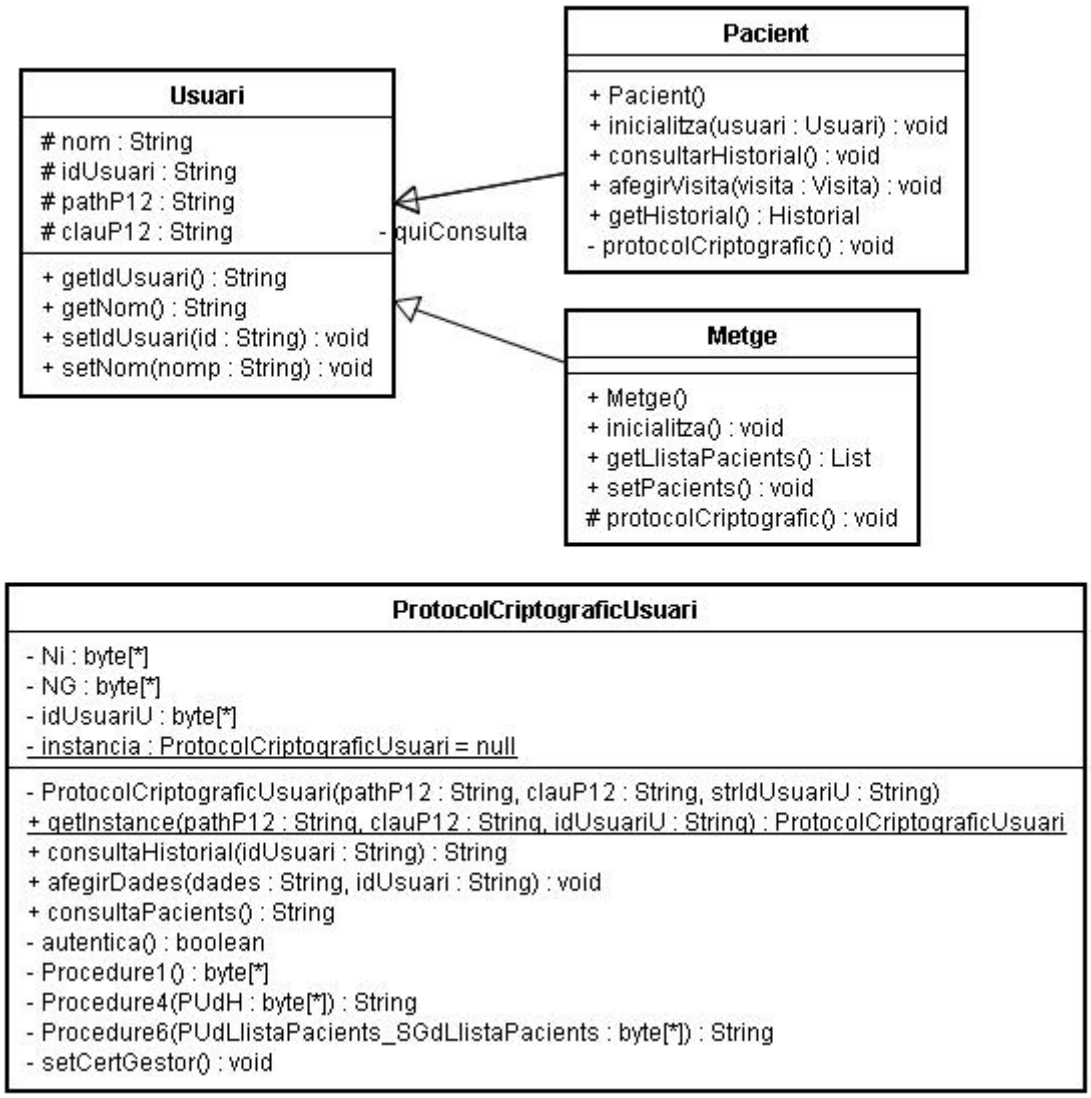

Figura 7.2: Classes de la part client en el punt actual de desenvolupament.

Classes de la part servidor de l'aplicació:

