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

Alumne
Daniel Colomer Travé
Enginyeria Informàtica

Director de projecte
Jordi Castellà Roca
Doctor en Informàtica

Gener 2008

Agraïments

Vull donar les gràcies a totes aquelles persones relacionades amb la UOC que d'una o una altra manera han fet que el meu pas com a estudiant d'Enginyeria Informàtica fos una experiència enriquidora, plena i satisfactòria. Aquí es poden englobar els estudiants, consultors i tutors que al llarg d'aquests anys han provocat en mi, conscient o inconscientment, la il·lusió continuada per l'estudi.

En segon terme haig de donar les gràcies als meus pares, la meva germana i els meus sogres, per la seva actitud sempre positiva i animadora i pel seu recolzament moral durant les hores baixes.

Però sobretot vull agrair a Rocío, la meva parella, la seva paciència i comprensió, doncs sense la seva complicitat m'hauria estat impossible arribar a assolir aquesta fita.

Resum

Aquest projecte final de carrera (a partir d'ara PFC) pertany a l'àrea de la seguretat informàtica. En particular es tracta de portar a terme un projecte informàtic de programari dins del sector sanitari, on cal donar solució a les necessitats de seguretat de la informació clínica dels pacients que tenen tant aquests com el personal mèdic responsable.

Els historials mèdics son les dades amb significació mèdica referents a un pacient, al tractament al qual se'l sotmet i a l'evolució de la seva infermetat. Òbviament, és tracta d'informació sensible, regulada per la Llei de Protecció de Dades de Caràcter Personal (LOPD), la qual considera aquestes dades com informació de nivell molt confidencial i objecte de les més estrictes mesures de seguretat.

D'una altra banda tenim què a l'hora de gestionar aquesta informació, tant metges com pacients, troben molts avantatges en els mitjans informàtics i telemàtics, atès que s'obté més eficiència organitzativa, es redueixen els costos, s'estalvia temps, etc.

Per tant, una entitat mèdica que vulgui oferir aquests serveis amb infraestructures informàtiques ha de dissenyar i implementar el seu sistema basant-se en esquemes criptogràfics, atès que tant els ordinadors com les xarxes de comunicació per defecte no incorporen els mecanismes de seguretat necessaris.

Així doncs, aquest treball presenta una solució basada en criptosistemes de clau pública, certificats i signatures digitals, emprant Java com a llenguatge de programació. Per a estendre la funcionalitat quant a seguretat del Java Developer Kit (JDK) s'utilitza la llibreria criptogràfica IAIK (Institute for Applied Information Processing and Communication).

Per el intercanvi de dades entre els actors que conformen el sistema s'ha utilitzat intensivament la tecnologia XML i com a protocol de comunicació remot s'ha optat per RMI (Remote Method Invocation), atès què és un mecanisme senzill i eficient.

Quant a la persistència de la informació s'utilitza MySQL com a Sistema Gestor de Bades de Dades (SGBD) i respecte a les interficies d'usuari s'ha optat per interficies gràfiques emprant SWT (Standard Widget Toolkit)

Índex

Capítol 1. Introducció

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

1.2. Objectius del PFC

1.3 Enfocament i mètode seguit

1.4 Planificació del projecte

1.5 Esquema general de l'aplicació

1.6 Productes obtinguts

1.7 Breu descripció de la resta dels capítols de la memòria

1.7.1 Instal·lació del programari base

1.7.2 Creació d'una PKI

1.7.3 Esquema criptogràfic

1.7.4 Anàlisi i disseny de l'aplicació

1.7.5 Representació de les dades amb XML

1.7.6 Comunicació dels components amb RMI

1.7.7 Gestió de la persistència amb MySQL

1.7.8 Interfícies d'usuari

1.7.9 Joc de proves

1.7.10 Conclusions

1.7.11 Bibliografia i annexos

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

Ningú no dubta de les bondats que la Societat de la Informació, gracies a la incorporació de les TIC (tecnologies de la informació i la comunicació), ens ofereix avui en dia. Els ordinadors i les xarxes de comunicació permeten el intercanvi immediat de dades sense necessitat de mobilitat física i això obre una llista d'aplicacions quasi interminable.

Son varis els factors que estan facilitant l'ús de les noves tecnologies en tots els àmbits socials. La voluntat política dels governs d'apropar les TI als ciutadans i a les empreses, així com la davallada de preus any rere any de les connexions de banda ampla, els serveis i els components informàtics, son dos dels factors més importants.

El sector sanitari, el qual té una importància cabdal, sempre ha estat capdavanter a l'hora d'aplicar les innovacions tecnològiques als seus processos de negoci. En particular, en aquest PFC es tracta el tema de la gestió dels historials mèdics, assegurant el flux de les dades entre els pacients i el personal mèdic.

La informació continguda als historials clínics té uns requeriments legals força exigents quant a seguretat i, malauradament les solucions informàtiques que estan més a l'abast tenen un propòsit general i no satisfan les necessitats específiques.

Així doncs, l'objectiu d'aquest PFC és l'assegurament de la informació gestionada per una aplicació distribuïda en una xarxa de comunicacions oberta (pública).

Per a entendre en què consisteix l'assegurament de la informació cal assimilar els següents quatre conceptes clau:

- Confidencialitat: La informació només ha de ser accessible per les persones autoritzades. Així doncs, un historial mèdic únicament pot ser consultat pel personal mèdic autoritzat i pel propi pacient.

- Autenticació: Aquesta propietat evita que ningú no pugui suplantar la identitat d'un metge autoritzat o del propi pacient.

- Integritat: Assegurar que la informació dels historials no pugui modificar-se per cap persona no autoritzada sense que quedi de manifest.

- No repudi: Aquesta propietat evita que cap persona autoritzada pugui desdir-se d'una acció portada a terme per ella en el sistema per a la qual te permís.

Per resoldre aquests aspectes es tenen els criptosistemes de clau pública, els certificats i les signatures digitals, els quals possibiliten, entre els seus serveis, protocols i components, la utilització d'un medi insegur, com és el cas de les xarxes obertes, per establir comunicacions segures.

1.2. Objectius del PFC

L'objectiu d'aquest treball consisteix en dissenyar i portar a terme un sistema informàtic de programari que permeti la gestió dels historials mèdics dels pacients dins d'una xarxa de comunicacions.

Aquesta gestió s'ha de realitzar amb totes les garanties de seguretat exigibles a aquest tipus de dades. Per aquest motiu s'utilitza una infraestructura de clau pública (PKI), seguint un model pla, això és un entorn on només hi ha una entitat de certificació en la qual confien tots els usuaris del sistema.

En aquest treball es posen en pràctica gran part dels coneixements adquirits durant la carrera, tant els coneixements tècnics i tecnològics com els de gestió. Es comença fent una planificació de les activitats que s'han de dur a terme; es passa per una fase de documentació inicial, per a realitzar després l'anàlisi i el disseny de classes propi d'una aplicació orientada a objectes.

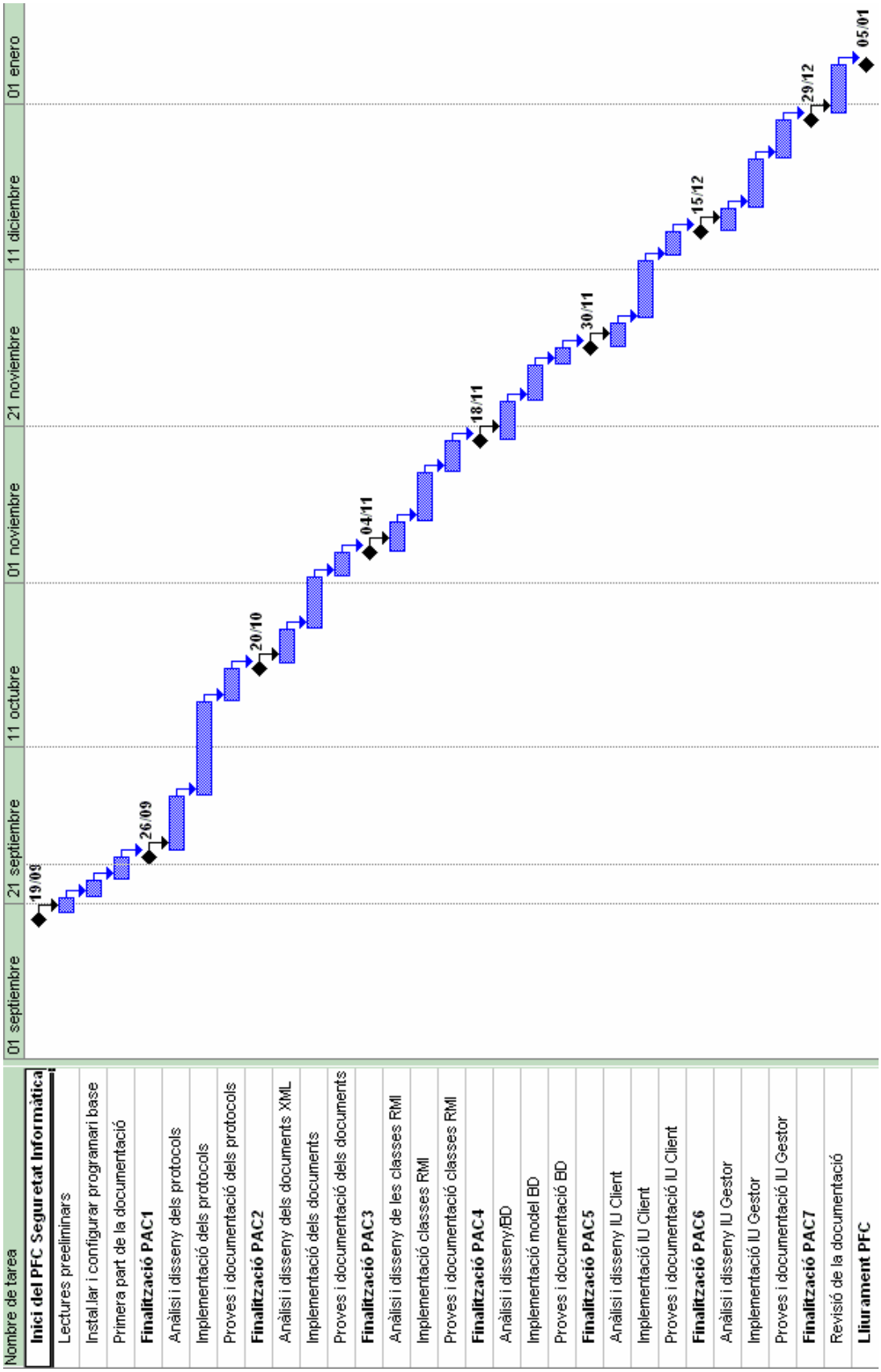
Posteriorment s'implementa en Java el treball anterior, incorporant tecnologies com XML i RMI per tal de complir amb els requeriments del sistema. Tot això sense descuidar les interfícies d'usuari ni la gestió de la persistència de les dades.

Respecte a la documentació, aquesta es va generant durant tot el procés de desenvolupament, amb la qual cosa, les últimes sessions es dediquen per a redactar les conclusions i corregir imprecisions.

1.3 Enfocament i mètode seguit

El PFC s'ha planificat per realitzar-se en una sèrie d'etapes consecutives, llevat de la fase de documentació que dura tota la vida del projecte.
Es dissenya i implementa cada fase de treball per després provar el seu funcionament i depurar els possibles errors. Quan la fase següent està enllestida cal integrar-la amb la anterior, amb les corresponents proves d'integració, per tal de veure que tot el conjunt continua lliure d'errors.
D'aquesta mena de desenvolupaments se'n diu implementació incremental, atès què l'etapa n-1 serveix de punt de partida per l'etapa n i n sempre té una complexitat més gran què n-1 (està més a prop del resultat final).

