Implementació d'un esquema criptogràfic per gestionar de forma segura els historials mèdics dels pacients a través d'una xarxa de comunicacions

Alumne: David Mateos García

Enginyeria en Informàtica

Consultor: Jordi Castellà-Roca

08/01/2007

Dedicado a Sandra.

Resum

Aquest PFC s'emmarca dins de l'àrea de seguretat informàtica. Entenem per seguretat informàtica el conjunt constituït per diverses metodologies, documents, programari i maquinari que determinen que els accessos als recursos d'un sistema informàtic siguin duts a terme exclusivament pels elements autoritzats a fer-ho.

A més aquest projecte també s'endinsa dins de l'àrea de l'informàtica mèdica.La medicina, com qualsevol altre sector, també ha introduït las Tecnologies de la Informació i les Comunicacions (TIC) en els seus processos sanitaris. En el sector de la salut, un dels aspectes més importants i necessaris per a brindar un adequat servei i atenció mèdica amb qualitat és la disponibilitat de la informació, ja sigui per a la presa de decisió clínica, sanitària o de direcció, així com, en la docència i la investigació. El processament d'aquests sistemes d'informació amb la utilització de les noves tecnologies de la informació i les comunicacions permet l'optimització de tots els processos amb la finalitat de proporcionar millors serveis de salut i contribuir d'aquesta forma a assolir millores en la salut individual i col·lectiva i una millor qualitat de vida a la població.

La seguretat i confidencialitat en la salut són dos temes que van de la mà, units intrínsicament a l'atenció de salut que es brinda a la població. En el marc sanitari, la seguretat es relaciona amb la protecció física de la informació, inclosa la protecció contra la pèrdua accidental, així com, contra l'alteració no autoritzada, mentre que la confidencialitat, es relaciona amb la garantia que solament les persones amb responsabilitat per les tasques clíniques tinguin accés a la informació dels pacients

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. Les dades mèdiques d'una persona són altament confidencials, i només poden ser accedides pel propi pacient o per personal mèdic autoritzat.

L'objectiu d'aquest projecte és dissenyar i implementar un gestor d'historial mèdics de pacients a través d'una xarxa de comunicacions que garanteixi aquests punts:

- Privadesa: Només el personal autoritzat pot accedir als historials.
- Integritat: Les dades no poden ser modificades sense autorització.
- No-repudi: Les dades inserides per un metge no poden ser repudiades.
- Autenticitat. S'ha de garantir la identitat del metge que ha inserit les dades.

Paraules clau

- Criptografia
- DTD
- IAIK
- Integritat
- Java
- JDOM
- JUnit
- MySQL
- PKI
- Privadesa
- RMI
- SQL
- UML
- XML

Àrea de PFC

Aquest projecte s'emmarca en l'àrea de Seguretat.

CONTINGUT

LLISTAT DE FIGURES

Introducció

Justificació del PFC i context en el qual es desenvolupa

El passat segle XX es va manifestar per profunds canvis socials i tecnològics, caracteritzats per la integració al desenvolupament humà de noves tecnologies per al tractament de la informació que ha influït amb nous paradigmes en les comunicacions, el coneixement i les relacions socials.

Aquesta anomenada “revolució de la informació” té la seva major expressió en l'ús de xarxes informàtiques com suport necessari per al intercanvi d'informació, que modifiquen els criteris d'espai i temps i amb això globalitzen i agiliten els processos en les diferents esferes de la societat, convertint la informació en el producte més utilitzat, la qual cosa ens duu en el present segle a plantejar-nos una “Societat de la informació” per al intercanvi del coneixement humà.

Els documents, en l'actualitat són considerats com qualsevol objecte material (manuscrits, edicions impreses i digitals, registres en paper, vídeo, disquet, DVD o altre mitjà electrònic, monedes, obres d'art, etc.) que registra o fixa algun coneixement i pugui ser inclòs en una recopilació. Aquells en els quals es consigna la informació dels pacients i les accions que es realitzen en funció de la seva salut, són cridats arxius mèdics, d'ells el mes important és l’historial mèdic del pacient, on es recull directament pel metge tota la informació requerida per a l'atenció del mateix.

Aquests documents, suportats tradicionalment en paper i en alguns llocs, a partir de les últimes dècades del passat segle, en suport electrònic estan sotmesos a normes i procediments per al seu tractament i constituïxen les fonts primàries i secundàries fonamentals per a la presa de decisió en la salut a nivell clínic, operatiu i estratègic.

El caràcter obert i públic de les xarxes de transmissió de dades i les vulnerabilitats existents en els serveis i protocols utilitzats, així com la facilitat del seu ús amb fins il·lícits, tant per intrusos com per usuaris legítims de les mateixes, facilita la realització d'atacs i l'ocurrència d'incidents que afecten la confidencialitat, integritat i disponibilitat de la informació.

La justificació principal d’aquest projecte és la de poder tenir un sistema de gestió d’historial mèdics que sigui remot i segur, que pugi garantir als pacients la confidencialitat, integritat i autenticitat de les seves dades.

Aquest projecte s'emmarca en el context de la implementació d'una eina d'ús remot que es pugui utilitzar en l'espai actual de les comunicacions globals que suposa la xarxa Internet, aconseguint l'aprofitament dels avantatges que suposa l'accés en qualsevol moment a la informació.

Objectius del PFC

L'objectiu d'aquest PFC és dissenyar i implementar un gestor d'historial mèdics de pacients segur a través d'una xarxa de comunicacions que garanteixi aquests punts:

- Privadesa: Només el personal autoritzat (el metge) pot accedir a les dades dels historials mèdics del pacient.
- Integritat: Les dades dels historials mèdics del pacient no poden ser modificades sense autorització.
- No-repudi: Les dades inserides en l'historial mèdic del pacient per un metge no poden ser repudiades.
- Autenticitat. S'ha de garantir la identitat del metge que ha inserit les dades en l'historial mèdic del pacient.

Per aconseguir aquest objectius de seguretat es farà servir les eines que ens proporciona la PKI (*Public Key Infraestructure*): xifrat asimètric i signatura digital. També es dissenyaran uns protocols que permetran desenvolupar les accions més comunes en la gestió de historials mèdics: inserció de noves dades, consulta, etc.

Perquè es pugi realitzar la gestió de les dades remotament, es farà servir el protocol de comunicació RMI (*Remote Method Invocation*) per intercanviar les dades entre el sistema client i el sistema servidor.

Les dades s'enregistraran de manera persistent en una base de dades, per poder consultar i auditar els historials mèdics posteriorment.

També s'ha dissenyat una interfície d'usuari per metges, pacients i administradors per aconseguir una interacció senzilla i còmoda amb l'aplicatiu.

Enfocament i mètode seguit.

El projecte s'ha dividit en diferents fases per tal de fer una realitzar una implementació incremental, es a dir, en cada fase s'implementaran la funcionalitat corresponent, es faran les proves necessàries, es documentarà i s'integrarà amb la fase anterior. Amb aquesta divisió en subprojectes, a cada fase es fixen uns objectius parcials a acomplir que un cop completats són el punt de partida de la següent fase. D'aquesta manera s'aconsegueix reduir la complexitat global del sistema, en parts mes petites i mes clares, facilitant les tasques d'implementació.

El projecte consta de set fases:

1 Instal·lació del programari.

2 Creació d'una PKI.

3 Esquema criptogràfic.

4 Representació de les dades (XML).

5 Comunicació de components (RMI).

6 Gestió de la informació (base de dades).

7 Interfícies d'usuari: client i servidor.

Les proves i la documentació del projecte s'han realitzat *ad-hoc* durant el desenvolupament de cadascuna de les fases.

Planificació del projecte

La primera tasca realitzada en aquest projecte ha sigut definir una planificació inicial. Aquesta planificació ha consistit en la divisió del projecte en les diferents etapes seguint el cicle de vida clàssic, concretant les diferents subtasques de cada etapa. Es van establir uns punts de control del treball realitzat al final de cada etapa, per a permetre dur una anàlisi exhaustiva sobre l'estat del projecte en un moment determinat i així poder determinar si es compleixen les previsions tant a nivell funcional com temporal.

Durant aquesta planificació inicial també es van fer els càlculs corresponents de temps per a la finalització del projecte (*Figura Intro-1*), així com els recursos necessaris per al seu desenvolupament.

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	Inici	0 días	mié 20/09/06	mié 20/09/06	
2	**Planificiació inicial del projecte**	1 día	mié 20/09/06	mié 20/09/06	1
3	**Anàlisi de requeriment**	**3 días**	**jue 21/09/06**	**sáb 23/09/06**	**2**
4	Lectura exhaustiva de l'enunciat	1,5 días	jue 21/09/06	vie 22/09/06	
5	Revisió de Junit	0,5 días	vie 22/09/06	vie 22/09/06	4
6	**Determinació dels requeriments**	**1 día**	**sáb 23/09/06**	**sáb 23/09/06**	**5**
7	Requeriments funcionals	0,5 días	sáb 23/09/06	sáb 23/09/06	
8	Requeriments no funcionals	0,5 días	sáb 23/09/06	sáb 23/09/06	7
9	Revisió dels requisits del sistema	0 días	sáb 23/09/06	sáb 23/09/06	6
10	**Instalació programari**	**1 día**	**dom 24/09/06**	**dom 24/09/06**	**3**
11	Instalació IAIK	0,5 días	dom 24/09/06	dom 24/09/06	
12	Instalació PKI	0,5 días	dom 24/09/06	dom 24/09/06	11
13	**Esquema criptogràfic**	**27 días**	**lun 25/09/06**	**dom 22/10/06**	**10**
14	Especificació/Disseny	7 días	lun 25/09/06	dom 01/10/06	
15	Implementació	16 días	lun 02/10/06	mié 18/10/06	14
16	Tests unitaris	4 días	jue 19/10/06	dom 22/10/06	15
17	Revisió del esquema criptogràfic	0 días	dom 22/10/06	dom 22/10/06	16
18	**Representació de les dades**	**13 días**	**lun 23/10/06**	**dom 05/11/06**	**13**
19	Disseny XML	3 días	lun 23/10/06	mié 25/10/06	
20	Implementació	8 días	jue 26/10/06	vie 03/11/06	19
21	Test Unitaris	2 días	sáb 04/11/06	dom 05/11/06	20
22	Revisió de la representació de les dades	0 días	dom 05/11/06	dom 05/11/06	21
23	**Comunicació dels components**	**14 días**	**lun 06/11/06**	**dom 19/11/06**	**18**
24	Especificació/Disseny	3 días	lun 06/11/06	mié 08/11/06	
25	Implementació	9 días	jue 09/11/06	vie 17/11/06	24
26	Test Unitaris	2 días	sáb 18/11/06	dom 19/11/06	25
27	Revisió de la comunicació del components	0 días	dom 19/11/06	dom 19/11/06	26
28	**Gestió de la informació**	**14 días**	**lun 20/11/06**	**dom 03/12/06**	**23**
29	Instal·lacio MySQL	0,5 días	lun 20/11/06	lun 20/11/06	
30	Disseny BD	3,5 días	lun 20/11/06	jue 23/11/06	29
31	Implementació	8 días	vie 24/11/06	vie 01/12/06	30
32	Test Unitaris	2 días	sáb 02/12/06	dom 03/12/06	31
33	Revisió de la gestió de la informació	0 días	dom 03/12/06	dom 03/12/06	32
34	**Interficies**	**24 días**	**lun 04/12/06**	**dom 31/12/06**	**32**
35	Vista client	12 días	lun 04/12/06	dom 17/12/06	
36	Vista gestor	12 días	lun 18/12/06	dom 31/12/06	35
37	Revisió de les intefícies	0 días	dom 31/12/06	dom 31/12/06	36
38	**Documentació**	**6 días**	**mar 02/01/07**	**dom 07/01/07**	**34**
39	Revisió documentació	4 días	mar 02/01/07	vie 05/01/07	
40	Presentació	2 días	sáb 06/01/07	dom 07/01/07	39
41	**Lliurament PFC**	1 día	lun 08/01/07	lun 08/01/07	38
42	Fi del projecte	0 días	lun 08/01/07	lun 08/01/07	41

Figura Intro-1. Planificació inicial del projecte.

Com es veurà a *Seguiment del projecte*, pàg. 26, aquesta planificació no va poder ser seguida al peu de la lletra. Malgrat això, la desviació temporal tampoc ha estat massa important i en cap moment s'han hagut de retallar funcionalitats importants per a poder complir la data de lliurament del projecte.

Una vegada acabades aquestes fases de forma satisfactòria podem donar per finalitzat el projecte. Cal recordar que aquest projecte no abasta un futur manteniment de l'aplicació.

Representació de l'escenari

L'arquitectura de l'aplicació és una arquitectura client-servidor, on els clients són els metges i els pacients i els servidors són els gestors. Els clients fan peticions com, per exemple, demanar un historial, modificar un historial, etc. al gestor, el qual accepta les peticions i recupera/enregistra a la base de dades la informació corresponent.

A la *Figura Intro-2* es representa gràficament l'escenari de com funciona tot el sistema de gestió dels historials mèdics.

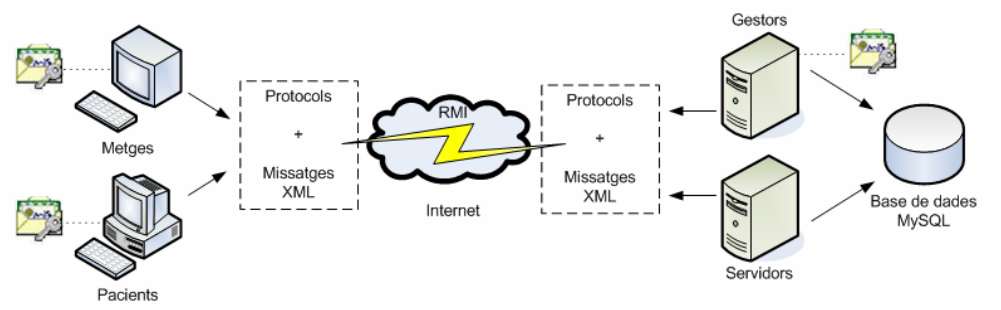

Figura Intro-2. Representació de l'escenari

Els elements que intervenen en aquest escenari són :

- ***Smart-cards***: Targetes intel·ligents que permeten l'accés segur al sistema. Per simplificar el projecte, es faran servir PKCS#12s en comptes de *smart-cards*.

  Seguint aquest model les claus privades no viatgen mai per la xarxa, i són sempre a la màquina local.

- **Metges/Pacients**: Ordinadors on s'executa l'aplicació client de gestió de historials.

  El programari client no té accés directe a la base de dades si no és mitjançant el servidor de gestió de historials.

- **Gestor**: Ordinador que fa de servidor on s'executa l'aplicació servidor de gestió de historials. Té un servidor RMI que permet les comunicacions amb els clients.

- **Servidor de base de dades**: Ordinador que gestiona la base de dades de l'aplicació de gestió de historials

La comunicació entre Metges/Pacients i el Gestor es fa seguint els protocols definits a l'esquema criptogràfic. El protocol de comunicació utilitzat per executar crides remotes és el RMI.

Productes obtinguts

Un cop desenvolupat, el sistema consta dels següents productes:

- Infraestructura de clau pública (PKI) que permet signar i xifrar dades i identificar als usuaris i al gestor.
- Aplicatiu per administrador: Aplicatiu que permet gestionar els usuaris i el servidor del gestor.
- Aplicatiu per pacients: Aplicatiu que permet als pacients consultar el seu propi historial mèdic.
- Aplicatiu per metges: Aplicatiu que permet als metges gestionar els historials mèdics dels pacients que tenen assignats.
- Base de dades: On s'enregistren les dades mèdiques i de la qual podem recuperar dades i en un futur fet tota mena de minera de dades.

A més em obtingut tot un sèrie de protocols i de llibreries de classes que permetran en un futur una fàcil ampliació de les funcionalitats del sistema.

Breu descripció dels altres capítols de la memòria

En els següents capítols de la memòria s'expliquen els conceptes, dissenys i implementacions duts a terme a cada fase. A continuació es fa una breu descripció del contingut d'aquest capítols inclosos en la memòria:

Introducció a la infraestructura de clau pública (PKI)

Capítol introductori a la criptografia i a la infraestructura de clau pública.

Estructura del projecte

Estructuració que s'ha fet durant el projecte i el resultat del seguiment realitzat.

Esquema criptogràfic

Disseny dels diferents protocols criptogràfics i de com es faran servir en l'aplicaciu. El conjunt de protocols és el que s'anomenem esquema criptogràfic.

Disseny de l'aplicació

Disseny UML de l'arquitectura amb la que es basarà el sistema.

Representació de les dades: XML

S'explica el format que de les dades de les visites i els historial mèdics. S'expliquen les decisions de disseny i com s'ha implementat mitjançant Java i JDOM.

Comunicació dels components: RMI

S'introdueixen els conceptes de comunicació RMI i com s'han aplicat al disseny del sistema gestor d'historial mèdics.

Gestió de la informació: base de dades

Es dóna el model per defineix la estructura de la base de dades. S'expliquen les decisions de disseny i com s'ha implementat la base de dades mitjançant MySQL i Java.

Interfície d'usuari

Aquest capítol explica com s'ha dissenyat la interfície d'usuari. També s'explica el funcionament dels aplicatius per metges i pacients i de l'aplicatiu per l'administrador del sistema.

Jocs de proves

En aquest capítol es detalla la metodologia que s'ha fet servir per realitzar les proves del sistema.

Valoració econòmica

Resum de la valoració econòmica del projecte.

Treball futur

En aquest capítol s'expliquen les millores que no s'han acabat implementant perquè no estaven en els objectius d'aquest projecte, però que podrien ser interesant implementar-les en un futur per millorar les funcionalitats i característiques del sistema actual.

Conclusions

Capítol on es fa balanç dels objectius acomplerts, i s'expliquen les consideracions mes importants en el desenvolupament del projecte

Bibliografia i annexos

En la bibliografia hi ha les referències a tota la documentació que s'ha consultat durant el projecte.

En els annexos s'inclou una referència sobre com instal·lar i posar en marxa el sistema.

CAPÍTOL 1

Introducció a la infraestructura de clau pública (PKI)

Quan parlem de criptografia aplicada, hem de tenir clar quin és el nostre objectiu. El xifrat de les dades (o encriptació) és potser la funcionalitat més coneguda de la criptografia. No obstant això ens hem de preguntar, per què volem xifrar? Quines prestacions disposarà una aplicació que disposi de xifrat enfront d'una altra que no disposi?

Si contestéssim immediatament aquestes preguntes, segurament ens deixaríem de plantejar els problemes des d'una perspectiva més àmplia, deixant de banda alguns aspectes tan o més importants que el propi xifrat.

Per això a continuació es discutiran quines són les amenaces que poden sofrir les dades que usen les aplicacions o quines són les amenaces que poden sofrir les transmissions, per a deixar per a més endavant quines són les prestacions que ens ofereix la criptografia.

Amenaces

Podem agrupar les amenaces en quatre grups:

- Espionatge o robatori: És el fet de copiar unes dades confidencials. L'espionatge o el robatori és indetectable, en el sentit que les dades no queden manipulats de cap manera, així que el propietari de les dades no es pot adonar.
- Manipulació o modificació: És el fet de modificar (eliminar, ampliar o substituir) el contingut de les dades. El propietari de les dades podria adonar-se del fet si disposés d'una copia total o resumida de les dades en un magatzem alternatiu no manipulat.
- Falsificació o manipulació d'autoria: És el fet de manipular les proves d'autoria d'una informació. En cas de que les dades disposin d'indicadors sobre qui va ser el seu autor, una falsificació implica la manipulació voluntària d'aquestes proves. Usualment aquesta amenaça s'aplica en dades transmeses, però també pot afectar a dades emmagatzemades.

- Repudi d'autoria: És la negació de la redacció de les dades. Aquesta amenaça la sofreix el receptor de les dades quan el creador nega que els ha fet. Usualment aquesta amenaça s'aplica en dades transmeses.

Mesures

Perquè aquestes amenaces no tinguin efecte, fa mancada prendre les mesures apropiades. Les mesures poden ser físiques o lògiques.

Les mesures físiques poden consistir a protegir les dades, tancant-les dintre d'algun tipus de "caixa forta".

No obstant això, les mesures que ens interessen són les lògiques. La criptografia ens ofereix un ventall d'algorismes que, combinats entre si, ens permeten una protecció efectiva contra els diferents tipus d'amenaces. Les tres eines criptogràfiques més comunes són el xifrat, que ens permetrà protegir les dades contra l'espionatge, el resum o *hash*, que ens permetrà protegir la integritat de les dades contra manipulacions i la signatura, que ens permet verificar l'autoria de les dades protegint-los contra la falsificació i el repudi d'autoria.

Les eines que ens posa a l'abastament la criptografia estan dividides en dos conjunts:

- criptografia de clau simètrica,
- criptografia de clau pública (o criptografia asimètrica).

Criptografia de clau simètrica

La criptografia de clau simètrica engloba una sèrie d'algorismes que ens permeten processar dades (cadenes de bits) i extreure un resultat (altra cadena de bits). L'operació bàsica tindria una descripció com aquesta:

```
R = f( M )
```

on *M* serien les dades, cadena de bits o missatge, *f( )* seria la funció criptogràfica i *R* seria la seqüència resultant.

Les funcions *f( )* criptogràfiques que contenen tota la informació original (funcions de xifrat) donen seqüències *R* resultants de la mateixa longitud que *M*, mentre que les funcions que generen resultats que no contenen la informació original (com les funcions de resum o *hash*) tenen un resultat *R* de longitud fixa (usualment pocs bytes).

Xifrat

Durant segles s'ha usat la criptografia per a protegir la informació confidencial. En temps dels romans, els missatges s'enviaven lacrats i xifrats, amb un algorisme que avui es coneix com algorisme del Cèsar.

Aquest algorisme consistia en una substitució lletra per lletra de tot el missatge per altres lletres segons indicava una taula de substitució. Aquesta taula de substitució es construïa simplement agafant totes les lletres de l'alfabet llatí i desplaçant-les *k* unitats.

L'algorisme del Cèsar servirà per a introduir diferents conceptes comuns als algorismes de xifrat actuals.

Per a començar, la funció de processat del missatge es pot expressar d'una manera més especifica com:

```
R = f( M ) = e( k, M )
```

La funció *e( )* o funció de xifrat és parametrizable amb *k* que és el factor de rotació de la taula. En general, tots els algorismes de xifrat disposen del paràmetre *k*, de manera que podem tenir diferents funcions de xifrat *f( )* sense haver de canviar l'algorisme, solament s'ha de canviar el paràmetre. Aquest paràmetre es diu clau de xifrat.

Perquè una algorisme de xifrat *e( )* sigui útil, fa falta que existeixi una funció inversa *d( )* o funció de desxifrat que compleixi:

```
M = d( x, R )
```

ò

```
M = d( x, e( k, M ) )
```

En criptografia simètrica, és necessari que *k* = *x*, o el que és equivalent que *x* sigui fàcilment inferible a partir de *k*. Per tant:

```
M = d( k, e( k, M ) )
```

En el cas de l'algorisme del Cèsar, l'algorisme de desxifrat seria idèntic al de xifrat però aplicant la *k* en sentit invers *(-k)*.

Hi ha un punt important a considerar: les condicions d'aplicació de l'algorisme de xifrat. Si utilitzem un algorisme de xifrat és per a protegir la informació d'ulls estranys. Això vol dir que la informació estarà en algun moment viatjant en un entorn hostil o canal insegur (per exemple Internet).

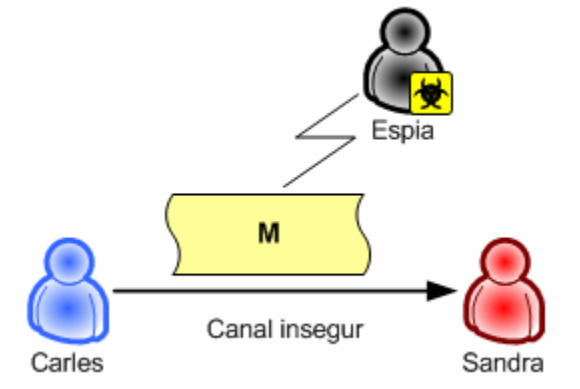

Figura 1-1. Enviament de un missatge per un canal insegur

Si Carles envia el missatge *M* a Sandra mitjançant el canal insegur, ni Carles ni Sandra tenen cap garantia de qualsevol espia pugui llegir, copiar o manipular el missatge *M*. No obstant això, aquest atac no és cap problema si enviem el missatge *M* xifrat amb una clau *k*:

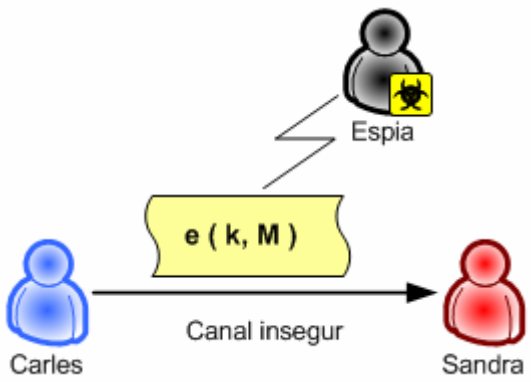

Figura 1-2. Enviament de un missatge xifrat per un canal insegur

Carles i Sandra han de compartir la mateixa clau *k*. Però per fer-lo fa falta que algun dels dos la inventi i la comuniqui a l'altre. Aquesta comunicació inicial no pot fer-se a través del canal insegur, si no que s'ha de fer a través d'un canal segur, que només sigui accessible per Carles i Sandra i per ningú més.

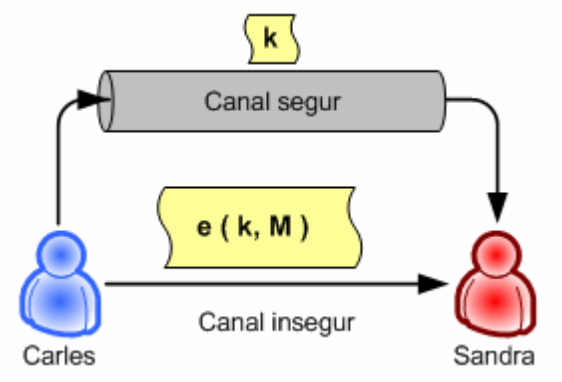

Figura 1-3. Enviament de la clau por un canal segur

Exemples de canal segur poden ser, una conversa (cara a cara o telefònica), un sobre postal, etc. Dependrà del nivell de seguretat que vulguem donar al sistema.

Tipus de xifrat

El xifrat perfecte o xifrat de Vernam és l'únic considerat com insuperable, malgrat que no és pràctic. El funcionament és trivial: solament fa mancada fer una or-exclusiva de la clau *k* sobre el missatge *M*:

$$R = e(k, M) = k \oplus M$$

Per a desxifrar solament es necessari reutilitzar la *k* i tornar a fer el mateix, però ara sobre la seqüència xifrada:

```
M = d( k, M ) = k ⊕ R
```

El problema és que la clau ha de tenir la mateixa longitud que el missatge, i a més, fa falta que, per a cada missatge transmès, la clau sigui completament diferent. Aquests dos fets fan que el xifrat de Vernam no sigui pràctic, ja que a priori els interlocutors s'han d'intercanviar la clau d'una manera segura, però si la longitud de la clau és igual que del missatge per què no s'envien el missatge per aquest canal segur?

Per a crear un algorisme de xifrat pràctic ens podem basar en el de Vernam. Usant algorismes generadors de seqüències seudo-aleatòries, podem aconseguir emular el xifrat de Vernam, però sense haver d'intercanviar una clau k tan gran. Tot algorisme generador de seqüències seudo-aleatòries, necessita una entrada o llavor que li permeti generar seqüències aleatòries diferents:

```
seq-aleat = genAleat( llavor, longitud)
```

Els usuaris només necessiten compartir la llavor, per a generar la mateixa seqüència aleatòria (suposant que tinguin el mateix algorisme generats de seqüències seudo-aleatòries).

```
R = i( *genAleat( k, lM ), M )
```

on , *lM* és la longitud del missatge *M*.

Els algorismes de xifrat que usen aquesta tècnica es diuen algorismes de xifrat de flux. Un dels més usats és el RC4 (*Rivest Cipher 4*).

El segon gran grup d'algorismes de xifrat són els algorismes de xifrat en bloc. Aquests algorismes treballen sobre grups de bits d'entrada (usualment 64 o 128 bits per bloc), traient un bloc de sortida de la mateixa mesura. Els bits de bloc de sortida són una combinació d'operacions booleanes i desplaçaments de registre en funció de la clau de xifrat.

L'algorisme de xifrat en bloc més usat és el DES (*Data Encryption Standard*). El DES usa claus de 56 bits i blocs de 64 bits. La longitud de la clau del DES és el seu únic problema. 256 bits està en el límit del trencable per força bruta amb la tecnologia actual. No és recomanable crear un nou programari que usi DES. Això no significa que el DES estigui en desús. Encadenant el DES tres vegades s'aconsegueix un algorisme igual de resistent a la criptoanàlisi com el DES però amb una clau de 168 bits (3*56). És l'anomenat Triple-DES o 3DES.

Resum o *hash*

El resum o *hash* ens permet obtenir una seqüència de longitud finita (normalment curta) de bits que en principi és única per a cada missatge *M*.

```
H = h( M )
```

La funció de *hash h( )* no és parametrizable, per tant de cada missatge M només es pot obtenir una única seqüència *H* o *hash* del missatge. El *hash* és com l'empremta digital del missatge M. Cada missatge diferent, té un *hash* diferent. Addicionalment,

no existeix la funció inversa. Per tant, no hi ha forma d'extreure el missatge original a partir del *hash*.