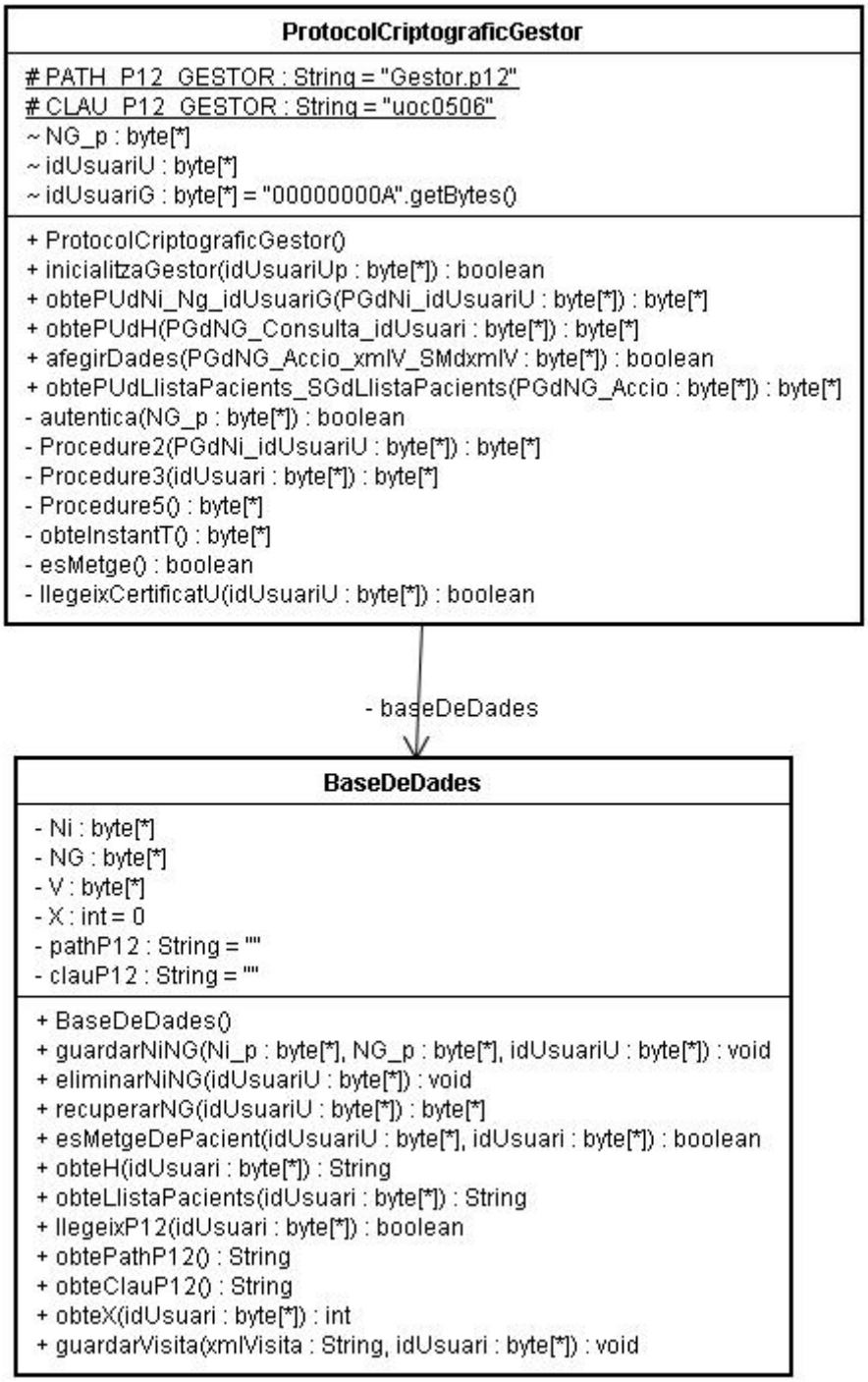

Figura 7.3: Classes de la part servidor en el punt actual de desenvolupament.

Capítol 8: Interfície gràfica d'usuari.

8.1. Introducció

L'objectiu de la interfície és la comunicació entre els usuaris, pacients i metges en el cas d'aquest projecte, i l'aplicació. En aquest projecte concret, la interfície és la part menys important, i la seva existència és justifica sobretot per poder realitzar les proves dels diferents casos d'ús amb comoditat, claredat, i fent-se una idea de les principals característiques que tindria la interfície en un cas real. Per tant s'ha fet una interfície el màxim de simple, però que permet provar tots els casos d'ús amb facilitat.

8.2. Llibreria gràfica SWT.

Per a desenvolupar la interfície gràfica d'usuari (*GUI*), s'ha utilitzat la llibreria *SWT* (*Standart Widgets Toolkit*). *SWT* és una llibreria de codi obert de components gràfics per a Java, que proveeix un accés eficient i portable a les característiques de la interfície d'usuari dels sistemes operatius pels quals es desenvolupa programari.

Aquesta llibreria gràfica és una ampliació (*plugin*) de l'entorn integrat de desenvolupament utilitzat per aquest projecte: *Eclipse*, que juntament amb una altra ampliació, el *visual editor*, permet crear visualment les diferents pantalles i finestres, afegint directament els elements anomenats *widgets*, com botons, llistes, caixes de text, etc... des d'una barra d'eines. I posteriorment, crear el codi per gestionar les accions provocades per l'accionament d'aquests elements, com la introducció de text, prémer un botó, etc...

8.3. Interfície gràfica del pacient.

La interfície gràfica del pacient, només ha permetre la comunicació amb l'usuari pel cas d'ús de mostrar l'historial d'un pacient. Per tant, només consta de 3 pantalles o finestres:

Pantalla d'autenticació

En aquesta pantalla s'escriu el NIF, que com s'ha vist en els protocols criptogràfics, és l'identificador d'usuari, en aquest cas el pacient.

Figura 8.1: Pantalla d'autenticació d'usuari.

Pantalla de visualització de l'historial mèdic.

En aquesta pantalla el pacient pot llegir el contingut de les diferents visites del seu historial mèdic, i abandonar l'aplicació mitjançant el botó "sortir".