1.4 Planificació del projecte



1.5 Esquema general de l'aplicació

A la imatge 1-1 es pot observar com l'aplicació està formada per tres capes:

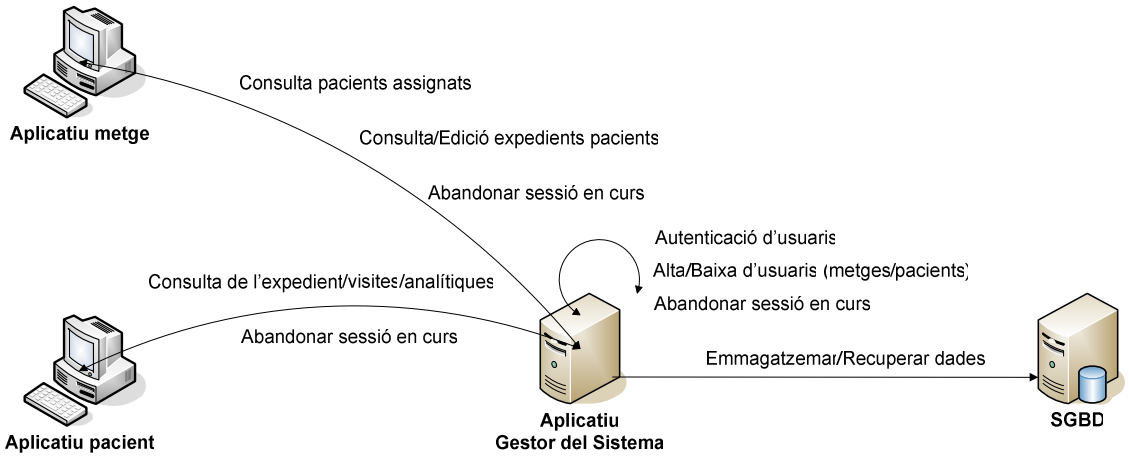


Figura 1-1. Esquema d'alt nivell dels actors i els serveis de l'aplicació.

- A la primera capa es té el programa client, que pot presentar-se amb la variant metge, pacient, però al cap i a la fi, una aplicació client.

- A la segona capa es té el programa servidor, el qual rep el conjunt de sol·licituds de la part client, les processa i si s'escau, retorna els resultats.

- A la tercera capa hi és el sistema de emmagatzemament, sobre el què es recolza el gestor del sistema per tal de recuperar i desar la informació de l'aplicació.

Les aplicacions client funcionen sobre màquines físiques independents; així, el més segur és què un metge utilitzi el programa des de l'ordinador del seu despatx a l'hospital i un pacient revisi el seu expedient des de l'ordinador de casa seva.

L'aplicació del gestor del sistema funciona sobre una màquina independent, però sense interactuar directament amb cap usuari, llevat de l'administrador de l'aplicació que dona d'alta als usuaris del sistema, doncs és tracta d'un automatisme concebut per funcionar per si mateix.

Cal tenir en compte que la segona i la tercera capa, malgrat que el dibuix inciti a pensar el contrari, poden coexistir perfectament a una mateixa màquina física. Tot dependrà de les necessitats de implementació a l'entorn de producció.
En un entorn de proves l'aplicació pot funcionar perfectament en una sola màquina física.

1.6 Productes obtinguts

El principal resultat del PFC ha estat una aplicació de gestió dins de l'àmbit sanitari que permet a metges i pacients fer diverses operacions sobre els historials mèdics d'aquests últims.

Es tracta d'un programa d'arquitectura distribuïda, fet en Java i utilitzant diverses llibreries de tercers. El programa té dos mòduls: el mòdul d'usuari i el del gestor del sistema. El primer té dos usos diferenciats: metge i pacient, mentre que el segon és un automatisme encarregat d'efectuar les operacions suportades pel sistema: recuperar i emmagatzemar informació assegurant les propietats de la informació. No obstant, necessita una interfície d'usuari perquè un administrador pugui registrar nous metges i pacients, afegir certificats, treure informació estadística, etc.

Acompanya a l'aplicació una breu presentació en PowerPoint i la memòria del projecte en format Word, document on s'expliquen amb profunditat tots els aspectes de treball realitzat, sempre dins del marc teòric de la criptografia com a garantia de seguretat en les comunicacions en xarxa.

Però el PFC ha generat altres productes al llarg de les seves etapes. Així es té una infraestructura de clau pública (PKI) amb la què és possible crear tants certificats com es necessitin. D'una altra banda, es té una Base de Dades MySQL amb la informació dels expedients mèdics, els certificats digitals i de les relacions entre metges i pacients, etc.

1.7 Breu descripció de la resta dels capítols de la memòria

En aquest apartat s'explica el contingut de la resta dels capítols de la memòria per tal donar una visió global del document.

1.7.1 Instal·lació del programari base

Explicació del sistemes operatius utilitzats, els programes instal·lats, les versions emprades i altres detalls.

1.7.2 Creació d'una PKI

Es defineix i es detalla què és una Infraestructura de Clau Pública. Es mostren les captures gràfiques de les sortides per pantalla durant el procés de creació de la PKI.

1.7.3 Esquema criptogràfic

La finalitat d'aquest apartat és estudiar exhaustivament els procediments i protocols que son necessaris per a complir amb els requeriments de seguretat de l'aplicació.

1.7.4 Anàlisi i disseny de l'aplicació

A l'etapa d'anàlisi, mitjançant els casos d'ús, els diagrames i models de l'UML s'obtenen les especificacions en termes orientats a l'objecte. A la fase de disseny això s'adapta amb vista a la implementació concreta del PFC: llenguatge Java+IAIK, SGBD MySQL, etc.

1.7.5 Representació de les dades amb XML

XML és un Llenguatge d'Etiquetat Extensible que permet el intercanvi d'una gran varietat de dades. La seva funció principal es descriure dades (no mostrar-les como és el cas del HTML). Aquesta tecnologia serveix per a estructurar, emmagatzemar i intercanviar informació, per tant es l'eina utilitzada per a transferir dades durant l'execució dels protocols criptogràfics.

1.7.6 Comunicació dels components amb RMI

RMI, Invocació Remota de Mètodes, forma part de Java estàndard i ve inclòs en el JDK des de la versió 1.1. Es tracta d'un mecanisme de comunicació entre aplicacions que no es basa en l'ús d'un client web, ni en formularis HTML, ni en extensions d'un servidor web, ni en l'ús d'una connexió HTTP. La Màquina Virtual de Java (JVM) permet la invocació de mètodes entre objectes de diferents JVM o de diferents ordinadors com si fos la mateixa màquina.

Aquesta aplicació està concebuda de manera distribuïda, és a dir, que cada component del sistema (el programa del pacient o del metge, el programa del gestor, el SGBD) pot funcionar en una màquina física diferent dins d'una xarxa de comunicacions, però alhora tot funciona amb la coherència d'una aplicació monolítica.

1.7.7 Gestió de la persistència amb MySQL

La informació inherent a un historial mèdic ha de ser emmagatzemada per tal d'estar disponible en qualsevol moment, així com tot un conjunt d'informació relacionada.
En aquest capítol es descriu el procés seguit per a poder interactuar des de Java amb el SGBD MySQL.

1.7.8 Interfícies d'usuari

En aquest apartat es detalla les decisions preses a l'hora de dissenyar i implementar les interfícies d'usuari, les qual permeten interactuar amb l'aplicació d'una manera més eficient i amable.

1.7.9 Joc de proves

Les proves, tant unitàries com d'integració, son una part fonamental dins del cicle de vida del programari per tal d'aconseguir un producte de qualitat.
En aquest capítol es descriu el procés seguit en aquesta etapa tan important.

1.7.10 Conclusions

Es fa un balanç del treball realitzat des d'un punt de vista crític. Es comenten possibles ampliacions que no s'han pogut implementar per quedar fora de l'abast del projecte.

1.7.11 Bibliografia i annexos

Tanca el PFC un ampli apartat on hi ha tot un seguit de documents miscel·lanis que complementen o detallen la resta de capítols de la memòria. És força interessant l'últim apartat on es dona solució als possibles problemes que poden aparèixer al compilar i/o executar l'aplicació.

Capítol 2. Programari base

2.1 Introducció

2.2 Sistemes Operatius

2.3 Cronologia de programes instal·lats

2.1 Introducció

Una aplicació d'aquestes característiques requereix d'altres programaris que facilitin les diferents etapes del projecte. En aquest capítol s'explica breument quins son aquests programes i es justifica el seu ús dins del PFC.

2.2 Sistemes Operatius

L'aplicació ha estat totalment desenvolupada sobre el sistema operatiu Windows XP Professional amb Service Pack 2. Tot i això s'ha emprat Linux Suse Professional 9.3 per l'execució del programa openssl i dur a terme la generació de les claus de xifratge, els certificats digitals i els arxius PKCS12.

2.3 Cronologia de programes instal·lats

Java Development Kit: Entorn de programació i d'execució Java

Instal·lació del JDK de Java (jdk-6u2-windows-i586-p.exe). Això ens permet compilar i executar programes escrits amb Java.

Programari per ampliar les funcions criptogràfiques del JDK

Per ampliar les funcions criptogràfiques dels JDK s'utilitza la llibreria IAIK. Aquest software disposa d'algorismes més especialitzats i potents que els propis de Java.
Cal copiar la llibreria criptogràfica "iaik_jce_full.jar" al directori arrel de l'Eclipse (C:\Eclipse 3.1) .

Preparació de l'entorn de desenvolupament i d'execució

S'han modificat les variables d'entorn PATH i CLASSPATH per tal d'incloure les rutes del Java i de les llibreries que es faran servir al llarg del projecte.

Restricció de longitud de claus criptogràfiques de Java

Per a poder emprar en Java qualsevol longitud de clau s'utilitzen les polítiques de seguretat de Java(TM) Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 6. Descomprimir el fitxer "jce_policy-6.zip" i copiar els fitxers "local_policy.jar" i "US_export_policy.jar" al directori:
C:\Archivos de programa\Java\jdk1.6.0_02\jre\lib\security i al directori C:\Archivos de programa\Java\jre1.6.0_02\lib\security.

Eclipse 3.1 com a entorn de desenvolupament Java

Instal·lació del IDE Eclipse (eclipse-SDK-3.1.2-win32.zip). Aquest software és un dels IDE més potents i robusts de tot el mercat.

Planificació del Projecte

Instal·lació del Microsoft Project 2003 Professional per a confeccionar la planificació del PFC.

Disseny d'esquemes i quadres sinòptics

Instal·lació del Microsoft Visio 2003 Professional per tal d'elaborar els diferents gràfics que apareixen en aquesta memòria.

API per la gestió de documents XML

JDOM permet crear i llegir documents XML des de Java d'una manera àgil i senzilla.
Un cop descarregada la llibreria jdom.jar cal copiar-la al directori arrel de l'Eclipse (C:\Eclipse 3.1).

RMI. Invocació Remota de Mètodes

Quant al protocol de comunicació entre client i servidor no cal considerar cap configuració ni instal·lació extra, atès que ve incorporat amb la instal·lació del JDK.

Base de dades

Instal.lar MySQL Server 5.0 i les utilitats per l'administració i gestió. Cal copiar el connector Java per MySQL 'mysql-connector-java-3.1.8-bin.jar ' al directori arrel de l'Eclipse (C:\Eclipse 3.1).

Interficie gràfica d'usuari

Aquesta es basa en la llibreria gràfica 'Standard Widget Toolkit' (SWT) desenvolupada per IBM. Per a poder desenvolupar des del IDE Eclipse cal situar el fitxer swt.jar en el mateix directori (C:\Eclipse 3.1).

Capítol 3. PKI, Infraestructura de Clau Pública

3.1 Introducció

3.2 Creació de la PKI del PFC

3.1 Introducció

Una PKI, del angles Public Key Infraestructure, es tradueix per Infraestructura de Clau Pública i fa referència al conjunt d'elements de maquinari, programes, polítiques i procediments de seguretat que permeten l'execució de transaccions electròniques amb garanties mitjançant operacions criptogràfiques tals com el xifrat, la signatura digital, el no repudi, etc.

La finalitat d'un infraestructura de clau pública és gestionar eficientment les claus criptogràfiques i els certificats per tal de dur a terme les propietats de seguretat de la informació: confidencialitat, autenticació, integritat i no repudi.

En una PKI el certificat és l'objecte que pren una major rellevància, atès què legitima la correspondència entre les claus de xifratge i el propietari de les mateixes. Així es té que la majoria de les operacions disponibles en una PKI son del tipus "generar petició de certificat", "generar certificat", "signar certificat", "revocar certificat", etc.

La problemàtica més gran quant el xifratge de clau pública és l'atac anomenat "man-in-the-middle". Imaginem el següent escenari: L'emissor E vol enviar un missatge al receptor R; per tant E necessita la clau pública de R (PK_R) per tal de xifrar el missatge i que només pugui ser desxifrat per R mitjançant la seva clau privada (SK_R). Aleshores tenim que quan E consulta una font pública per a obtenir PK_R intervé una entitat fraudulenta F que vincula de manera transparent per a E la seva pròpia clau pública (PK_F) a la identitat de R. A la figura 3-1 es pot observar gràficament l'explicació anterior.