L'ús habitual del *hash* és realitzar comprovacions d'integritat de les dades. Si calculem el *hash* d'unes dades en un lloc a part (posem un paper), és possible poder verificar posteriorment que les dades no han estat modificats, recalculant el *hash* i comparant el resultat amb el *hash* prèviament calculat (el del paper).

$$H = h(M) \neq H' = h(M')$$

$$\forall M' \neq M$$

La funció *h( )* ha de retornar resultats petits. Si la grandària del *hash* fora de l'ordre de la mida del missatge *M*, no tindria molt sentit usar-lo, ja que una còpia del missatge seria una prova suficient de la seva integritat i la mida d'aquesta còpia seria òbviament del mateix ordre que el missatge (això sí, copiar-lo en un paper potser sigui poc pràctic).

No obstant això, el fet que el *hash* tingui una longitud fixa, fa que el nombre de *hashes* resultants de *h( )* sigui finit. En canvi, el nombre de missatges sobre els quals es pot aplicar la funció de *hash* és infinit, per tant, és obvi que haurà missatges que comparteixin el mateix *hash*. Les funcions de *hash* criptogràfiques fan que la probabilitat de trobar un missatge alternatiu *M'* a un missatge *M* atès que doni el mateix *hash H* sigui molt baixa.

Tipus de *hash*

Els algoritmes de *hash* més coneguts son el MD5 i el SHA1. El MD5, o *Message Digest #5*, genera una seqüència de 128 bits a partir del missatge original. El SHA1, o *Secure Hash Algorithm #1*, produeix un *hash* de 160 bits. Tots dos apliquen funcions booleanes en blocs de 512 bits cada cop. Aquestes operacions, comparades amb els algoritmes de xifrat en bloc, no depenen de cap clau, només dels bloc de 512 bits dels que es composa el missatge.

Criptografia de clau pública

El principal problema del xifrat amb criptografia simètrica és que fa falta un canal segur per a poder intercanviar la clau. Si a més el nombre d'usuaris augmenta, fa falta distribuir la clau de manera segura entre tots ells, amb el perill que algun dels intercanvis la clau quedi exposada a terceres persones.

A més del problema de la integritat i l'autenticitat, fins i tot implementats amb *hashes* criptogràfics, les claus són difícils de compartir amb uns altres, sobre tot si és un grup nombrós de gent.

La diferència bàsica de la criptografia de clau pública o criptografia asimètrica, respecte a la criptografia simètrica, es basa en que la primera usa dues claus diferents per al procés de xifrat i desxifrat, i la segon utilitza sempre la mateixa.

- Clau simètrica:

$$R = e(k, M) \quad \Leftrightarrow \quad M = d(k, R)$$

- Clau asimètrica:

```
R = e( k1, M )  ⇔   M = d( k2, R )
```

Xifrat

Les claus *k1* i *k2* es generen com un parell de manera simultània, en un procés que es coneix com generació del parell de claus. Aquest procés és realitzat pel propietari de les claus, i és aproximadament anàleg al d'escollir la clau de xifrat en criptografia simètrica.

La clau *k1* la usarem per a xifrar dades, i la clau *k2* per a desxifrar-los. Si el propietari publica la clau *k1*, i manté en secret la *k2*, tenim un mecanisme ideal per a rebre informació xifrada de qualsevol, sense haver de compartir cap secret.

La publicació de clau pública no compromet en cap moment la privadesa de la clau privada, ja que no és viable trobar la segona a partir de la primera.

A causa del diferent ús de les claus a cada parella la hi crida, clau pública (*kpub*), i clau privada (*kprv*).

El xifrat queda com:

```
R = e( kpub, M )
```

i el desxifrat com:

```
M = d( kprv, R )
```

És obvi que la destrucció i el robatori de la clau privada son els perill més grans del sistema. La destrucció faria impossible el desxifrat dels missatges, i el robatori que l'il·lícit nou propietari de la clau pogués desxifrar la nostra informació personal.

Signatura

Si apliquem el procés anterior però intercanviant les claus:

```
R = e( kprv, M )

M = d( kpub, R )
```

El que obtenim, a primera vista, és una situació absurda, qualsevol pot desxifrar *R*. Però si parem atenció, veurem que tot el món sabrà que solament nosaltres podem haver-lo signat. Aquest procés es coneix com signatura digital.

La signatura digital ens garanteix que el missatge no ha estat manipulat després de ser signat, és a dir garantim la integritat del missatge, i que ha estat generat pel propietari de la clau, és a dir, hem garantir l'autenticitat del missatge.

Xifrat i signatura pràctics

La criptografia de clau pública usualment no s'utilitza sola, degut al fet que la velocitat de procés és molt baixa en comparació de la criptografia simètrica. En la

pràctica s'utilitzen sempre els dos tipus de criptografia de manera combinada per a millorar les prestacions de velocitat de procés, però sense perdre cap de les virtuts.

Per a millorar la velocitat de xifrat, el procés es realitza en dos passos:

1 Es genera una clau simètrica *k*, de manera aleatòria (clau de sessió). Amb ella es xifra el missatge *M*:

```
R1 = esim( k, M )
```

2 Es xifra la clau de sessió amb la clau pública del destinatari:

```
R2 = easim( kpub, k )
```

El que s'envia és la concatenació de *R1* i *R2*. El receptor simplement ha de desxifrar la clau de sessió, i amb ella la resta del missatge. A més, aquest mecanisme té un efecte lateral beneficiós: per a enviar dades xifrades per a múltiples destinataris, no fa falta enviar blocs independents xifrats amb cadascuna de les claus, només es necessari xifrar les dades amb una única clau de sessió i després xifrar la clau de sessió independentment per a cada destinatari.

El procés de signat també es fa en dos passos:

1 S'extreu el *hash* del missatge a signar:

```
h = h( M )
```

2 Se signa el *hash*:

```
R = easim( kprv, h )
```

El que s'envia en aquest cas és la concatenació de dos elements: el missatge *M*, i la signatura *R* (no fa falta el *hash h* ja que el receptor el pot tornar a calcular).

En recepció, calculem *h* a partir de *M*:

```
h1 = h( M )
```

i sobre la signatura *R* rebuda apliquem la clau pública del signant:

```
h2 = easim( kpub, R )
```

Totes dues versions del *hash* han de ser iguals:

```
h1 = h2 = h
```

En cas contrari, o bé el missatge *M* ha estat manipulat, o bé la signatura va ser realitzada per algun altre.

Tipus d'algorismes de clau pública

Si el concepte de criptografia simètrica es coneix des de temps dels romans, el de criptografia asimètrica no apareix fins a finals del segle XX. Whitfield Diffie i Martin Hellman van publicar la idea i des de llavors han aparegut diferents implementacions. La matemàtica que suporten tots ells es basa en la solució de problemes complexos, com la factorització de grans nombre o la matemàtica discreta. Fins a ara, tots els algorismes inventats són massa lents per a ser usats

per a xifrar o signar les dades directament, de manera que cal aplicar-los en dos passos tal com s'ha explicat anteriorment.

Els algorismes de clau pública més utilitzats són:

- **RSA** (Rivest, Shamir, Adelm): és l'algorisme més usat dels existents Ron Rivest, Adi Shamir i Leonard Adelman van dissenyar un algorisme basat en la dificultat de factoritzar nombres molt grans. Major longitud, major seguretat. En l'actualitat les claus que s'usen són de més de 512 bits (ja considerades com insegures), sent les de 1024 bits considerades amb una seguretat suficient.
- **DSA** (*Digital Signature Algorithm*): Algorisme patrocinat pel govern dels Estats Units aplicable a la signatura digital (no serveix per a xifrar). Es basa també en la dificultat de càlcul de logaritmes discrets. La mesura de les claus usades és equivalent a les de RSA.
- **Diffie-Hellman**: Algorisme per a l'intercanvi de claus simètriques (és a dir, solament serveix per al xifrat). És el primer algorisme de clau pública (1976).
- **Corbes el·líptiques**: Permet implementar tant el xifrat com la signatura. Encara que va ser proposada a mitjans dels anys vuitanta, les aplicacions pràctiques van aparèixer en els noranta. La longitud de les claus és molt inferior a les de RSA per a una seguretat equivalent, i la velocitat de procés és major. És previsible que tinguin un ús important en els sistemes portàtils (terminals mòbils com telèfons o PDA).

Infraestructura de clau pública (PKI)

La criptografia de clau pública permet evitar el fet de compartir les claus secretes, i compartir informació confidencial entre grups grans de gent. Fins i tot ens permet incorporar els algorismes de control d'integritat i autenticitat.

Però, com podem verificar la identitat dels participants en una comunicació? Com podem estar segurs de que les claus públiques pertanyen a qui ens sembla que pertanyen? Aquests punts són fonamentals si volem aprofitar les capacitats de la criptografia asimètrica comentades en el paràgraf anterior

La Infraestructura de Clau Pública PKI (*Public Key Infrastucture*) és una combinació de procediments, algorismes i estructures de dades que ens permeten assegurar la identitat dels participants d'un intercanvi de dades usant criptografia.

Per a comprovar l'autoria d'un missatge no és suficient amb verificar la signatura digital amb la corresponent clau pública. Fa falta “lligar” aquesta clau pública a dades que ens permetin identificar el seu titular. La manera més fàcil de fer-lo és que algú ens signi el conjunt en una única estructura de dades: el certificat.

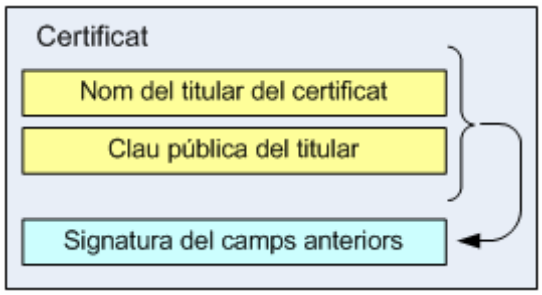

Figura 1-4. Estructura de un certificat.

La signatura dels dos primers camps assegura la integritat contra manipulacions indesitjables. Això fa, que quan rebem un missatge signat podrem saber el nom de qui ho ha signat, i quan vulguem enviar dades xifrades, sabrem quina és la clau del destinatari de manera fiable.

Ara bé, qui ha de signar el certificat? Està clar que el titular del certificat no, ja que així qualsevol podria autosignar certificats amb noms arbitraris.

Autoritat de Certificació (CA)

Fa falta que algú s'encarregui de signar aquests documents, i que aquest algú mereixi la confiança de tots els participants. A aquest algú li anomenem Autoritat de Certificació o CA (acrònim de *Certification Authority*).

La confiança en la CA vindrà donada per dos fets:

1. La CA segueix uns procediments de verificació de les dades del titular suficients, i comprova que la seva clau pública és realment seva
2. La clau privada de firma de la CA (o clau privada de certificació) es manté sota unes mesures de seguretat suficients.

El compliment del segon punt varia àmpliament, i ens podem trobar amb CAs que tinguin la clau en un fitxer per a un funcionament online, fins a una altre que té la clau privada protegides amb claus de pas (criptografia simètrica), dividida en trossos, repartit en diferents operadors, etc.

El primer punt és, no obstant, més homogèniament implementat. La comprovació que la clau és pròpia es fa sempre automàticament, només fa falta que el titular demandant de certificació signi una petita seqüència de dades i l'enviï. Aquesta seqüència de dades, que normalment conté el nom del titular, es diu petició de certificació. La verificació de les dades d'afiliació del titular s'han de comprovar amb la presentació de documentació acreditativa, i moltes vegades es fa mitjançant personal dedicat.

Certificat X.509 v1

El format del certificat ve especificat per la norma ITU-T X.509. Aquest estàndard especifica una estructura (SEQUENCE) en ASN.1 formada pels següents camps:

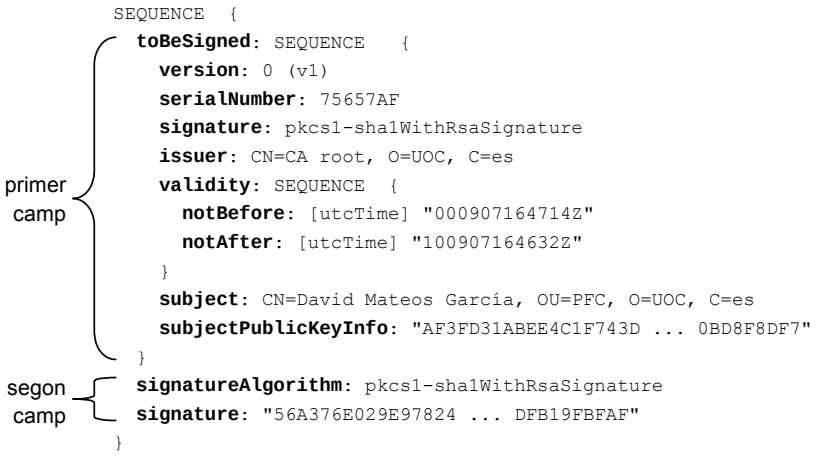

Figura 1-5. Notació ASN.1 de un certificat X.509 v1

El segon camp és la signatura del primer. L'autor de la signatura (el propietari de la clau privada) és la CA emissora del certificat.

Dintre del primer camp (`toBeSigned`) trobem tota la informació necessària per a identificar el titular del certificat (camp `subject`), i el seu emissor (camp `issuer`). Ambdós camps són del tipus Nom Distintiu o DN (*Distinguished Name*) que a part de contenir el nom del titular, conté camps addicionals com l'organització a la qual pertany o el país.

Els certificats emesos per una mateixa CA poden tenir el mateix DN en el camp `subject`. No obstant això, usualment, els certificats amb un mateix DN pertanyen a un mateix titular.

Addicionalment, l'emissor qualifica el certificat amb un nombre de sèrie (camp `serialNumber`). En una determinada CA, només fa falta el nombre de sèrie per a identificar un certificat emès per ella. El camp `signature` (dintre de `toBeSigned`) i el camp `signatureAlgorithm` especifiquen el format de la signatura del certificat (el segon camp de la seqüència principal). Usualment la signatura és una RSA sobre un *hash* SHA1 o MD5. El camp `version` pot valer 0 (versió 1) o 2 (versió 3), la versió 2 gairebé s'ha fet servir. El camp `subjectPublicKeyInfo` conté la clau pública del titular del certificat.

Finalment, mereix un comentari a part el camp `validity`. És una seqüència de dues dates que delimiten el període de validesa del certificat (normalment entre un i quatre anys). Si usem un certificat per a xifrar informació al seu titular, solament ho podem fer si el certificat està dintre del seu període de validesa, és a dir, si el certificat no ha caducat. Més delicat és quan el certificat s'usa per a verificar una signatura. Perquè la signatura sigui vàlida fa falta que la signatura s'hagi realitzat durant el període de validesa del certificat i és necessari que en el moment de la verificació de la signatura, el certificat segueixi estant dintre del seu període de validesa. Aquest fet, implica que la informació signada ha de ser re-signada periòdicament cada vegada que el certificat caduqui.

Llavors, per què caduquen els certificats? no seria més fàcil eliminar el període de validesa? La raó per la qual els certificats caduquen, i que s'hagin d'anar renovant periòdicament és per a facilitar la seva gestió. El certificat és la prova d'identitat del titular, i ve fortament lligada amb el secret amb el qual el titular manté la clau privada

associada en secret. Durant la vida del certificat (el seu període de validesa), aquest pot deixar de ser vàlid (deixem de treballar en l'empresa, canviem de departament o la clau privada potser a deixat de ser secreta). Per tant, cada CA sap que una tant per cent dels certificats emesos s'han de treure de circulació per que algun dels seus camps ha deixat de ser vàlid. En aquest cas diem que el certificat s'ha de revocar.

La revocació s'ha de fer pública en el mateix moment que aquesta es produeixi. La publicació dels certificats revocats és treball que també fa la CA, emetent llistes negres, o llistes de certificats revocats o CRL (*Certificate Revokation List*). Aquestes llistes contenen tots els certificats revocats no caducats en el moment d'emissió de la CRL. La CRL també està signada per la CA, i té un format en ASN.1.

Gairebé hem respost a la pregunta de per què caduquen els certificats. Si no caduquessin, les CRLs creixerien indefinidament. A mesura que caduquen els certificats, anem buidant les CRLs, i vam evitar que aquestes llistes siguin massa grans. Si una vegada caducat un certificat, aquest no ha estat revocat, el titular ha de procedir a una petició de renovació. Usualment, la renovació és l'emissió d'un certificat amb el mateix camp `subject`, amb una clau pública igual o diferent, un període de validesa de similar durada, i un nou nombre de sèrie..

Certificat X.509 v3

En l'actualitat pràcticament tots els certificats emesos són versió 3. Si s'ha dedicat un apartat als certificats versió 1, és perquè els versió 3 són gairebé idèntics. Únicament la seqüència `toBeSigned` té tres camps addicionals. Els dos primers, `issuerUid` i `subjectUid`, van ser incorporats en la versió 2. Teòricament servien per a posar informació addicional, però aquesta no quedava bé identificada, i això feia que els certificats X.509 v2 fossin inintel·ligibles entre sistemes diferents. En l'actualitat, aquests camps estan gairebé sempre buits.

Aquest problema es va resoldre en la versió 3 amb el camp `extensions`, que és una seqüència de longitud arbitrària que conté elements de tipus `extension`. La millora, respecta a `issuerUid` i `subjectUid` és que cada extensió va inequívocament identificada amb un *oid* (*Object Identifier*, identificador d'objecte) únic. Cada extensió és una seqüència de tres camps:

- `extnId`: *oid* que identifica què es aquella extensió.
- `critical`: Booleà que diu si la extensió es de compressió ineludible o no. Es a dir, si la extensió està marcada com crítica, qui processi el certificat ha d'entendre-la, i si no rebutjar el processat (si el procés es una signatura, la signatura no es vàlida). Si la extensió es no-crítica, el processat de la extensió es opcional.
- `extValue`: es un `OctetString` que encapsula la codificació en DER del ASN.1 del valor de la extensió.

Hi ha moltes extensions. A més cada fabricant les va creant per als seus propòsits particulars (Microsoft i Netscape tenen vàries per a distingir si el certificat és de signatura o xifrat, per a e-mail, per a autenticació, TLS, etc.). Algunes de les més comunes són:

- `subjectAltName`: Nom alternatiu del titular. Usualment és una adreça web o de correu, i s'usa en servidors de web i CAs. És una alternativa al camp `subject` del certificat, encara que aquesta extensió mai té precedència sobre aquest camp.
- `keyUsage` (crítica): Es un camp de bits que indica si el certificat es de firma, xifrat, etc.
- `extendedKeyUsage` (crítica): Similar a l'anterior, però aquí es tracta d'una seqüència de *oid* de manera que és extensible. Els valors més comuns són `serverAuth`, `clientAuth`, `codeSigning`, `emailProtection`, `timeStamping`,...
- `cRLDistributionPoints` (crítica): Indica el punt (o els punts) on hem d'anar a buscar la CRL per a comprovar si el certificat ha estat revocat.

Autoritats de certificació y jerarquies de certificació

Fins a ara s'ha parlat fonamentalment dels certificats d'usuari. Però també les CAs tenen claus per a signar els certificats o les CRLs, i aquestes claus han de ser tan fiables com les dels usuaris. Per això les claus públiques de les CAs també s'haurien de publicar en certificats i el format serà el mateix que per als usuaris: el X.509

Certificats de CA

El certificat de la CA, si és versió 1, és pràcticament idèntic al d'un usuari. L'emissor (camp `issuer`) i el titular (camp `subject`) d'un certificat d'una CA tindran el mateix DN. És a dir, la pròpia CA s'ha emès un certificat per a ella mateixa. La clau pública del certificat ens permetrà validar la signatura del propi certificat. És el que es diu un certificat autosignat.. Les CAs que tenen certificats autosignats es diuen CAs arrel (o *root CAs*).

No totes les CAs són arrel. Hi ha també CAs subordinades, els certificats de les quals no estan signats per elles mateixes, si no per altres CAs de jerarquia superior. Aquestes CAs de jerarquia superior poden estar també signades per altres CAs subordinades, etc., fins a arribar a una CA arrel (autosignada).

Un certificat de CA versió 3 tampoc varia molt del d'un usuari final. Usualment disposa de l'extensió `basicConstraints`, amb el camp CA igual a cert (*true*) i el camp `pathLenConstraint` amb un enter que limita el nombre de CAs subordinades que poden penjar d'aquesta.

Jerarquies de certificació i relacions de confiança

Si treballem sobre una mateixa CA és fàcil verificar la validesa d'un certificat qualsevol de la mateixa CA, ja que disposem de la clau pública d'aquesta i, per tant, podem verificar tots els certificats que hagi emès.

Si volem enviar un missatge xifrat a un usuari del com disposem el seu certificat, però que ha estat emès per CA desconeguda, com ens podem fiar que el certificat correspongui lícitament al seu titular o, per contra, ho ha creat fraudulentament un impostor? Hauríem d'instal·lar el certificat de la seva CA (arrel o no) en el nostre

programari perquè pogués verificar la signatura del certificat. Aquest procediment d'instal·lació implica una acció important: estem confiant en els certificats emesos per aquesta CA. Per a cada àmbit hauríem d'instal·lar la CA corresponent, i declarar-la com de confiança.

Algú es podria pregunta per què no tenim una CA arrel única, que signi totes del món. Des del punt de vista del programari PKI seria el millor que podria passar, ja simplificaria el treball dels usuaris. Però a efectes pràctics sembla que hi ha massa interessos creats en les empreses emissores de certificats perquè s'esculli una única CA com arrel.

Hi ha arrels úniques, però són entorns tancats com en els mitjans de pagament. Un exemple serien VISA i MasterCard que utilitzen una arrel única i diversos nivells de CAs subordinades per al seu protocol SET (*Secure Electronic Transaction*).

CAPÍTOL 2

Estructura del projecte

Aquest projecte s'ha estructurat seguint el cicle de vida clàssic après en les assignatures de Enginyeria del Programari, ja que és el procés de enginyeria amb el que es compte amb més experiència. En l'elecció també ha influït el fet de no haver de presentar cap prototip a l'usuari, i el fet que no haurà canvis ni nous requeriments durant cap de les fases de desenvolupament del projecte, ja que tots els requeriments ens són disponibles des del començament (partim de l'enunciat del PFC).

Seguint el cicle de vida clàssic el projecte es dividirà en les diferents fases que ens indica aquest procés: enginyeria del sistema (anàlisi de requeriments), anàlisis (especificació), disseny, codificació (implementació), proves i documentació final.

Al inici de l'especificació del projecte, aquest s'ha dividit en cinc subprojectes (veure *Especificació*, pàg. 23). En cadascun d'aquest projectes s'aplica el cicle de vida clàssic.

Les proves i la documentació es faran ad-hoc juntament amb les diferents fases. El manteniment del programari no serà tingut en compte per a la realització d'aquest projecte.

Anàlisis de requeriments

Per a l'anàlisi de requeriments, cal recordar que ens limitem als requeriments especificats en l'enunciat del projecte. És a dir, tots els requeriments funcionals són disponibles a priori. En aquesta fase ha estat bàsic familiaritzar-se amb l'especificació dels esquemes criptogràfics.

Durant aquest fase també s'han estudiat les diferents eines que posteriorment es farien servir durant el desenvolupament del projecte i s'ha procedit a la seva instal·lació i posta en marxa.

Especificació

Al fer l'especificació també partim de l'avantatge de disposar d'uns requeriments fixos, ja que ens serà més simple descriure el comportament extern del sistema, des del punt de vista de l'usuari o de l'entorn. L'especificació s'ha fet en UML (*Unified Modeling Language*, Llenguatge Unificat de Modelatge) ja que, a més de ser el llenguatge d'especificació amb el qual es contava amb més experiència, els mòduls

que podem construir amb ells ens permet fer una fàcil correspondència amb el llenguatge de programació que es volia fer servir per a la implementació (Java). Això implica que l'especificació, el disseny i la codificació s'ha realitzat completament orientada a objectes. Així mateix, UML cobreix també la part de documentació corresponent a l'arquitectura del sistema i els seus detalls i l'expressió de requisits.

Per el desenvolupament de l'especificació s'ha fet servir una eina CASE, el *Visual Paradigm for UML* de l'empresa Visual Paradigm (http://www.visual-paradigm.com). Aquesta eina permet automatitzar la construcció, el posterior manteniment i la documentació del programari, ja que dona suport en aquestes activitats. A més amb *Visual Paradigm for UML* es pot fer servir la notació UML, el que permet fer servir aquesta eina per realitzar els diagrames de casos d'ús, de classes i de seqüència. Ens em decantat *Visual Paradigm for UML* sobre altres eines CASE per ser un programari amb el que es contava amb una certa experiència i es tenien coneixements sobre les seves possibilitats i les seves limitacions.

Durant la fase d'especificació s'ha dividit el projecte en cinc subprojectes, per a poder desenvolupar el sistema d'una manera més simple (tècnica del "Divideix i Venceràs").

Aquests cinc subprojectes són:

- **Esquema criptogràfic**: Implementació dels protocols criptogràfics associats a les diferents accions que poden sol·licitar metges i pacients.

  En aquest subprojecte s'implementa l'esquema criptogràfic de forma lineal, es a dir, sense comunicació entre les diferents parts.

- **Representació de les dades**: Disseny del format en que s'emmagatzemen i s'intercanvien les dades entre les aplicacions i/o protocols.

  En aquest subprojecte s'utilitza XML (*eXtensible Markup Language*) per a codificar les dades que s'envien durant l'execució dels diferents protocols criptogràfics.

- **Comunicació dels components**: Desenvolupament del programari per la comunicació entre els diferents components client-servidor

  La comunicació dels diferents components del sistema tradicionalment suposaria el disseny d'un protocol o mecanisme de comunicació. Per evitar la sobrecarrega de feina, i donat que la part essencial del projecte és l'esquema criptogràfic, per aquest subprojecte s'ha fet servir la tecnologia RMI (*Remote Method Invocation*).

- **Gestió de la informació**: Disseny i implementació de la base de dades on es guarda la informació sobre els usuaris i els historials mèdics.

  En aquest subprojecte es fa servir el com a sistema gestor de bases de dades el programari lliure MySQL.

- **Interfícies**: Interfícies de usuari per metges, pacients i gestor.

De aquesta manera podem dir que encara que la planificació del projecte està basada en una PBS (*Product Breakdown Structure*), es a dir, orientada al producte, cadascun del subprojectes té una estructura WBS (*Work Breakdown Structure*), es a dir, orientada al procés .

Donat que aquests cinc subprojectes no poden ser considerats com sistemes independents, hi ha que fer proves d'integració entre ells. Per donar com finalitzat un subprojecte hi ha que provar que funciona correctament juntament el sistema realitzat en el(s) subproject(es) anterior(s).

Implementació

La plataforma per al qual s'ha desenvolupat el projecte és el IBM PC, amb el sistema operatiu Windows XP. El llenguatge de programació que s'ha usat durant la fase de codificació ha estat el llenguatge Java, ja que permet la programació orientada a objectes (el que permet l'ús de tot el desenvolupat anteriorment). L'entorn de programació (IDE) escollit per a la implementació ha estat el Ultra Edit-32 v12.10.

Donat que Java no és un llenguatge multiplataforma, la naturalesa del projecte fa fàcil una migració a altres sistemes que tinguin una màquina virtual Java disponible (com Linux o Solaris) ja que no fa un ús de recursos exclusius de la plataforma Windows, que faria molt més complexa l'exportació

Proves

Durant tota la implementació del projecte s'han construït tests unitaris de manera *ad-hoc* juntament amb la implementació per comprovar la correctesa de tot el codi desenvolupat.

Durant la fase de proves s'han sotmès als components (esquema criptogràfic, XML, RMI, esquema de base de dades, interfícies) a una bateria de proves per garantir la seva correcta integració.

Per la realització dels test unitaris s'ha decidit fer servir el *framework* JUnit. JUnit es un *framework* simple i *open source* per escriure i executar tests.

Documentació

La memòria del projecte s'ha fet a mida que es realitzaven les diferents fases del projecte. La documentació tècnica (documentació de classes i mètodes) s'ha realitzar mitjançant l'eina JavaDoc durant el desenvolupament del codi del projecte.

Finalment, a la fase de documentació pròpiament dita, s'han acabat detalls i unificats estils de la memòria del projecte i s'ha preparat la presentació.

Per la realització de la documentació no-tècnica (memòria i presentació) s'ha fet servir la *suite* ofimàtica Microsoft Office de Microsoft i el programari Adobe Acrobat d'Adobe (per realitzar el PDF). S'ha decidit fer servir aquestes eines sobre altres possibilitats (OpenOffice.org o LaTeX) perquè es comptava amb més experiència amb la *suite* de Microsoft.

Seguiment del projecte

Com era d'esperar, la planificació inicialment feta no va ser possible portar-la a terme al peu de lletra.

La falta d'experiència en algunes eines ha fet que algunes fases duressin més de l'esperat, encara que les desviacions van ser mínimes. El principals canvis en la planificació van ocórrer a meitat del projecte.

A principis del més de desembre es va fer una replanificació, en aquesta replanificació es va decidir:

- Implementar un sistema per mantenir l'estat de la connexió de manera que el gestor sàpiga a quin usuari correspon cada connexió (veure *Autenticació, pàg.* 30). Aquesta millora afecta a la segureta i sembla una característica important pel sistema. Això implica que una tasca posterior haurà d'ajustar-se a un nou calendari.

- La interfície d'usuari es faria per línea de comandes (veure *Interfície d'usuari*, pàg. 83). Encara que aquest tipus d'interfície es més austera, pot donar plena funcionalitat. A més, es considera que l'objectiu del projecte es implementar funcions de seguretat, no el disseny d'interfícies gràfiques. Amb aquest canvi la planificació del projecte torna a ajustar-se al temps previst.