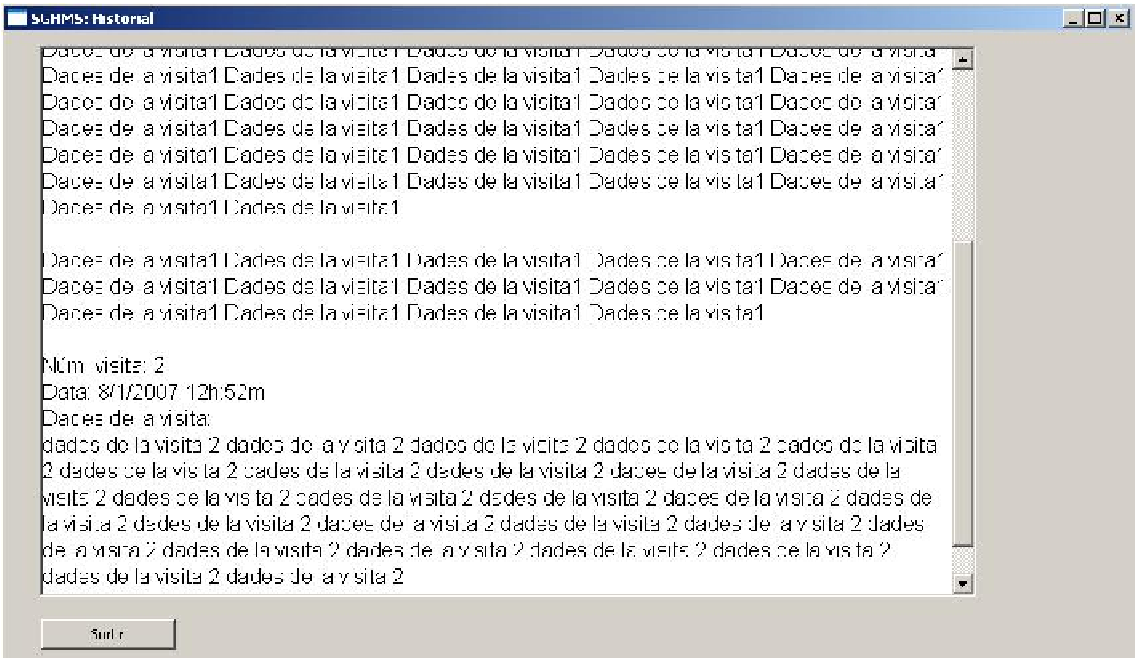

Figura 8.2: Pantalla de visualització de l'historial.

Pantalla de visualització d'errors.

En aquesta pantalla ens adonem que hi hagut un error, i podem llegir les explicacions que el sistema dóna sobre aquest error.

Figura 8.3: Pantalla de visualització d'errors.

8.4. Interfície gràfica del metge.

La interfície gràfica del metge, consta de més pantalles, ja que en aquest projecte el metge protagonitza més casos d'ús. Per tant, una vegada el metge s'ha autenticat amb una pantalla idèntica a la de la interfície gràfica del pacient, arriba a una pantalla on hi ha una llista dels seus pacients assignats. Les pantalles d'autenticació, de visualització d'errors i de visualització de l'historial d'un pacient assignat són idèntiques a les de la interfície gràfica del pacient.

Pantalla de visualització de llistat de pacients assignats.

A partir d'aquesta pantalla, podrà visualitzar l'historial del pacient seleccionat o afegir-hi noves visites. Mitjançant el botó "sortir" el metge pot abandonar l'aplicació.