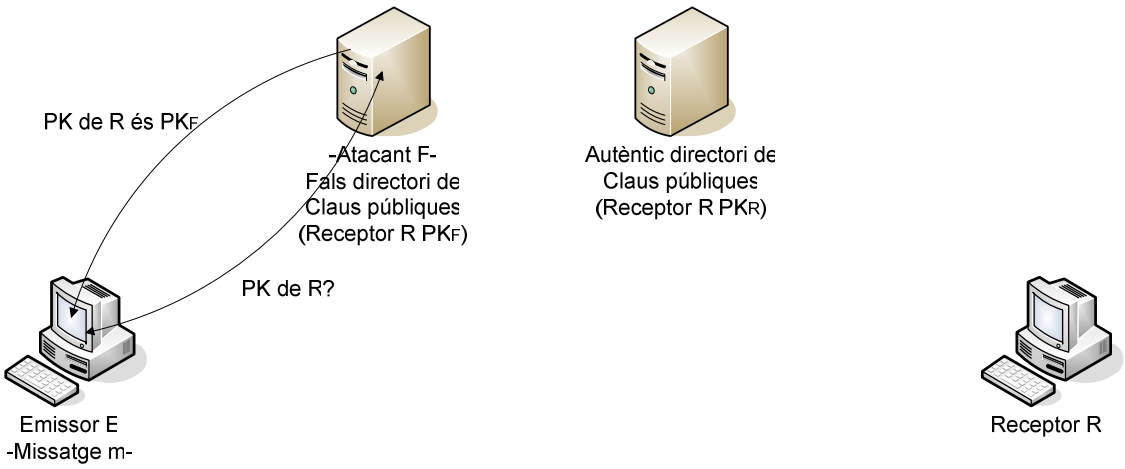


Figura 3-1. Atac man-in-the-middle, part 1 de 3.

Llavors E xifra el missatge amb PK_F i l'envia cap a R, però F el captura i aplica la seva clau privada (SK_F), amb la qual cosa ja pot llegir el contingut. A la figura 3-2 s'il·lustra aquet fet.

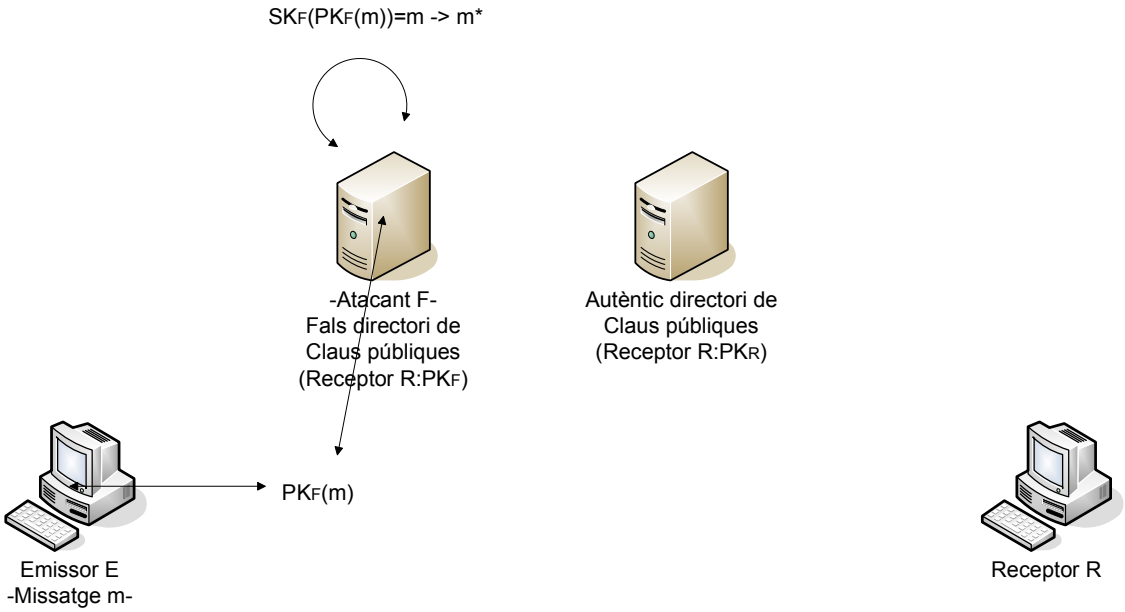


Figura 3-2. Atac man-in-the-middle, part 2 de 3.

Després F pot tornar a xifrar el missatge (original o manipulat) amb la clau pública del receptor autèntic (PK_R) i enviar-li a aquest. Finalment R rep el missatge sense assabentar-se de res. A la figura 3-3 queda pales aquest procés.

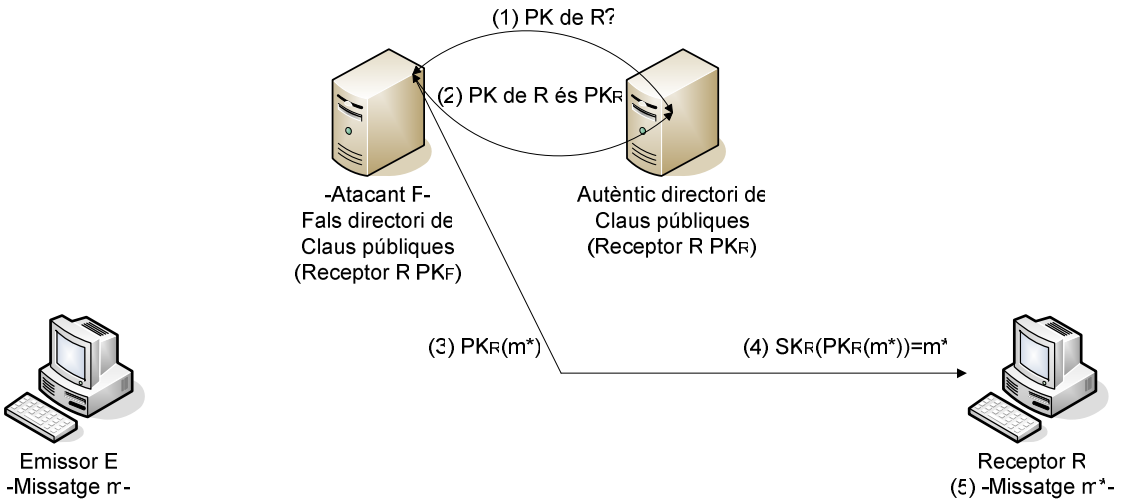


Figura 3-3. Atac man-in-the-middle, part 3 de 3.

Generalment, en una PKI hi ha una autoritat de certificació anomenada CA. Una altra figura és l'autoritat de registre, anomenada RA. Quan un usuari vol obtenir un certificat, primer de tot generen un parell de claus (la privada i la pública) i envien una petició a la RA, aleshores aquesta, si es compleixen els requeriments, fa la validació pertinent i envia una petició de certificat a la CA, qui finalment emet el certificat d'usuari. A la figura 3-4 es mostra aquest cicle:

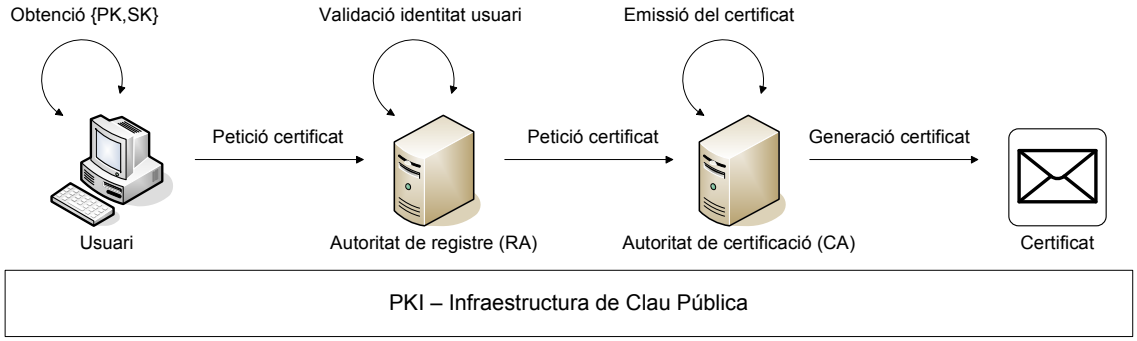


Figura 3-4. Cicle de vida bàsic de l'obtenció d'un certificat d'usuari.

Existeixen diferents models de PKI, sent els més representatius el model distribuït de la xarxa de confiança, el model pla i el model jeràrquic.

En el model distribuït de la xarxa de confiança no cal una tercera entitat que garanteixi la identitat dels usuaris. És adequat per entorns petits atès que cada usuari crea i signa certificats per els altres usuaris de la xarxa. D'aquesta manera treballa el programa PGP (Pretty Good Privacity).

El model pla és dins l'abast de les Intranets, doncs hi ha una única autoritat de certificació que posseeix un certificat que ella mateixa ha generat (certificat autosignat) i en qual tots els usuaris confien.

El model jeràrquic és emprat en àmbits que transcendeixen les Intranets. És l'ampliació del model pla, això és existeixen moltes autoritats certificació, però organitzades de tal manera que les autoritats de certificació de nivells superiors certifiquen a les de nivells inferiors. Es té igualment una autoritat arrel amb el seu certificat autosignat i amb la qual confien tots el components de la PKI.

3.2 Creació de la PKI del PFC

En aquest PFC s'utilitza el model pla i el programari emprat per crear la PKI és el openssl per Linux a la seva versió 0.9.8e. El sistema operatiu és un Suse Linux Professional 9.3.

A continuació s'explica tot el procés necessari per a obtenir els fitxers de claus, certificats i els contenidors PKCS12.

Un cop descomprimit el fitxer PKI.tgz (lliurat amb l'enunciat del PFC) s'obté l'estructura de directoris i fitxers que es mostra a la figura 3-5.

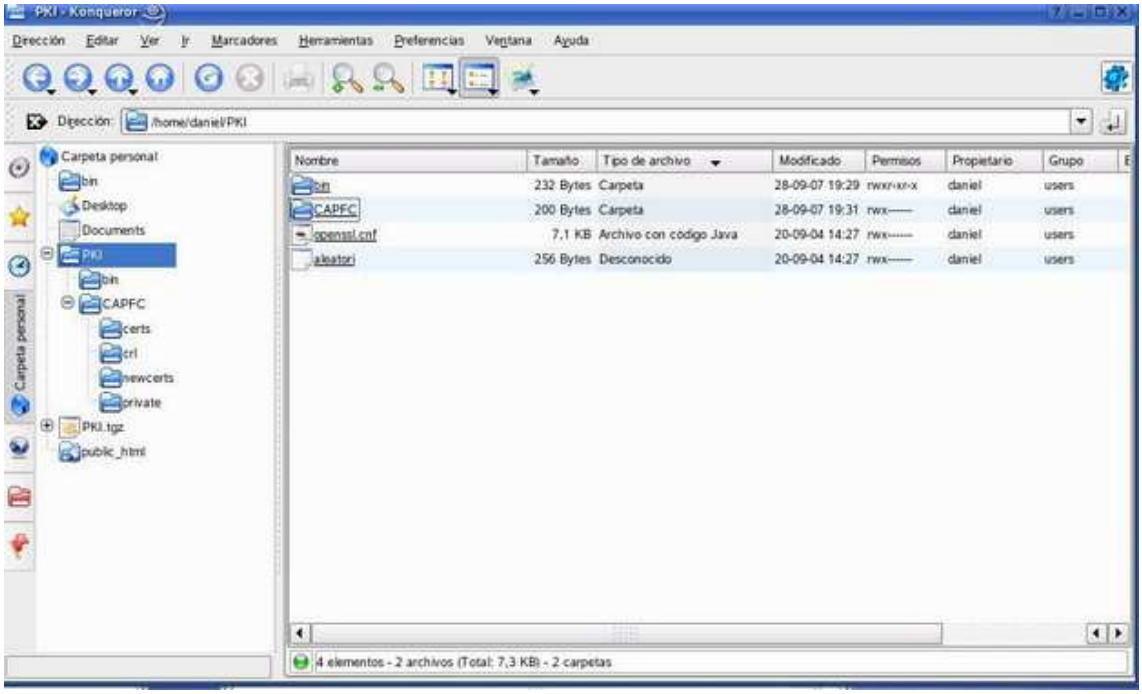


Figura 3-5. Estructura de directoris openssl.