D'aquesta manera, l'ocupació temporal de les diferents fases i etapes del projecte ha quedat així:

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	Inici	0 días	mié 20/09/06	mié 20/09/06	
2	**Planificiació inicial del projecte**	1 día	mié 20/09/06	mié 20/09/06	1
3	**Anàlisi de requeriment**	**3 días**	**jue 21/09/06**	**sáb 23/09/06**	**2**
4	Lectura exhaustiva de l'enunciat	1,5 días	jue 21/09/06	vie 22/09/06	
5	Revisió de Junit	0,5 días	vie 22/09/06	vie 22/09/06	4
6	**Determinació dels requeriments**	**1 día**	**sáb 23/09/06**	**sáb 23/09/06**	**5**
7	Requeriments funcionals	0,5 días	sáb 23/09/06	sáb 23/09/06	
8	Requeriments no funcionals	0,5 días	sáb 23/09/06	sáb 23/09/06	7
9	Revisió dels requisits del sistema	0 días	sáb 23/09/06	sáb 23/09/06	6
10	**Instalació programari**	**1 día**	**dom 24/09/06**	**dom 24/09/06**	**3**
11	Instalació IAIK	0,5 días	dom 24/09/06	dom 24/09/06	
12	Instalació PKI	0,5 días	dom 24/09/06	dom 24/09/06	11
13	**Esquema criptogràfic**	**29 días**	**lun 25/09/06**	**mar 24/10/06**	**10**
14	Especificació/Disseny	7 días	lun 25/09/06	dom 01/10/06	
15	Implementació	17 días	lun 02/10/06	jue 19/10/06	14
16	Tests unitaris	5 días	vie 20/10/06	mar 24/10/06	15
17	Revisió del esquema criptogràfic	0 días	mar 24/10/06	mar 24/10/06	16
18	**Representació de les dades**	**15 días**	**mié 25/10/06**	**jue 09/11/06**	**13**
19	Disseny XML	3 días	mié 25/10/06	vie 27/10/06	
20	Implementació	10 días	sáb 28/10/06	mar 07/11/06	19
21	Test Unitaris	2 días	mié 08/11/06	jue 09/11/06	20
22	Revisió de la representació de les dades	0 días	jue 09/11/06	jue 09/11/06	21
23	**Comunicació dels components**	**14 días**	**vie 10/11/06**	**jue 23/11/06**	**18**
24	Especificació/Disseny	3 días	vie 10/11/06	dom 12/11/06	
25	Implementació	9 días	lun 13/11/06	mar 21/11/06	24
26	Test Unitaris	2 días	mié 22/11/06	jue 23/11/06	25
27	Revisió de la comunicació del components	0 días	jue 23/11/06	jue 23/11/06	26
28	**Gestió de la informació**	**16 días**	**vie 24/11/06**	**lun 11/12/06**	**23**
29	Instal·lacio MySQL	0,5 días	vie 24/11/06	vie 24/11/06	
30	Disseny BD	3,5 días	vie 24/11/06	lun 27/11/06	29
31	Implementació	9 días	mar 28/11/06	jue 07/12/06	30
32	Test Unitaris	3 días	sáb 09/12/06	lun 11/12/06	31
33	Revisió de la gestió de la informació	0 días	lun 11/12/06	lun 11/12/06	32
34	**Replanificació**	0 días	jue 23/11/06	jue 23/11/06	23
35	**Revisió esquema criptogràfic (sessions)**	**7 días**	**mar 12/12/06**	**lun 18/12/06**	**28**
36	Especificació/Disseny	2 días	mar 12/12/06	mié 13/12/06	
37	Implementació	3 días	jue 14/12/06	sáb 16/12/06	36
38	Test Unitaris	2 días	dom 17/12/06	lun 18/12/06	37
39	**Interficies**	**11 días**	**mar 19/12/06**	**dom 31/12/06**	**35**
40	Vista client	6 días	mar 19/12/06	dom 24/12/06	
41	Vista gestor	5 días	mié 27/12/06	dom 31/12/06	40
42	Revisió de les intefícies	0 días	dom 31/12/06	dom 31/12/06	41
43	**Documentació**	**6 días**	**mar 02/01/07**	**dom 07/01/07**	**39**
44	Revisió documentació	4 días	mar 02/01/07	vie 05/01/07	
45	Presentació	2 días	sáb 06/01/07	dom 07/01/07	44
46	**Lliurament PFC**	1 día	lun 08/01/07	lun 08/01/07	43
47	Fi del projecte	0 días	lun 08/01/07	lun 08/01/07	46

Figura 2-6. Seguiment del projecte

Recursos destinats al projecte

Com en tot projecte es requereixen una sèrie de recursos tant en el seu disseny com en la seva implementació. Aquests recursos destinats al projecte han estat els següents:

- Recursos hardware:
  - Ordenador personal PC-Pentium IV.
- Recursos software:
  - Sistema operatiu *Windows XP Professional Service Pack 2* (http://ww.microsoft.com).
  - *MS Word 2002* de Microsoft (http://ww.microsoft.com) per la documentació del projecte.
  - *Adobe Acrobat 7.0 Professional* de Adobe (http://www.adobe.com) per la documentació del projecte.
  - *MS Project Professional 2003* de Microsoft (http://ww.microsoft.com) per la planificació del projecte.
  - *MS Powerpoint 2002* de Microsoft (http://ww.microsoft.com) per la presentació del projecte.
  - *MS Visio Professional 2003* de Microsoft (http://ww.microsoft.com) pel disseny de figures.
  - *Visual Paradigm for UML Community Edition v5.2* de Visual Paradigm (http://www.visual-paradigm.com) pels diagrames UML.
  - *UltraEdit-32 v12.10a* de IDM Computer Solutions Inc. (http://ww.ultraedit.com) per la implantació.
  - *OpenSSL v0.9.8c* (http://www.openssl.org) com programari de PKI.
  - *JDOM v1.0* (http://jdom.org) com llibreria per manipular XML en Java.
  - *JUnit v4.1* (http://junit.org) per les proves (tests unitaris).
  - *MySQL 4.0.13* (http://www.mysql.com) com a sistema gestor de base de dades.

CAPÍTOL 3

Esquema criptogràfic

Cada tipus d'acció o servei del nostre sistema es realitza d'acord a un protocol criptogràfic. El conjunt de protocols que es fan servir s'anomena esquema criptogràfic.

En aquest apartat es descriuen l'esquema criptogràfic desenvolupat, esquema que ens permet garantir les necessitats de seguretat d'un historial mèdic que pot ser gestionat a través d'una xarxa de comunicacions.

Requisits de seguretat

Com em vist al resum del projecte, el sistema a de complir les següents propietats:

- **Autenticitat**: S'ha de garantir quin es l'usuari que està accedint al sistema. S'ha de garantir a l'usuari que està accedint a un sistema legítim.
- **Integritat**: No es poden manipular les dades dels historials mèdics emmagatzemats a la base dades. Les comunicacions entre client i servidor també han de complir aquest propietat.
- **No-repudi**: Un metge no pot repudiar les dades d'un historial mèdic que ha inserit.
- **Privadesa**: Només les persones autoritzades poden accedir a les dades dels historials mèdics, tant si les dades s'accedeixen per l'aplicatiu client del sistema com si es fa servir un programari per accedir directament a les dades emmagatzemades en la base de dades. Les comunicacions entre client i servidor també han de complir aquesta propietat.

Notació emprada en els protocols

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

- K: Clau d'un criptosistema simètric.
- M: Missatge.
- $E_K(M)$: 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*, (funció de *hash*).
- N_i: Valor aleatori.
- $Id_{Entitat}$: Identificador d'*Entitat* dins del sistema. Correspon al *hash* SHA1 del certificat de l'*Entitat*.

Protocols criptogràfics

Autenticació

Els usuaris poden autenticar-se un únic cop al principi o cada cop que realitzin una operació.

En el primer cas caldria mantenir un estat de la connexió de manera que el gestor sàpiga a quin usuari correspon cada connexió. Aquest mètode és eficient perquè només cal fer una autenticació cada cop i es més amigable per l'usuari final.

Finalment s'ha decidit implementar aquesta primera opció. Quan l'usuari s'autentica per primer cop al sistema, el sistema gestor genera i es guarda el testimoni (N_G) associat al usuari. El testimoni es guarda xifrat per garantir la seva autenticitat i integritat. Posteriorment l'usuari presentarà aquest testimoni i el sistema comprova a quin usuari està associat. El testimoni deixa de ser vàlid després de que un usuari tanqui la sessió o després de que passi un cert temps configurat sense que l'usuari interactui amb el sistema.

Autenticació_usuari (Id_usuari_U , P_G)

Aquest mètode conté una part de l'autenticació del protocol de Needham-Schroeder. El mètode serà utilitzat pel gestor.

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 Retornar $P_G[N_i, Id\_usuari_U]$.

Autenticació_gestor ($P_G[N_i, Id\_usuari_U]$)

Aquest mètode conté una altra part de l'autenticació del protocol de Needham-Schroeder. El mètode serà utilitzat pels metges i pacients.

1 Desxifrar $P_G[N_i, Id\_usuari_U]$ amb S_G, i obtenir; N_i i Id_usuari_U;

2 Obtenir el certificat de *U* a partir de Id_usuari_U. Suposem que 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;

3 Obtenir un valor de forma aleatòria, N_G;

4 Obtenir l'instant de temps actual *T*;

5 Xifrar N_G i *T* amb la clau pública P_G de *G*, obtenint el testimoni, $P_G[N_G, T]$;

6 Guardar a la base de dades el testimoni, Id_usuari_U i $P_G[N_G, T]$;

7 Xifrar N_i, N_G, Id_usuari_U, amb la clau pública P_U de *U*, $P_U[N_i, N_G, Id\_usuari_U]$;

8 Retornar $P_U[N_i, N_G, Id\_usuari_U]$.

Comprovar_testimoni (Id_usuari_U, N_G)

Aquest mètode permet recuperar les dades d'un testimoni guardat a la base de dades i comprovar si el testimoni es vàlid. El testimoni està guardat a la base de dades xifrat, el que garanteix la seva autenticitat i integritat.

1 Recuperar de la base de dades el testimoni,$P_G[N_G, T]$, associat de Id_usuari_U;

2 Si no existeix cap testimoni associat l'usuari, retornar error;

3 Obtenir l'instant de temps actual *T*;

4 Desxifrar $P_G[N_G, T]$ amb S_G, i obtenir; N'_G i *T'*;

5 Si $N'_G = N_G$:

a) Si *T'* més *timeout_testimoni* es menor que *T*:

a) Xifrar N_G i *T* amb la clau pública P_G de *G*, obtenint el testimoni, $P_G[N_G, T]$

b) Guardar a la base de dades els valors Id_usuari_U i $P_G[N_G, T]$;

b) Si no, retornar error;

6 Si no, retornar error;

Tancar sessió

Quan els usuaris vulguin sortir de l'aplicatiu client, han de tancar la sessió que han obert amb el servidor. Això correspon bàsicament amb esborrar el testimoni guardat en base dades.

1 *U* envia a *G* una petició de tancar sessió.

2 *G* esborra de base de dades Id_usuari_U i $P_G[N_G, T]$; relacionats amb U

Consulta d'un historial

En el protocol **Consulta d'un historial** 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 és Id_usuari_G i correspon al *hash* del seu 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.

El protocol **Consulta d'un historial** consta del següents passos:

1 *U* realitza les operacions següents:

a) Executar el mètode *Autenticació_usuari (* Id_usuariU , PG) (pàg. 30) amb la clau pública P_G, 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 mètode *Autenticació_gestor (* PG[Ni, Id_usuariU]) (pàg. 30) amb $P_G[N_i, Id\_usuari_U]$, i obtenir $P_U[N_i, N_G, Id\_usuari_U]$;

b) Enviar $P_U[N_i, N_G, Id\_usuari_U]$ a *U*;

3 *U* realitza les operacions següents:

a) Desxifrar $P_U[N_i, N_G, Id\_usuari_U]$ amb la clau privada S_U, i obtenir N'_i, N_G, Id_usuari_U;

b) Si $N'_i = N_i$:

a) Xifrar Id_usuari_U, N_G, *Consulta*, *Id_usuari* amb la clau pública P_G de *G*, $P_G[Id\_usuari_U, Id\_usuari_U, N_G, Consulta, Id\_usuari]$. *Consulta* indica que es vol consultar l'historial de l'usuari identificat amb *Id_usuari*;

b) Enviar $P_G[N_G, Consulta, Id\_usuari]$ a *G*;

c) Si no, retornar error;

4 *G* realitza les operacions següents:

a) Desxifrar $P_G[Id\_usuari_U, N_G, Consulta, Id\_usuari]$ amb la clau privada S_G, i obtenir Id_usuari_U, N'_G, *Consulta* i *Id_usuari*;

b) Recuperar N_G de la base de dades. Durant el mètode *Autenticació_gestor (PG[Ni, Id_usuariU])* (pàg. 30) N_G ha estat guardat a la base de dades;

c) Executar el mètode *Comprovar_testimoni (Id_usuariU, NG)* (pàg. 31). Si no, retornar error, fer:

a) Si (Id_usuari_U = *Id_usuari*) o (Id_usuari_U és metge i *Id_usuari* és un pacient de Id_usuari_U) [(1)] fer:

A. Executar el mètode *Obtenir_historial (Id_usuari, PU)* (pàg. 33) amb *Id_usuari* i P_U, i obtenir P_U *[H]*;

B. Enviar P_U *[H]* a *U*.

b) Si no, retornar error;

d) Si no, retornar error;

e) Esborrar la informació sobre el testimoni N_G de la base de dades;

5 *U* realitza les operacions següents:

a) Executar el mètode *Desxifrar_historial (PU [H], SU)* (pàg. 33) amb P_U [H], i obtenir H;

b) Mostrar *H*.

(1) El camp `Organizational Unit Name` del certificat de Id_usuari_U indica si l'usuari és Metge o Pacient, i amb la base de dades podem saber els pacients assignats a un metge.

Obtenir_historial (Id_usuari, P_U)

Aquest mètode 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.

1 Buscar l'historial *H* corresponent a *Id_usuari*;

2 Desxifrar la part de *H* que està 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].*

Desxifrar_historial (P_U [H], S_U)

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

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*.

Consulta dels pacients assignats a un metge

Amb el protocol **Consulta dels pacients assignats a un metge**, un metge pot obtenir el llistat del seus pacients. A la descripció només s'envien el identificadors d'usuari.

El protocol Consulta dels pacients assignats a un metge consta del següents passos:

1 *M* realitza les operacions següents:

a) Executar el mètode *Autenticació_usuari (* Id_usuariU , PG) (pàg. 30) amb la clau pública P_G, 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 mètode *Autenticació_gestor (* PG[Ni, Id_usuariU]) (pàg. 30) amb $P_G[N_i, Id\_usuari_M]$, i obtenir $P_M[N_i, N_G, Id\_usuari_M]$;

b) Enviar $P_M[N_i, N_G, Id\_usuari_M]$ a *M*;

3 *M* realitza les operacions següents:

a) Desxifrar $P_M[N_i, N_G, Id\_usuari_M]$ amb la clau privada S_M, i obtenir N'_i, N_G, Id_usuari_M;

b) Si $N'_i = N_i$:

a) Xifrar Id_usuari_U, N_G, *Llista_pacients*, *Id_usuari* amb la clau pública P_G de *G*, $P_G[Id\_usuari_U, N_G, Llista\_pacients, Id\_usuari]$. *Llista_pacients* un llistat dels identificadors d'usuari corresponents als pacients del metge identificat amb *Id_usuari*;

b) Enviar $P_G[Id\_usuari_U, N_G, Llista\_pacients, Id\_usuari]$ a *G*;

c) Si no, retornar error;

4 *G* realitza les operacions següents:

a) Desxifrar $P_G[Id\_usuari_U, N_G, Llista\_pacients, Id\_usuari]$ amb la clau privada S_G, i obtenir Id_usuari_U, N'_G, *Llista_pacients i Id_usuari*;

b) Recuperar N_G de la base de dades. Durant el mètode *Autenticació_gestor (* PG[Ni, Id_usuariU]) (pàg. 30) N_G han estat guardats a la base de dades;

c) Executar el mètode *Comprovar_testimoni (Id_usuariU, NG)* (pàg. 31). Si no, retornar error, fer:

a) Si Id_usuari_M és metge[(2)] fer:

A. Executar el mètode *Obtenir_llistat_pacients (* Id_usuari, *PU*) (pàg. 35) amb Id_usuari_M i P_M, i obtenir $P_M[\{Id\_usuari_1, ..., Id\_usuari_N\}]$ i $S_G[\{Id\_usuari_1, ..., Id\_usuari_N\}]$;

B. Enviar a *U* $P_M[\{Id\_usuari_1, ..., Id\_usuari_N\}, S_G[\{Id\_usuari_1, ..., Id\_usuari_N\}]$;

b) Si no, retornar error;

d) Si no, retornar error;

e) Esborrar la informació sobre el testimoni N_G de la base de dades;

5 *M* realitza les operacions següents:

a) Executar el mètode *Desxifrar_llistat_pacients (PU[{Id_usuari1, ...,* Id_usuariN}], SG[{Id_usuari1, ..., Id_usuariN}]) (pàg. 35) amb P_M*[{Id_usuari*$_1$*, ..., Id_usuari*$_N$*}]* i S_G*[{Id_usuari*$_1$*, ..., Id_usuari*$_N$*}]*, i obtenir *{Id_usuari*$_1$*, ..., Id_usuari*$_N$*}*;

b) Mostrar {Id_usuari$_1$, ..., Id_usuari$_N$}.

(2) El camp `Organizational Unit Name` del certificat de *Id_usuari*$_U$ indica si l'usuari és Metge o Pacient, i amb la base de dades podem saber els pacients assignats a un metge.

Obtenir_llistat_pacients (Id_usuari, P_U)

Amb aquest mètode, el gestor *G* obté el llistat dels pacients assignats a un metge *Id_usuari*, xifrant-la amb la clau corresponent al metge.

1 Cercar a la base de dades 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. Obtenim així 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*$_N$*}]]*.

Desxifrar_llistat_pacients (P_U[{Id_usuari$_1$, ..., Id_usuari$_N$}], S_G[{Id_usuari$_1$, ..., Id_usuari$_N$}])

Aquest mètode permet a un usuari obtenir la llista dels seus pacients i verificar que ha estat generada pel Gestor *G*.

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 es correcta retornar *{Id_usuari*$_1$*, ..., Id_usuari*$_N$*}*.

Inserció de dades a l'historial mèdic

En el protocol **Inserció de dades a l'historial mèdic** es suposa que prèviament a la inserció de les dades el metge *M* ha consultat l'historial del pacient *P* i per tant coneix el identificador de *Id_usuari*$_U$.

Aquest protocol està pensat únicament per afegir una nova visita *V* a l'historial. El gestor *G* un cop rep un visita *V* d'un pacient *P* verifica que ha estat signada pel metge *M* assignat al pacient. A continuació afegeix la visita *V* 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 es pot saber l'instant de la visita i l'ordre que han seguit.

La visita *V*, el temps *T* i el número de sèrie *X* son signats pel gestor *G*. Si un atacant elimina un registre al mig de l'historial es detectaria perquè hi hauria un salt en el un número de sèrie de les visites. La marca temporal *T* no protegeix l'historial de la eliminació d'una visita. L'atacant no podria 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 la última visita es detectaria perquè hi hauria un salt entre l'última visita i el número de sèrie *X* signat.

El protocol **Inserció de dades a l'historial mèdic** consta del següents passos:

1 *M* realitza les operacions següents:

a) Executar el mètode *Autenticació_usuari (* Id_usuariU , PG) (pàg. 30) amb la clau pública P_G, 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 mètode *Autenticació_gestor (* PG[Ni, Id_usuariU]) (pàg. 30) amb $P_G[N_i, Id\_usuari_M]$, i obtenir $P_M[N_i, N_G, Id\_usuari_M]$;

b) Enviar $P_M[N_i, N_G, Id\_usuari_M]$ a *M*;

3 *M* realitza les operacions següents:

a) Desxifrar $P_M[N_i, N_G, Id\_usuari_M]$ amb la clau privada S_M, i obtenir N'_i, N_G, Id_usuari_M;

b) Si $N'_i = N_i$:

a) Obtenir les dades de la visita *V*. La visita hauria d'incloure com a minin Id_usuari_P;

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

c) Xifrar N_G, *Inserir_visita, 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*;

d) Enviar P_G [N_G, Inserir_visita, V, S_M [V]] a G;

c) Si no, retornar error.

4 *G* realitza les operacions següents:

a) Desxifrar P_G [Id_usuari_m N_G, Inserir_visita, V, S_M [V]] amb la clau privada S_G i obtenir Id_usuari_m N'_G, Inserir_visita, V i S_M [V];

b) Recuperar N_G de la base de dades. Durant el mètode *Autenticació_gestor (* PG[Ni, Id_usuariU]) (pàg. 30) N_G ha estat guardat a la base de dades;

c) Executar el mètode *Comprovar_testimoni (Id_usuariU, NG)* (pàg. 31). Si no, retornar error, fer:

   a) Obtenir Id_usuari_P a partir de V;

   b) Verificar que Id_usuari_M és metge(3);

   c) Verificar que Id_usuari_P és un pacient assignat a Id_usuari_M(4);

   d) 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 la 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 P_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 base de dades $P_G\ [V, S_M[V]]$, $X + 1$, T, $S_G[V, S_M[V], T, X + 1]$ i $S_G[X + 1, Id\_usuari_P]$;

   e) Si no, retornar error.

d) Si no, retornar error.

(3) El camp `Organizational Unit Name` del certificat de Id_usuari_U indica si l'usuari és Metge o Pacient, i amb la base de dades podem saber els pacients assignats a un metge.

(4) Consultant la base de dades podem saber quins pacients que té assignat un metge.

CAPÍTOL 4

Disseny de l'aplicació

Abans d'implementar, s'han realitzat uns models del sistema que s'anava a desenvolupar. Aquests models proporcionen els plànols del sistema. A través d'aquest modelatge s'aconsegueix visualitzar com es vol que sigui el sistema, veure com serà el seu comportament, ens proporciona plantilles que guien en la construcció (implementació) i documenten les decisions preses.

Per a realitzar el modelatge s'ha escollit UML (*Unified Modeling Language*), que és un llenguatge estàndard per a escriure plànols de programari. Les especificacions de UML proporcionen una base semàntica que inclou a tots els elements de tots els models d'un sistema, i relaciona cada element amb uns altres de manera consistent. Els diagrames de UML són simples projeccions visuals d'aquesta base, i cada diagrama revela un aspecte específic interessant del sistema.

El disseny del sistema s'ha realitzat seguint una arquitectura en tres capes:

- **Capa presentació**: Rep les peticions de l'usuari, ordena l'execució de les accions i comunica el resultat obtingut a l'usuari.

  Aquesta capa correspondrien les classes del paquet `View` i es dissenyarà en el apartat *Interfície* d'usuari, pàg. 83.

- **Capa domini**: Rep els esdeveniments de la capa de presentació. Controla la validesa dels esdeveniments i executa les accions demandades, realitzant les consultes necessàries. Aquesta capa ignora on es guarden les dades i com es presentaran a l'usuari.

  Aquest capítol està dedicat principalment al disseny d'aquesta capa.

- **Capa de gestió de dades**: Aquesta capa permet que el domini ignori on estan emmagatzemades les dades (usuaris, historials mèdics) i sobre que dispositiu (disc, base de dades, etc.). S'encarrega de transformar les operacions conceptuals en operacions físiques.

  Aquesta capa correspondrien les classes `Manager.SecureTokenStore`, `Manager.SecureCaseHistoryStore` i `Manager.UserDataStore` (veure *Diagrama de classes*, pág. 42) i es dissenyaran en el apartat *Gestió de la informació: base de dades*, pàg. 77.

Model conceptual

Aquest model és la representació dels conceptes significatius en el domini del sistema i inclou els diagrames estàtics per als objectes del domini.

El diagrama de classes és el més utilitzat en el modelatge de sistemes orientats a objectes. Aquests diagrames s'utilitzen per a modelar la vista de disseny estàtica del sistema i les seves relacions.

Diagrama de desplegament

El diagrama de desplegament (*deployment diagram*) ens mostra les relacions físiques entre els components de programari i de maquinari de un sistema.

Cada node de diagrama de desplegament representa alguna classe d'unitat de còmput, normalment una peça de maquinari. Les relaciones entre nodes ens mostren les connexions, les rutes de comunicació, a través de les quals interactua el sistema. Els components representen mòduls físics de codi.

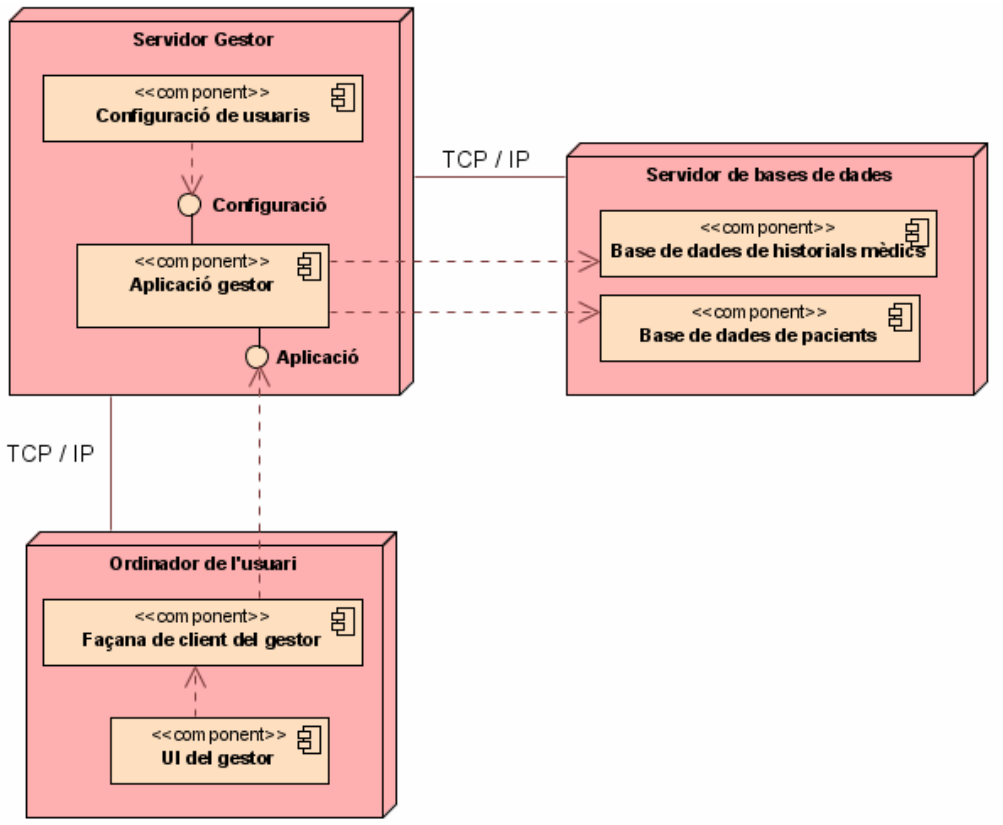

Figura 4-1.Diagrama de desplegament.

Diagrama de paquets

El diagrama de paquets presenta els paquets de classes que hi ha en el sistema i les relacions entre ells.

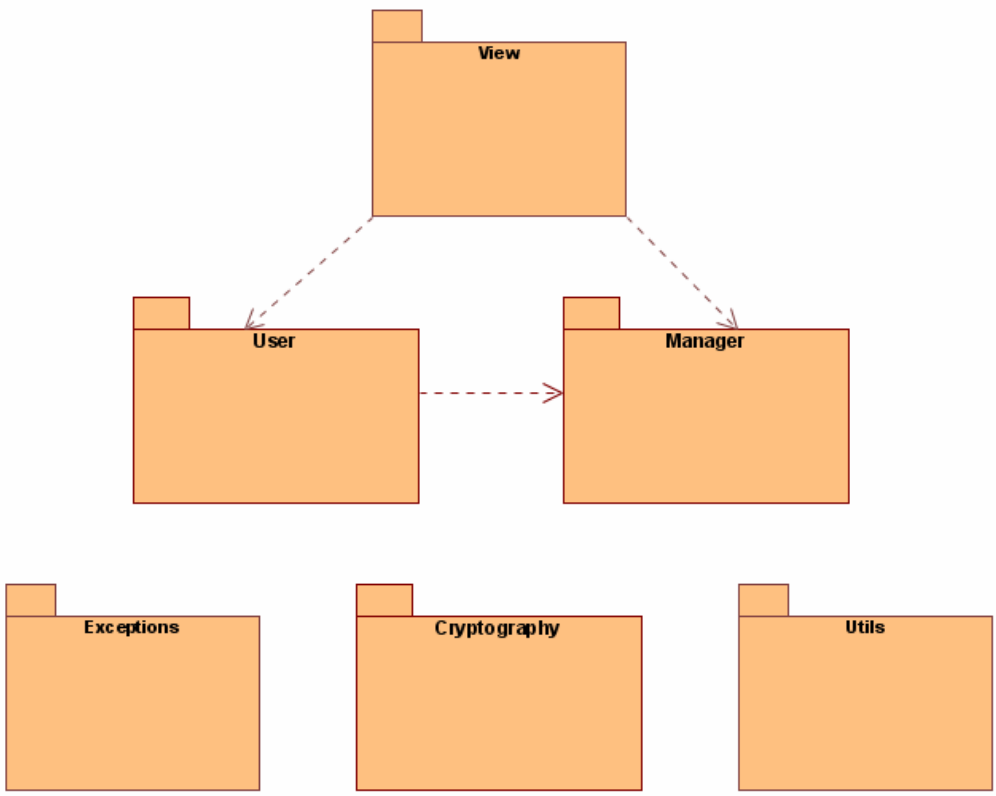

Figura 4-2.Diagrama de paquets.

La responsabilitat de cada paquet es resumeix així:

- **`View`**: Paquet on s'inclouen les classes que implementen les interfícies amb l'usuari. Aquest paquet es descriu en l'apartat *Interfície* d'usuari, pàg. 83.
- **`Client`**: Paquet on s'inclouen les classes que representen la part client del sistema (l'aplicatiu que faran servir metges i pacients).
- **`Manager`**: Paquet on s'inclouen les classes que representen la part servidor del gestor.
- **`Cryptography`**: Paquet on s'inclouen les classes per realitzar les operacions criptogràfiques (xifrat, signatura, accés als camps de certificats i PKCS#12, etc.
- **`Utils`**: Paquet on s'inclouen les classes d'utilitat que farà servir tot el sistema (com constants, mètodes per convertir a base-64, etc.).
- **`Exceptions`**: Paquet on s'inclouen les classes que identifiquen els diferents errors que pot donar el sistema.