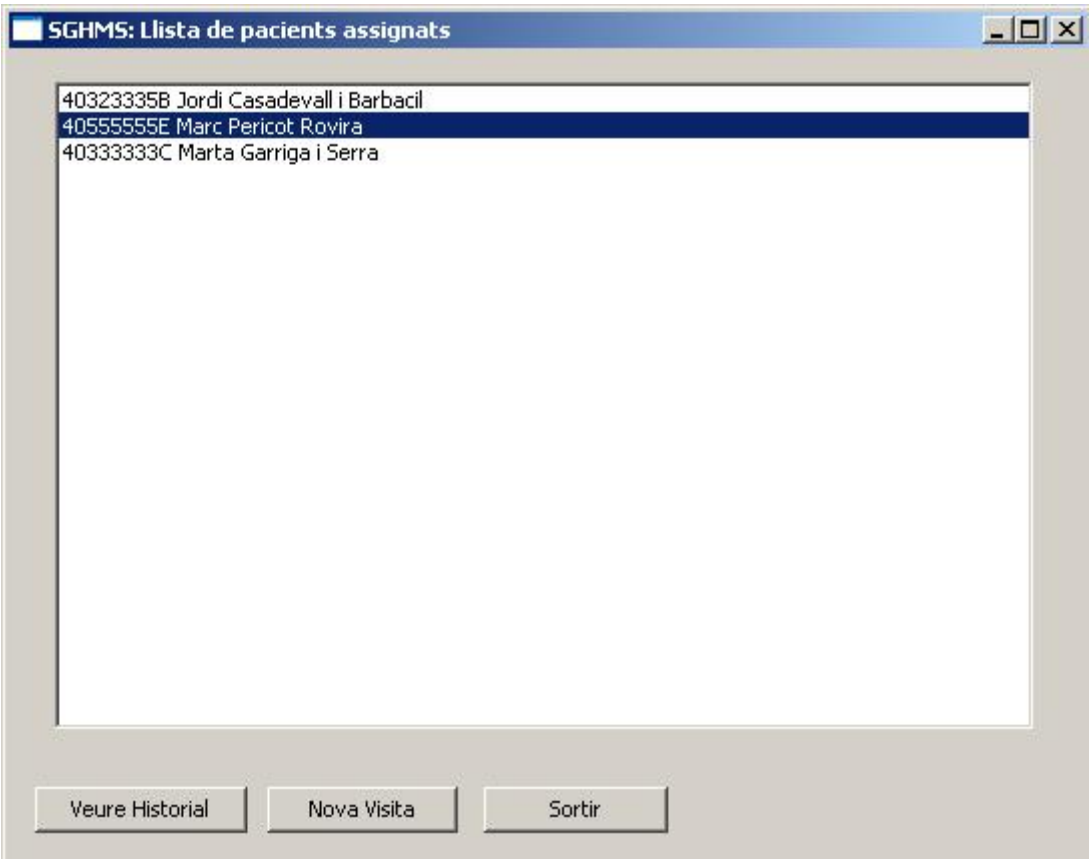

Figura 8.4: Pantalla de visualització de llistat de pacients assignats.

Pantalla d'escriure les dades d'una nova visita.

En aquesta pantalla el metge escriu les dades corresponents a una nova visita al pacient. En qualsevol moment pot avortar aquesta escriptura mitjançant el botó "sortir".

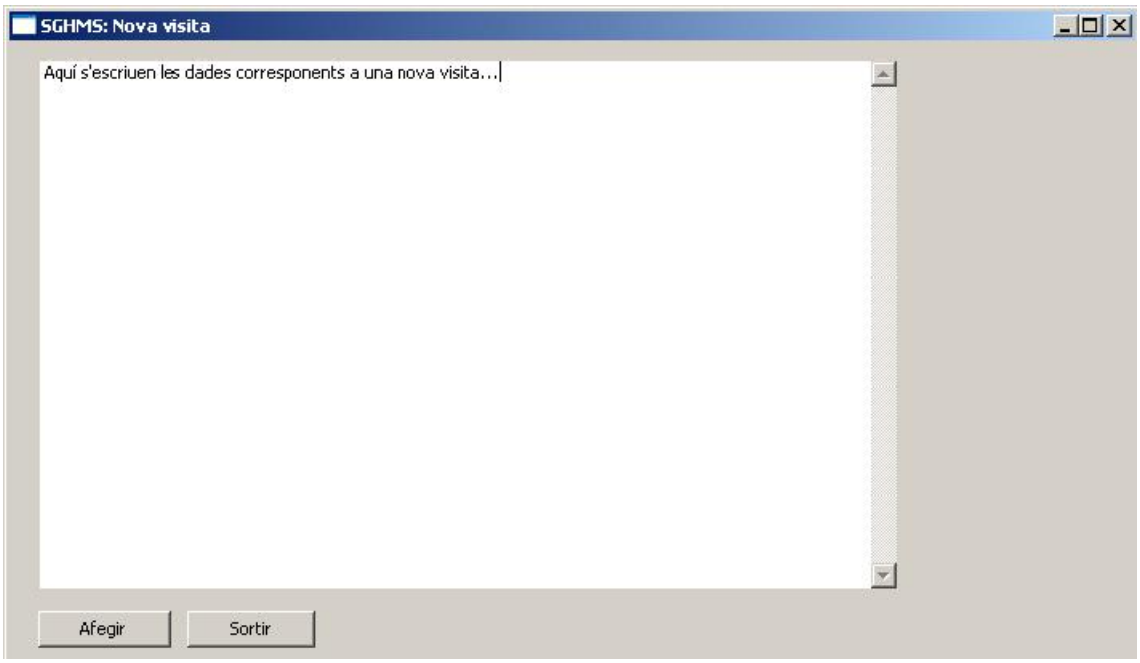

Figura 8.5: Pantalla d'escriure dades d'una nova visita.

8.5. Implementació

Per a la implementació de la interfície gràfica d'usuari, s'ha fet ús del patró de d'arquitectura *MVC* (Model Vista Controlador). Aquest patró es basa en dividir el sistema en 3 en 3 tipus de components des del punt de vista de la programació de la interfície d'usuari.

- **Models:** Encapsulen l'estat del sistema. El seu nivell d'abstracció és el domini del problema. És on implementem la lògica de negoci. En el cas d'aquest projecte, els models comencen en les classes *Metge, Pacient, Historial, i Visita,* des de les quals podem obtenir i gestionar tota la informació.

- **Vistes:** Presenten les dades als usuaris i en recullen les interaccions per enviar-les als controladors. En el cas d'aquest projecte, les classes *GUIError, GUIAutentiacio, GUIHistorial, GUILlistaPacients i GUIAfegirDades* formen part d'aquest grup.

- **Controlador:** Estableixen la relació entre les accions de l'usuari i els esdeveniments del sistema. També decideixen quines vistes es mostren als usuaris, i obtenen del model les dades que es mostraran a la vista. En aquest projecte, aquest component correspon a la classe *ControladorGUI*, que serà la classe instanciada per les aplicacions: *AplicacioPacient.class i AplicacioMetge.class*.