Per a dur a terme totes les operacions que conduiran a l'obtenció dels certificats es fan servir uns scripts (proporcionats amb l'enunciat del PFC) que fan més còmoda la utilització del openssl. A l'annex A s'expliquen en detall aquest programes.

A continuació es comenta el funcionament d'aquests programes:

- generarClaus: Crea una parella de claus del criptosistema RSA. El poden emprar tant els usuaris com altres components de la infraestructura.
- generaCertificatAutosignat: S'utilitza per a crear el certificat de la CA. Així, a partir d'un parell de claus s'obté un certificat autoritzat on la CA queda legitimada per ella mateixa.
- generaPeticioCertificat: Es fa servir quan un usuari ja ha generat el seu parell de claus i vol que l'Autoritat de Registre li gestioni l'obtenció d'un certificat.
- generaCertificat: La CA l'utilitza per a crear el certificat d'usuari demanat per la RA.
- generaPKCS12: Permet obtenir un fitxer contenidor amb el parell de claus del usuari, el seu certificat i el certificat de la CA.

Ara que ja es té el programa operatiu el primer que s'ha de fer és configurar l'Autoritat de Certificació. Aleshores es generen el parell de claus de la CA, tenint en compte què en aquest cas convé emprar una clau de xifratge de 2048 bits donada la especial sensibilitat d'aquest component de l'arquitectura. A la figura 3-6 s'observa la sortida de l'execució d'aquesta operació.

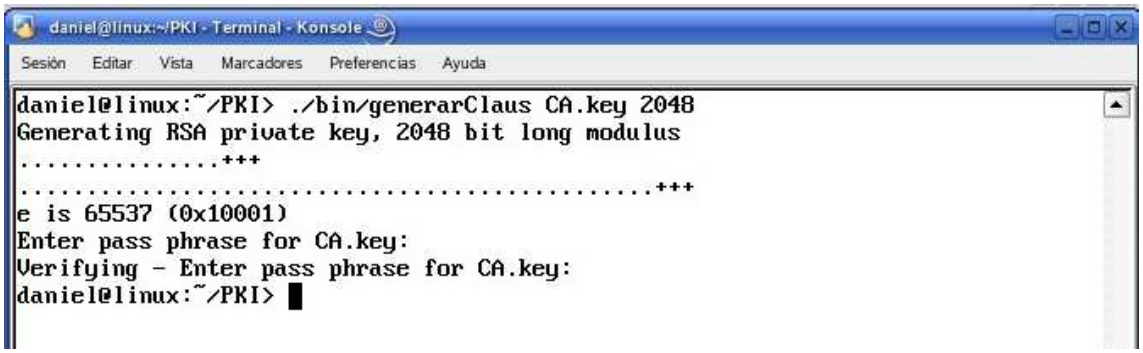


Figura 3-6. Sortida de l'execució de l'script generarClaus per la CA.

A continuació cal generar el certificat autosignat de l'Autoritat de Certificació, tal i com es mostra a la figura 3-7.

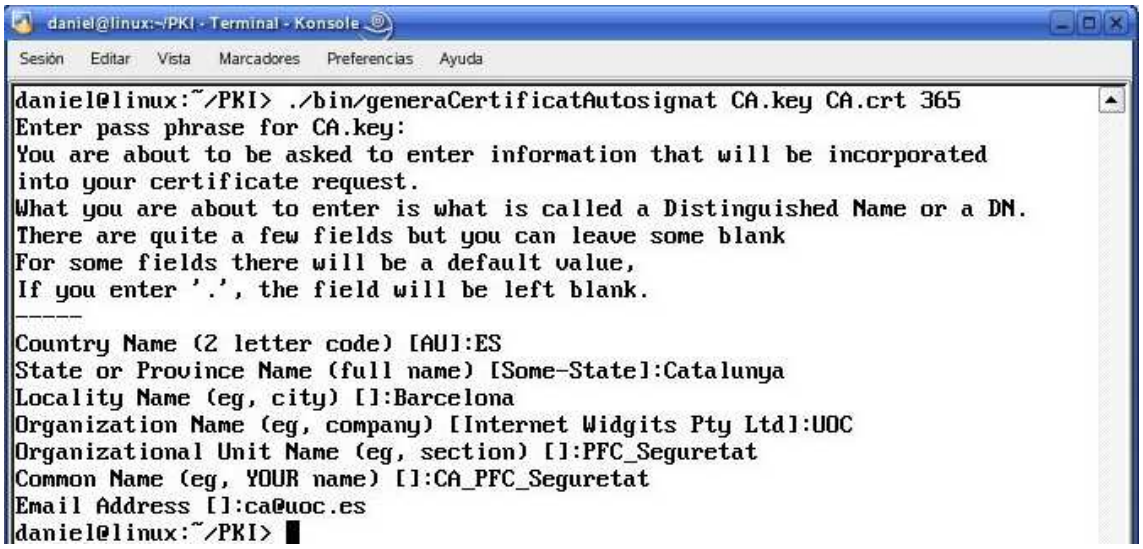


Figura 3-7. Sortida de l'execució de l'script generaCertificatAutosignat.

Resultat de les operacions anteriors s'han creat dos fitxers: un amb el parell de claus i l'altre amb el certificat autosignat. La CA ja és operativa per gestionar la PKI. A la figura 3-8 es mostren com apareixen aquest fitxers a l'arrel del directori PKI.

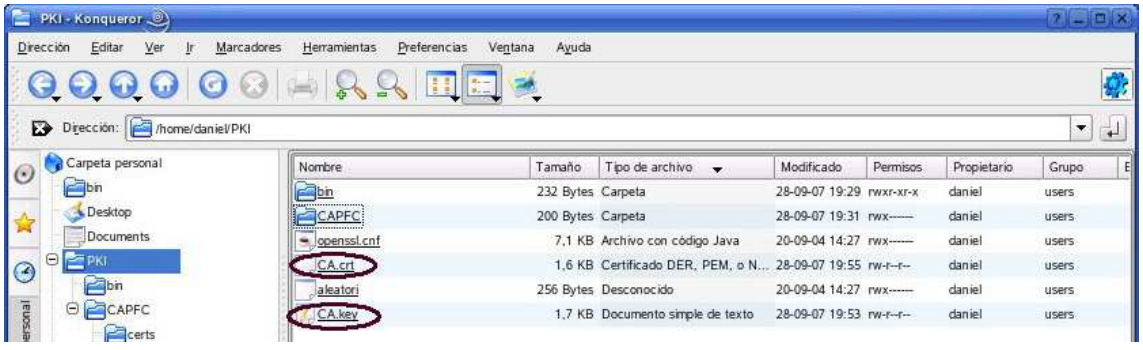


Figura 3-8. Creació de CA.key i CA.crt.

Un cop es té la CA configurada es procedeix a crear el parell de claus de cada usuari del sistema. En aquest cas és suficient amb una longitud de 1024 bits (figura 3-9).

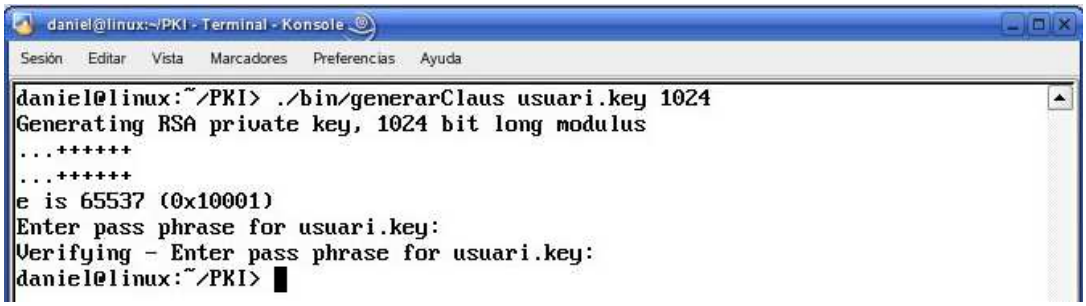


Figura 3-9. Sortida de l'execució de l'script generarClaus per a un usuari.

Ara s'ha de realitzar la petició de certificat a l'Autoritat de Registre (figura 3-10).

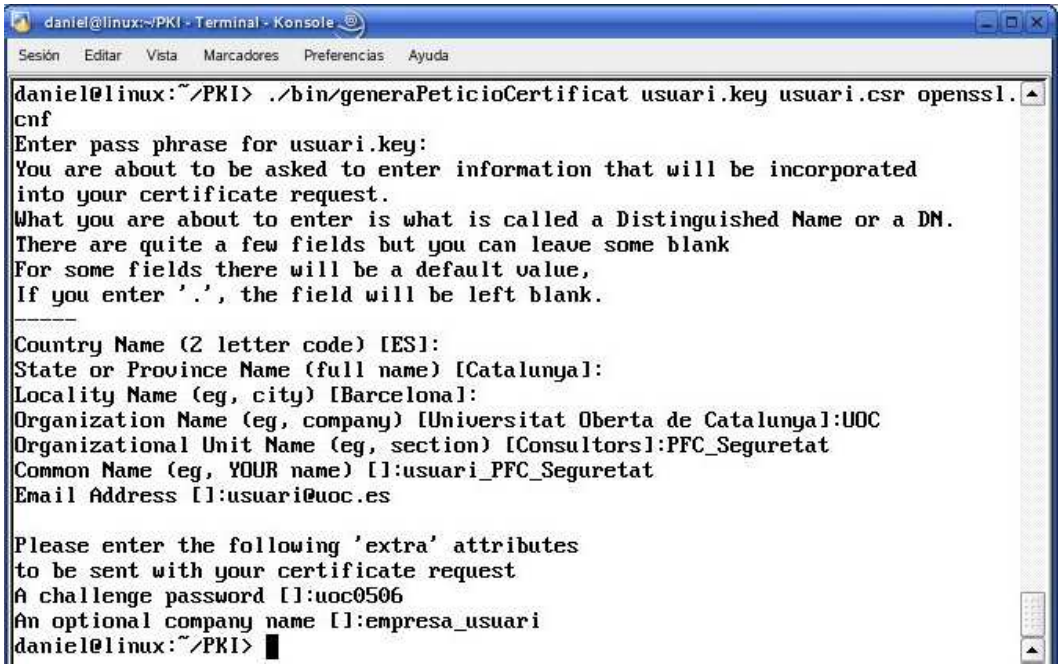
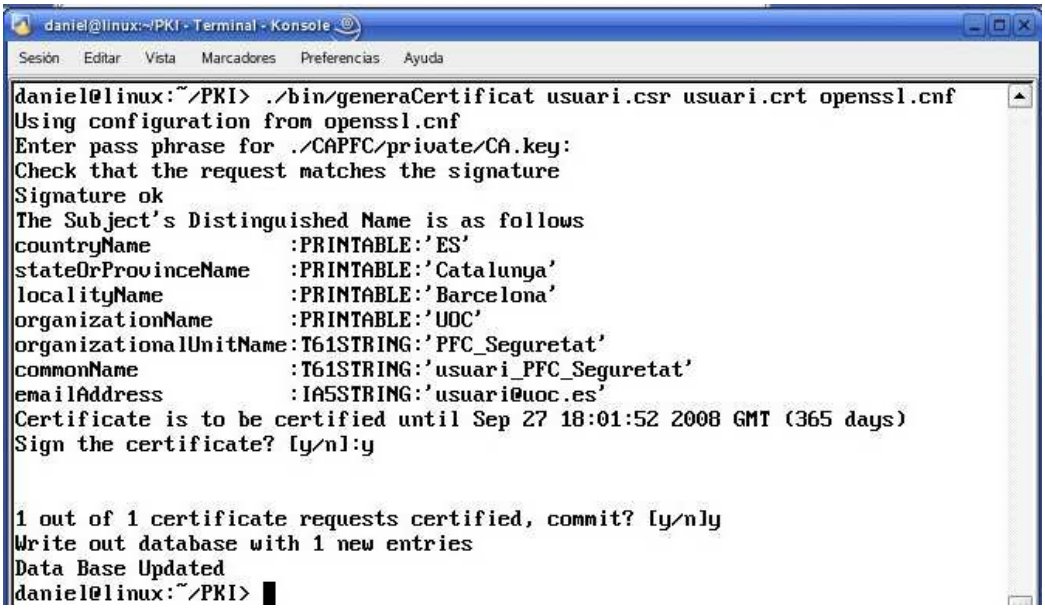


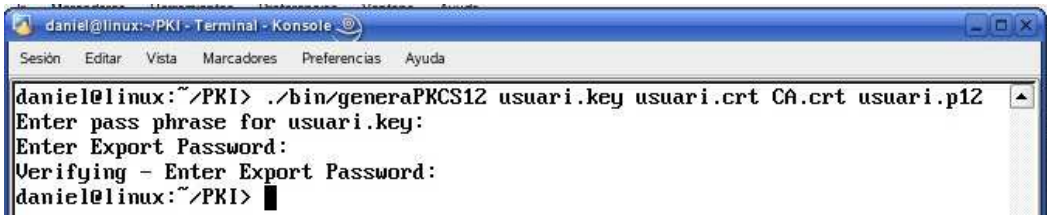
Figura 3-10. Sortida de l'execució de l'script generaPeticioCertificat.