Els paquets `Cryptography`, `Utils` i `Exceptions` poden ser accedits des de qualsevol dels altres paquets.

Diagrama de classes

El diagrama de classes presenta un conjunt de classes i les relacions entre elles. S'utilitza per a descriure la vista estàtica del sistema.

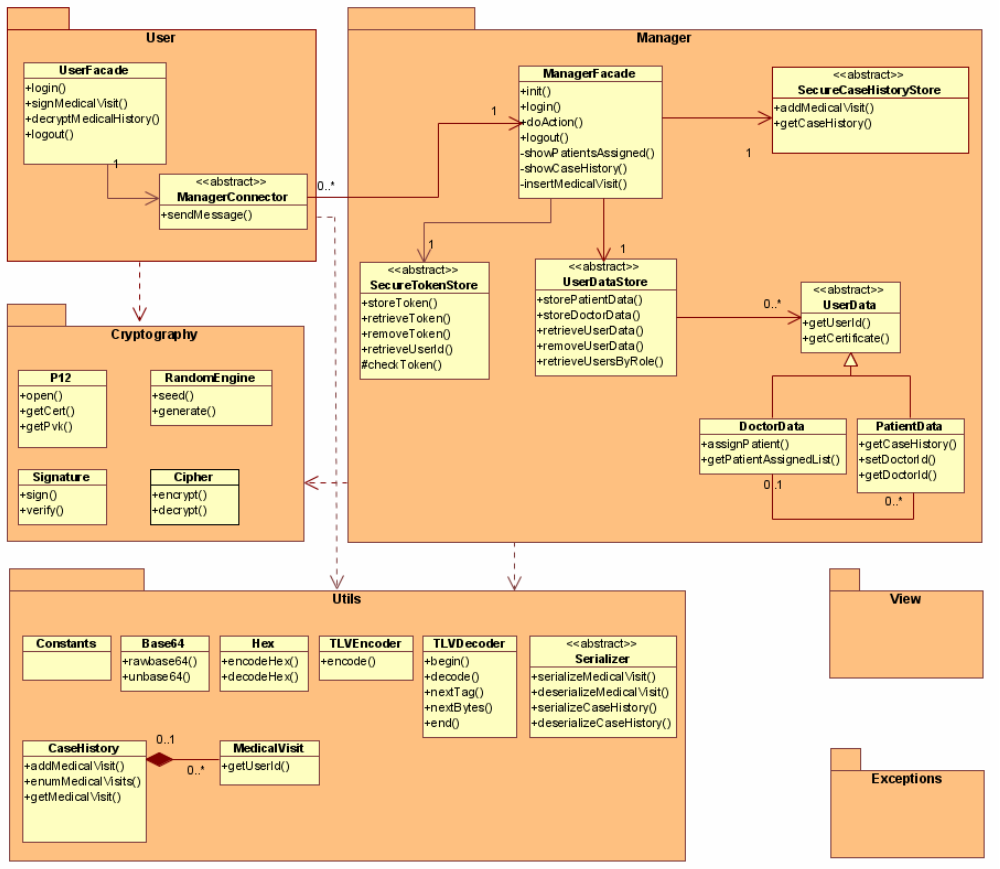

Figura 4-3.Diagrama de classes.

Es pot observar que es fa servir el patró Facade (classes `UserFacade` i `ManagerFacade`), ja que això en proporciona unes classes que donen una interfície unificada per el conjunt de interfícies de cada subsistema.

Les classes del paquets `Cryptography`, `Utils` i `Exceptions` son accessibles des de qualsevol altre classe.

Model de casos d'ús

Cap sistema es troba aïllat. Qualsevol sistema interactua amb actors humans i mecànics que ho utilitzen amb algun objectiu i que esperen que el sistema funcioni

de forma predictible. El model de casos d'ús especifica el comportament del sistema i descriu un conjunt de seqüències d'accions que executa un sistema per a produir un resultat observable de valor per a un actor

Diagrama de casos d'ús

Els diagrames de casos s'utilitzen per al modelatge dels aspectes dinàmics del sistema. Mostren un conjunt de casos d'ús, actors i les seves relacions.

Els diagrames de casos d'ús s'empren per a visualitzar el comportament d'un sistema, de manera que els usuaris puguin comprendre com utilitzar aquest element.

El diagrama de casos d'ús del projecte és el següent:

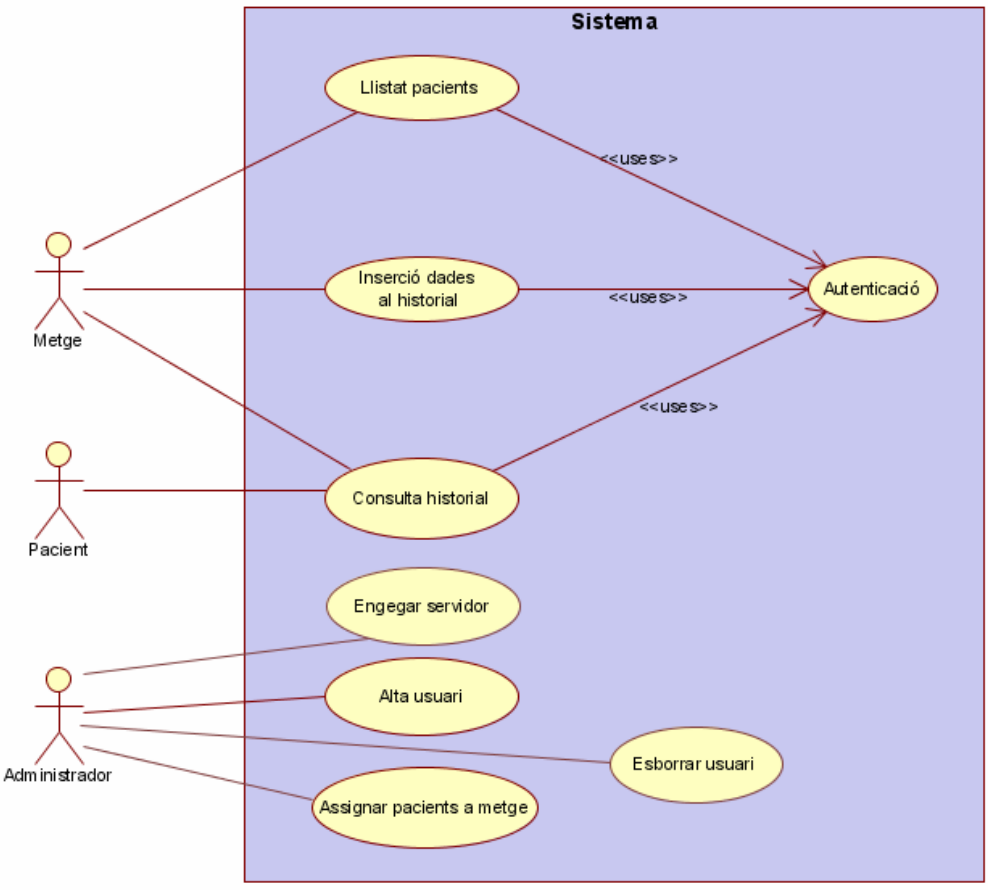

Figura 4-4.Diagrama de casos d'ús.

Veiem que hi han tres actors:

- **Metge**: Representa als usuaris que tenen el rol 'metge'. Tenen accés de lectura i escriptura a les dades dels pacients que tenen assignats.
- **Pacient**: Representa als usuaris que tenen el rol 'pacient'. Tenen accés de lectura a les seves dades.

- **Administrador**: Es el encarregat de donar d'alta els usuaris (metges i pacients).

En tots els casos d'ús de metges i pacients, aquest s'han d'autenticar en el sistema per fer-ho servir. ´

En el cas de l'administrador, per aquest projecte, es considera que no cal que s'autentiqui. Es considera es suficient amb que l'administrador sàpiga com obrir el PKCS#12 del gestor i que el lloc on està instal·lada l'aplicació està securitzat físicament, es a dir, només els administradors tenen accés físic a l'aplicació d'administració.

Contractes dels casos d'ús

Un cas d'ús és un document que descriu una seqüència d'esdeveniments que realitza un actor usant el sistema per a completar un procés que té algun valor per a ell.

En aquest apartat es passarà a descriure de manera exhaustiva la utilitat i els efectes en el sistema de cadascun dels casos d'ús definits en el diagrama de casos d'ús.

Cas d'us: Autenticació

Cas d'ús: Autenticació

Actors: Metge o Pacient

Descripció: Es actor s'autentica en el sistema, indicant quina es l'acció que vol realitzar.

Tipus: Primari

Seqüència típica de esdeveniments:

Accions del actor	*Respostes del sistema*
1 El cas d'us comença quan un actor vol realitzar una acció en el sistema.	
2 L'actor envia les seves credencials al sistema.	
	3 El sistema comprova les credencials de l'actor.
	4 El sistema envia a l'actor les credencials del sistema i un desafiament.
5 L'actor comprova les credencials del sistema.	
6 L'actor envia al sistema la resposta al desafiament.	

Accions del actor	*Respostes del sistema*
	7 El sistema comprova la resposta de l'actor al desafiament.

Alternatives:

3. Error al comprovar les credencials (usuari no acreditat).

3. Usuari no existent en el sistema.

5. Error al comprovar les credencials (missatge modificat durant l'enviament).

6. Error en la resposta del desafiament.

Cas d'us: Tancar sessió

Cas d'ús: Tancar sessió

Actors: Metge o Pacient

Descripció: Es actor vol tancar la sessió al sistema.

Tipus: Primari

Seqüència típica de esdeveniments:

Accions del actor	*Respostes del sistema*
8 El cas d'us comença quan un actor vol tancar la seva sessió al sistema.	
	9 El sistema comprova les credencials de l'actor.
	10 El sistema treu l'usuari del sistema.

Alternatives:

Cas d'us: Llista pacients

Cas d'ús: Xifrar llista pacients

Actors: Metge

Descripció: El sistema retorna la llista de pacients del metge

Tipus: Primari

Seqüència típica de esdeveniments:

Accions del actor	*Respostes del sistema*
1 El cas d'us comença quan un metge vol el llistat dels seus pacients	
2 El metge s'autentica contra el sistema (veure *Cas d'us: Autenticació*, pàg. 44).	
	3 El sistema obté la llista de pacients del sistema.
	4 El sistema signa la llista de pacients.
	5 El sistema xifra contra l'actor la llista de pacients i la signatura.
	6 El sistema envia el resultat del xifrat.
7 El metge desxifra el missatge enviat pel sistema.	
8 El metge verifica la signatura del missatge.	

Alternatives:

2. Error en l'autenticació en el sistema.

7. Error al desxifrar el missatge.

8. Error al verificar el missatge.

Cas d'us: Inserció de dades al historial

Cas d'ús: Inserció de dades al historial

Actors: Metge

Descripció: El metge afegeix les dades d'una visita al historial de un pacient.

Tipus: Primari

Seqüència típica de esdeveniments:

Accions del actor	*Respostes del sistema*
1 El cas d'us comença quan un metge vol afegir dades al historial de un pacient.	
2 El metge s'autentica contra el sistema (veure *Cas d'us: Autenticació*, pàg. 44).	

Accions del actor	*Respostes del sistema*
3 El metge signa les dades de la visita mèdica.	
4 El metge xifra contra el sistema les dades de la visita i la signatura.	
5 El metge envia el resultat del xifrat.	
	6 El sistema desxifra el missatge enviat pel metge.
	7 El sistema verifica que el pacient indicat en la visita està assignat al metge.
	8 El sistema verifica la signatura del missatge.
	9 El sistema signa les dades de la visita.
	10 El sistema xifra contra el sistema les dades de la visita i la signatura.
	11 El sistema emmagatzema el resultat del xifrat.

Alternatives:

2. Error en l'autenticació en el sistema.

6. Error al desxifrar el missatge.

7. El pacient no està assignat al metge.

8. Error al verificar el missatge.

Cas d'us: Consulta d'un historial

Cas d'ús: Consulta d'un historial

Actors: Metge o Pacient

Descripció: El metge afegeix les dades d'una visita al historial de un pacient.

Tipus: Primari

Seqüència típica de esdeveniments:

Accions del actor	*Respostes del sistema*
1 El cas d'us comença quan l'actor vol consultar l'historial de un pacient.	

Accions del actor	*Respostes del sistema*
2 El metge s'autentica contra el sistema (veure *Cas d'us: Autenticació*, pàg. 44).	
3 L'actor xifra contra el sistema l'identificador del pacient del que vol l'historial	
4 L'actor envia el resultat del xifrat al sistema.	
	5 El sistema desxifra el missatge enviat per l'actor.
	6 El sistema comprova si l'actor és un metge o si es un pacient que consulta el seu propi historial.
	7 El sistema busca l'historial corresponent al pacient consultat.
	8 El sistema desxifra les dades de l'historial.
	9 El sistema xifra contra l'actor les dades de l'historial.
	10 El sistema envia el resultat del xifrat l'actor.
11 L'actor desxifra el missatge enviat pel sistema.	
12 Per cada entrada (visita) de l'historial, l'actor verifica: a) La signatura del metge que ha afegit la visita. b) La signatura que ha fet el sistema quan ha enregistrat la visita.	

Alternatives:

2. Error en l'autenticació en el sistema.

5. Error al desxifrar el missatge.

6. L'actor no està donat d'alta al sistema. L'actor es un pacient i no està consultat el seu historial.

11. Error al desxifrar el missatge.

12. Error al verificar el missatge

Cas d'us: Engegar servidor

Cas d'ús: Engegar servidor

Actors: Administrador

Descripció: El administrador engega el servidor.

Tipus: Primari

Seqüència típica de esdeveniments:

Accions del actor	*Respostes del sistema*
1 El cas d'us comença quan l'actor vol engegar el servidor.	
2 L'administrador obre el PKCS#12 del servidor indicant la contrasenya.	
	3 El sistema guarda a memòria el PKCS#12 obert

Alternatives:

2. Error al obrir el PKCS#12 del sistema gestor.

Cas d'us: Alta usuari

Cas d'ús: Alta usuari

Actors: Administrador

Descripció: El administrador dona d'alta un nou usuari en el sistema.

Tipus: Primari

Seqüència típica de esdeveniments:

Accions del actor	*Respostes del sistema*
1 El cas d'us comença quan l'actor vol donar d'alta un nou usuari en el sistema.	
2 L'administrador indica al sistema el certificat que identifica al nou usuari.	
	3 El sistema comproba que el certificat pertany a la unitat organitzativa 'Metges' o 'Pacients'.
	4 El sistema dona d'alta el nou usuari

Alternatives:

3. L'usuari no pertany a una unitat organitzativa adequada.

4. Usuari ja donat d'alta.

Cas d'us: Esborrar usuari

Cas d'ús: Esborrar usuari

Actors: Administrador

Descripció: El administrador dona d'alta un nou usuari en el sistema.

Tipus: Primari

Seqüència típica de esdeveniments:

Accions del actor	*Respostes del sistema*
1 El cas d'us comença quan l'actor vol esborrar un usuari del sistema.	
2 L'administrador indica al sistema quin usuari es vol donar de baixa	
	3 El sistema esborra l'usuari del sistema.

Alternatives:

3. L'usuari no existeix.

Cas d'us: Assignar pacients a metge

Cas d'ús: Assignar pacients a metge

Actors: Administrador

Descripció: El administrador nous pacients a un metge.

Tipus: Primari

Seqüència típica de esdeveniments:

Accions del actor	*Respostes del sistema*
1 El cas d'us comença quan l'actor vol assignar pacients a un metge.	
2 L'administrador indica al sistema quin pacients vol assignar i a quin metge	
	3 El sistema assigna els pacients al metge indicat.

Alternatives:

3. Algun dels usuaris indicats no existeix.

Model de comportament del sistema

En qualsevol sistema, els objectes interactuen entre si passant-se missatges. Una interacció és un comportament que inclou un conjunt de missatges que s'intercanvien diversos objectes dintre d'un context per a obtenir un propòsit. El model de comportament del sistema especifica l'aspecte dinàmic de les col·laboracions dels objectes.

Per a exposar aquests procediments, el model de comportament del sistema es basa en els diagrames de seqüència del sistema, que mostren els esdeveniments que els actors invoquen al sistema i permeten identificar les operacions del sistema. Per a descriure formalment l'efecte que tenen aquestes operacions s'especifiquen contractes per a cadascuna d'aquestes operacions del sistema.

Diagrames de seqüència del sistema

Els diagrames de seqüència del sistema mostren els esdeveniments que provoquen els actors al sistema en els diferents casos d'ús. Aquests diagrames permeten identificar les operacions i els esdeveniments que els actors o el sistema invoquen.

Un diagrama de seqüència mostra, en un escenari particular d'un cas d'ús, els esdeveniments generats pels actors externs, el seu ordre i els esdeveniments interns del sistema (operacions del sistema) que resulten de les crides. Els diagrames de seqüència es defineixen segons el traçat típic d'esdeveniments d'un cas d'ús.

Seguint aquesta filosofia s'han dissenyat els diagrames de seqüència per a cadascun dels casos d'ús llistats en a l'apartat *Diagrama de casos d'ús*, pàg. 43 que es mostra a continuació.

Engegar servidor

El següent diagrama correspon a la seqüència d'esdeveniments pel cas d'us 'Autenticació' (veure *Cas d'us: Engegar servidor*, pàg. 49)

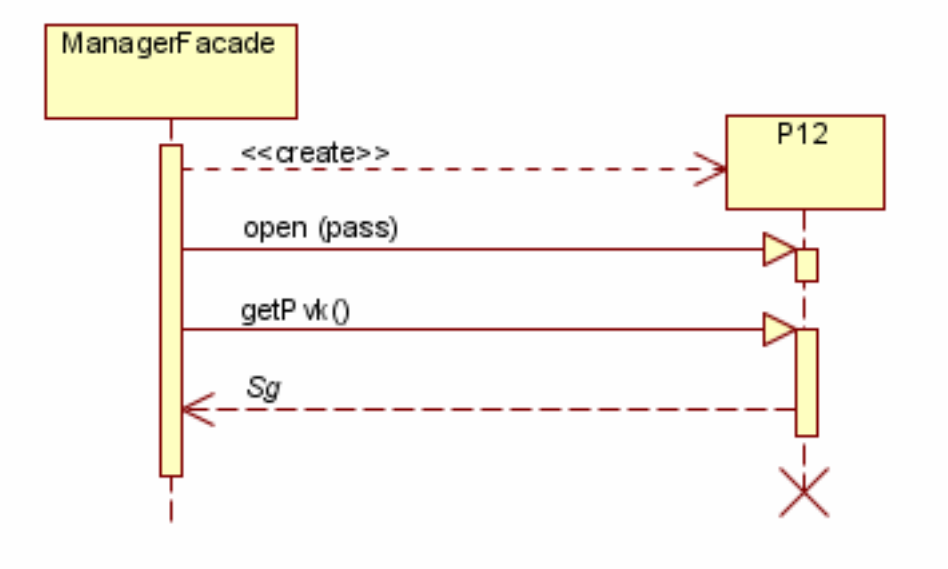

Figura 4-5. Diagrama de seqüència del cas d'us "Engegar servidor"

Gestió de peticions

El següent diagrama correspon a la seqüència d'esdeveniments quan la part servidor rep una petició de la part client:

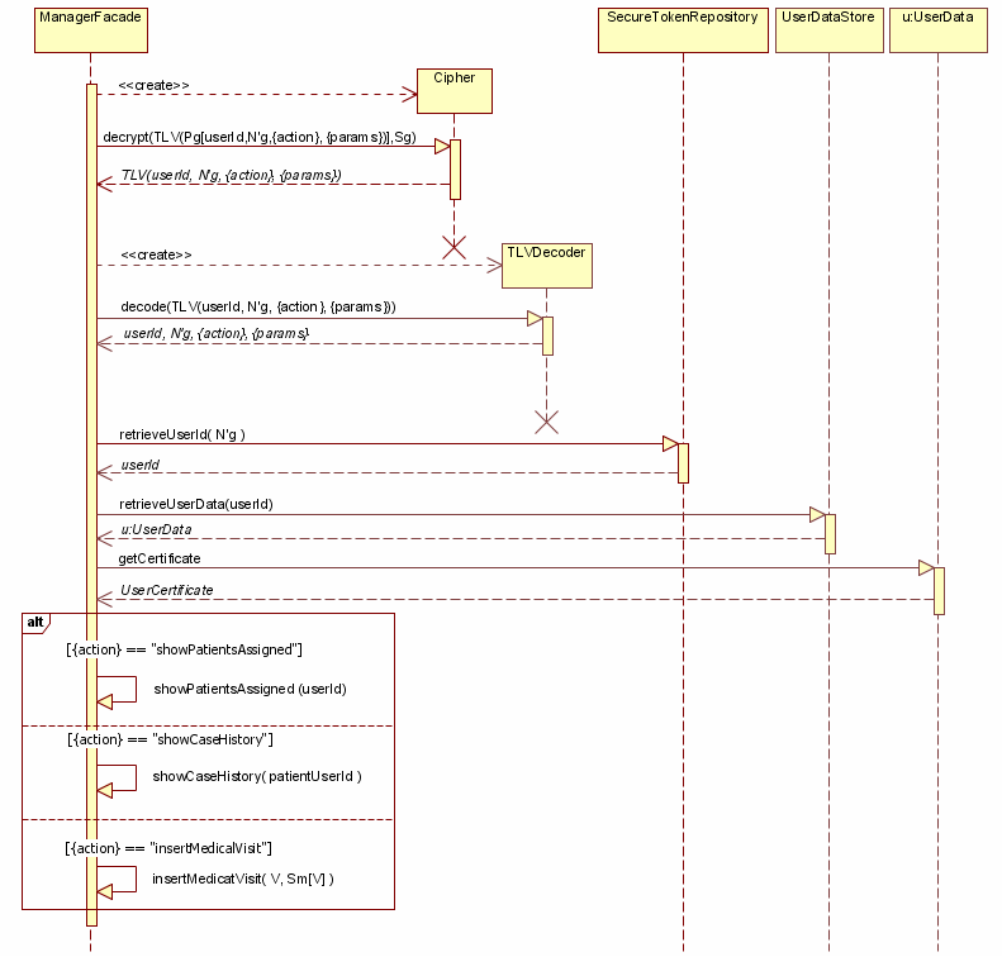

Figura 4-6. Diagrama de seqüència per la gestió de peticions.

Autenticació

El següent diagrama correspon a la seqüència d'esdeveniments pel cas d'us 'Autenticació' (veure *Cas d'us: Autenticació, pàg.* 44*)* en l'aplicatiu client des de que l'actor comença al cas d'us (veure *Autenticació_usuari (Id_usuariU , PG)*, pàg. 30, i *Autenticació_gestor (PG[Ni, Id_usuariU])*, pàg. 30).:

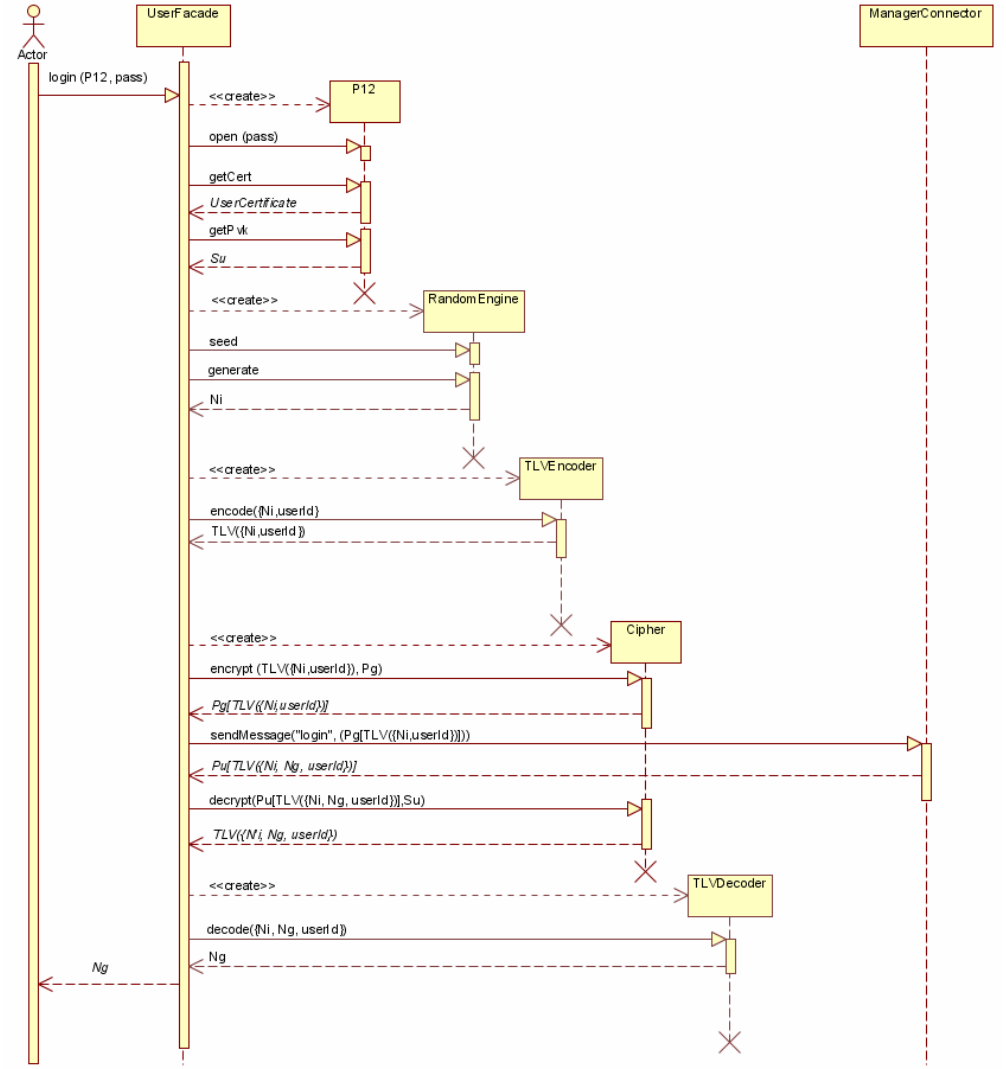

Figura 4-7. Diagrama de seqüència del cas d'us "Autenticació" (part client).

El següent diagrama correspon a la seqüència d'esdeveniments pel cas d'us 'Autenticació' (veure *Cas d'us: Autenticació, pàg.* 44*)* en l'aplicatiu servidor des de que es rep una petició d'autenticació (veure *Autenticació_gestor (PG[Ni, Id_usuariU])*, pàg. 30).

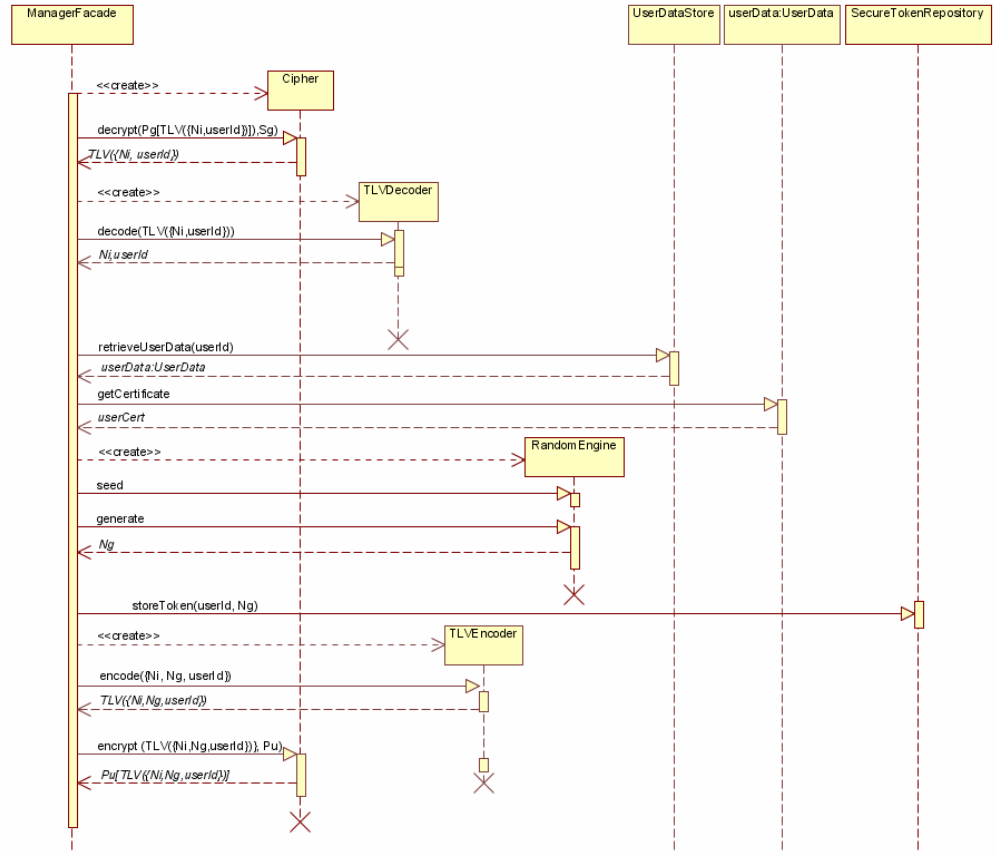

Figura 4-8. Diagrama de seqüència del cas d'us "Autenticació" (part servidor)

El següent diagrama es mostra en detall com el la part servidor del sistema emmagatzema de forma segura el testimoni del client que s'acaba d'autenticar (veure punt 6 de *Autenticació_gestor (PG[Ni, Id_usuariU])*, pàg. 31).