Gràcies a la utilització d'aquest patró d'arquitectura, s'aconsegueix que es compleixin propietats desitjables en el disseny d'aplicacions de programari:

- La cohesió de les classes és més alta perquè totes les seves responsabilitats estan relacionades.
- Podem suportar més d'una tecnologia de visualització. Podríem ampliar-ho a interfícies web o per a petits dispositius com organitzadors personals (*PDA*) o altres dispositius mòbils.
- Els desenvolupadors es poden especialitzar en un domini concret, ja sigui el domini de l'aplicació o el domini de desenvolupament d'interfícies gràfiques.
- Podem actualitzar la lògica de visualització independentment de la lògica de negoci, o actualitzar la lògica de negoci sense haver de modificar la lògica de visualització.

A la classe *ControladorGUI* s'ha aplicat el patró de disseny "Singleton", ja explicat i aplicat en el capítol 4 d'aquesta memòria, per tal de tenir una única instància d'aquesta classe encarregada de fer de pont entre les classes de lògica de negoci i les classes visuals de l'aplicació.

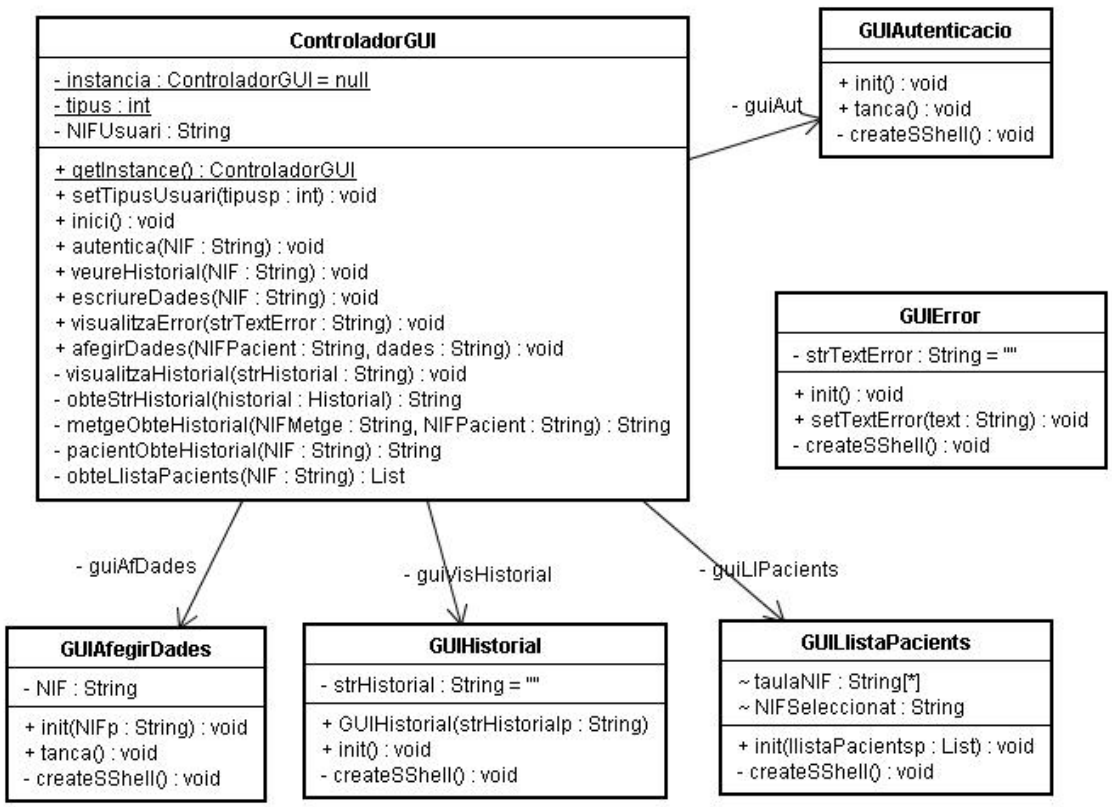

Figura 8.6: Diagrama de classes de la interfície gràfica d'usuari.

Capítol 9: Valoració econòmica.

El nombre d'hores dedicades al desenvolupament d'aquest projecte i la seva documentació, ha estat de 230 hores, de les quals 100 han estat emprades en l'anàlisi, disseny i documentació, i 130 hores per al desenvolupament de codi i realització de proves unitàries.

En funció d'aquest nombre d'hores i d'un preu/hora diferanciat per a programació i anàlisi i disseny, obtindríem el cost final. Suposant 60 €/h per l'anàlisi, disseny i documentació i 40 €/h per el desenvolupament de codi i realització de proves unitàries, el cost final es calcularia de la següent manera:

100h*60 €/h + 130h*50 €/h = **12.500 €**

En un cas real, seria necessari calcular també el cost de la implantació del sistema gestor en un servidor, incloent la base de dades i el programari necessari, com l'entorn d'execució en Java (*JRE*), que ja inclou *RMI* (comunicació entre components). I també el cost d'instal·lació i posada a punt de la base de dades *MySQL* en el servidor.