Amb l'script invocat a la figura 3-11 l'usuari obté finalment el certificat.



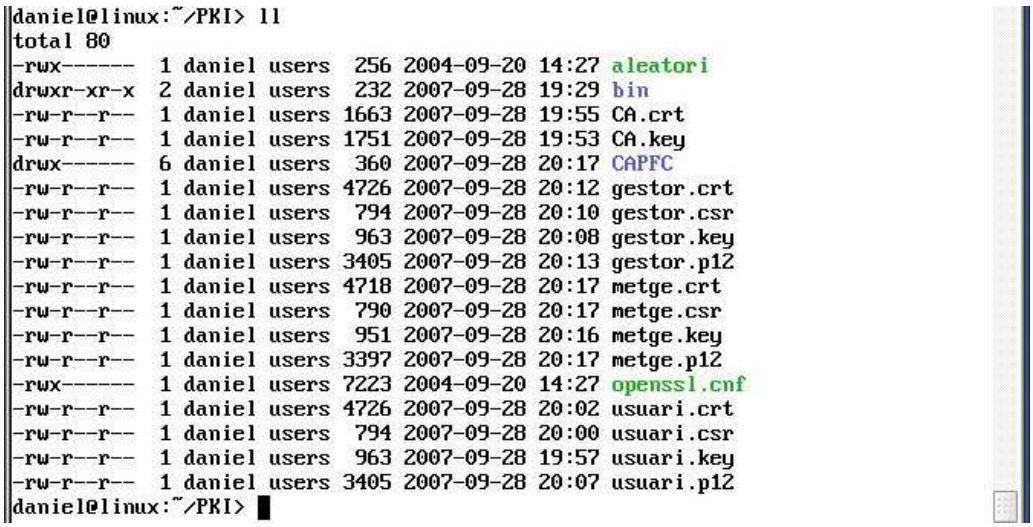
3-11. Sortida de l'execució de l'script generaCertificat.

Ara només resta generar el fitxer contenidor PKCS12 (figura 3-12).



3-12. Sortida de l'execució de l'script generaPKCS12.

Finalment, un cop generats els certificats de tots els usuaris del sistema, el directori PKI hauria de quedar de la manera semblant a la figura 3-13.



3-13. Llista de fitxers generats amb l'openssl.

Capítol 4. Esquema criptogràfic

4.1 Serveis que ofereix l'aplicació

4.2 Notació

4.3 Identificació i el certificat digital

4.4 Protocols criptogràfics

4.4.1 Consulta d'un expedient mèdic

4.4.2 Consulta dels pacients assignats a un metge

4.4.3 Inserció de dades al historial mèdic

4.1 Serveis que ofereix l'aplicació

Les operacions principals que l'aplicació ha de presentar son:

- Alta d'usuaris: metges i pacients.
- Consultar els expedients dels pacients i afegir visites.
- Consultar els pacients assignats a un metge.
- Assignar metges a pacients.
- Abandonar l'aplicació de manera segura.

Aquests serveis es fonamenten en una sèrie de protocols criptogràfics el conjunt dels quals rep el nom d'esquema criptogràfic i que garanteix les propietats de la seguretat: confidencialitat, autenticitat, integritat i no-repudi.

4.2 Notació

Per a descriure els protocols se segueix la notació de la taula 4-1:

$(P_{Entitat}, S_{Entitat})$	Parella de claus asimètriques propietat d'Entitat, on P correspon a la clau pública i S a la clau 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 (funció hash) del missatge M.
U	És l'entitat usuari, que pot ser un pacient o un metge.
G	És l'entitat gestor del sistema.
N_i	És un valor aleatori obtingut per l'usuari.
N'_i	És (presumptament)el valor aleatori N_i obtingut per l'usuari després de desxifrar informació rebuda.
N_g	És un valor aleatori obtingut pel gestor del sistema
N'_g	És (presumptament) el valor aleatori N_i obtingut pel gestor del sistema després de desxifrar informació rebuda.
Id_usuari_u	És el identificador únic del usuari.
id_usuari_g	És el identificador únic del gestor del sistema.
P_u/S_u	És la clau pública/secreta de l'usuari.
P_g/S_g	És la clau pública/secreta del gestor del sistema.

Taula 4-1. Notació emprada en els protocols.

4.3 Identificació i el certificat digital

A la taula 4-2 es presenta el format del tipus de certificats que s'utilitzen a l'aplicació.
A la primera columna apareixen els atributs i a la segona s'han omplert uns valors d'exemple.

Country Name	ES
State of Province Name	Barcelona
Locality Name	Badalona
Organization Name	UOC
Organizational Unit Name	Metges
Common Name	Dr. Colomer
dnQualifier	46695124
Email Address	d.colomer@uoc.es

Taula 4-2. Certificat digital d'exemple.

Cada actor del sistema disposa d'un identificador d'usuari únic (a la pràctica s'utilitza el DNI) que li permet ser identificat pel gestor del sistema. Associat a aquest identificador es té un certificat digital, també únic.

El gestor del sistema es basa en el identificador d'usuari per a recuperar el corresponent certificat i així autenticar a l'entitat que demana una operació, alhora que l'utilitza per avaluar quin tipus d'usuari és (metge, pacient o administrador) i contemplar els seus privilegis.

Per exemple, a la figura 4-3 es pot observar com una aplicació client de tipus pacient intenta fer una operació de modificació sobre el seu expedient i el gestor li denega la petició al veure que l'usuari no és un metge.

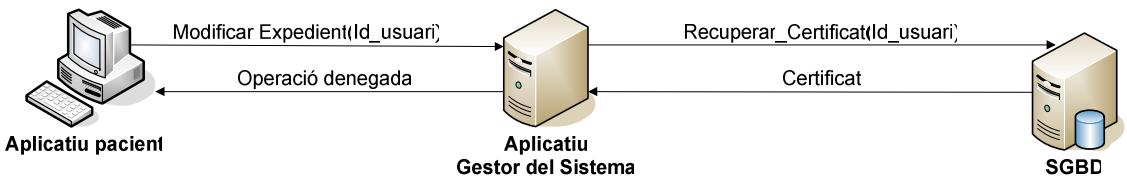


Figura 4-3. Exemple de denegació d'operació per manca de privilegis.

4.4 Protocols criptogràfics

4.4.1 Consulta d'un expedient mèdic

L'expedient mèdic d'un pacient pot ser consultat pel propi pacient o per un metge autoritzat, tenint aquest últim a més a més potestat per afegir dades. Tanmateix, el propi metge pot no estar autoritzat per a consultar certes dades del document, per tractar-se de informació al marge de la seva competència.

Deixant definit en què consisteix la consulta del historial clínic es procedeix ara a l'anàlisi d'un protocol que el fa possible: el protocol 1:

Protocol 1

Notació particular del protocol:

Id_usuari: És el identificador de l'expedient que es vol consultar.
H: Expedient que es vol consultar.

Procés:

Observació: Els passos del 1 al 8 conformen el protocol d'autenticació recíproca, és a dir, que tant l'usuari com el gestor del sistema queden mútuament autenticats.

Pas 1: U obté un valor N_i aleshores xifra aquest valor juntament amb el seu Id_usuari_u amb la clau P_g i li envia aquesta informació al gestor del sistema.

$$P_g[N_i, id\_usuari_u] \rightarrow G$$

Pas 2: G per la seva banda, el primer que fa es desxifrar amb la seva clau privada la informació rebuda:

$$S_g[P_g[N_i, id\_usuari_u]] = Ni, id\_usuari_u$$

Pas 3: Llavors a partir de id_usuari_u aconsegueix el certificat digital de U i consegüentment la seva clau pública P_u.

Pas 4: Després obté un valor N_g i li envia a U aquest valor, juntament amb N_i i amb id_usuari_g, tot xifrat amb la clau pública de U:

$$P_u[N_i, N_g, id\_usuari_g] \rightarrow U$$

Pas 5: Quan U rep la informació desxifra amb S_u:

$$S_u[P_u[N_i, N_g, id\_usuari_g]] = N'_i, N_g, id\_usuari_g$$

Pas 6: Aleshores, si $N_i = N'_i$ llavors U sap que G és autèntic. Per tant, pot enviar-li la petició de consulta, especificant quin és l'expedient que es vol consultar. Per a fer això ha de xifrar amb P_g aquests valors:

$$P_g[N_g, operacio\_consulta, id\_usuari] \rightarrow G$$

En canvi, si $N_i <> N'_i$ llavors U sap que alguna cosa no ha anat bé i retorna un error.

Pas 7: Quan G rep la informació anterior desxifra amb S_g i obté:

$$S_g[P_g[N_g, operacio\_consulta, id\_usuari] = N'_g, operacio\_consulta, id\_usuari$$

Pas 8: Aleshores, si $N_g = N'_g$ llavors G sap que U és autèntic i es té l'autenticació bilateral.
Per tant ara ha de comprovar els següent:

- Si U és un pacient, l'expedient que està demanant ha de ser el seu .
- Si U és un metge, l'expedient que està demanant ha de pertànyer a un pacient seu.

Cal contemplar que si $N_g <> N'_g$ llavors G retorna un error.

Pas 9: Si es compleix qualsevol de les dues condicions anteriors, G busca l'expedient H a partir del id_usuari (identificador únic d'un expedient), aleshores desxifra amb la seva clau privada S_g la part de l'expedient que estigui xifrada i el xifra sencer amb la clau pública d'U.

Pas 10: Finalment G envia a U l'expedient:

$$P_u[H] \rightarrow U$$

Pas 11: Quan U rep la informació anterior la desxifra amb S_u i obté:

$$S_u[P_u[H]] = H$$

Pas 12: Aleshores per a cada entrada del historial que està signada verifica la signatura digital de M, de G i la seqüència i si aquestes comprovacions son correctes el procés conclou exitosament.

4.4.2 Consulta dels pacients assignats a un metge

Un metge atén a un o més pacients, per tant ha de poder consultar qui son aquestes persones. Amb el protocol 2 es mostra com es pot fer això sense vulnerar les propietats de la informació:

Protocol 2

Procés:

Els passos del 1 al 5 son idèntics al del protocol 1.

Pas 6: Aleshores, si $N_i = N'_i$ llavors U sap que G és autèntic. Per tant, pot enviar-li la petició de consulta, especificant que es vol la relació de identificadors dels seus pacients. Per a fer això ha de xifrar amb P_g aquests valors:

$$P_g[N_g, \text{operacio\_llista\_id\_pacients}] \rightarrow G$$

En canvi, si $N_i <> N'_i$ llavors U sap que alguna cosa no ha anat bé i retorna un error.

Pas 7: Quan G rep la informació anterior desxifra amb S_g i obté:

$$S_g[P_g[N_g, \text{operacio\_llista\_id\_pacients}] = N'_g, \text{operacio\_llista\_id\_pacients}$$

Pas 8: Aleshores, si $N_g = N'_g$ llavors G sap que U és autèntic i es té l'autenticació bilateral.
Per tant ara ha de comprovar si id_usuari_u és un metge (mitjançant el certificat de id_usuari_u). Si això és així llavors ha d'obtenir (recuperar de la BD) tots els pacients assignats a id_usuari_u: $\{\text{id\_usuari}_1, ...\text{id\_usuari}_n\}$ i signar aquesta llista amb la seva clau privada S_g:

$$S_g [\{\text{id\_usuari}_1, ...\text{id\_usuari}_n\}]$$

Cal contemplar que si $N_g <> N'_g$ o id_usuari_u no és metge llavors G retorna un error.