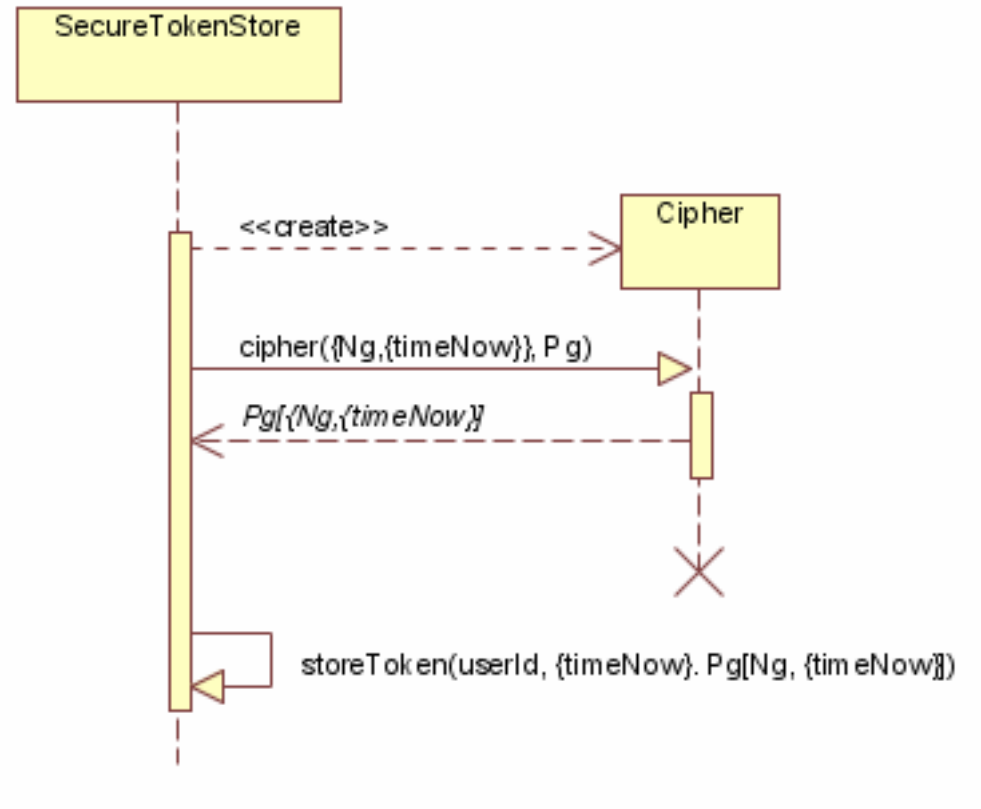

Figura 4-9. Diagrama de seqüència del cas d'us "Autenticació" (part servidor, emmagatzematge del testimoni).

El següent diagrama es mostra en detall com el la part servidor del sistema comprova el testimoni que ha rebut del client (veure *Comprovar_testimoni (Id_usuariU, NG)*, pág. 31).

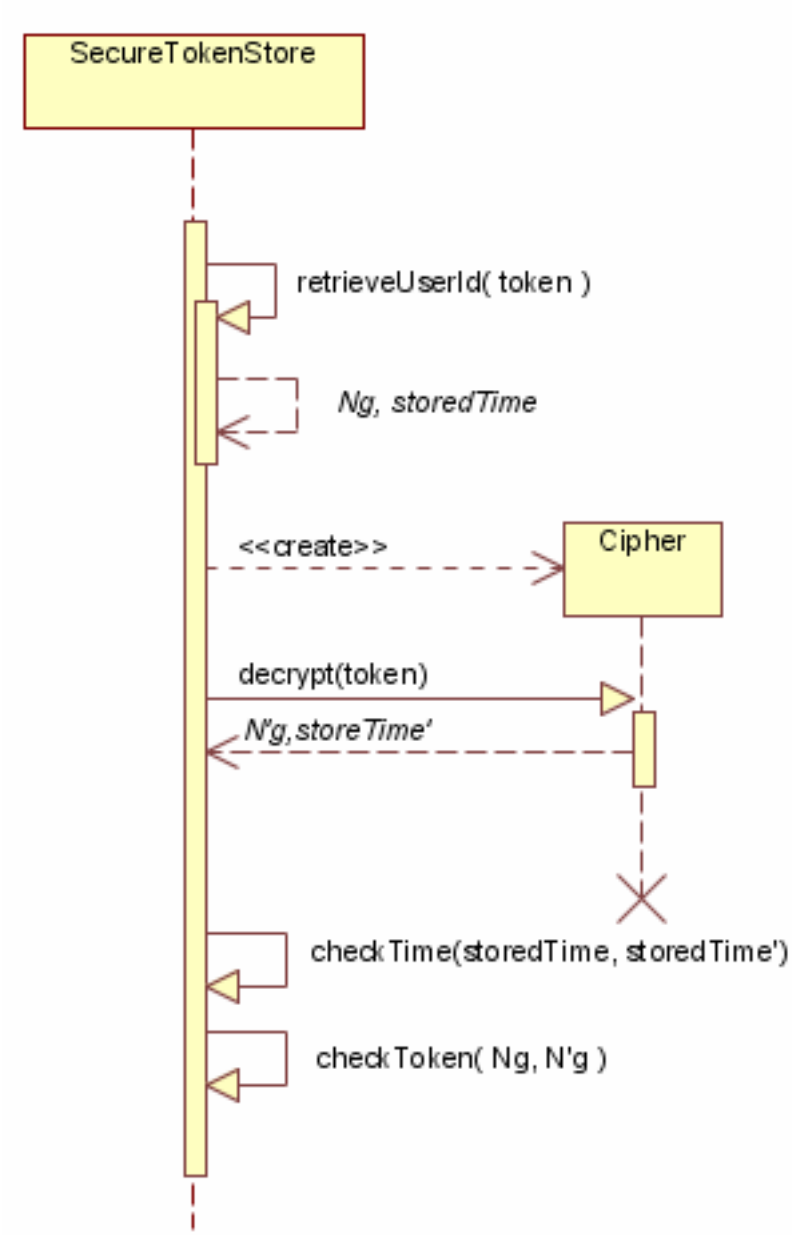

Figura 4-10. Diagrama de seqüència del cas d'us "Autenticació" (part servidor, obtenció del testimoni)

Tancar sessió

El següent diagrama correspon a la seqüència d'esdeveniments pel cas d'us 'Tancar sessió' (veure *Cas d'us: Tancar sessió, pàg.* 45*)* en l'aplicatiu client des de que l'actor comença al cas d'us (veure *Tancar sessió*, pàg. 31).

.

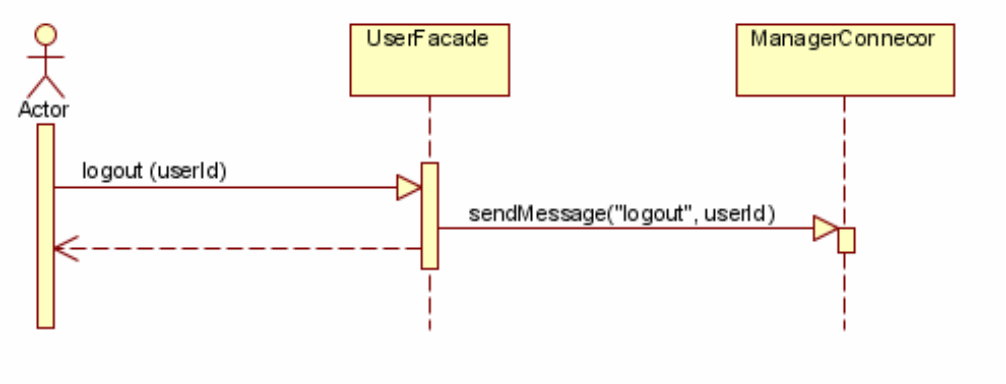

Figura 4-11. Diagrama de seqüència del cas d'us "Tancar sessió" (part client)

El següent diagrama correspon a la seqüència d'esdeveniments pel cas d'us 'Tancar sessió' (veure *Cas d'us: Tancar sessió, pàg.* 45*)* en l'aplicatiu servidor des de que es rep una petició de tancar sessió (veure *Tancar sessió*, pàg. 31).

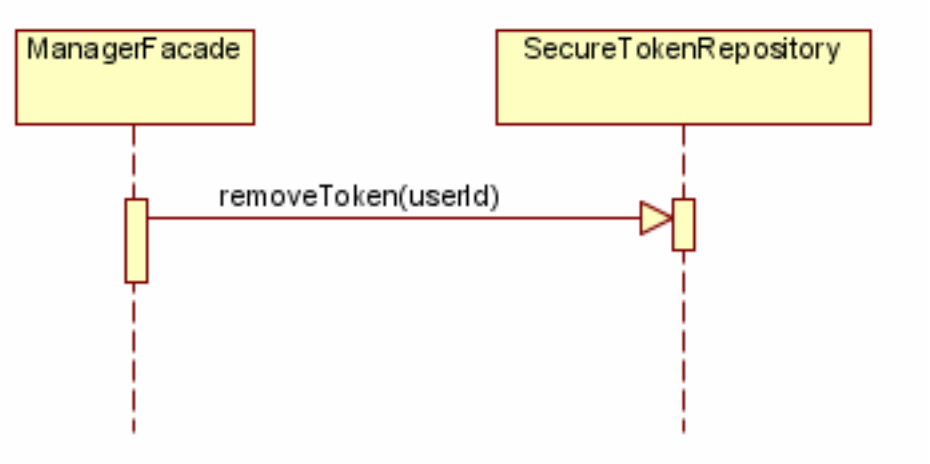

Figura 4-12. Diagrama de seqüència del cas d'us "Tancar sessió" (part servidort)

Llista pacients

El següent diagrama correspon a la seqüència d'esdeveniments pel cas d'us 'Llista pacients' (veure *Cas d'us: Llista pacients, pàg.* 45*)* en l'aplicatiu client des de que l'actor comença al cas d'us (veure *Consulta dels pacients assignats a un metge*, pàg. 33).

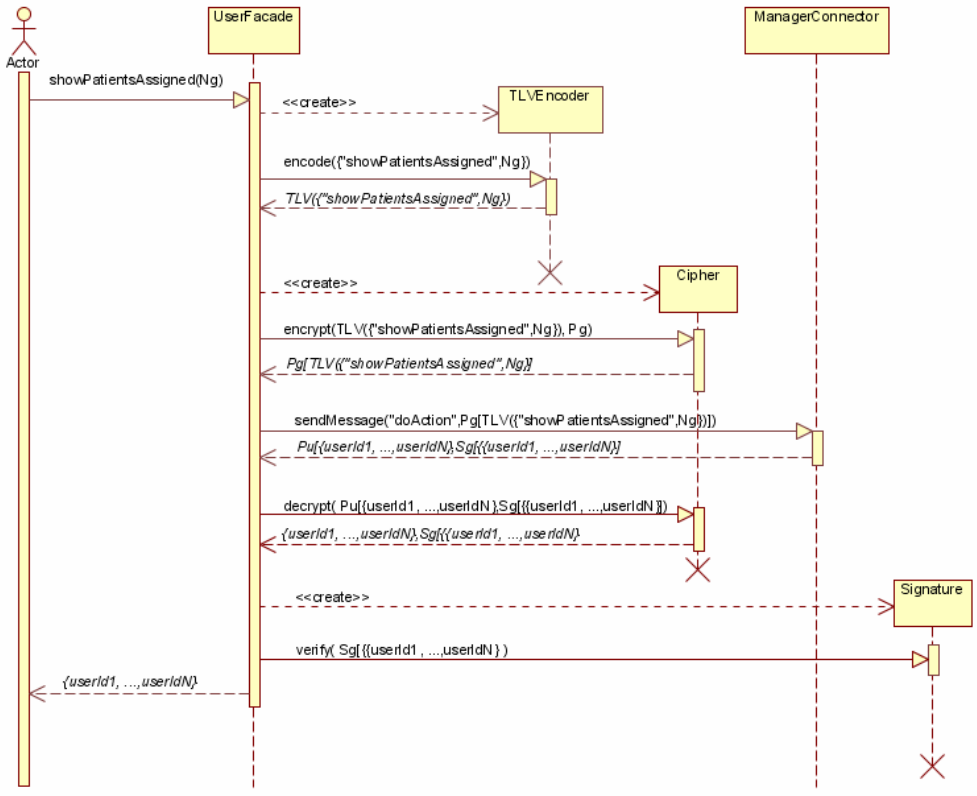

Figura 4-13. Diagrama de seqüència del cas d'us "Llista pacients" (part client).

El següent diagrama correspon a la seqüència d'esdeveniments pel cas d'us 'Llista pacients' (veure *Cas d'us: Llista pacients, pàg.* 45*)* en l'aplicatiu servidor des de que es rep una petició d'obtenció del llista de pacients (veure *Obtenir_llistat_pacients (Id_usuari, PU)*, pág. 35 i *Desxifrar_llistat_pacients (PU[{Id_usuari1, ..., Id_usuariN}], SG[{Id_usuari1, ..., Id_usuariN}])*, pàg 35).

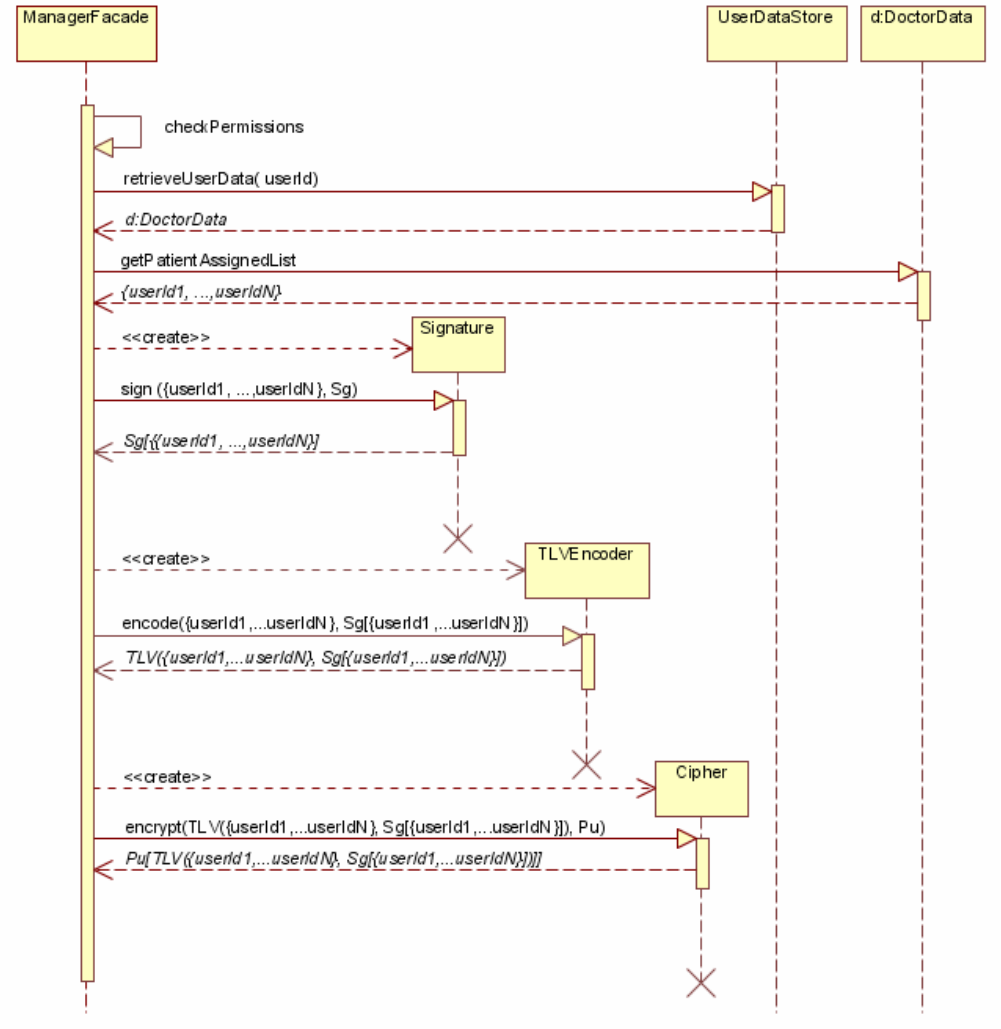

Figura 4-14. Diagrama de seqüència del cas d'us "Llista pacients" (part servidor).

Inserció de dades a l'historial

El següent diagrama correspon a la seqüència d'esdeveniments pel cas d'us 'Inserció de dades al historial' (veure *Cas d'us: Inserció de dades al historial*, pàg. 46*)* en l'aplicatiu client des de que l'actor comença al cas d'us (veure *Inserció de dades a l'historial mèdic*, pàg. 35).

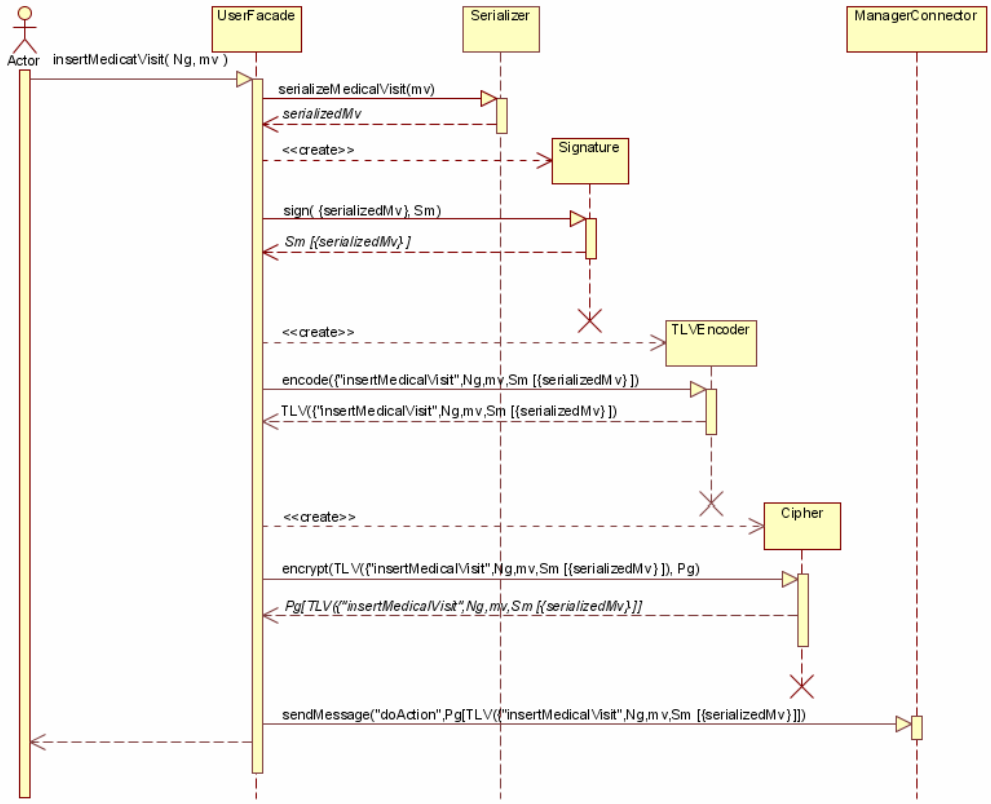

Figura 4-15. Diagrama de seqüència del cas d'us "Inserció de dades al historial" (part client)

El següent diagrama correspon a la seqüència d'esdeveniments pel cas d'us 'Llista pacients' (veure *Cas d'us: Inserció de dades al historial*, pàg. 46*)* en l'aplicatiu servidor des de que es rep una petició d'inserció de dades al historial (veure *Inserció de dades a l'historial mèdic*, pàg. 35).

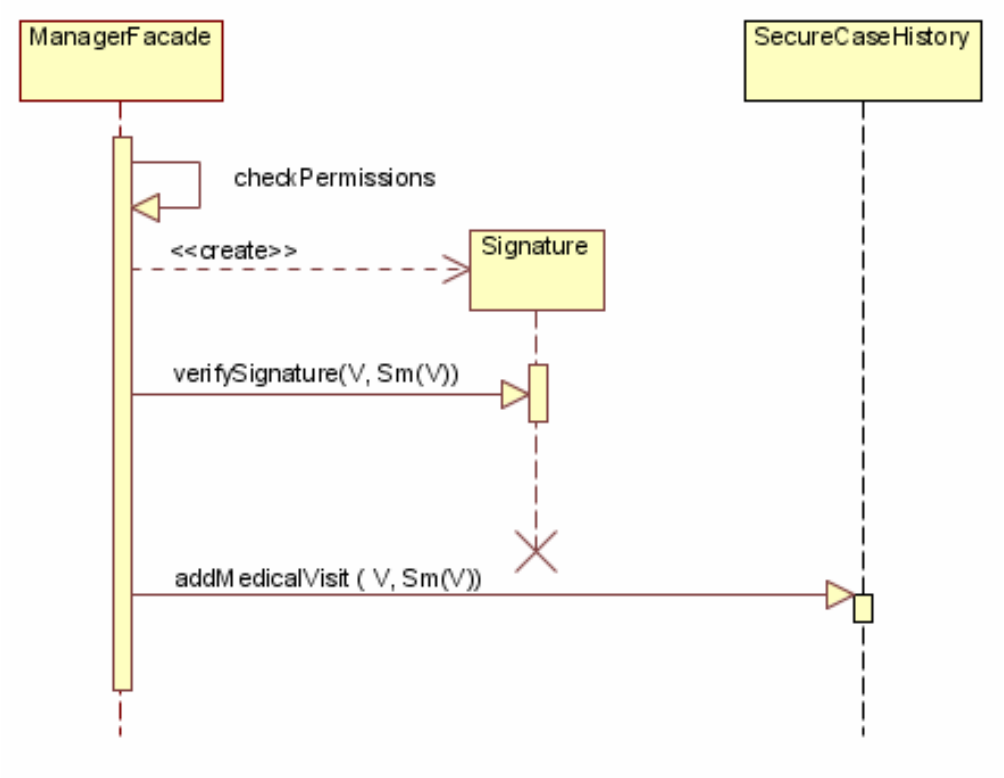

Figura 4-16. Diagrama de seqüència del cas d'us "Inserció de dades al historial" (part servidor).

El següent diagrama es mostra en detall com el la part servidor del sistema emmagatzema de forma segura les dades de la nova visita mèdica (veure *Inserció de dades a l'historial mèdic*, pàg. 35)

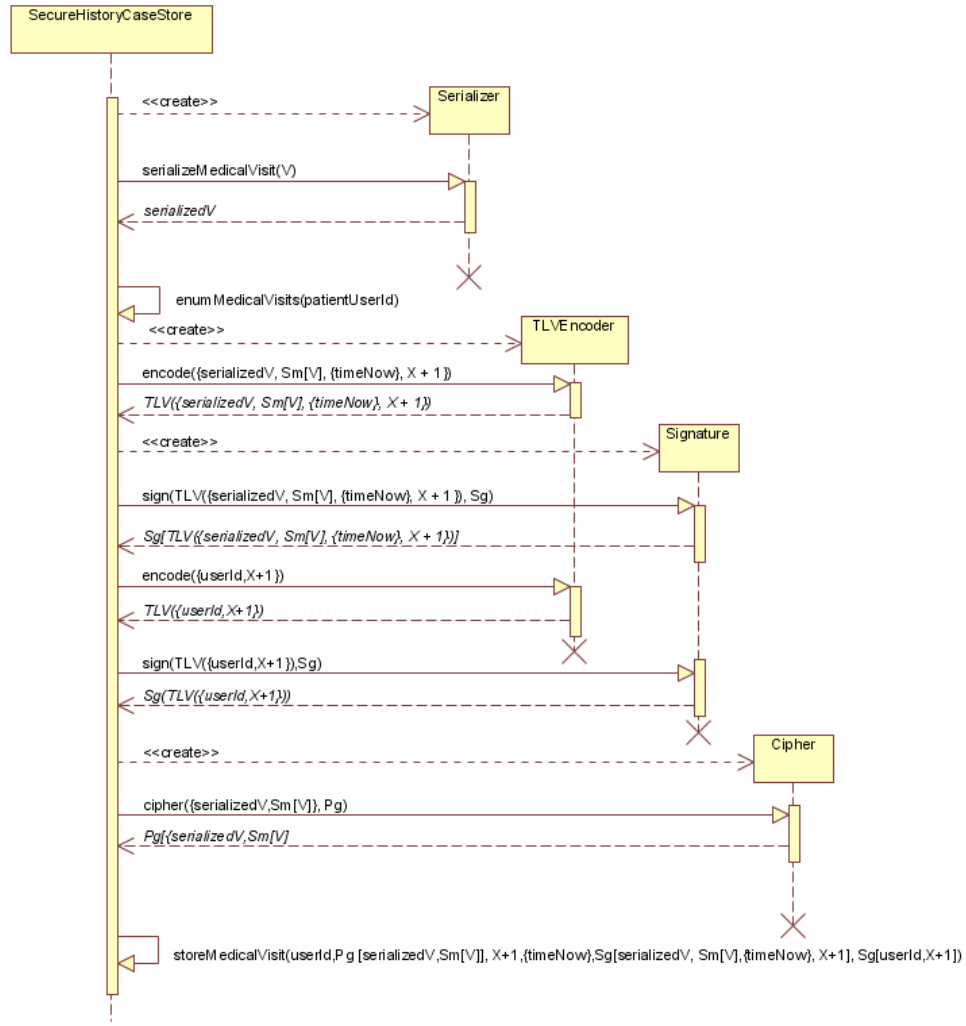

Figura 4-17. Diagrama de seqüència del cas d'us "Inserció de dades al historial" (part servidor, emmagatzematge de les dades).

Consulta d'un historial

El següent diagrama correspon a la seqüència d'esdeveniments pel cas d'us 'Consulta d'un historial l' (veure *Cas d'us: Consulta d'un historial*, pàg. 47*)* en l'aplicatiu client des de que l'actor comença al cas d'us (veure *Consulta d'un historial*, pàg. 32).

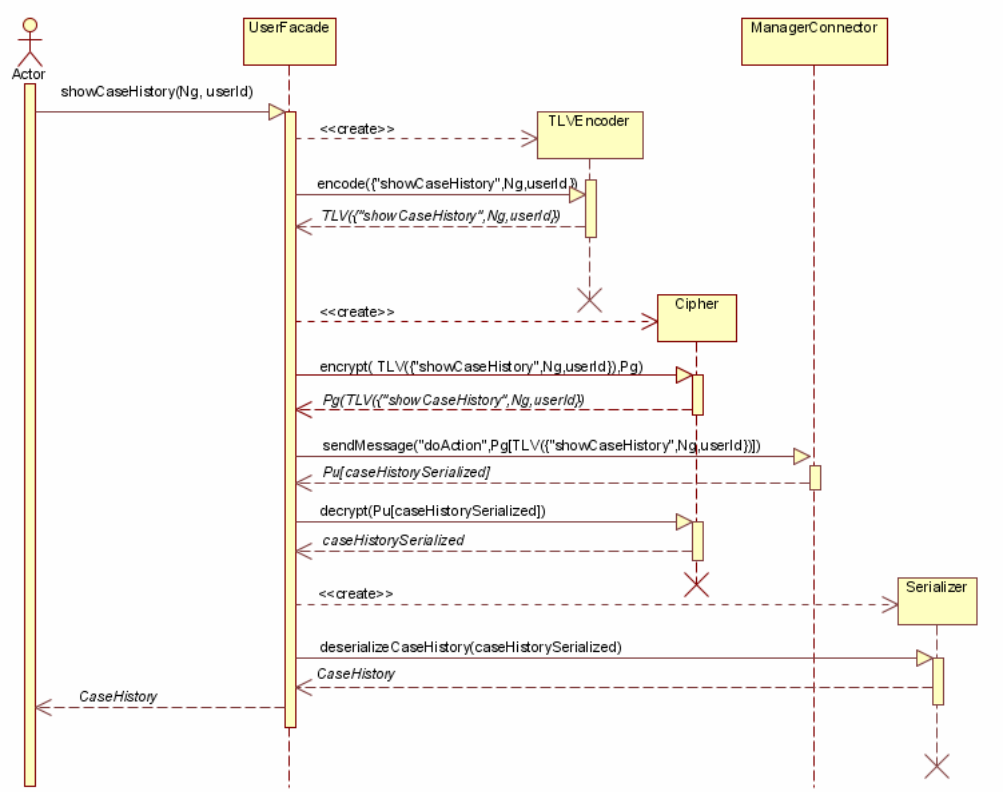

Figura 4-18. Diagrama de seqüència del cas d'us "Consulta d'un historial" (part client)

El següent diagrama correspon a la seqüència d'esdeveniments pel cas d'us 'Consulta d'un historial l' (veure *Cas d'us: Consulta d'un historial*, pàg. 47*)* en l'aplicatiu servidor des de que es rep una petició de consulta d'un historial (veure *Inserció de dades a l'historial mèdic*, pàg. 35).