Capítol 10: Conclusions.

Els objectius proposats a l'inici del projecte, s'han complert ja que s'ha aconseguit desenvolupar un sistema de gestió d'historials mèdics segurs, anomenat SGHMS, que permet a pacients i metges, la consulta i actualització de les visites dels historials, complint les propietats de seguretat explicades al capítol 4, apartat 4.2 "Requisits de seguretat", representant les dades en *XML*, i guardant la informació en una base de dades *MySQL*.

El desenvolupament d'aquest projecte ha permès al seu autor aprofundir, i fer recerca sobre els següents temes:

- Ús d'una *PKI*, mitjançant *OpenSSL* per a la creació de certificats i *PKCS#12*.
- Desenvolupament d'esquemes *UML* amb l'aplicació de lliure distribució: *Jude Community*.
- Programació orientada a objectes en llenguatge *Java* a partir d'un disseny realitzat en *UML.*
- Desenvolupament de codi en *Java* utilitzant tipus de dades per representar textos xifrats i signatures. *Byte[ ], Base64*, ...
- Ús de la llibreria criptogràfica *IAIK* en el llenguatge de programació *Java*.
- Ús de *JDOM*, per manipular en dades representades en *XML,* mitjançant el llenguatge de programació *Java*
- Ús del sistema de comunicació entre components en *Java: RMI* per al desenvolupament d'aplicacions distribuïdes, accés a mètodes en objectes remots.
- Accés a bases de dades *MySQL,* des d'aplicacions programades en llenguatge *Java*.
- Desenvolupament d'interfícies gràfiques d'usuari (*GUI*)*,* fent ús de la llibreria gràfica *SWT* i fent ús del *Visual Editor*, ampliació de l'entorn integrat de programació *en Java: Eclipse*, que permet desenvolupar interfícies visualment afegint elements com botons, caixes de text, etc... directament des d'una barra d'eines.

En termes generals, ha estat una magnífica oportunitat de posar en pràctica importants coneixements adquirits durant els estudis d'enginyeria en informàtica, així com la realització de recerca en tecnologies molts interessants de seguretat, programació distribuïda, representació i emmagatzematge de dades, mitjançant un cas innovador amb moltes possibilitats de ser implementat com a projecte real.

Glossari de termes

- **Base 64**: Codificació que empra només 6 bits per caràcter. D'aquesta manera tenim 64 possibles valors. En el nostre cas permet transmetre cadenes que contenen les signatures com a cadena de caràcters sense fer malbé el contingut de les mateixes. Base64 també s'utilitza molt en comunicacions amb dades binàries provinents de text, com ara en el correu electrònic.
- **CA:** Autoritat certificadora que emet certificats digitals.
- **Cas d'ús**: Diagrama pertanyent a les especificacions UML que permet veure gràficament i per a cada actor del sistema, les accions que pot dur a terme i les relacions d'aquestes accions amb el sistema.
- **Certificat**: Arxiu que conté les dades que donen fe de l'autenticitat de la persona o entitat que el presenta.
- **Clau**: Peça d'informació que s'utilitza en criptografia simètrica per a xifrar i desxifrar un missatge. En criptografia asimètrica la clau pot ser pública o privada. La clau pública s'utilitza per xifrar missatges o verificar una signatura. La clau privada s'utilitza per desxifrar o per signar unes dades. La longitud en bits de la clau sovint ens dóna una idea de la robustesa del sistema que emprem.
- **Eclipse:** Entorn de desenvolupament i editor de Java.
- **Entorn de desenvolupament**: Conjunt d'eines que permeten el desenvolupament d'un programari de manera integrada, des de la redacció del codi, fins la posterior compilació i l'execució a passos (debugging) per verificar els errors.
- **Funció de Hash**: Resum amb pèrdua que dona lloc a una seqüència de longitud fixa a partir d'unes dades sense importar la longitud d'aquestes. Les seves propietats permeten la seva utilització alhora de verificar la integritat de les dades. Les funcions de hash també són utilitzades per assignar posicions en temes de cerca i ordenació.
- **IAIK:** Biblioteca de classes criptogràfiques per Java.
- **Java:** Llenguatge de programació de SUN Microsystems.
- **JDOM:** Biblioteca de classes per manipular *XML* amb Java.
- **Jude Community:** Eina de disseny UML.
- **Lliure distribució**: Se'n diu així del programari que s'ofereix lliurement sense la necessitat de que l'usuari final aboni una quantitat de diners per a la seva utilització. Normalment, amb la distribució s'inclou el codi font al que l'usuari hi te accés a modificar-lo i redistribuir-lo si així ho desitja.
- **MySQL:** Gestor de bases de dades relacionals SQL.
- **OpenSSL:** Programari per gestionar plataformes PKI.
- **PKI:** Infraestructura criptogràfica de clau pública.
- **PKCS#12:** Standard que permet representar informació d'identitat personal, claus privades, públiques i certificats entre d'altres.
- **RMI:** Invocació remota de mètodes de Java.
- **SQL:** Llenguatge de consulta de bases de dades.
- **UML:** Metodologia de disseny d'aplicacions informàtiques.
- **SWT:** Llibreria de components gràfics.
- **XML:** Llenguatge de representació de dades.