Pas 9: Ara ha de xifrar amb la clau pública P_u la informació signada anteriorment juntament amb la llista plana de pacients i enviar-li a U:

$$P_u[\{\text{id\_usuari}_1, ...\text{id\_usuari}_n\}, S_g [\{\text{id\_usuari}_1, ...\text{id\_usuari}_n\}]] \rightarrow U$$

Pas 10: Quan U rep la informació anterior la desxifra amb S_u i obté:

$$S_u[\ P_u[\{id\_usuari_1, ...id\_usuari_n\}, S_g\ [\{id\_usuari_1, ...id\_usuari_n\}]]\ ] =$$
$$= \{id\_usuari_1, ...id\_usuari_n\}, S_g\ [\{id\_usuari_1, ...id\_usuari_n\}]$$

Pas 11: Ara U necessita verificar la signatura digital de G amb P_g:

$$P_g[\ S_g\ [\{id\_usuari_1, ...id\_usuari_n\}]\ ]$$

I si la comprovació és correcta se sap que la llista d'identificadors obtinguda ha estat realment generada per G:

$$\{id\_usuari_1, ...id\_usuari_n\}$$

4.4.3 Inserció de dades al historial mèdic

Un metge ha de poder afegir dades a l'expedient d'un pacient. Amb el protocol 3 queda resolta aquesta operació:

Protocol 3

Notació particular del protocol:

P: És el pacient al qual pertany l'expedient.
Id_usuari_u: És el identificador únic d'usuari de U (presumiblement un metge).
Id_usuari_p: És el identificador únic d'usuari de P.
H: És el expedient en el que s'han d'inserir dades.
V: És una visita; son les dades que s'han d'incorporar al expedient H.
T: Marca de temps que s'assigna a cada visita V.
X: Número (seqüencial) de sèrie que s'assigna a cada visita V.

Procés:

Els passos del 1 al 5 son idèntics als del protocol 1 i protocol 2.

Pas 6: Aleshores, si $N_i = N'_i$ llavors U sap que G és autèntic i ha de fer el següent:

Obtenir les dades de la visita V (que com a mínim contindrà el identificador únic del pacient, Id_usuari_p) i signar-les amb la seva clau privada: $S_u[V]$
Amb la clau pública de G, P_g, cal xifrar els valors N_g, V, $S_u[V]$ i l'operació necessària per a poder inserir dades a l'expedient del pacient: operacio_inserir_visita.

Llavors se li envia aquesta informació a G:

$$P_g[N_g, V, S_u[V], operacio\_inserir\_visita] \rightarrow G$$

En canvi, si $N_i <> N'_i$ llavors U sap que alguna cosa no ha anat bé i retorna un error.

Pas 7: Quan G rep la informació anterior desxifra amb S_g i obté:

$$S_g[P_g[N_g, V, S_u[V], operacio\_inserir\_visita] = N'_g, V, S_u[V], operacio\_inserir\_visita$$

Pas 8: Aleshores, si $N_g = N'_g$ llavors G sap que U és autèntic i es té l'autenticació bilateral. Per tant ara ha de fer les següents operacions:

- Obtenir id_usuari_p a partir de V.
- Verificar que id_usuari_u és metge.
- Verificar que id_usuari_p és un pacient assignat a id_usuari_u.

Cal contemplar que si $N_g <> N'_g$ llavors G retorna un error.

Pas 9: Si les comprovacions anteriors son certes, aleshores G ha de fer:

a) Amb la clau pública de U, P_u, verificar la signatura digital $S_u[V]$.
b) Obtenir el instant de temps actual T.
c) Obtenir el número de sèrie X de l'ultima visita del historial H del pacient id_usuari_p.
d) Incrementar X en una unitat (X+1).
e) Amb la seva clau privada, S_g, signar V, $S_u[V]$, T, X+1, obtenint: S_g [V, $S_u[V]$, T, X+1]
f) Xifrar amb la seva clau pública, P_g, els valors V i $S_u[V]$, obtenint: $P_g[V, S_u[V]]$
g) Amb la seva clau privada, S_g, signar els valors: id_usuari_p, X+1, obtenint: S_g [X+1, id_usuari_p]
h) Finalment es guarda a la BD: $P_g[V, S_u[V]]$, T, X+1, S_g [V, $S_u[V]$, T, X+1] i S_g [X+1, id_usuari_p]

En canvi, si les comprovacions no són correctes G retorna un error.

Capítol 5. Anàlisi i disseny de l'aplicació

5.1 Els casos d'ús dels requisits

5.1.1 Casos d'ús corresponents al pacient i al gestor

5.1.2 Casos d'ús corresponents al metge i al gestor

5.2 Diagrama de classes bàsic

5.3 Anàlisi i Disseny

5.3.1 Diagrames de seqüències

5.3.2 Diagrama de classes incloent la implementació dels protocols criptogràfics

5.3.3 Classes per a dur a terme els serveis i les operacions dels protocols

5.3.4 Diagrama complet: classes criptogràfiques i classes de servei

5.3.5 Paquets de classes i subsistemes

5.1 Els casos d'ús dels requisits

A continuació es presenten i s'expliquen els diagrames dels casos d'us pels actors de l'aplicació.

5.1.1 Casos d'ús corresponents al pacient i al gestor

A la figura 5-1 es mostra el diagrama de casos d'ús de l'aplicació del pacient:

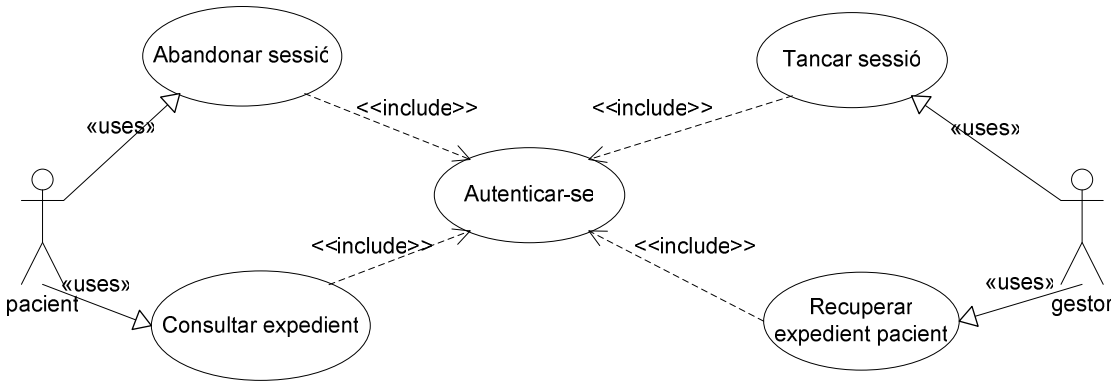


Figura 5-1. Diagrama de casos d'ús corresponent al pacient.

De la imatge anterior cal observar com les relacions <<include>> indiquen que un cas d'ús inclou a un altre. Per exemple, "Consultar expedient" inclou necessàriament "Autenticar-se".

5.1.1.1 Cas d'ús "Autenticar-se"

Resum de la funcionalitat: Certificar davant del gestor del sistema l'autenticitat del pacient i viceversa.

Paper dins el treball de l'usuari: És un cas d'ús bàsic, doncs l'autenticació és condició sine qua non per a dur a terme qualsevol operació.

Actors: pacient i gestor del sistema.

Precondició:

- El pacient ha d'estar registrat dins del sistema, això és, el gestor ha de disposar del seu certificat digital i a més a més, el pacient vol demanar una nova operació, per exemple consultar el seu expedient mèdic.

- Respecte el gestor, a aquest li estan demanant una nova operació, per exemple recuperar un expedient de la base de dades.

Postcondició: En una primera etapa el pacient autentica al gestor i en una segona aquest últim autentica al pacient, tenint llavors l'autenticació bilateral.

Cursos alternatius: Falla l'autenticació, bé perquè el pacient no autentica al gestor o perquè aquest últim no autentica al pacient.

Descripció detallada: El pacient vol demanar una operació al gestor, llavors comença el protocol d'autenticació, on en una primera fase el gestor ha de certificar la seva autenticitat davant el pacient i en una segona a l'inrevés. Si tots dos s'autentiquen es pot portar a terme l'operació objecte de l'autenticació. En cas contrari no es dur a terme aquesta operació.

5.1.1.2 Cas d'ús "Consultar expedient"

Resum de la funcionalitat: El pacient obté el seu propi historial mèdic per a consultar.

Paper dins el treball de l'usuari: És la principal operació que pot portar a terme un pacient.

Actors: pacient.

Precondició: L'expedient mèdic ha d'existir i el pacient ha d'estar autenticat.

Postcondició: El pacient rep l'expedient.

Cursos alternatius: El pacient demana un expedient que no li pertany i conseqüentment no el pot recuperar.

Descripció detallada: Un cop que s'estableix l'autenticació bilateral amb el gestor del sistema, el pacient sol·licita l'obtenció del seu propi expedient mèdic per a poder consultar-ho. Quan el rep el mostra per pantalla.

5.1.1.3 Cas d'ús "Abandonar sessió"

Resum de la funcionalitat: El pacient sol·licita sortir de l'aplicació.

Paper dins el treball de l'usuari: És una operació bàsica que dur a terme qualsevol usuari.

Actors: pacient.

Precondició: El pacient ha d'estar autenticat, ja que si no fos així un usuari malintencionat podria sol·licitar la desconnexió d'un altre usuari.

Postcondició: El pacient ha informat al gestor que vol sortir del programa.

Cursos alternatius: -

Descripció detallada: El pacient vol sortir del programa, aleshores sol·licita abandonar l'aplicació, però haurà d'estar autenticat perquè el gestor del sistema tingui la certesa de que efectivament pot eliminar la sessió d'aquest pacient i no d'un altre.

5.1.1.4 Cas d'ús "Tancar sessió"

Resum de la funcionalitat: El gestor rep la petició de sortir del programa per part del pacient.

Paper dins el treball de l'usuari: És una operació bàsica que dur a terme el gestor del sistema.

Actors: gestor del sistema.

Precondició: El pacient i el gestor han d'estar autenticats mútuament.

Postcondició: El pacient rep la confirmació per part del gestor de que pot sortir de l'aplicació.

Cursos alternatius: -

Descripció detallada: El gestor rep la petició de sortir del programa per part del pacient, aleshores dur a terme les comprovacions pertinents i li confirma a aquell que ja pot sortir.

5.1.1.5 Cas d'ús "Recuperar expedient pacient"

Resum de la funcionalitat: El gestor rep la petició per part del pacient de recuperar el seu expedient mèdic.

Paper dins el treball de l'usuari: És una operació bàsica que dur a terme el gestor del sistema.

Actors: gestor del sistema.

Precondició: L'expedient ha d'existir i el pacient i el gestor han d'estar autenticats mútuament.

Postcondició: El pacient rep el seu historial per part del gestor.

Cursos alternatius: No es pot recuperar el historial perquè es produeix alguna mena de problema amb la seguretat, per exemple amb les signatures dels certificats digitals.

Descripció detallada: El gestor rep la petició de recuperar el historial mèdic del pacient, aleshores dur a terme les comprovacions pertinents i li envia el seu expedient.

5.1.2 Casos d'ús corresponents al metge i al gestor

A la figura 5-2 es mostra el diagrama de casos d'ús de l'aplicació del metge:

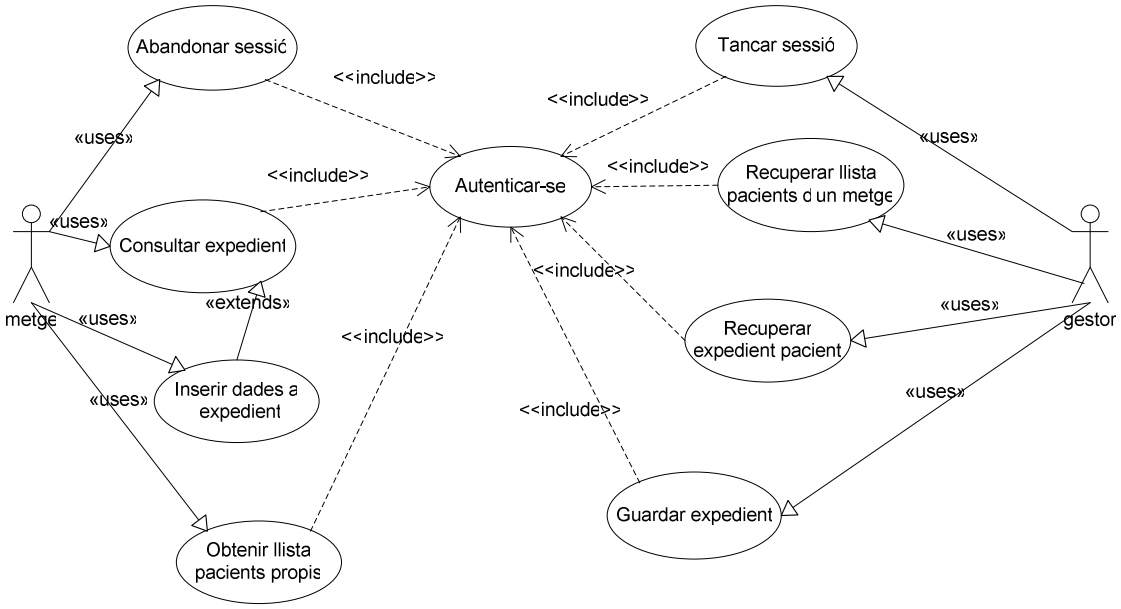


Figura 5-2. Diagrama de casos d'ús corresponent al metge.