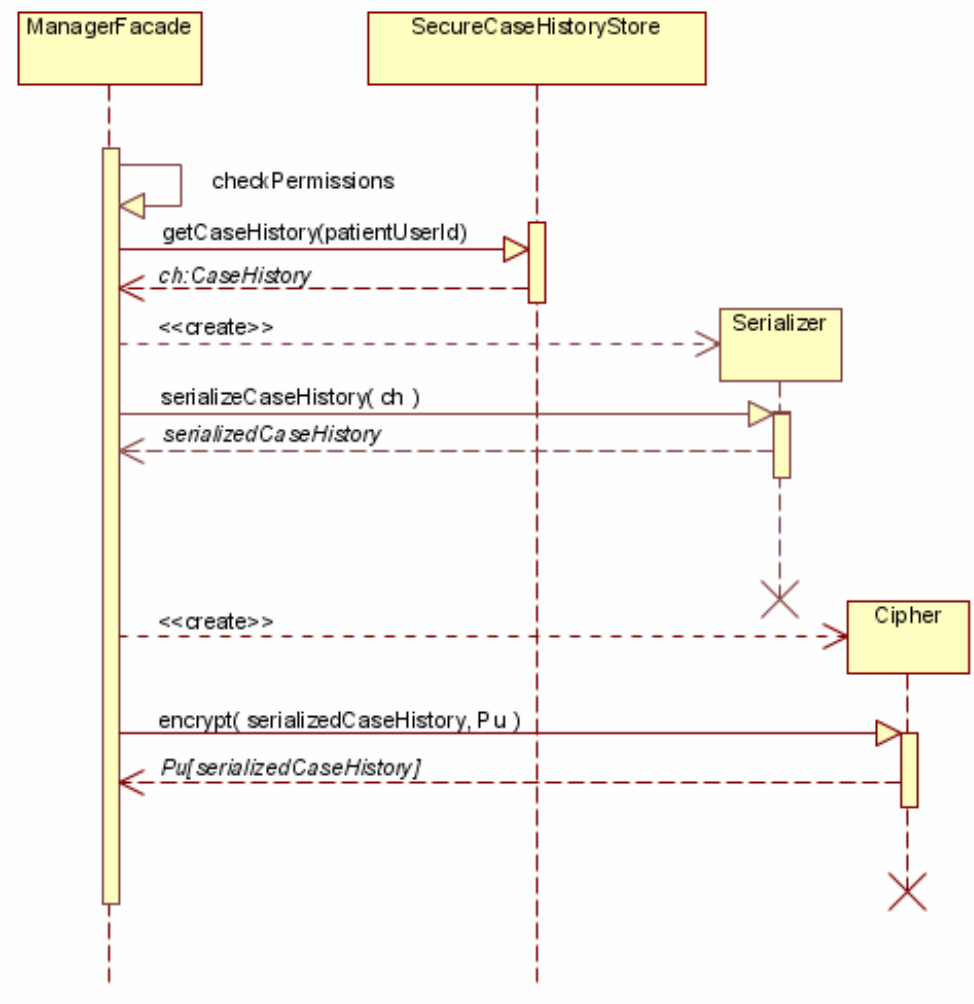

Figura 4-19. Diagrama de seqüència del cas d'us "Consulta d'un historial" (part servidor).

El següent diagrama es mostra en detall com el la part servidor del sistema recupera la informació del historial mèdic, comprovant la integritat i l'autenticitat de les dades.

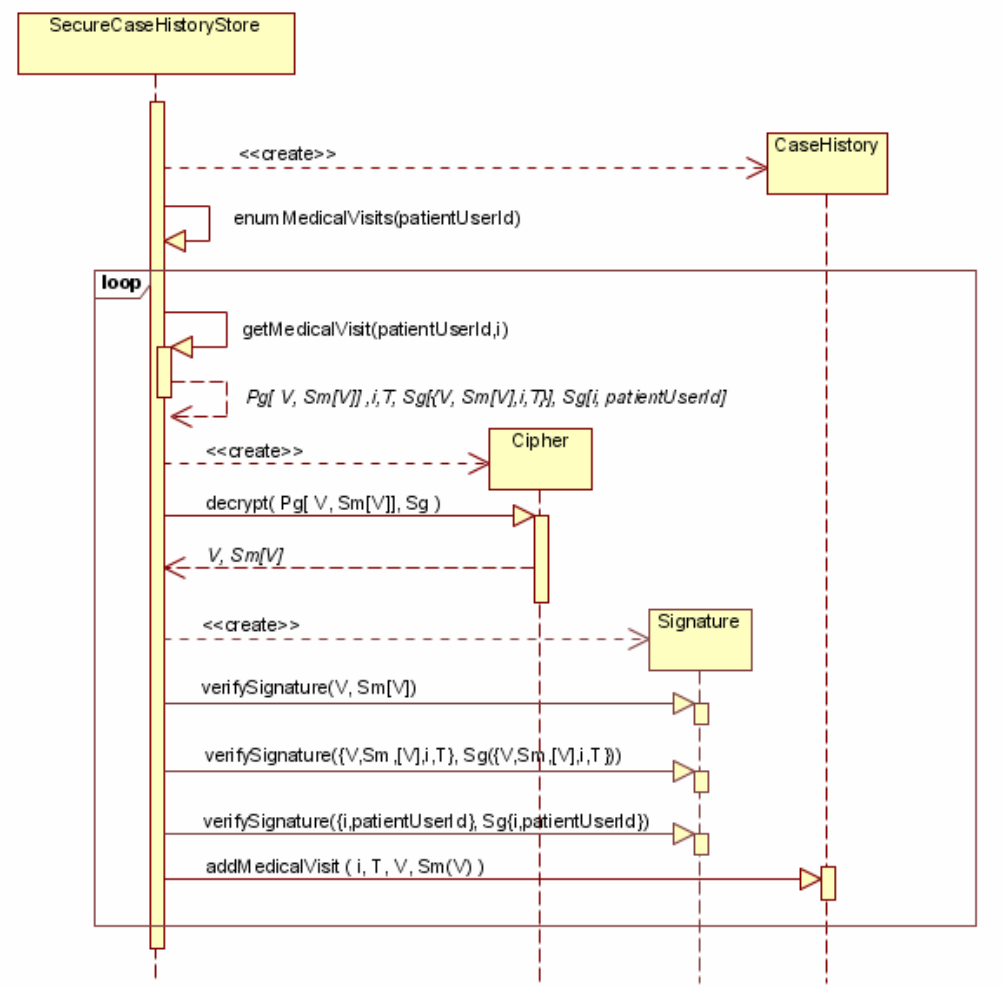

Figura 4-20. Diagrama de seqüència del cas d'us "Consulta d'un historial" (part servidor, obtenció de l'historial).

CAPÍTOL 5

Representació de les dades: XML

L'XML, acrònim de *eXtensible Markup Language* (llenguatge de marques extensible), és un metallenguatge extensible d'etiquetes desenvolupat pel World Wide Web Consortium (W3C). És una simplificació i adaptació del SGML i permet definir la gramàtica de llenguatges específics (de la mateixa manera que HTML és al seva vegada un llenguatge definit per SGML). Per tant XML no és realment un llenguatge en particular, sinó una manera de definir llenguatges per a diferents necessitats. Alguns d'aquests llenguatges que fan servir XML per a la seva definició són XHTML, SVG i MathML.

XML no ha nascut només per a la seva aplicació en Internet, sinó que es proposa com un estàndard per a l'intercanvi d'informació estructurada entre diferents plataformes. Es pot usar en bases de dades, intercanvi d'informació entre organitzacions, editors de text, fulls de càlcul i gairebé qualsevol cosa imaginable. XML és una tecnologia senzilla que té a la seu voltant altres que la complementen i la fan molt més gran i amb unes possibilitats majors. Té un paper molt important en l'actualitat ja que permet la compatibilitat entre sistemes per a compartir la informació d'una manera segura, fiable i fàcil.

Es diu documents XML "ben formats" (de l'anglès *well formed*) als documents que compleixen amb totes les definicions bàsiques de format i poden, per tant, ser analitzats correctament per qualsevol *parser* (analitzador) que compleixi amb la norma. Se separa això del concepte de validesa, que implica que no només el document està bé format sinó que també la seva estructura es correspon amb la definida en un document extern (expressada com DTD o com XML Schemas).

Una restricció del format en documents XML és que ha de estar en format text. Si les aplicacions han d'intercanviar-se algun missatge que hagi contingui dades binaries, com per exemple una signatura digital o un missatge xifrat, s'hauran de representar com si fossin text. Per aquest projecte, això s'ha fet codificant aquestes dades en base 64.

Els documents XML, opcionalment, poden estar validats. Es diu validar, en el sentit que ha de complir una estructura predeterminda: l'esquema o DTD (*Document Type Definition*).Un document DTD és una descripció d'estructura i sintaxi d'un document XML o SGML. La seva funció bàsica és la descripció del format de dades, per a usar un format comú i mantenir la consistència entre tots els documents que utilitzin el mateix DTD. D'aquesta forma, aquests documents poden ser validats, coneixen l'estructura dels elements i la descripció de les dades que porta amb si cada document, i poden a més compartir la mateixa descripció i forma de validació dintre d'un grup de treball que usa el mateix tipus d'informació.

Per a manipular les estructures XML s'ha fet servir la llibreria JDOM. JDOM és una llibreria *open source* per a manipulacions de dades XML optimitzat para Java.

Format dels documents

L'ús de XML ha estat molt útil per la implementació de la representació de les dades que es guardaran xifrades a base de dades, els historials mèdics.

Les dades que conformen una visita mèdica són orientatives de les possibilitats que ens pot donar XML. En un projecte real, la llista camps, valors admissibles i, en general, l'estructura de l'XML s'hauria d'acordar amb el client, ja que segons el tipus de visites mèdiques que es facin (revisions mèdiques de treball, revisions mèdiques per persones grans, revisions a malats de càncer, etc.) aquesta estructura variarà.

Visita mèdica

És el document XML que s'insereix quan un metge afegeix una nova visita mèdica. Aquest document ha de complir amb el següent DTD:

```
<!ELEMENT MedicalVisit
(UserId,Date,Weight,Height,MinBloodPressure,MaxBloodPressure,HeartRate,Respira
toryRate,OxigenSaturation,Temperature,CapillaryGlycemia,Observations)>
<!ELEMENT UserId (#PCDATA)>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT Weight (#PCDATA)>
<!ELEMENT Height (#PCDATA)>
<!ELEMENT MinBloodPressure (#PCDATA)>
<!ELEMENT MaxBloodPressure (#PCDATA)>
<!ELEMENT HeartRate (#PCDATA)>
<!ELEMENT RespiratoryRate (#PCDATA)>
<!ELEMENT OxigenSaturation (#PCDATA)>
<!ELEMENT Temperature (#PCDATA)>
<!ELEMENT CapillaryGlycemia (#PCDATA)>
<!ELEMENT Observations (#PCDATA)>
```

La descripció dels camps es:

- `UserId`: Identificador del pacient.
- `Date`: Data de quan es va fer la visita mèdica.
- `Weight`: Pes del pacient.
- `Height`: Alçada del pacient.
- `MinBloodPressure`: Tensió arterial mínima del pacient.
- `MaxBloodPressure`: Tensió arterial màxima del pacient.
- `HeartRate`: Freqüència cardíaca del pacient.
- `RespiratoryRate`: Freqüència respiratòria del pacient..

- `OxigenSaturation`: Saturació d'oxigen del pacient.
- `Temperature`: Temperatura del pacient.
- `CapillaryGlycemia`: Glucèmia capil·lar del pacient.
- `Observations`: Observacions rellevants sobre la visita mèdica.

Historial mèdic

És el document XML que es guarda i recupera a base de dades amb l'historial mèdic del pacient. Aquest document ha de complir amb el següent DTD:

```
<!ELEMENT CaseHistory (MedicalVisit*)>
```

on `MedicalVisit` té la estructura vista en el apartat *Visita mèdica*, pàg. 68.

Diagrama de classes

El diagrama de classes ampliat amb la representació de les dades és el següent:

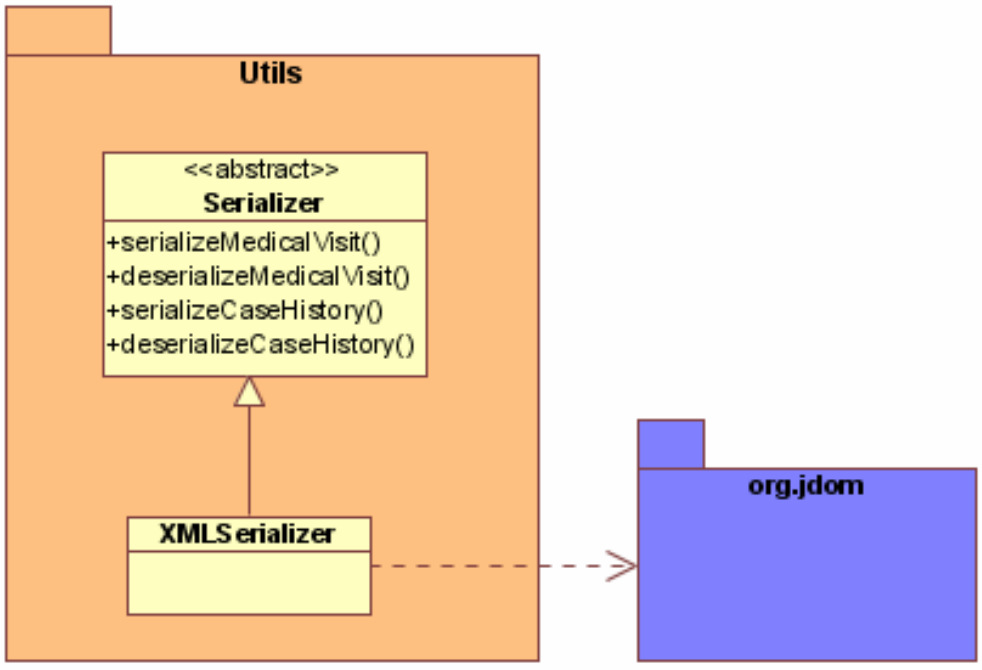

Figura 5-1.Diagrama de classes amb la representació de les dades (XML).

Per a la implementació amb Java s'ha fet servir la llibreria JDOM. Aquesta llibreria porta les classes i mètodes necessaris per manipular i validar documents XML. El procés d'instal·lació d'aquest programari està descrit a *Instal·lació JDOM*, pàg. 113.

La clase `Utils.XMLSerializer` té la responsabilitat de serialitzar/deserialitzar les visites i els historials mèdics a XML i de validar que compleixen els DTD indicats a *Visita mèdica*, pàg. 68 i *Historial mèdic*, pàg. 69.

CAPÍTOL 6

Comunicació dels components: RMI

La comunicació del diferents components és una part essencial del projecte. Els metges han de poder consultar els historials mèdics i inserir noves visites mèdiques de forma remota. Els pacients han de poder accedir als seus propis historials mèdics. La comunicació dels diferents components del sistema tradicionalment suposaria el disseny d'un protocol o mecanisme de comunicació. Per evitar la sobrecàrrega de feina, i donat que la part essencial és l'esquema criptogràfic es va optar per la utilització de la tecnologia RMI.

RMI són les sigles de *Remote Method Invocation*. Java incorpora aquesta tecnologia a l'API estàndard. RMI permet a un objecte que s'executa en una màquina virtual de Java cridar a mètodes d'objectes que estan en màquines virtuals diferents, ja sigui en el mateix ordinador o en un de remot.

RMI consta d'un servidor on s'executen diferents instàncies de les classes servidores que es necessiten i d'uns clients que criden a aquestes classes o mètodes servidores. Una aplicació client típica obté una referència d'un o més objectes remots en el servidor i en crida els seus mètodes. Les aplicacions que volen emprar els mètodes remots únicament necessiten conèixer la interfície del servidor, és a dir, els mètodes que ofereix la classe que està al servidor. La implementació d'aquesta interfície està oculta i el client no arriba mai a saber quin és codi que s'està executant.

En el projecte s'ha decidit d'utilització de RMI sobre altres tecnologies (com Web Services) per aconseguir una reducció del temps de desenvolupament.

Funcionament de la comunicació amb RMI

L'arquitectura RMI pot veure's com un model de quatre capes:

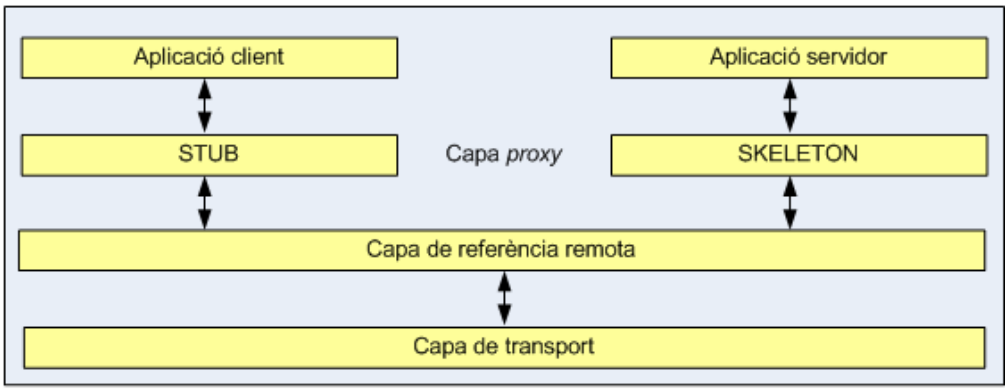

Figura 6-1.Arquitectura RMI.

La primera capa és la d'**aplicació** i es correspon amb la implementació real de les aplicacions client i servidor. Aquí tenen lloc les crides a alt nivell per a accedir i exportar objectes remots. Qualsevol aplicació que vulgui que els seus mètodes estiguin disponibles per al seu accés per clients remots ha de declarar aquests mètodes en una interfície que estengui `java.rmi.Remote`. Aquesta interfície s'usa bàsicament per a "marcar" un objecte com remotament accessible. Una vegada que els mètodes han estat implementats, l'objecte ha de ser exportat. Això pot fer-se de forma implícita si l'objecte estén la classe `UnicastRemoteObject` (paquet `java.rmi.server`), o pot fer-se de forma explícita amb una cridada al mètode `exportObject()` del mateix paquet.

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

La capa 3 és la de **referència remota**, i és responsable del maneig de la part semàntica de les invocacions remotes. També és responsable de la gestió de la replicació d'objectes i realització de tasques específiques de la implementació amb els objectes remots, com l'establiment de les persistències semàntiques i estratègies adequades per a la recuperació de connexions perdudes. En aquesta capa s'espera una connexió de tipus *stream* (*stream-oriented connection*) des de la capa de transport.

La capa 4 és la de **transport**. És la responsable de realitzar les connexions necessàries i maneig del transport de les dades d'una màquina a una altra. El protocol de transport subjacent per a RMI és JRMP (*Java Remote Method Protocol*), que solament és "comprès" per programes Java.

El funcionament de la tecnologia RMI és el següent. El servidor RMI estableix un contracte amb les aplicacions remotes. Això vol dir que el servidor informa de les classes i mètodes que estaran disponibles. Aquest pas es fa creant una interfície que serà coneguda per tothom. Alhora el servidor RMI té la implementació de la interfície.

Quan la classe està implementada s'ha de procedir a crear els punts de connexió que utilitzaran els aplicatius remots. El resultat d'aquest procediment crea dues classes que els clients han de conèixer. Aquestes classes es coneixen com Stub-

Skeleton. Un cop preparat, les aplicacions remotes ja poden accedir a fer les instancies dels objectes remots.

Per a poder fer invocacions remotes entre clients i servidor , cal un mecanisme per a localitzar objectes remots , el registre RMI (`rmiregistry`) , on el servidor publica la interfície amb els mètodes remots, i els clients (mitjançant la interfície) els poden invocar

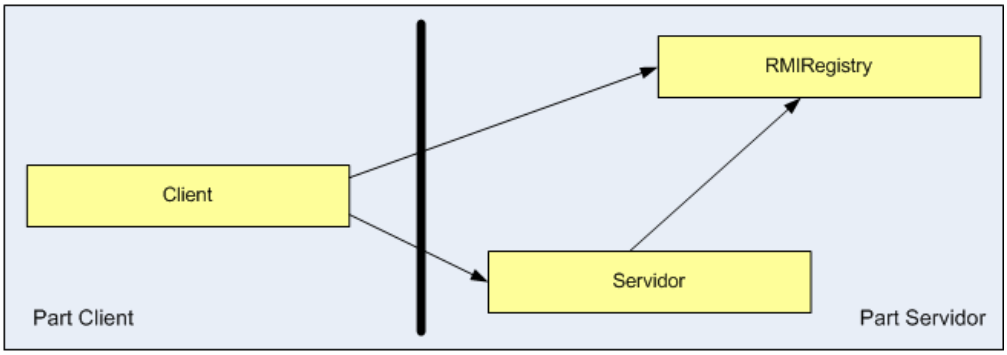

Figura 6-2. Representació d'una crida remota mitjançant RMI.

Diagrama de classes

El diagrama de classes ampliat per la comunicació amb RMI es el següent:

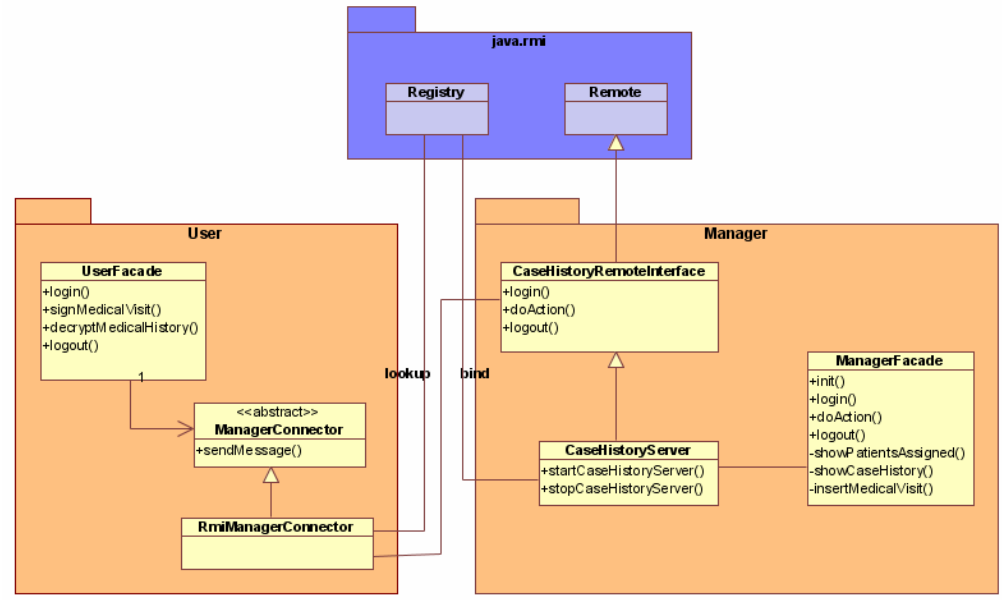

Figura 6-3. Diagrama de classes que intervenen en la comunicació amb RMI.

Descripció de les classes utilitzades

La classe `Manager.CaseHistoryRemoteInterface` defineix d'interfície remota.

```
public interface CaseHistoryRemoteInterface extends Remote {
    byte[] login( byte[] message ) throws RemoteException;
    byte[] doAction( byte[] message ) throws RemoteException;
    byte[] logout( byte[] message ) throws RemoteException;
}
```

La classe `Manager.CaseHistoryServer` es registra (`bind`) al registre RMI (classe `java.rmi.Registry`) e implementa els mètodes de `Manager.CaseHistoryRemoteInterface`.

```
public class CaseHistoryServer implements CaseHistoryRemoteInterface {
    private ManagerFacade m_managerFacade ;

     public CaseHistoryServer ( ManagerFacade managerFacade )
     {
          m_managerFacade = managerFacade ;
     }

   public byte[] login(byte[] message ) throws RemoteException
     {
          try
          {
               return m_managerFacade.login( message ) ;
          }
          catch ( CaseHistoryException e )
          {
               throw new RemoteException( e.toString() ) ;
          }
     }
     ...
     public void startCaseHistoryServer( )
     {
          CaseHistoryRemoteInterface stub = (CaseHistoryRemoteInterface)
              UnicastRemoteObject.exportObject( this, 0 );
          LocateRegistry.getRegistry().bind("CaseHistoryRemoteInterface",
                                            stub);
     }
     ...
```

La classe `User.RmiManagerConnector` busca al registre RMI el *stub* `Manager.CaseHistoryRemoteInterface` per realitzar les crides remotes.

```
public byte[] sendMessage( String messageType, byte[] message ) throws
ProcessException
{
     try
     {
          Registry registry = LocateRegistry.getRegistry(m_host);
```

```
            CaseHistoryRemoteInterface stub = (CaseHistoryRemoteInterface)
                                registry.lookup("CaseHistoryRemoteInterface");
            if ( messageType == "login" )
                return stub.login( message ) ;
            else if ( messageType == "doAction" )
                return stub.doAction ( message ) ;
            else // if ( messageType == "logout" )
                return stub.logout ( message ) ;
        }
...
```

CAPÍTOL 7

Gestió de la informació: base de dades

La gestió de la informació es fa necessària ja que ens permet mantenir l'estat persistent sobre els usuaris i els historials mèdics del pacients. El gestor ha de poder guardar i recuperar informació sobre com identificar els usuaris (el seus certificats) i poder guardar de forma segura (garantint la privadesa, la integritat i l'autenticitat) les visites mèdiques que insereixen els metges.

Una base de dades ens permetrà que les dades siguin perdurables en el temps. Una vegada donats d'alta els usuaris o afegides dades mèdiques, aquestes es podran continuar consultat en sessions posteriors. També amb programes de *Data Ware House* i tècniques de mineria de dades es podria extreure informació de les dades emmagatzemades, com quines són les afeccions més comuns, a quina edat, si hi ha metges amb sobrecarrega de treball, etc.

Model relacional de base de dades

A la següent figura està representat el model relacional per la implementació de la persistència de dades.

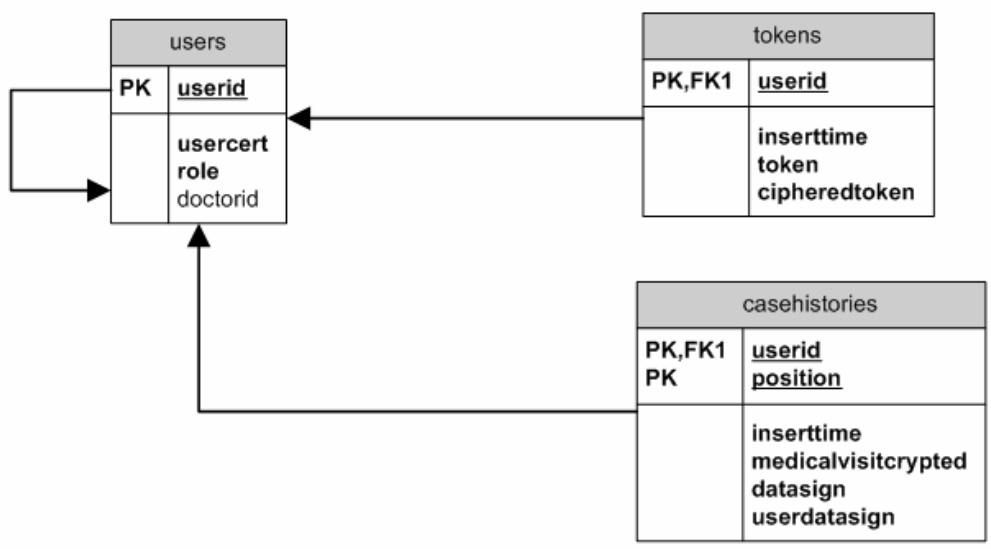

Figura 7-1.Model relacional de base de dades.

Les relacions són les següents:

- `users` ► `users`: Relació 1 a 0, 1. Relaciona els usuaris de tipus pacient de la taula `users` amb seu metge. Per a un usuari només pot haver com a màxim un metge.
- `users` ► `token`: Relació 1 a 0, 1. Relaciona els usuaris de la taula `users` amb el testimonis de la taula `tokens`. Per a un usuari només pot haver coma màxim un testimoni.
- `users` ► `casehistories`: Relació 1 a *. Relaciona els usuaris de la taula `users` amb les visites mèdiques de la taula `casehistories`. Per a un usuari pot haver més d'una visita mèdica. El conjunt ordenat de visites mèdiques d'un usuari conforma el seu historial mèdic.

Implementació de la base de dades

Hem usat el gestor de base de dades MySQL perquè és de lliure distribució (i gratuït) i està disponible en múltiples plataformes (Linux, Windows i Mac entre altres). La instal·lació del programari necessari es pot veure a *Instal·lació MySQL*, pàg. 114, i *Instal·lació Java JDBC de MySQL*, pàg. 115. L'script que crea la base de dades buida està descrita a *Creació de les taules buides de* la base dades, pàg. 116.

Encara que segons el disseny no faria falta que la base de dades suporti transaccions, s'ha decidit que ho faci per facilitar futures ampliacions en les característiques del projecte. S'ha fet servir l'*Engine Storage InnoDB* perquè proporciona al MySQL un entorn transaccional amb les capacitats de *commit*, *rollback*, *crash* i *recovery*.

Les dades binaries (dades xifrades, signatura) es podrien haver emmagatzemar en format binari fent servir el tipus de dades BLOB (*Binary Large Object*) per a gestionar dades binàries grans. Es va provar, però es van experimentar problemes amb la recuperació i posterior comparació (desxifrat i comparació de signatura), així que finalment es va optar guardar aquestes dades en un camp de text en base 64. Com que no tots els SGDB tenen el tipus de dades BLOB (o tenen limitacions sobre aquest tipus de camps), aquesta decisió també facilitarà una futura migració de MySQL a altre tipus de SGDB (com Oracle o Informix).

Un altre tipus de dades que hem de tenir en compte és el temps. S'ha triat el tipus de dades `bigint` en comptes de l'altre tipus (com el `datetime`) que ens proporciona MySQL. Aquesta decisió s'ha fet per permetre que sigui més fàcil una futura migració de MySQL a altre tipus de SGDB.

Taula `users`

Aquesta taula emmagatzema les dades sobre els usuaris. Els camps es resumeixen en la següent taula:

Camp	*Tipus*	*Descripció*
`userid`	Caràcter	Identificador de l'usuari.

Camp	*Tipus*	*Descripció*
`usercert`	Binari	Certificat del usuari.
`role`	Caràcter	Rol del usuari (pacient o metge).
`doctorid`	Caràcter	Identificador del metge de l'usuari.

Taula `tokens`

Aquesta taula emmagatzema de forma integra les dades sobre els testimonis del usuaris que estan autenticats (tenen una sessió oberta) en el sistema. Els camps es resumeixen en la següent taula:

Camp	*Tipus*	*Descripció*
`userid`	Caràcter	Identificador de l'usuari.
`inserttime`	Data	Data d'inserció del testimoni.
`token`	Caràcter	Testimoni en clar i en base 64. Necessari per la navegació a la taula.
`cipheredtoken`	Binari	Testimoni xifrat amb la clau del gestor i en base 64. Dona la integritat al camp `token`.