Bibliografia

- **OpenSSL: The Open Source toolkit for SSL/TLS**
  www.openssl.org
- **Eclipse - an open development platform**
  www.eclipse.org
- **Apunts de criptografia i signatura digital de la UOC, any 2003**
- **Apunts d'enginyeria del software i UML de la UOC, any 2004**
- **The J2SE Development Kit (JDK)**
  java.sun.com/downloads
- **Java Remote Method Invocation (Java RMI)**
  http://java.sun.com/products/jdk/rmi/
- **The JDOM XML API**
  www.jdom.org/docs/apidocs/index.html
- **The MySQL database server**
  www.mysql.com/documentation/index.html
- **The Unified Modeling Language: UML**
  www.uml.org
- **SWT: The Standard Widget Toolkit**
  www.eclipse.org/swt/

Annèx A: Fitxer de configuració per a PKI.

```
#
# OpenSSL example configuration file.
# This is mostly being used for generation of certificate requests.
#

# This definition stops the following lines choking if HOME isn't
# defined.
HOME                    = .
RANDFILE        = $ENV::HOME/.rnd

# Extra OBJECT IDENTIFIER info:
#oid_file               = $ENV::HOME/.oid
oid_section     = new_oids

# To use this configuration file with the "-extfile" option of the
# "openssl x509" utility, name here the section containing the
# X.509v3 extensions to use:
# extensions        =
# (Alternatively, use a configuration file that has only
# X.509v3 extensions in its main [= default] section.)

[ new_oids ]

# We can add new OIDs in here for use by 'ca' and 'req'.
# Add a simple OID like this:
# testoid1=1.2.3.4
# Or use config file substitution like this:
# testoid2=${testoid1}.5.6

####################################################################
#####
[ ca ]
default_ca    = CA_default      # The default ca section

####################################################################
#####
[ CA_default ]

dir             = ./CAPFC       # Where everything is kept
certs           = $dir/certs    # Where the issued certs are kept
crl_dir         = $dir/crl      # Where the issued crl are kept
database        = $dir/index.txt        # database index file.
new_certs_dir   = $dir/newcerts         # default place for new certs.
```

```
certificate     = $dir/CA.crt # The CA certificate
serial          = $dir/serial           # The current serial number
crl             = $dir/crl.pem                  # The current CRL
private_key   = $dir/private/CA.key # The private key
RANDFILE    = $dir/private/.rand  # private random number file

x509_extensions     = usr_cert              # The extentions to add to the cert

# Extensions to add to a CRL. Note: Netscape communicator chokes on V2
CRLs
# so this is commented out by default to leave a V1 CRL.
# crl_extensions      = crl_ext

default_days = 365                  # how long to certify for
default_crl_days= 30                        # how long before next CRL
default_md      = sha1                # which md to use.
preserve      = no                  # keep passed DN ordering

# A few difference way of specifying how similar the request should look
# For type CA, the listed attributes must be the same, and the optional
# and supplied fields are just that :-)
policy          = policy_match

# For the CA policy
[ policy_match ]
countryName                 = match
stateOrProvinceName       = optional
organizationName    = match
organizationalUnitName    = optional
commonName                  = supplied
emailAddress                = optional

# For the 'anything' policy
# At this point in time, you must list all acceptable 'object'
# types.
[ policy_anything ]
countryName                 = optional
stateOrProvinceName       = optional
localityName          = optional
organizationName    = optional
organizationalUnitName    = optional
commonName                  = supplied
emailAddress                = optional

####################################################################
#####
[ req ]
default_bits          = 1024
default_keyfile       = privkey.pem
distinguished_name          = req_distinguished_name
```

```
attributes              = req_attributes
x509_extensions     = v3_ca         # The extentions to add to the self signed cert

# 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     = Catalunya

localityName                    = Locality Name (eg, city)
localityName_default            = Barcelona

0.organizationName              = Organization Name (eg, company)
0.organizationName_default      = Universitat Oberta de Catalunya

# we can do this but it is not needed normally :-)
#1.organizationName             = Second Organization Name (eg, company)
#1.organizationName_default     = World Wide Web Pty Ltd

organizationalUnitName          = Organizational Unit Name (eg, section)
organizationalUnitName_default  = Consultors

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

emailAddress                    = Email Address
emailAddress_max        = 40

# SET-ex3                       = SET extension number 3
```

```
[ 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

# 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			= "Seguretat en Xarxes de Computadors"

# 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

[ v3_req ]

# Extensions to add to a certificate request

basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment

[ v3_ca ]


# Extensions for a typical CA


# PKIX recommendation.

subjectKeyIdentifier=hash

authorityKeyIdentifier=keyid:always,issuer:always

# This is what PKIX recommends but some broken software chokes on critical
# extensions.
#basicConstraints = critical,CA:true
# So we do this instead.
basicConstraints = CA:true

# Key usage: this is typical for a CA certificate. However since it will
# prevent it being used as an test self-signed certificate it is best
# left out by default.
# keyUsage = cRLSign, keyCertSign

# Some might want this also
# nsCertType = sslCA, emailCA

# Include email address in subject alt name: another PKIX recommendation
# subjectAltName=email:copy
# Copy issuer details
# issuerAltName=issuer:copy

# DER hex encoding of an extension: beware experts only!
# obj=DER:02:03
# Where 'obj' is a standard or added object
# You can even override a supported extension:
# basicConstraints= critical, DER:30:03:01:01:FF

[ crl_ext ]
```

```
# CRL extensions.
# Only issuerAltName and authorityKeyIdentifier make any sense in a CRL.

# issuerAltName=issuer:copy
authorityKeyIdentifier=keyid:always,issuer:always
```