En aquest apartat no s'expliquen els casos d'ús:

- Abandonar sessió
- Autenticar-se
- Consulta expedient
- Recuperar expedient pacient
- Tancar sessió

atès que estan inclosos a l'apartat anterior 5.1.1 (diagrama de casos d'ús corresponents al pacient i al gestor).

5.1.2.1 Cas d'ús "Inserir dades a l'expedient"

Resum de la funcionalitat: El metge disposa de nova informació que ha d'afegir a l'expedient d'un pacient.

Paper dins el treball de l'usuari: És una operació bàsica per el metge.

Actors: metge.

Precondició: El metge i el gestor han d'estar autenticats mútuament i el metge té informació per afegir a un expedient d'un pacient seu.

Postcondició: El metge ha informat al gestor del sistema de quina és la informació que cal afegir a l'expedient.

Cursos alternatius: Error en el protocol de seguretat.

Descripció detallada: El metge necessita incorporar nova informació a l'expedient d'un dels seus pacients. Aleshores sol·licita al gestor del sistema la inserció d'aquesta informació en el historial clínic.

5.1.2.2 Cas d'ús "Obtenir llista pacients propis"

Resum de la funcionalitat: El metge necessita obtenir una relació del pacients que visita.

Paper dins el treball de l'usuari: És una operació bàsica per el metge.

Actors: metge.

Precondició: El metge i el gestor han d'estar autenticats mútuament i el metge té assignat almenys un pacient.

Postcondició: El metge ha informat al gestor del sistema de que vol la llista dels seus pacients.

Cursos alternatius: -

Descripció detallada: El metge informa al gestor del sistema de la necessitat d'obtenir una relació de tots els pacients que estan a càrrec seu.

5.1.2.3 Cas d'ús "Recuperar llista pacients d'un metge"

Resum de la funcionalitat: El gestor obté la llista de pacients assignats al metge i li envia a aquest.

Paper dins el treball de l'usuari: És una operació bàsica per el metge.

Actors: gestor.

Precondició: El metge i el gestor han d'estar autenticats mútuament i el metge té assignat almenys un pacient.

Postcondició: El gestor li envia al metge la llista dels seus pacients.

Cursos alternatius: -

Descripció detallada: El gestor del sistema li envia al metge la relació de tots els pacients que estan a càrrec seu. Aquest rep la llista i la mostra per pantalla.

5.1.2.4 Cas d'ús "Guardar l'expedient del pacient"

Resum de la funcionalitat: El gestor rep la sol·licitud del metge per tal d'emmagatzemar un expedient que ha estat modificat.

Paper dins el treball de l'usuari: És una operació bàsica que ha de realitzar el gestor del sistema.

Actors: metge.

Precondició: El metge i el gestor han d'estar autenticats mútuament i el metge té informació per afegir a un expedient d'un pacient seu.

Postcondició: El metge ha informat al gestor del sistema de quina és la informació que cal afegir a l'expedient.

Cursos alternatius: Error en el protocol de seguretat.

Descripció detallada: El metge necessita incorporar nova informació a l'expedient d'un dels seus pacients. Aleshores sol·licita al gestor del sistema la inserció d'aquesta informació en el historial clínic.

5.2 Diagrama de classes bàsic

Segons els casos d'ús vistos fins ara i sense detallar els protocols, es té el següent diagrama de classes:

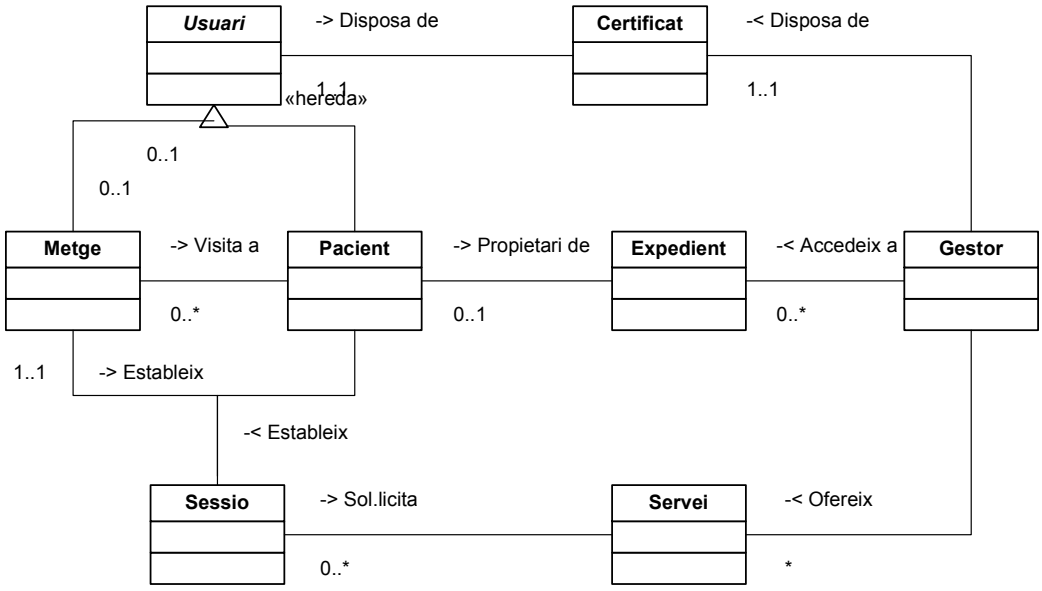


Figura 5-3. Primera aproximació al diagrama de classes.

De la figura anterior s'observa com Usuari és una classe abstracta, de la qual hereten les dues subclasses Metge i Pacient. Alhora es té que hi ha una associació entre Metge i Pacient, atès que un metge pot tenir varis pacients al seu càrrec.

També es pot veure com la classe Usuari estableix una relació d"associació amb la classe Certificat indicant que un usuari té un certificat. El mateix passa amb la classe Gestor i la classe Certificat, atès que el gestor té un certificat.

D'una altra banda es té una altra relació d'associació entre la classe Pacient i la classe Expedient, on s'indica que un pacient es propietari de cap o d'un expedient.

La classe Gestor també estableix una relació d'associació amb la classe Expedient, indicant que el gestor del sistema pot accedir a varis expedients mèdics.

També es pot apreciar com la classe Metge estableix una relació amb la classe Sessió, indicant que un metge té una sessió. El mateix passa entre la classe Pacient i la classe Sessió.
Per Sessió s'entén el conjunt d'atributs que caracteritzen una connexió concreta (un identificador de metge o de pacient, una data, una hora, un protocol de comunicacions, etc).

La classe Sessió i la classe Servei estableixen una relació, atès que una sessió sol·licita cap o varis serveis (consultar, inserir, desconnectar, etc.). Per últim es té una relació entre la classe Gestor i la classe Servei, on queda pales que el gestor ofereix varis serveis o operacions.

5.3 Anàlisi i Disseny

Els diagrames de casos d'ús anteriors son informació textual i poc formalitzada. L'objectiu d'aquesta etapa d'anàlisi és identificar les classes i objectes que seran la base de la implementació del programari.

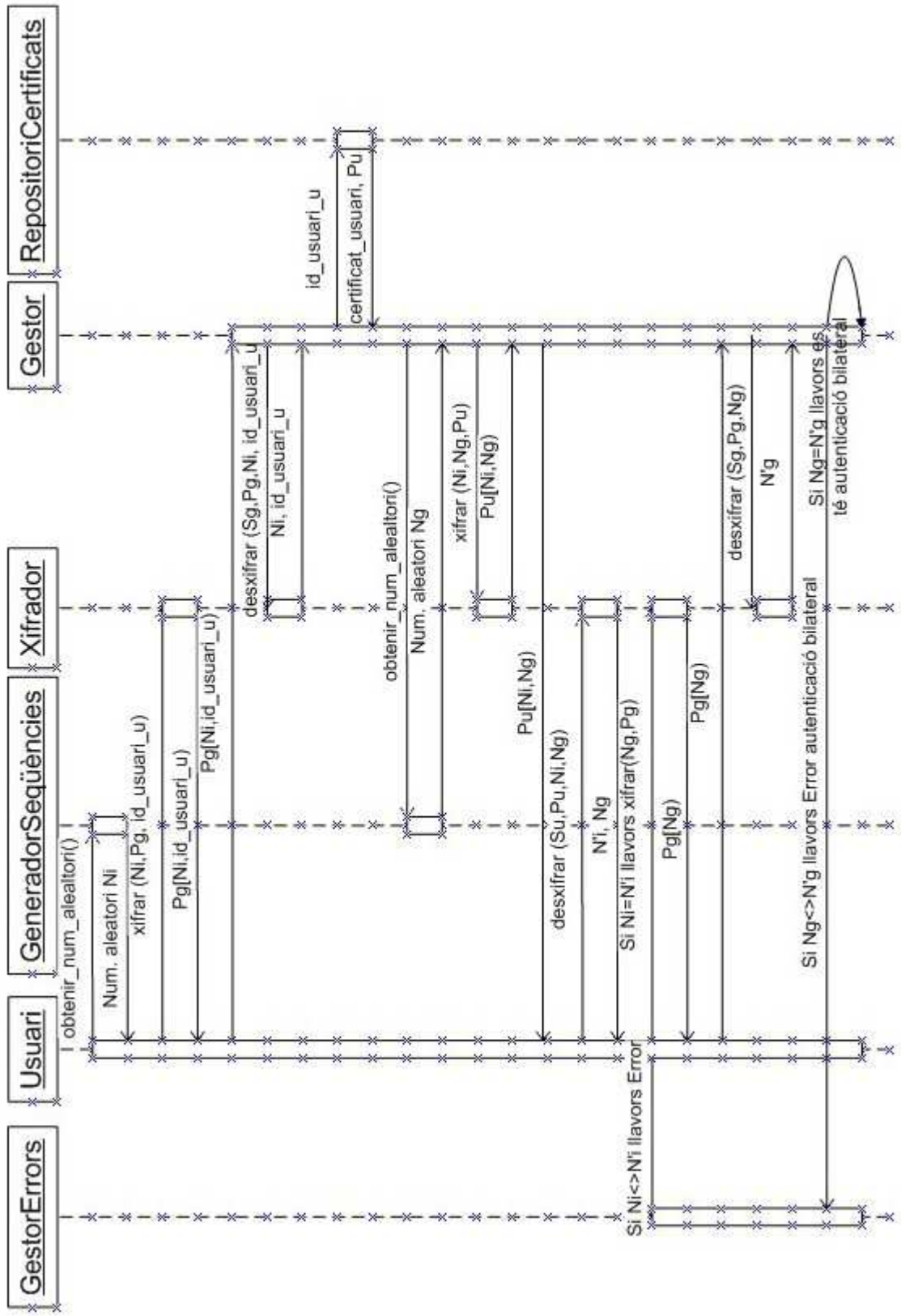
5.3.1 Diagrames de seqüències

L'objectiu principal de l'etapa d'anàlisi és fer l'especificació formal dels casos d'ús i una manera d'aconseguir aquesta formalització és mitjançant els diagrames de seqüències, atès que permeten detallar les interaccions entres les classes.