Taula `casehistories`

Aquesta taula emmagatzema de forma integra i privada les dades sobre les visites mèdiques dels pacients. Els camps es resumeixen en la següent taula:

Camp	*Tipus*	*Descripció*
`userid`	Caràcter	Identificador de l'usuari.
`position`	Numèric	Posició de la visita mèdica al historial mèdic.
`medicalvisitcrypted`	Caràcter	Dades de la visita mèdica xifrades amb la clau del gestor i en base 64.
`datasign`	Caràcter	Signatura del sistema gestor de les dades de la visita mèdica en base 64.
`userdatasign`	Caràcter	Signatura de l'usuari de les dades de la visita mèdica en base 64.

Diagrama de classes

A la següent figura està representat el diagrama de classes (simplificat) on apareixen les noves classes que donen funcionalitat per la gestió de la informació, es a dir, accés a la base de dades.

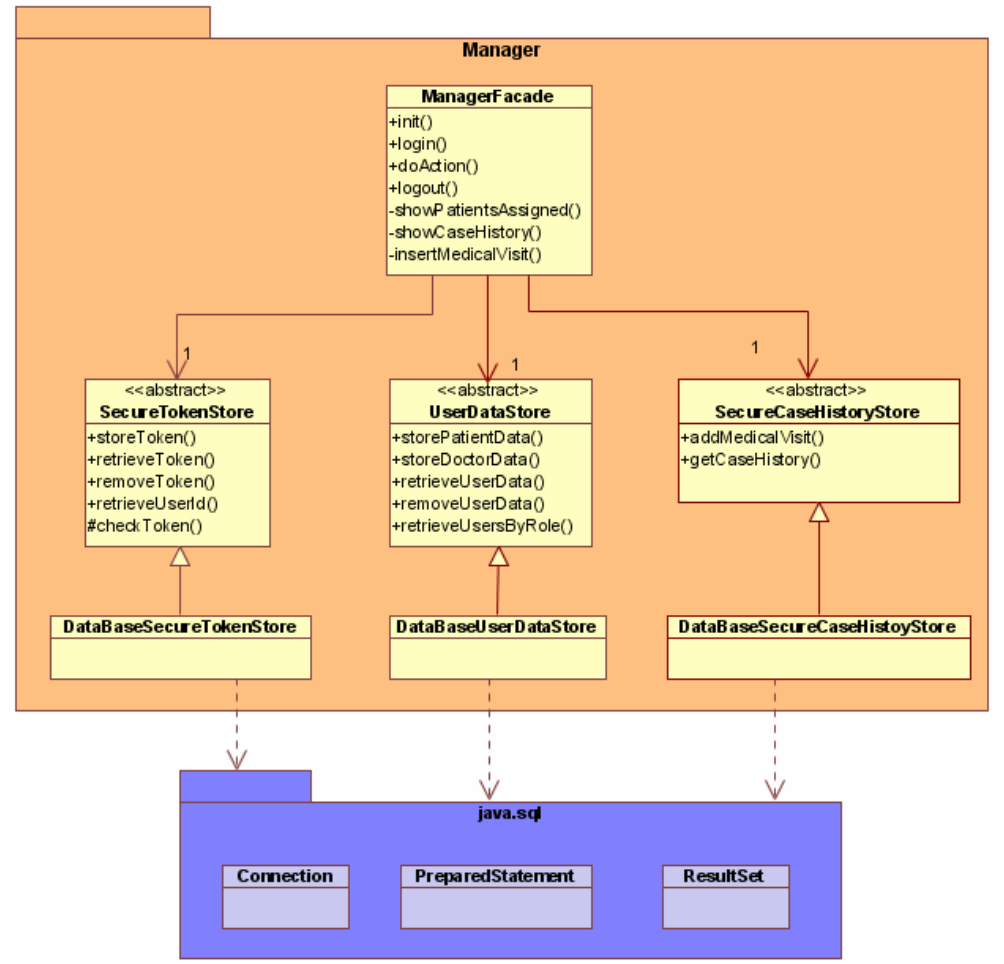

Figura 7-2. Diagrama de classes que intervenen en la gestió de la informació (base de dades).

No hem necessitat instal·lar de cap programari addicional, hem usat JDBC proporcionat per l'API estàndard de Java.

L'accés a les taules a la base de dades es fa mitjançant una classes específiques a les quals només accedeix la part servidor del gestor.

S'han afegit tres noves classes:

- `Manager.DataBaseUserDataStore`: Es la responsable de gestionar la informació sobre els usuaris.

- `Manager.DataBaseSecureCaseHistoryStore`: Es la responsable de gestionar de forma segura la informació sobre els historials mèdics Aquesta classe només s'encarrega de la comunicació amb la base de dades, mentre que la classe de la que hereta (`Manager.SecureCaseHistoryStore`) té la responsabilitat de xifrar, signar, desxifrar i verificar les signatures.

- `Manager.DataBaseSecureTokenStore`: Es la responsable de gestionar de forma segura la informació sobre els testimonis Aquesta classe només s'encarrega de la comunicació amb la base de dades, mentre que la classe de la que hereta (`Manager.SecureTokenStore`) té la responsabilitat de xifrar, signar, desxifrar, verificar les signatures i verificar la caducitat dels testimonis.

CAPÍTOL 8

Interfície d’usuari

La interfície d'usuari és la forma que els usuaris poden comunicar-se amb l’aplicatiu, i comprèn tots els punts de contacte entre l'usuari i el programari.

Donat que la interfície d’usuari no es punt crític en el desenvolupament del projecte, s’ha decidit simplificar fent un interfície en línea de comandes per afavorir el desenvolupament de noves capacitats criptogràfiques (veure *Seguiment del projecte*, pàg. 26).

La interfície en línea de comandes s’ha realitzat mitjançant menús, de tal forma que, encara que sigui una interfície austera comparada amb les interfícies gràfiques, aquesta sigui totalment funcional per l’usuari.

D’aquesta manera s’han fet dos aplicatius, un per l’administrador de sistema i un altre per els metges i pacients.

Diagrama de classes

A la següent figura està representat el diagrama de classes (simplificat) on apareixen les noves classes que donen funcionalitat per la interfície d'usuari.

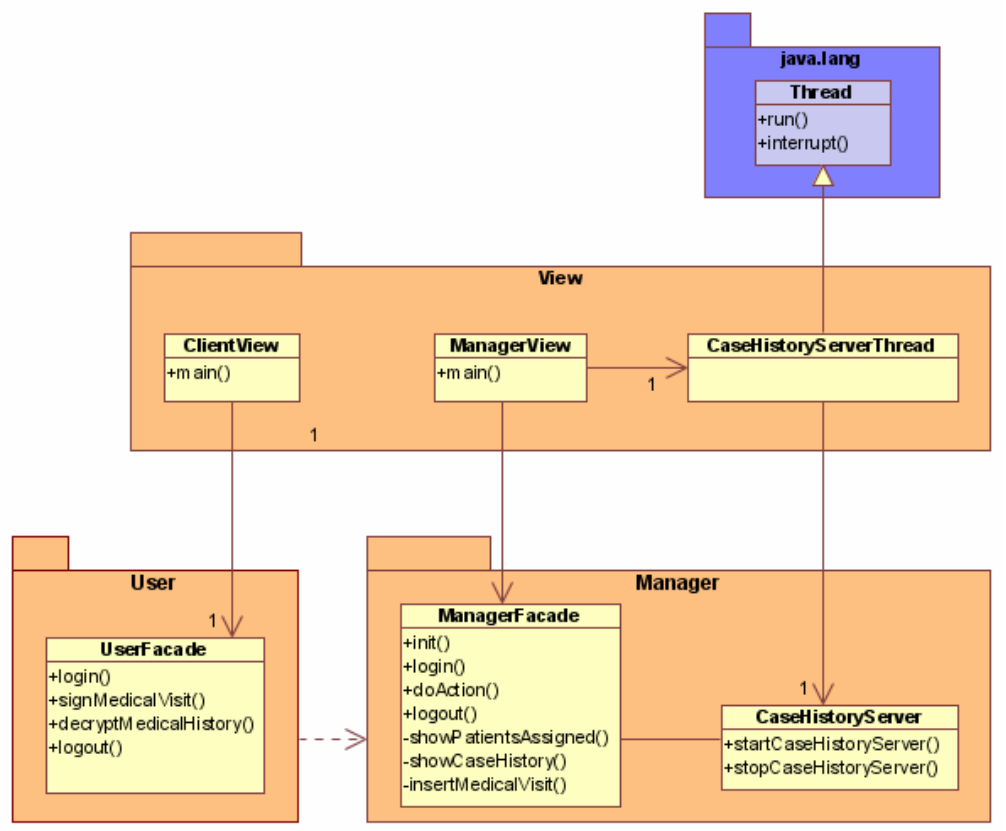

Figura 8-1.Diagrama de classes que intervenen en la interfície d'usuari.

S'han afegit tres noves classes:

- `View.ClientView`: Dona la interfície d'usuari a metges i pacients.
- `View.ManagerView`: Dona la interfície d'usuari per l'administrador del sistema.
- `View.CaseHistoryServerThread`: Permet engegar i parar el servidor en un altre fil (*thread*), de tal manera que l'aplicació principal (`View.ManagerView`) pot continuar funcionant.

Gestor del sistema

Aquest aplicatiu ens permet:

- Engegar i aturar el servidor: Veure *Engegar/aturar el servidor*, pàg. 85.
- Afegir i esborrar usuaris: Veure *Afegir usuaris*, pàg. 86 i *Esborrar usuaris*, pàg. 86.
- Assignar pacients a un metge: Veure *Assignar pacients a un metge*, pàg. 86.

Posta en marxa de l'aplicatiu gestor del sistema

Per posar en funcionament l'aplicatiu del gestor del sistema em d'executar la següent instrucció:

```
c:\PFC\project\src>java casehistory.view.ManagerView "..\data\seed"
```

El paràmetre indica la ruta on es troba l'arxiu amb la llavor de número pseudo-aleatoris.

A continuació ens demana la ruta del PKCS#12 del sistema gestor i la seva contrasenya:

```
Introduiu la ruta del PKCS#12 del gestor: C:\PFC\project\data\gestor.p12
Introduiu la contrasenya del gestor: demodemo
```

Una vegada fet això se'ns mostra el menú principal de l'aplicatiu:

```
**** Menú principal ****
1 – Engegar el servidor
2 - Afegir un nou usuari
3 - Assignar pacient a un metge
4 - Esborrar un usuari
0 - Sortir
Escolli una opció:
```

Engegar/aturar el servidor

Nota:
El servidor de registro RMI ha d'estar en funcionament per el correcte funcionament d'aquesta funcionalitat.

Per engegar el servidor, escollim l'opció 1 del menú principal. Se'ns mostrarà un missatge de informació:

```
El servidor s'ha engegat
Pulsi 'return' per continuar...
```

Al tornar al menú principal se'ns mostra l'opció per aturar el servidor:

```
**** Menú principal ****
1 – Aturar el servidor
2 - Afegir un nou usuari
3 - Assignar pacient a un metge
4 - Esborrar un usuari
0 - Sortir
Escolli una opció:
```

Si escollim l'opció per aturar el servidor, el servidor s'atura i se'ns mostra un missatge de informació:

```
El servidor s'ha aturat
Pulsi 'return' per continuar...
```

Afegir usuaris

Per afegir un nou usuari, escollim l'opció 2 del menú principal. Se'ns demanarà el certificat del nou usuari:

```
**** Afegir un nou usuari ****
Introduiu la ruta del certificat de l'usuari: C:\PFC\project\data\metge1.crt
Afegit usuari 11111111-M amb rol METGE.
Pulsi 'return' per continuar...
```

Una vegada afegit l'usuari ens mostrarà informació sobre l'identificador de l'usuari i el rol que té a l'aplicació.

Assignar pacients a un metge

Per assignar pacients a un metge, escollim l'opció 3 del menú principal. Se'ns que escollim al metge al qual volem assignar un nou pacient:

```
**** Assignar pacient a un metge ****
Llista de metges:
1 - 11111111-M
2 - 22222222-M
0 - Tornar
Escolli un metge:
```

Una vegada escollit un metge, ens presenta la llista de pacients i amb quin metge estan actualment assignats:

```
Llista de usuaris:
1 - 33333333-P assignat a 22222222-M
2 - 44444444-P no assignat
0 - Tornar
Escolli un pacient al que assignar al metge 11111111-M : 2
Pacient 44444444-P assignat al metge 11111111-M
Pulsi 'return' per continuar...
```

Al escollir un pacient, queda assignat al metge i se'ns presenta un missatge d'informació.

Esborrar usuaris

Per esborrar un nou usuari, escollim l'opció 4 del menú principal. Se'ns presentarà una llista amb els usuaris donats d'alta en el sistema i el seu rol:

```
**** Esborrar un usuari ****
Llista de usuaris:
1 - 33333333-P (pacient)
2 - 44444444-P (pacient)
3 - 11111111-M (metge)
4 - 22222222-M (metge)
0 - Sortir
Escolli un usuari:
```

Una vegada escollit l'usuari que es vol esborrar, ens demanarà confirmació per fer-ho. Si confirmem l'esborrat, l'usuari es eliminat del sistema i se'ns presenta un missatge d'informació:

```
**** Esborrar un usuari ****
Llista de usuaris:
1 - 33333333-P (pacient)
2 - 44444444-P (pacient)
3 - 11111111-M (metge)
4 - 22222222-M (metge)
0 - Sortir
Escolli un usuari: 1
Està segur de que vol esborrar al pacient 33333333-P? (sí\no): s
Pacient 33333333-P esborrat.
Pulsi 'return' per continuar...
```

Sortir de l'aplicatiu

Per sortir de l'aplicatiu, escollim l'opció 0 del menú principal. Se'ns demanarà confirmació per sortir del programa:

```
**** Menú principal ****
1 - Aturar el servidor
2 - Afegir un nou usuari
3 - Assignar pacient a un metge
4 - Esborrar un usuari
0 - Sortir
Escolli una opció: 0
Està segur de que vol sortir? (sí\no): s
El servidor s'ha aturat
Adéu!
```

Si estava engegat, el servidor s'atura al sortir de l'aplicatiu.

Metge

Aquest aplicatiu permet a un metge:

- Veure quins pacients té assignats.
- Afegir noves visites mèdiques a pacients que tingui assignats: Veure *Afegir visites mèdiques*, pàg. 88.
- Mostrar l'historial mèdic de pacients que tingui assignats. Veure *Mostrar historial mèdic*, pàg. 89.

Posta en marxa de l'aplicatiu per metges

Per posar en funcionament l'aplicatiu del metge em d'executar la següent instrucció:

```
c:\PFC\project\src>java casehistory.view.ClientView localhost "..\data\seed"
".. \data\gestor.crt"
```

El primer paràmetre indica la URL del servidor, el segon la ruta on es troba l'arxiu amb la llavor de números pseudo-aleatoris i el tercer el certificat del sistema gestor.

Nota:
El servidor de sistema ha d'estar engegat (veure Engegar/aturar el servidor, pàg. 85, per el correcte funcionament d'aquest aplicatiu.

A continuació ens demana la ruta del PKCS#12 del metge i la seva contrasenya:

```
Introduiu la ruta del vostre PKCS#12: C:\PFC\project\data\metge1.p12
Introduiu la vostre contrasenya: demodemo
```

Una vegada s'ha autenticat el metge, se'ns mostra la llista de pacients que té assignat el metge:

```
**** Llista de pacients del metge 11111111-M ****
1 - 33333333-P
2 - 44444444-P
0 - Sortir
Escolli un pacient:
```

Al escollir un pacients se'ns mostra el menú d'accions que es poden fer sobre un pacient:

```
Pacient: 33333333-P
1 - Afegir visita mèdica
2 - Mostrar historial mèdic
0 - Tornar al llistat de pacients
Escolli una opció:
```

Afegir visites mèdiques

Per afegir una nova visita mèdica, escollim l'opció 1 del menú del pacient. Se'ns demanaran les dades de la visita:

```
**** Afegir visita mèdica al pacient 33333333-P ****
Pes: 70
Alçada: 168
Tensió arterial max.: 8
Tensió arterial min.: 12
Freqüència cardíaca: 72
Freqüència respiratoria: 14
Saturació de oxígen: 99
Temperatura: 36.5
Glucemia capilar: 70
Observacions: Tot OK!
Visita médica afejida!
Pulsi 'return' per continuar...
```

Una vegada inserides totes les dades, la visita mèdica es guarda i se'ns presenta un missatge d'informació.

Mostrar historial mèdic

Per mostrar l'historial mèdic del pacient, escollim l'opció 2 del menú del pacient seleccionat. Se'ns mostren la llista de visites mèdiques que ha rebut el pacient i la data en que es van fer:

```
**** Mostrar historial mèdic del pacient 33333333-P ****
1 - Visita mèdica 1 - Thu Jan 04 21:51:31 CET 2007
2 - Visita mèdica 2 - Thu Jan 04 21:59:58 CET 2007
3 - Visita mèdica 3 - Thu Jan 04 22:01:03 CET 2007
4 - Visita mèdica 4 - Thu Jan 04 22:01:16 CET 2007
0 – Tornar
```

Si seleccionem una de les visites, se'ns mostren les dades que es van inserir en aquella visita:

```
Data: Thu Jan 04 21:51:31 CET 2007
Pes: 70
Alçada: 168
Tensió arterial max.: 8
Tensió arterial min.: 12
Freqüéncia cardíaca: 72
Freqüéncia respiratoria: 14
Saturació de oxígen: 99
Temperatura: 36.5
Glucemia capilar: 70
Observacions: Tot OK!
Pulsi 'return' per continuar...
```

Sortir de l'aplicatiu

Per sortir de l'aplicatiu, escollim l'opció 0 del menú principal:

```
**** Llista de pacients del metge 11111111-M ****
1 - 33333333-P
2 - 44444444-P
0 - Sortir
Escolli un pacient: 0
Adéu 11111111-M!
```

Pacient

Aquest aplicatiu ens permet veure l'historial mèdic del pacient que s'ha autenticat:

Posta en marxa de l'aplicatiu per pacients

Per posar en funcionament l'aplicatiu del pacient em d'executar la següent instrucció:

```
c:\PFC\project\src>java casehistory.view.ClientView localhost "..\data\seed"
".. \data\gestor.crt"
```

El primer paràmetre indica la URL del servidor, el segon la ruta on es troba l'arxiu amb la llavor de número pseudo-aleatoris i el tercer el certificat del sistema gestor.

Nota:
El servidor de sistema ha d'estar engegat (veure Engegar/aturar el servidor, pàg. 85, per el correcte funcionament d'aquest aplicatiu.

A continuació ens demana la ruta del PKCS#12 del metge i la contrasenya:

```
Introduiu la ruta del vostre PKCS#12: C:\PFC\project\data\pacient1.p12
Introduiu la vostre contrasenya: demodemo
```

Una vegada s'ha autenticat el pacient, se'ns mostra el menú principal:

```
**** Menú principal ****
1 - Mostrar historial mèdic
0 - Sortir
Escolli una opció:
```

Mostrar historial mèdic

Per mostrar l'historial mèdic del pacient autenticat, escollim l'opció 1 del menú principal. Se'ns mostren la llista de visites mèdiques que ha rebut el pacient i la data en que es van fer:

```
**** Mostrar historial mèdic del pacient 33333333-P ****
1 - Visita mèdica 1 - Thu Jan 04 21:51:31 CET 2007
2 - Visita mèdica 2 - Thu Jan 04 21:59:58 CET 2007
3 - Visita mèdica 3 - Thu Jan 04 22:01:03 CET 2007
4 - Visita mèdica 4 - Thu Jan 04 22:01:16 CET 2007
0 – Tornar
```

Si seleccionem una de les visites, se'ns mostren les dades que es van inserir en aquella visita:

```
Data: Thu Jan 04 21:51:31 CET 2007
Pes: 70
Alçada: 168
Tensió arterial max.: 8
Tensió arterial min.: 12
Freqüéncia cardíaca: 72
Freqüéncia respiratoria: 14
Saturació de oxígen: 99
Temperatura: 36.5
Glucemia capilar: 70
Observacions: Tot OK!
Pulsi 'return' per continuar...
```

Sortir de l'aplicatiu

Per sortir de l'aplicatiu, escollim l'opció 0 del menú principal.

```
**** Menú principal ****
1 - Mostrar historial mèdic
```

```
0 - Sortir
Escolli una opció: 0
Adéu 33333333-P!
```

CAPÍTOL 9

Jocs de proves

Per dels jocs de proves s'han realitzat tests unitaris sobre les principals classes del sistema. Els tests unitaris son programes que han de instanciar la classe que volem provar, criden als seus mètodes passant uns determinats paràmetres concrets i comproven que retornen els resultats adequats.

Els test unitaris aporten tota una sèrie d'avantatges:

- El conjunt de test unitaris proporciona constant retroalimentació de que cadascun dels components segueix funcionant correctament.
- Els test unitaris actuen com una documentació que no es queda obsoleta, al contrario que altres tipus de documentació.
- Quan el codi de producció és refactoritzat per alguna raó i passa els tests unitaris, queda clar que el codi continua funcionant, i el desenvolupador es pot moure a la següent tasca.
- El programari tendeix a estar millor dissenyat, això és, menys acoblat i més fàcilment mantenible, perquè el desenvolupador és lliure de fer decisions de disseny i refactoritzar en qualsevol moment amb la confiança que el programari funcionarà sempre que passi els tests unitaris.
- El conjunt de tests actua com una xarxa de seguretat contra regressions en els bugs: Si es troba un bug, el desenvolupador ha de crear un test que reproduexi el bug i després modificar el codi de producció per a eliminar el bug. En successives execucions dels test, totes les correccions de bugs són verificades.
- El temps de depuració es redueix.

Per la realització dels test unitaris s'ha decidit fer servir el *framework* JUnit. JUnit es un conjunt de classes (*framework*) simple i *open source* creat per Erich Gamma y Kent Beck que permet realitzar la execució de classes Java de manera controlada.

Els test unitaris s'han realitzat sobre les classes més representatives del sistema. Aquestes classes són:

- `Utils.MedicalVisit`: Serialització de les dades (XML).
- `Utils.CaseHistory`: Serialització de les dades (XML).
- `Manager.DataBaseUserDataStore`: Accés a base de dades (dades d'usuaris).
- `Manager.DataBaseSecureTokenStore`: Accés a base de dades (dades dels testimonis).

- `Manager.DataBaseSecureCaseHistoryStore`: Accés a base de dades (dades mèdiques).
- `Manager.CaseHistoryServer`: Accés al servidor remot (RMI).
- `User.UserFacade`: Funcionalitat general de sistema.

Per poder fer les proves necessàries a mida que avançava el projecte s'han tingut que fer certes classes auxiliars que simulaven classes que es desenvoluparien més endavant. Aquestes classes són:

- `Utils.TLVSerializer`: Serialització, per proves abans de la fase de representació de dades (XML).
- `User.LocalManagerConnector`: Accés a un servidor en memòria, per proves abans de la fase de comunicació dels components (RMI).
- `Manager.MemoryUserDataStore`: Accés a dades, per proves abans de la fase de gestió de la informació (base de dades).
- `Manager.MemorySecureTokenStore`: Accés a dades, per proves abans de la fase de gestió de la informació (base de dades).
- `Manager.MemorySecureCaseHistoryStore`: Accés a dades, per proves abans de la fase de gestió de la informació (base de dades).

Execució dels tests unitaris

Per executar els jocs de proves s'ha de tenir instal·lat tot el programari indicat a *Instal·lació del sistema*, pàg. 113.

Una vegada instal·lat el programari, creades les taules buides necessàries i compilat l'aplicatiu del projecte, s'han de compilar els test unitaris:

```
javac UserFacadeTest.java
javac MedicalVisitTest.java
javac CaseHistoryTest.java
javac CaseHistoryServerTest.java
javac DataBaseUserDataStoreTest.java
javac DataBaseSecureTokenStoreTest.java
javac DataBaseSecureCaseHistoryStoreTest.java
```

Per executar els tests unitaris s'han d'executar les següents instruccions:

```
javac UserFacadeTest.java
javac MedicalVisitTest.java
javac CaseHistoryTest.java
javac CaseHistoryServerTest.java
javac DataBaseUserDataStoreTest.java
javac DataBaseSecureTokenStoreTest.java
javac DataBaseSecureCaseHistoryStoreTest.java
```

El resultat de l'execució dels test unitaris a de ser el següent

```
c:\PFC\project\unittest>java UserFacadeTest
.***                                                                  ***
***                    Welcome to the IAIK JCE Library                ***
```

```
***                                                                     ***
*** This version of IAIK JCE is licensed for educational and research use ***
*** and evaluation only. Commercial use of this software is prohibited.   ***
*** For details please see http://jce.iaik.tugraz.at/sales/licences/.     ***
*** This message does not appear in the registered commercial version.    ***
***                                                                     ***

..........
Time: 5,609

OK (11 tests)


c:\PFC\project\unittest>java MedicalVisitTest
..
Time: 0,235

OK (2 tests)


c:\PFC\project\unittest>java CaseHistoryTest
..
Time: 0,234

OK (2 tests)


c:\PFC\project\unittest>java DataBaseUserDataStoreTest
.***                                                                     ***
***                    Welcome to the IAIK JCE Library                  ***
***                                                                     ***
*** This version of IAIK JCE is licensed for educational and research use ***
*** and evaluation only. Commercial use of this software is prohibited.   ***
*** For details please see http://jce.iaik.tugraz.at/sales/licences/.     ***
*** This message does not appear in the registered commercial version.    ***
***                                                                     ***

..
Time: 1,265

OK (3 tests)


c:\PFC\project\unittest>java DataBaseSecureTokenStoreTest
.***                                                                     ***
***                    Welcome to the IAIK JCE Library                  ***
***                                                                     ***
*** This version of IAIK JCE is licensed for educational and research use ***
*** and evaluation only. Commercial use of this software is prohibited.   ***
*** For details please see http://jce.iaik.tugraz.at/sales/licences/.     ***
*** This message does not appear in the registered commercial version.    ***
***                                                                     ***
```

```
..
Time: 2,156

OK (3 tests)


c:\PFC\project\unittest>java DataBaseSecureCaseHistoryStoreTest
.***                                                                          ***
***                    Welcome to the IAIK JCE Library                       ***
***                                                                          ***
*** This version of IAIK JCE is licensed for educational and research use ***
*** and evaluation only. Commercial use of this software is prohibited.   ***
*** For details please see http://jce.iaik.tugraz.at/sales/licences/.     ***
*** This message does not appear in the registered commercial version.    ***
***                                                                          ***

.
Time: 2,875

OK (2 tests)


c:\PFC\project\unittest>start rmiregistry

c:\PFC\project\unittest>java CaseHistoryServerTest
.***                                                                          ***
***                    Welcome to the IAIK JCE Library                       ***
***                                                                          ***
*** This version of IAIK JCE is licensed for educational and research use ***
*** and evaluation only. Commercial use of this software is prohibited.   ***
*** For details please see http://jce.iaik.tugraz.at/sales/licences/.     ***
*** This message does not appear in the registered commercial version.    ***
***                                                                          ***

.....
Time: 4,703

OK (6 tests)


c:\PFC\project\unittest>
```

El resultat de tots els 29 tests ha de ser **OK**.

CAPÍTOL 10

Valoració econòmica

En aquest apartat es passa a realitzar un càlcul aproximat del cost financer del desenvolupament del projecte.

A l'hora d'efectuar el càlcul del cost de mà d'obra, s'ha comptabilitzat l'hora a 45€ (es consideren aquestes xifres com sou d'un enginyer sènior).

Recurs	*Hores*	*Cost*
Mà d'obra	300 hores	13.500 €

Taula 10-1 Taula de costos de ma d'obra.

Per al càlcul del cost del programari s'han comptabilitzat el preu de la compra d'una llicència per una PIME de cadascuna de les aplicacions utilitzades per al desenvolupament del projecte.

Software	*Tasca*	*Cost*
Windows XP Professional Service Pack 2	Tot el projecte	460,95 €
MS Project	Planificació	868 €
Visual Paradigm for UML Community Edition	Disseny	118,50 $ = 90,5 € (1$=1,30€)
UltraEdit-32	Implementació	37,63 €
OpenSSL	Implementació	Gratuït
IAIK	Implementació	500 €
JDOM	Implementació	Gratuït
JUnit	Implementació	Gratuït
MySQL	Implementació	Gratuït
Microsoft Office	Documentació	670 €
Adobe Acrobat	Documentació	648,44 €
Total		**3.275,52 €**

Taula 10-2. Taula de costos del programari

Finalment caldria afegir els recursos hardware utilitzats:

Recurs	*Cost*
Ordinador personal PC	728,47 €

Taula 10-3. Taula de costos del maquinari

D'aquesta manera el cost total del projecte es resumeix en la següent taula:

Tipus de recurs	*Cost*
Mà d'obra	13.500
Software	3.275,52 €
Hardware	728,47 €
***Cost total de desenvolupament del projecte*[(1)]**	**17.503,99 €**

(1) En aquest cost caldria afegir els costos fixos, com llum, ADSL, etc.

Taula 10-4. Taula de costos totals del projecte

CAPÍTOL 11

Treball futur

Un projecte informàtic rares vegades es pot donar per acabat completament, sempre hi ha funcionalitats que no estan cobertes o millores que s'hi voldria afegir. Aquest capítol està dedicat a les possibles ampliacions que es podrien implementar en el sistema.