Annèx B: Script d'exportació de la base de dades MySQL

```
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;

/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;


--
-- Create schema sghms
--

CREATE DATABASE IF NOT EXISTS sghms;
USE sghms;

--
-- Definition of table `usuari`
--

DROP TABLE IF EXISTS `usuari`;
CREATE TABLE `usuari` (
  `id` varchar(9) NOT NULL default '',
  `nom` varchar(45) NOT NULL default '',
  `esPacient` int(10) unsigned NOT NULL default '0',
  `idMetge` varchar(9) default NULL,
  `Ni` text,
  `Ng` text,
  `X` int(10) unsigned default '0',
  PRIMARY KEY  (`id`)
) TYPE=MyISAM;

--
-- Dumping data for table `usuari`
--

/*!40000 ALTER TABLE `usuari` DISABLE KEYS */;
INSERT INTO `usuari` (`id`,`nom`,`esPacient`,`idMetge`,`Ni`,`Ng`,`X`) VALUES
 ('40323335B','Jordi Casadevall i Barbacil',1,'11111111A','res','res',2),
 ('40333333C','Marta Garriga i Serra',1,'11111111A','res','res',0),
 ('40444444D','Pere Espinosa i Grau',1,'22222222B',NULL,NULL,0),
 ('11111111A','Dr. Joan Casademont',0,NULL,'res','res',0),
```

```
 ('22222222B','Dr. Carles Martínez',0,'','res','res',0),
 ('40555555E','Marc Pericot Rovira',1,'11111111A','res','res',0),
 ('40666666F','Francesc Puig Huguet',1,'22222222B','res','res',0);
/*!40000 ALTER TABLE `usuari` ENABLE KEYS */;


--
-- Definition of table `visita`
--

DROP TABLE IF EXISTS `visita`;
CREATE TABLE `visita` (
  `idUsuari` varchar(9) NOT NULL default '',
  `X` int(10) unsigned NOT NULL default '0',
  `xmlVisita` text NOT NULL,
  PRIMARY KEY  (`idUsuari`,`X`)
) TYPE=MyISAM;


/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101                                                    SET
CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101                                                    SET
CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101                                                    SET
COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40101                                                    SET
CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
```

Annèx C: Joc de proves

Per a la realització de proves i compilació del codi font en un entorn amb sistema operatiu Windows XP, tot seguit indico l'estructura de directoris, llibreries i fitxers necessaris.

L'estructura de directoris del fitxer comprimit en format zip d'entrega del projecte és:

```
\bin
\doc
\pki
\src
```

Al directori \bin hi trobem les classe compilades, fitxers p12 i scripts necessaris per a l'execució de les aplicacions pel metge i el pacient.

Al directori \doc hi trobem la present documentació.

Al directori \pki hi trobem...

Al directori \src hi trobem el codi font en llenguatge Java

Requeriments de sistema

Cal tenir instal·lat l'entorn Java d'execució jre1.5_09 i desenvolupament jdk1.5_09 (si es volen compilar les classes) , ja que es requereix Java 5.0 degut a l'ús de tipus de dades parametritzats.

Cal tenir les llibreries:

iaik_jce_ful.jar (iaik)
jdom.jar (jdom, xml)
mysql-connector-java-3.0.8-stable-bin.jar (jdbc, connexió java-mysql)
swt.jar (components gràfics)

al directori /lib/ext del JRE i al /jre/lib/ext del JDK. I la llibreria DLL:

swt-win32-3235.dll (components gràfics swt)

Al directori /bin del JRE.

Caldrà tenir en funcionament una base de dades MySQL en el mateix ordinador on es troba el servidor gestor. Es pot crear l`estructura i dades d'usuaris de proves amb el dump del fitxer de l'Annex B: Script d'exportació de la base de dades.

Execució de les proves

Primer cal posar en marxa el gestor servidor que podria estar ubicat en un lloc físicament diferent sempre que canviem la IP en la crida RMI:

>rmic GestorRMI
>start rmiregistry 2001
>start java GestorRMI

Aquest procés es pot fer executant l'escript:

>ServidorGestor.bat

Una vegada tenim en funcionament el servidor o gestor, per executar l'aplicació del pacient, fem:

>java AplicacioPacient

Introduïm el NIF d'un pacient existent: "40323335B", i en veiem el seu historial. Si introduïm un NIF que no figuri a la base de dades com a usuari pacient, una finestra d'error ens avisarà.

>java AplicacioMetge

Introduïm el NIF d'un metge existent: 11111111A, i en veiem els seus pacients assignats. Seleccionem per exemple el "40323335B Jordi Casadevall", i mitjançant el botó corresponent, en podem veure l'historial o afegir-hi dades.