Criptografia i seguretat:

En l'apartat de criptografia i seguretat unes tasques futures serien:

- Autenticació dels usuaris mitjançant *smart-cards*, en comptes de PKCS#12 més contrasenya. Addicionalment també es podrien requerir dades biomètriques per identificar l'usuari. D'aquesta manera s'acomplirien les màximes de l'autenticació d'usuaris: una cosa que tens (l'*smart-card*), una cosa que saps (el pin de l'*smart-card*) i una cosa que ets (la dada biomètrica).
- Emmagatzematge de les claus del gestor en un HSM (*Hardware Security Module*), en comptes d'un PKCS#12 més contrasenya. Així afegiríem més seguretat física a la gestió de les claus del gestor.
- Possibilitat de configurar els algorismes de *hash*, xifrat i signatura que es fan servir en el sistema.
- Afegir el requeriment de que l'administrador s'hagi d'autenticar contra el sistema per gestionar-lo. Els administradors serien un tipus d'usuari més del sistema amb un rol específic.
- Permetre la comprovació dels certificats amb el certificat de la CA. Això voldria dir incloure un repositori de CAs de confiança.
- Comprovació de l'estat de revocació dels certificats dels usuaris. Es podria realitzar mitjançant la consulta de CRLs o delegar en una VA (*Validation Authority*) amb un protocol com l'OCSP (*Online Certificate Status Protocol*).
- Guardar les modificacions en els historials mèdics amb una marca de temps (*timestamp*) d'una TSA (*TimeStamp Authority*).
- Guardar qualsevol operació en el sistema en un registre de logs.

Comunicació dels components

En l'apartat de comunicació dels components unes tasques futures serien:

- Substituir la tecnologia RMI per Web Services. D'aquesta manera aconseguiríem que l'aplicatiu sigui més fàcilment integrable amb altres programaris.
- Ús del protocol SSL/TLS per la comunicació.

Gestió de la informació: base de dades

En l'apartat de gestió de la informació unes tasques futures serien:

- Suport de múltiples tipus de sistemes gestor de base de dades (Oracle, Informix, SQL Server, etc.).
- Afegir eines de treball per facilitar la minera de dades, mantenint els objectius de privadesa de les dades mèdiques (independència la identitat de l'usuari i les seves dades)

Interfície d'usuari

En l'apartat d'interfície amb l'usuari unes tasques futures serien:

- Creació d'una interfície gràfica. Idealment aquesta interfície gràfica es faria mitjançant una interfície web, així el clients remots no haurien d'instal·lar-se cap programari addicional per accedir al sistema.
- Possibilitat de disposar de l'aplicatiu en diversos idiomes, depenent de la de la configuració de l'usuari.

CAPÍTOL 12

Conclusions

Com a conclusions es fa un repàs de l'acompliment dels objectius plantejats al inici del projecte.

Conclusions generals

Criptografia i seguretat:

S'han definit un esquema criptogràfic que compleix les propietats de seguretat sobre els historials mèdics següents:

- Privadesa: Només pot accedir a les dades dels historials mèdics del pacient el personal autoritzat (el metge assignat al pacient o el propi pacient). Les dades mèdiques estan xifrades, el que garanteix la seva privadesa encara que se accedeixi directament a la base de dades.
- Integritat: Les dades dels historials mèdics del pacient no poden ser modificades sense autorització. Les dades mèdiques estan xifrades, el que ens protegeix de manipulacions.
- No-repudi: Les dades inserides en l'historial mèdic del pacient per un metge no poden ser repudiades. Les dades mèdiques estan signades digitalment pel mèdic que les ha inserides.
- Autenticitat. S'ha de garantir la identitat del metge que ha inserit les dades en l'historial mèdic del pacient. Les dades mèdiques estan identificades amb la signatura digitalment del mèdic que les ha inserides.

A més, durant el desenvolupament del projecte es va decidir ampliar una de les funcionalitat de l'esquema criptogràfic. Es va afegir el concepte de "sessió", el que permet als usuaris identificar-se un cop i fer totes les operacions que vulguin sense haver d'autenticar-se cada cop que tinguin que fer una acció. Aquesta decisió es va a costa de reduir les capacitats de la interfície d'usuari (s'ha fet una interfície en línea de comandes), veure *Seguiment del projecte*, pàg. 26.

Representació de les dades

El sistema gaudeix d'un sistema de representació de les dades que ens permet de guardar les dades (visites i historial mèdics) en documents XML per fer més senzilla la portabilitat entre els diferents aplicatius del projecte.

Aquest fitxer XML i les classes que ho gestionen són fàcilment personalitzables per ajustar-se a les necessitats particulars de l'entitat que faci servir el sistema. Per tant, aquest objectiu també es considera cobert amb el sistema entregat.

Comunicació dels components

S'ha assolit l'objectiu de poder accedir al sistema de manera remota, dividint l'aplicatiu en una part client (que fan servir metges i pacients) i una part servidora que es comuniquen entre si mitjançant RMI.

Gestió de la informació

Amb la instal·lació del sistema gestor de base de dades MySQL en la part servidor, i el disseny d'un model de base de dades, s'acompleix l'objectiu de mantenir l'estat persistent de les dades, mantenint les propietats de seguretat (privadesa, integritat, no-repudi i autenticitat).

A més el model de base de dades es fàcilment exportable a altres sistemes gestors de base de dades (Oracle, Informix, SQL Server, etc.).

Interfície d'usuari

Encara que la interfície d'usuari va quedar reduïda per ampliar les funcionalitats criptogràfiques, s'ha aconseguit una interfície senzilla i útil amb la que l'usuari pot accedir fàcilment a totes les funcionalitats del sistema.

Jocs de proves

Es disposa d'una bateria de tests unitaris per garantir sempre el correcte funcionament del codi i l'absència de bugs davant de futures modificacions i ampliacions del sistema.

Documentació

S'ha creat una documentació tècnica sobre l'arquitectura del sistema en UML i sobre l'API de les classes implementades mitjançant JavaDoc.

Opinió personal

Durant la realització d'aquest projecte he pogut aplicar gran part dels coneixements i tecnologies que he anat aprenent al llarg dels estudis de l'enginyeria superior en informàtica.

Les assignatures de enginyeria del programari i les relaciones amb seguretat sempre m'han agradat i aquest projecte m'ha donat l'oportunitat demostrar el meus coneixements partint de zero per realitzar un nou programari.

Durant la realització de projecte he tingut que aprendre i aprofundir en l'utilització de diverses tecnologies i eines, que m'han aportat nous coneixements:

- Consolidar els coneixements de les assignatures d'enginyeria del programari i criptografia.
- Aprofundir en el coneixement i aplicació de la metodologia UML i en l'eina Visual Paradigm.
- Aprofundir en l'ús del llenguatge Java.
- Aprendre l'ús de l'eina JavaDoc per la realització de la documentació tècnica (APIs).
- Aprendre a fer servir el *framework* JUnit per la creació i execució de test unitaris.
- Augmentar el coneixement sobre la llibreria criptogràfica IAIK.
- Aprendre l'ús de l'eina JDOM per manipular documents XML.
- Aprendre l'ús de RMI per comunicar sistemes de forma remota.
- Consolidar els coneixements sobre el disseny de base de dades i l'ús del sistema gestor de base de dades MySQL.

Potser com a crítica al projecte, m'hauria agradat un projecte una mica més enfocat a realització d'una gestió, planificació, anàlisi, especiació i disseny (per exemple, m'hauria agradat barallar-me més amb el disseny dels protocols criptogràfics), i no tan dirigit a la implementació (programació) del codi.

En definitiva, crec que aquest projecte m'ha servit per consolidar molts dels coneixements que he adquirit durant els estudis, aprendre noves coses sobre moltes eines que hi ha en el mercat i assumir el repte de realitzant jo mateix tots el passos per la creació del sistema informàtic.

Glossari

ASN.1 *m* Acrònim de *Abstract Syntax Notation* 1 (notació de sintaxis abstracta 1) Estàndard que defineix un formalisme per a l'especificació de tipus de dades abstractes. Aquesta notació es descriu en la norma internacional 8824.

API *f* Acrònim de *Application Programming Interface* (interfície de programació de aplicacions). Conjunt d'especificacions de comunicacions entre components *software*.

Autenticitat *f* Propietat de trobar-se, en relació amb la informació, en el mateix estat en què va ser produïda, sense modificacions no autoritzades; és sinònim d'integritat.

Autentificació *f* Comprovació de l'autenticitat.

Autoritat de certificació *f* Autoritat de confiança que emet certificats.

CASE Acrònim de *Computer Aided Software Engineering* (enginyeria de programari assistida per ordinador). Les eines CASE són diverses aplicacions informàtiques destinades a augmentar la productivitat en el desenvolupament de programari reduint el cost de les mateixes en termes de temps i de diners.

Clau *f* Paràmetre, normalment secret, que controla els processos de xifratge i/o de desxifrable.

Clau privada *f* Clau que permet realitzar la transformació criptogràfica inversa a la que s'obté amb una clau pública, i que és computacionalment inviable obtenir a partir d'aquesta última.

Clau pública *f* Clau que permet realitzar la transformació criptogràfica inversa a la que s'obté amb una clau privada, i que es pot obtenir fàcilment a partir d'aquesta última.

Clau simètrica *f* Clau que permet realitzar tant una transformació criptogràfica com la transformació inversa, és a dir, xifratge i desxifratge.

Autoritat de registre *f* Entitat que s'encarrega de les tasques més administratives de l'autoritat de certificació, com ara la confirmació de la identitat dels usuaris, la verificació que l'usuari posseeix la clau privada associada a la clau pública que es vol certificar, la publicació dels certificats en directoris, etc.

CA *f* Acrònim de *Certification Authority.* Veure autoritat de certificació.

Cadena de certificats *m* Llista de certificats, cadascun dels quals permet verificar l'autenticitat de la clau pública de la CA que ha emès l'anterior, fins arribar al certificat d'una CA arrel.

Certificat digital *m* Estructura de dades que vincula una identitat amb una clau pública. El certificat és emès i signat digitalment per una autoritat de certificació.

Criptografia *f* Ciència i estudi de l'escriptura secreta.

Criptologia *f* Denominació conjunta de la criptografia i de la criptoanàlisi.

Criptosistema de clau compartida *m* Criptosistema en què tant l'emissor com el receptor comparteixen una sola clau que fan servir tant per a xifrar com per a desxifrar.

Criptosistema de clau pública *m* Criptosistema en què cada usuari té una clau pública i una de privada; amb aquest criptosistema una parella d'usuaris es pot comunicar confidencialment sense compartir una clau secreta.

CRL *f* Acrònim de *Certificate Revocation List*. Veure llista de revocació de certificats.

DES *m* Acrònim de *Data Encryption Standard* (estàndard de xifrat de dades). Algorisme de xifrat per blocs, escollit en 1976 com FIPS als Estats Units.

Desxifratge *m* Procés de transformació del text xifrat en text en clar.

Digest *m* Nom que es dóna de vegades a un resum o *hash*.

DTD *m* Acrònim de *Document Type Definition* (definició de tipus de document). Descripció d'estructura i sintaxi d'un document XML o SGML.

Document Type Definition m Veure DTD.

Entitat final *f* Propietari d'un certificat que no és una autoritat de certificació.

Hash *m* Veure resum.

HSM *m* Acrònim de *Hardware Security Module* (mòdul de seguretat hardware). Dispositiu criptogràfic basat en maquinari que genera, emmagatzema i protegeix claus criptogràfiques i sol aportar acceleració maquinari per a operacions criptogràfiques.

IAIK *Toolkit* comercial que implementa funcions criptogràfiques.

IDE *m* Acrònim de *Integrated Development Environment* (entorn de desenvolupament integrat). Programa compost per un conjunt d'eines per a un programador.

Infraestructura de clau pública *f* Conjunt de maquinari, programari, persones, polítiques i procediments necessaris per a crear i gestionar certificats digitals basats en criptografia de clau pública.

Integritat *f* Propietat de no haver sofert, en relació amb la informació, modificacions ni supressions parcials no autoritzades.

Java *m* Llenguatge de programació orientat a objectes desenvolupat per Sun Microsystems a principis dels 90. A diferència dels llenguatges de programació convencionals, que generalment estan dissenyats per a ser compilats a codi nadiu,

Java és compilat en un *bytecode* que és executat (usant normalment un compilador JIT), per una màquina virtual Java.

Java Virtual Machine *f* Programa nadiu, és a dir, executable en una plataforma específica, capaç d'interpretar i executar instruccions expressades en un codi binari especial (el Java bytecode), el qual és generat pel compilador del llenguatge Java

JDOM Lliberia *open source* que manipular documents XML.

JVM Acrònim de *Java Virtual Machine*. Veure Java Virtual Machine.

Llenguatge unificat de modelatge *m* Llenguatge de modelatge de sistemes de programari més conegut i utilitzat en l'actualitat; encara quan encara no és un estàndard oficial, està recolzat en gran manera pel OMG (*Object Management Group*). És un llenguatge gràfic per a visualitzar, especificar, construir i documentar un sistema de programari. UML oferix un estàndard per a descriure un "plànol" del sistema (model), incloent aspectes conceptuals tals com processos de negocis i funcions del sistema, i aspectes concrets com expressions de llenguatges de programació, esquemes de bases de dades i components de programari reutilizables.

Llista de revocació de certificats *f* Llista emesa per les autoritats de certificació en la que es publiquen tots aquells certificats que han estat revocats.

MD5 Acrònim de *Message-Digest Algorithm 5* (algorisme de resum de missatge 5) Algorisme de resum criptogràfic de 128 bits àmpliament usat.

MySQL *m* Sistema de gestió de base de dades, multifil i multiusuari desenvolupat per MySQL AB desenvolupa MySQL sota la llicència GNU GPL.

No repudi *m* Propietat que garanteix la protecció de la informació contra denegació d'autoria per part del seu originador.

OpenSSL *m* Paquet d'eines d'administració i llibreries relaciones amb la criptografia, disponible sota llicència Apache-style.

Public key infrastructure Veure infraestructura de clau pública.

PKCS *m* Acrònim de *Public-Key Cryptography Standards*. Conjunt d'especificacions desenvolupades en els laboratoris RSA per tal d'establir una norma comuna a la indústria sobre els formats de les dades utilitzades en criptografia de clau pública.

PKCS#12 *m* Estàndard de sintaxis de intercanvi de informació personal. Defineix un format de fitxer usat comunament per a emmagatzemar claus privades amb el seu certificat de clau pública protegit mitjançant clau simètrica.

PKI *f* Acrònim de *Public Key Infrastructure*. Veure infraestructura de clau pública.

Privacitat *f* Dret de les persones a salvaguardar la seva intimitat, especialment pel que fa a les dades de què disposen les entitats públiques o privades.

RA *f* Acrònim de *Registration Authority*. Veure autoritat de registre.

RC4 Sistema de xifrat de flux molt utilitzat i s'usa en alguns dels protocols més populars com el TLS/SSL.

Requests for comments Conjunt de documents tècnics i notes organitzatives sobre Internet.

Resum *m* Una funció resum o hash es refereix a una funció o mètode para generar claus que representin de manera casi unívoca a un document, registre, arxiu, etc.

RFC Acrònim de *Requests For Comments*. Vegeu requests for comments.

RMI *m* Acrònim de *Remote Method Invocation* (invocació remota de métodos). Mecanisme ofert en Java para invocar un mètode remotament.

RSA Criptosistema de clau pública molt utilitzat, publicat per Rivest, Shamir i Adleman l'any 1978.

SHA1 m Acrònim de *Secure Hash Algorithm (a*lgorisme de resum segur). Algorisme de resum criptogràfic de 160 bits àmpliament usat.

Signatura digital *f* Procediment per a signar documents en format electrònic que consisteix en un algorisme de signatura privat del signatari i un algorisme públic per a la verificació de la signatura.

Sobre digital *m* Tècnica que combina la criptografia de clau compartida i la de clau pública per tal d'aprofitar la velocitat de la primera i la flexibilitat de la segona.

SQL *m* Acrònim de *Structured Query Language* (llenguatge de consulta estructurat). Llenguatge declaratiu d'accés a bases de dades relacionals que permet especificar diversos tipus d'operacions sobre les mateixes

Triple-DES m Algorismo que fa triple xifrat del DES. També es conegut como TDES o 3DES. Va ser desenvolupat per IBM en 1978.

UML Acrònim de *Unified Modeling Language*. Vegeu Llenguatge unificat de modelatge.

Verificació *f* Comprovació que una signatura és vàlida, és a dir, que ha estat efectuada pel pretès signatari. Ha de ser possible per a tothom, és a dir, no ha de requerir coneixement de paràmetres secrets.

Xifra *f* Mètode secret d'escriptura, mitjançant el qual un text en clar es transforma en un text xifrat.

Xifratge *m* Procés de transformació d'un text en clar en un text xifrat.

XML *m* Acrònim de l'expressió anglesa *eXtensible Mark-up Language*. Especificació que defineix una sintaxi i unes regles sobre l'ús d'etiquetes per a estructurar la informació.

XML Schema Descripció d'estructura i sintaxi d'un document XML o SGML.

Bibliografia

La bibliografia utilitzada durant aquest projecte es la següent

Llibres

Criptografia i PKI

- **Domingo Ferrer, Josep; Herrera Carlescomartí, Jordi; Rifà Pous, Helena** (2006). *Criptografia*. Universitat Oberta de Catalunya.
- **Herrera Carlescomartí, Jordi; García Alfaro, Joaquín; Perramón Tornil, Xavier** (2004). *Seguretat en Xarxes de Computadors*. Universitat Oberta de Catalunya.
- **Lucena López, Manuel J.** (2006). *Criptografía y Seguridad en Computadores (4ª edición, versión 0.7.5)*. http://wwwdi.ujaen.es/~mlucena/lcripto.html.
- **Menezes, A. J.; Dorschot, P. C.; Vanstone, S. A.** (2001). *Handbook of applied cryptography (5a. ed.)*. CRC Press.
- **Schneier, B.** (1996). *Applied cryptography, protocols, algorithms, and source code in C (2a. ed.)*. John Wiley & Sons.

Disseny

- **Booch, Grady; Rumbaugh, James, Jacobson, Ivar** (1999) *El Lenguaje Unificado de Modelado*. Addsion Wesley
- **Fowler, Martin; Scott, Kendall** (1999). *UML gota a gota*. Pearson Education.
- **Gamma, Erich; Helm, Richard; Johnson, Ralph, Vlissides, John** (2005). *Patrones de Diseño*. Pearson Education
- **Hunt, Andy; Thomas, Dave** (2003). *Pragmatic Unit Testing In Java with JUnit*. The Pragmatic Bookshelf.

Java

- **Morgan, Mike** (1998) *Descubre Java 1.2*. Prentice Hall

XML

- **Rusty Harold, Elliotte; Scott Means, W.** (2002). *XML in a Nutshell 2nd Edition*. O'Reilly & Associates, Inc.
- **van der Vlist, Eric** (2002). *XML Schema*. O'Reilly & Associates, Inc.

Pàgines web

Java

- *Java 2 Platform Standard Edition 5.0 API Specification.*
  http://java.sun.com/j2se/1.5.0/docs/api
- *Javadoc Tool Home Page.*
  http://java.sun.com/j2se/javadoc
- *JDOM.*
  http://www.jdom.org
- *JUnit, Testing Resources for Extreme Programming*
  http://junit.org

XML

- *Extensible Markup Language (XML) 1.0*

  http://mipagina.euskaltel.es/gsagarduy/rec-xml-es.html
- *Introducción al XML*

  http://quark.fe.up.pt/cursoxml/curso.pdf
- *Processing XML with Java*

  http://www.cafeconleche.org/books/xmljava/chapters/index.html

RMI

- *Programación distribuida con RMI*
  http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=rmi

Altres

- *Wikipedia, la enciclopedia libre*
  http://es.wikipedia.org
- *Wikipedia, the free encyclopedia libre*

http://en.wikipedia.org

- *Informàtica y medicina*

  http://www.facmed.unam.mx/eventos/seam2k1/2001/ponencia_abr_2k1.htm

- *Información, tecnología y ética en la salud*

  http://www.cecam.sld.cu/pages/rcim/revista_9/articulos_htm/eticaensalud.htm

APÈNDIX A

Instal·lació del sistema

En aquest apèndix es descriuen tots es passos que s'han de realitzar per instal·lar el sistema.

Instal·lació OpenSSL

Els passos per instal·lar OpenSSL són els següents:

1. (Linux) Descarregar el codi font de http://www.openssl.org/source/ i compilar.
2. (Windows) Descarregar el instal·lador de http://www.slproweb.com/products/Win32OpenSSL.html i instal·lar.

Instal·lació JDOM

Els passos per instal·lar JDOM són els següents:

1. Descarregar la versió 1.0 de JDOM de http://www.jdom.org/dist/binary.
2. Descomprimir el fitxer `jdom-1.0.zip`.
3. Compilar JDOM mitjançant la comanda `build.sh` (Linux) o `build.bat` (Windows).
4. Afegir al `CLASSPATH` el fitxer `<JDOM_DIR>/build/jdom.jar`.

Instal·lació IAIK

Els passos per instal·lar IAIK són aproximadament els següents:

1. Descarregar la versió última del JDK de SUN i instal·lar-lo. Cal vigilar que no s'utilitzi la màquina virtual de MS al compilar o executar.
2. Descarregar la última versió de IAIK de http://jce.iaik.tugraz.at/sic/download. Cal registrar-se però no suposa cap cost. Només es necessari descarregar l'arxiu `iaik_jce_full.jar`, que és la versió completa signada.

3. Descarregar les polítiques de seguretat de Java que permeten emprar qualsevol longitud de clau. Es pot descarregar a http://java.sun.com/javase/downloads/index.jsp.

4. (Windows) Copiar l'arxiu `iaik_jce_full.jar` als directoris:
   - `c:\Archivos de Programa\Java\jdk1.5.0\jre\lib\ext`
   - `c:\Archivos de Programa\Java\jre1.5.0\lib\ext`

5. (Linux) Copiar l'arxiu `iaik_jce_full.jar` al directori:
   - `$JAVA_HOME/jre/lib/ext`

6. (Windows) dins de l'arxiu jce_policy-1.5.0-beta2.zip hi ha els arxius:
   - `local_policy.jar`
   - `US_export_policy.jar`

   Copiar-los a:
   - `c:\Archivos de Programa\Java\jdk1.5.0\jre\lib\security`
   - `c:\Archivos de Programa\Java\jre1.5.0\lib\security`

7. (Linux) dins de l'arxiu `jce_policy-1.5.0-beta2.zip` hi els arxius:
   - `local_policy.jar`
   - `US_export_policy.jar`

   Copiar-los a:
   - `$JAVA_HOME/jre/lib/security`

Instal·lació JUnit

Els passos per instal·lar JUnit són aproximadament els següents:

1. Descarregar la versió 4.1 del JUnit de http://junit.org.
2. Descomprimir el fitxer `junit4.1.zip`.
3. Afegir al `CLASSPATH` el fitxer `<JUNIT_DIR>/junit-4.1.jar`. Afegir al `CLASSPATH` el directori `<JUNIT_DIR>`.

Instal·lació MySQL

Els passos per instal·lar MySQL són aproximadament els següents:

1. Descarregar la versió 4.0.13 de MySQL de la següent adreça

   http://downloads.mysql.com/archives/mysql-4.0/mysql-4.0.13-win.zip

2 Descomprimir el fitxer i executar el fitxer `setup.exe`.

3 Triar la instal·lació típica.

4 Una vegada instal·lat, anar al subdirectori `bin` del lloc on s'ha realitzat la instal·lació; i executar el programa: `winmysqladmin.exe`.

   a) Un cop executat, el primer que farà aquest programa és demanar el nom d'usuari i la clau.

   b) Això però, no crearà aquest usuari. Per a crear-lo, executar, des del directori `bin` el client de MySQL:

   ```
   mysql –u root
   ```

   c) Un cop fet això, teclegeu les següents comandes SQL per a crear el vostre usuari (substituïu USUARI i CLAU pels valors que li vulgueu donar):

   ```
   grant all privileges on *.* to USUARI@localhost identified by 'CLAU';

   flush privileges;
   ```

5 Configurar MySQL per a poder usar taules de tipus InnoDB. Les taules InnoDB donen suport per a transaccions.

   De les pestanyes que apareixen a la part superior de l'eina d'administració triar "my.ini Setup". Apareixerà una àrea de text amb el fitxer my.ini de configuració de MySQL.

   En primer lloc, triar `mysqld-max` dels botons que hi ha a l'esquerra. Guardar les modificacions fent clic sobre "save modification".

   Al àrea de text, a la secció `[mysqld]`, al final de tot, afegir la següent línia:

   ```
   innodb_data_file_path = ibdata1:10M:autoextend
   ```

   Amb això MySQL que guardarà totes les taules InnoDB a un fitxer anomenat ibdata1 que estarà situat al directori indicat per `datadir` (paràmetre definit al mateix fitxer de configuració). Aquest fitxer tindrà inicialment 10Mb i s'ampliarà en cas de ser necessari.

   Guardar les modificacions fent clic sobre "save modification".

Instal·lació Java JDBC de MySQL

Els passos per instal·lar Java JDBC de MySQL són aproximadament els següents:

1 Descarregar l Java JDBC de MySQL. de la següent adreça

   http://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-3.1.7.zip/from/pick

2 Descomprimiu l'arxiu `mysql-connector-java-3.1.7.zip`.

3 Modificar el `CLASSPATH` de manera que inclogui el fitxer `mysql-connector-java-3.1.7-bin.jar`.

Creació de les taules buides de la base dades

Els passos per crear les taules buides necessàries per els sistema són:

1 Instal·lar MySQL (veure *Instal·lació MySQL*, pàg. 114).

2 Obrir el client de MySQL i executar les següents comandes SQL:

```
drop table if exists users;
create table users(
     userid varchar(255) not null,
     usercert longtext not null,
     role varchar(10) not null,
     doctorid varchar(255),
     primary key(userid)
)type=InnoDB;

drop table if exists tokens;
create table tokens(
     userid varchar(255) not null,
     inserttime bigint not null,
     token longtext not null,
     cipheredtoken longtext not null,
     primary key(userid)
)type=InnoDB;

drop table if exists casehistories;
create table casehistories(
     userid varchar(255) not null,
     position decimal not null,
     inserttime bigint not null,
     medicalvisitcrypted longtext not null,
     datasign longtext not null,
     userdatasign longtext not null,
     primary key(userid,position)
)type=InnoDB;
```

Compilació del projecte

Una vegada instal·lat el programari anterior i creades les taules buides necessàries, s'ha de compilar les classes de l'aplicatiu del projecte.

Per fer-ho s'han d'executar les instruccions següents:

```
javac casehistory\exceptions\package-info.java
    javac casehistory\exceptions\DecoderException.java
    javac casehistory\exceptions\CaseHistoryException.java
    javac casehistory\exceptions\ProcessException.java
```

```
javac casehistory\exceptions\TokenExpired.java
javac casehistory\exceptions\NoTokenPresent.java
javac casehistory\exceptions\TokenNotMatch.java
javac casehistory\exceptions\UserNotExists.java
javac casehistory\exceptions\NoCaseHistoryPresent.java
javac casehistory\exceptions\UnexpectedTrailingBytes.java
javac casehistory\exceptions\UnexpectedTag.java
javac casehistory\exceptions\InvalidAction.java
javac casehistory\exceptions\InvalidActionData.java
javac casehistory\exceptions\ErrorOpeningP12.java
javac casehistory\exceptions\InvalidToken.java
javac casehistory\exceptions\InvalidCaseHistorySerialization.java
javac casehistory\exceptions\InvalidMedicalVisitSerialization.java
javac casehistory\exceptions\ErrorRetreivingCaseHistory.java
javac casehistory\exceptions\ErrorStoringToken.java
javac casehistory\exceptions\ErrorStoringMedicalVisit.java
javac casehistory\exceptions\ErrorEncrypting.java
javac casehistory\exceptions\ErrorDecrypting.java
javac casehistory\exceptions\ErrorVerifyingData.java
javac casehistory\exceptions\ProcessResponseException.java
javac casehistory\exceptions\InvalidManagerId.java
javac casehistory\exceptions\Unauthorized.java
javac casehistory\exceptions\UserTimeout.java
javac casehistory\exceptions\SerializationError.java
javac casehistory\exceptions\ErrorStoringUser.java
javac casehistory\exceptions\ErrorRetreivingToken.java
javac casehistory\exceptions\ErrorRetreivingUser.java
javac casehistory\exceptions\UserNotExists.java
javac casehistory\utils\package-info.java
javac casehistory\utils\Hex.java
javac casehistory\utils\MedicalVisit.java
javac casehistory\utils\CaseHistory.java
javac casehistory\utils\Serializer.java
javac casehistory\utils\TLVSerializer.java
javac casehistory\utils\XMLSerializer.java
javac casehistory\utils\TLVEncoder.java
javac casehistory\utils\TLVDecoder.java
javac casehistory\utils\Constants.java
javac casehistory\utils\Conversions.java
javac casehistory\utils\Base64.java
javac casehistory\cryptography\package-info.java
javac casehistory\cryptography\Cipher.java
javac casehistory\cryptography\P12.java
javac casehistory\cryptography\RandomEngine.java
javac casehistory\cryptography\Signature.java
javac casehistory\user\package-info.java
javac casehistory\user\ManagerConnector.java
javac casehistory\user\LocalManagerConnector.java
javac casehistory\user\RmiManagerConnector.java
javac casehistory\user\UserFacade.java
javac casehistory\manager\package-info.java
javac casehistory\manager\UserData.java
javac casehistory\manager\PatientData.java
```

```
javac casehistory\manager\DoctorData.java
javac casehistory\manager\UserDataStore.java
javac casehistory\manager\MemoryUserDataStore.java
javac casehistory\manager\DatabaseUserDataStore.java
javac casehistory\manager\SecureTokenStore.java
javac casehistory\manager\MemorySecureTokenStore.java
javac casehistory\manager\DatabaseSecureTokenStore.java
javac casehistory\manager\SecureCaseHistoryStore.java
javac casehistory\manager\MemorySecureCaseHistoryStore.java
javac casehistory\manager\DatabaseSecureCaseHistoryStore.java
javac casehistory\manager\ManagerFacade.java
javac casehistory\manager\CaseHistoryRemoteInterface.java
javac casehistory\manager\CaseHistoryServer.java
javac casehistory\view\package-info.java
javac casehistory\view\CaseHistoryServerThread.java
javac casehistory\view\ManagerView.java
javac casehistory\view\ClientView.java
```

Índex

E

F

H

I

J

L

Ll

M

N

O

P

R

S

T

U

V

W

X