A les pàgines següents es presenten els diagrames de seqüències dels principals casos d'ús de l'aplicació:

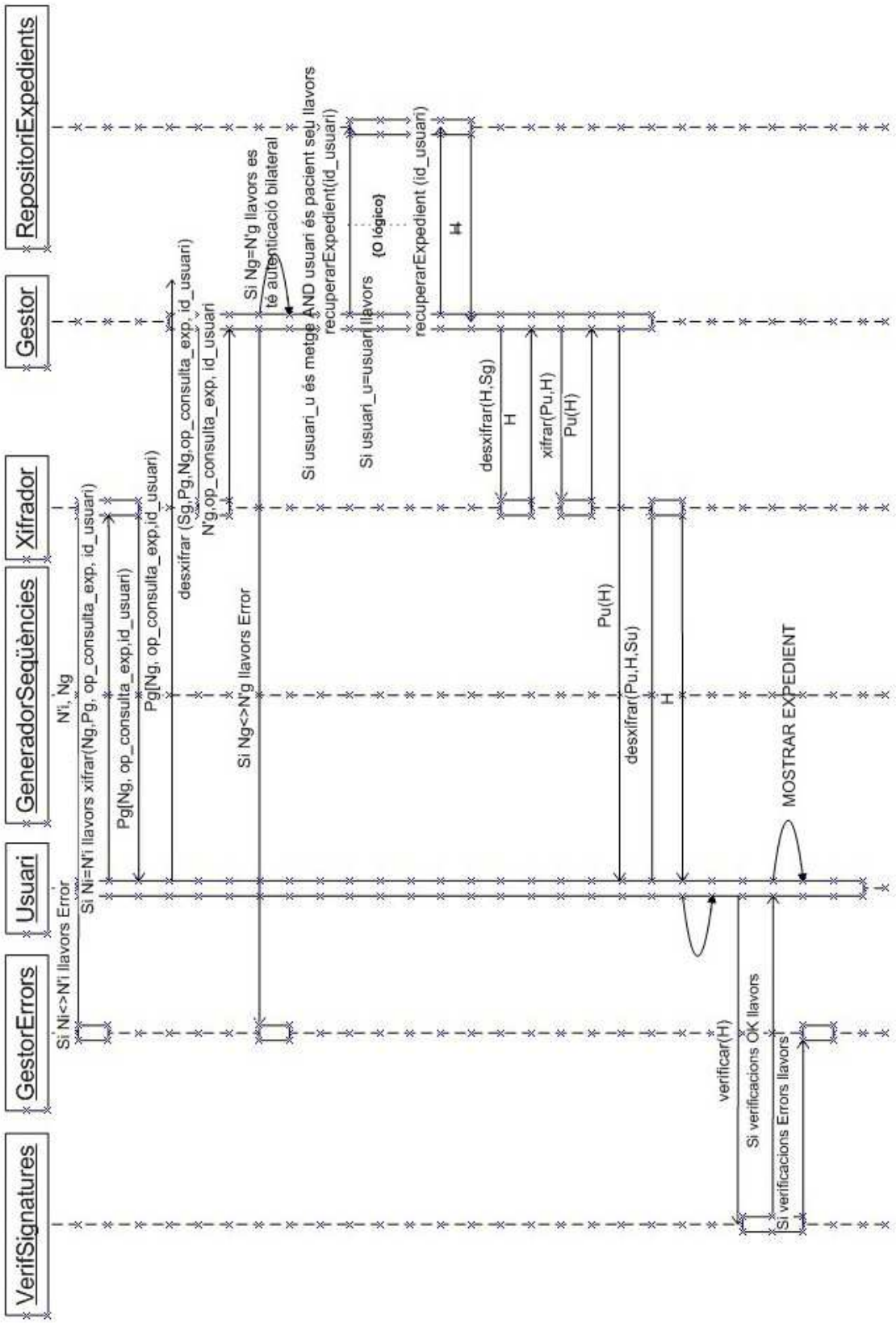
- Diagrama de seqüències dels cas d'ús "Autenticar-se".
- Diagrama de seqüències dels cas d'ús "Consultar expedient"
- Diagrama de seqüències dels cas d'ús "Obtenir llista pacients propis"
- Diagrama de seqüències dels cas d'ús "Inserir dades a l'expedient"

5.3.1.1 Diagrama de seqüències dels cas d'ús "Autenticar-se"



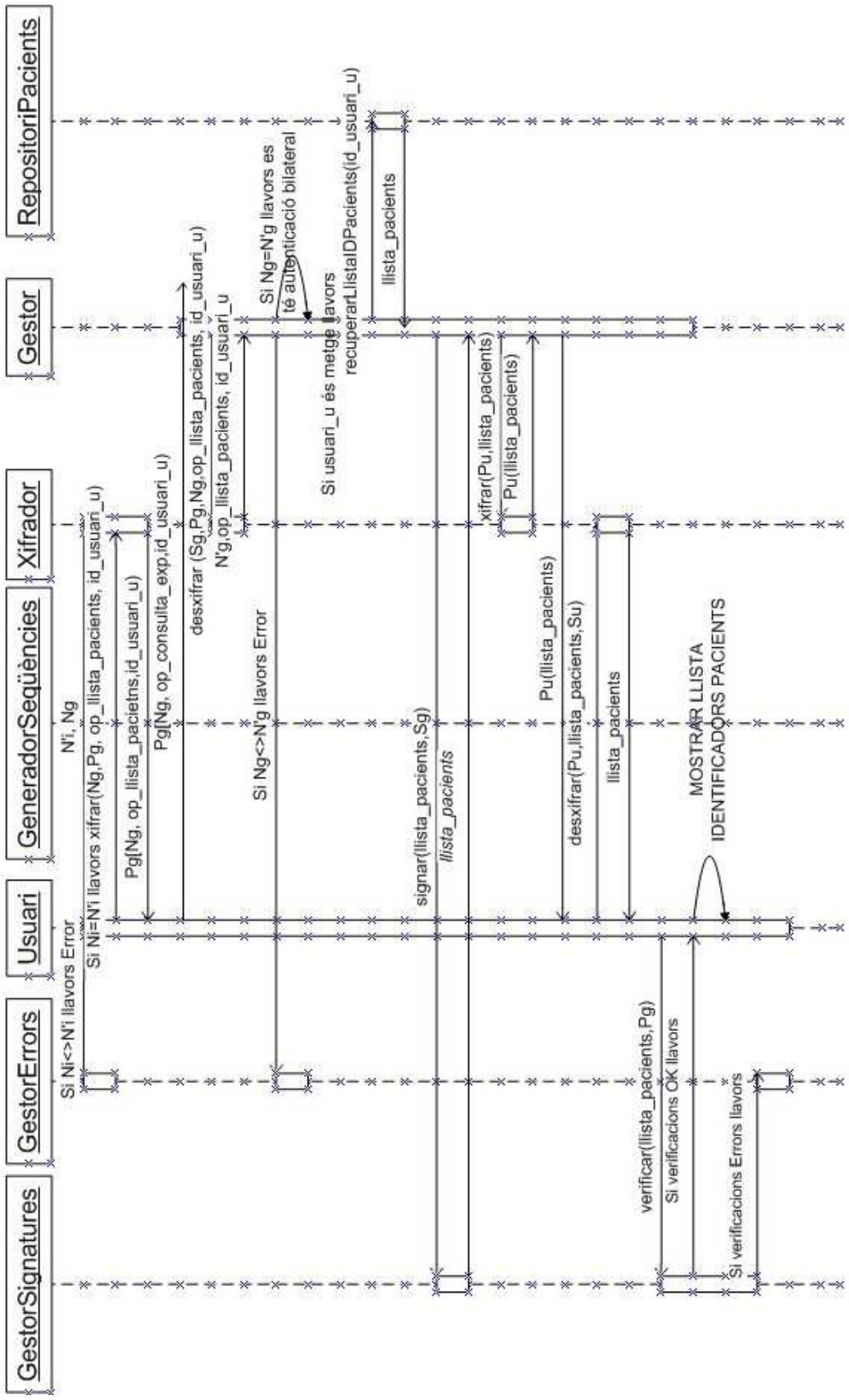
5.3.1.2 Diagrama de seqüències dels cas d'ús "Consultar expedient"

Nota: Es comença des del punt en que l'usuari és a punt d'autenticar al gestor del sistema.



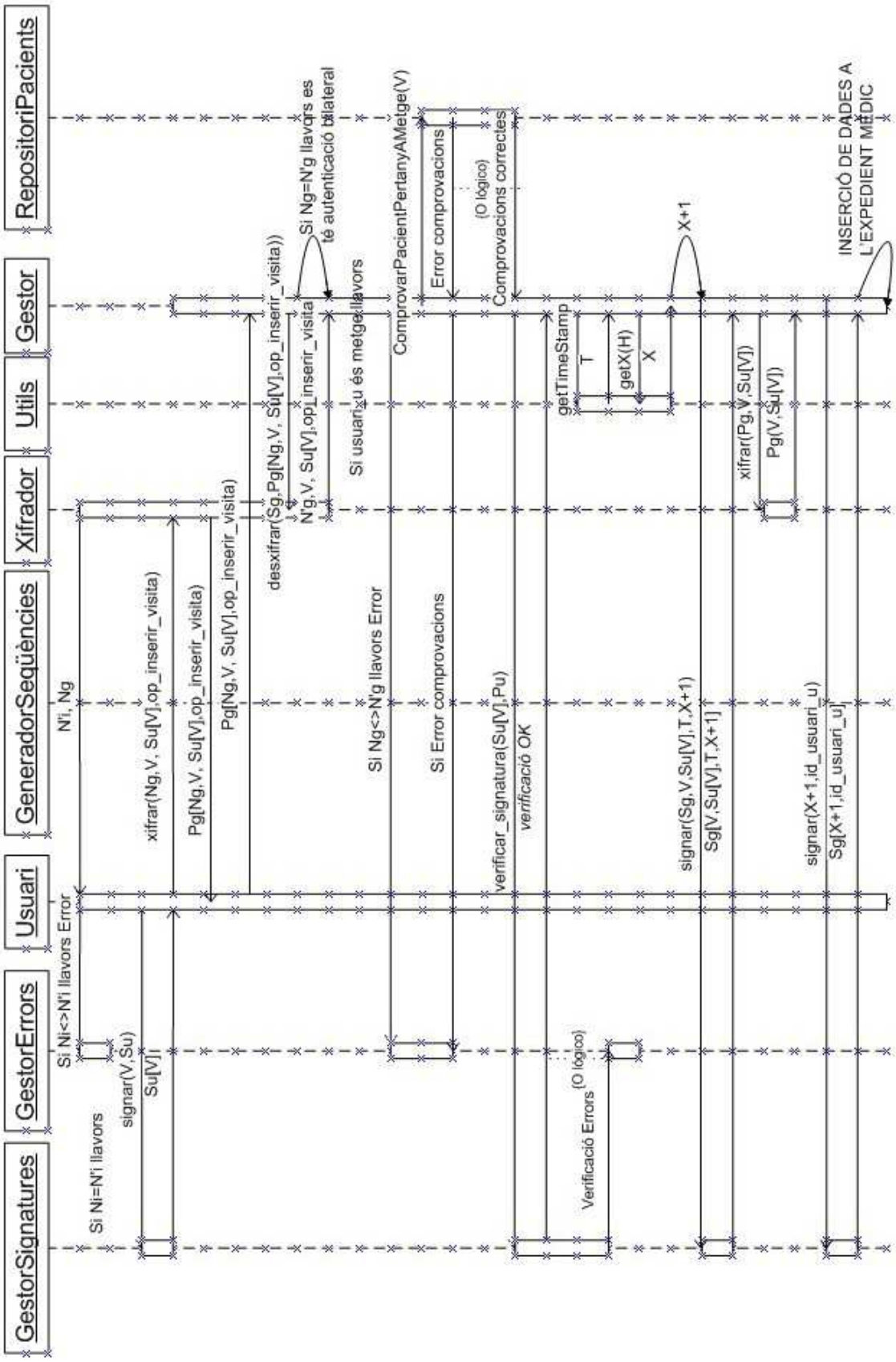
5.3.1.3 Diagrama de seqüències dels cas d'ús "Obtenir llista pacients propis"

Nota: Es comença des del punt en que l'usuari és a punt d'autenticar al gestor del sistema.



5.3.1.4 Diagrama de seqüències dels cas d'ús "Inserir dades a l'expedient"

Nota: Es comença des del punt en que l'usuari és a punt d'autenticar al gestor del sistema.



5.3.2 Diagrama de classes incloent la implementació dels protocols criptogràfics

Amb els diagrames de seqüències previs s'han expressat els casos d'ús de manera tal que han emergit totes les classes relacionades amb els protocols criptogràfics: Xifrador, GestorSignatures, GeneradorSequencies, Utils, etc.

Altres, com la classe P12 (encarregada de subministrar claus privades, públiques i certificats), s'han obviat en la representació dels diagrames de seqüències per tal de fer-los intel·ligibles, però apareixen en el diagrama de classes que es mostra en la figura 5-4:

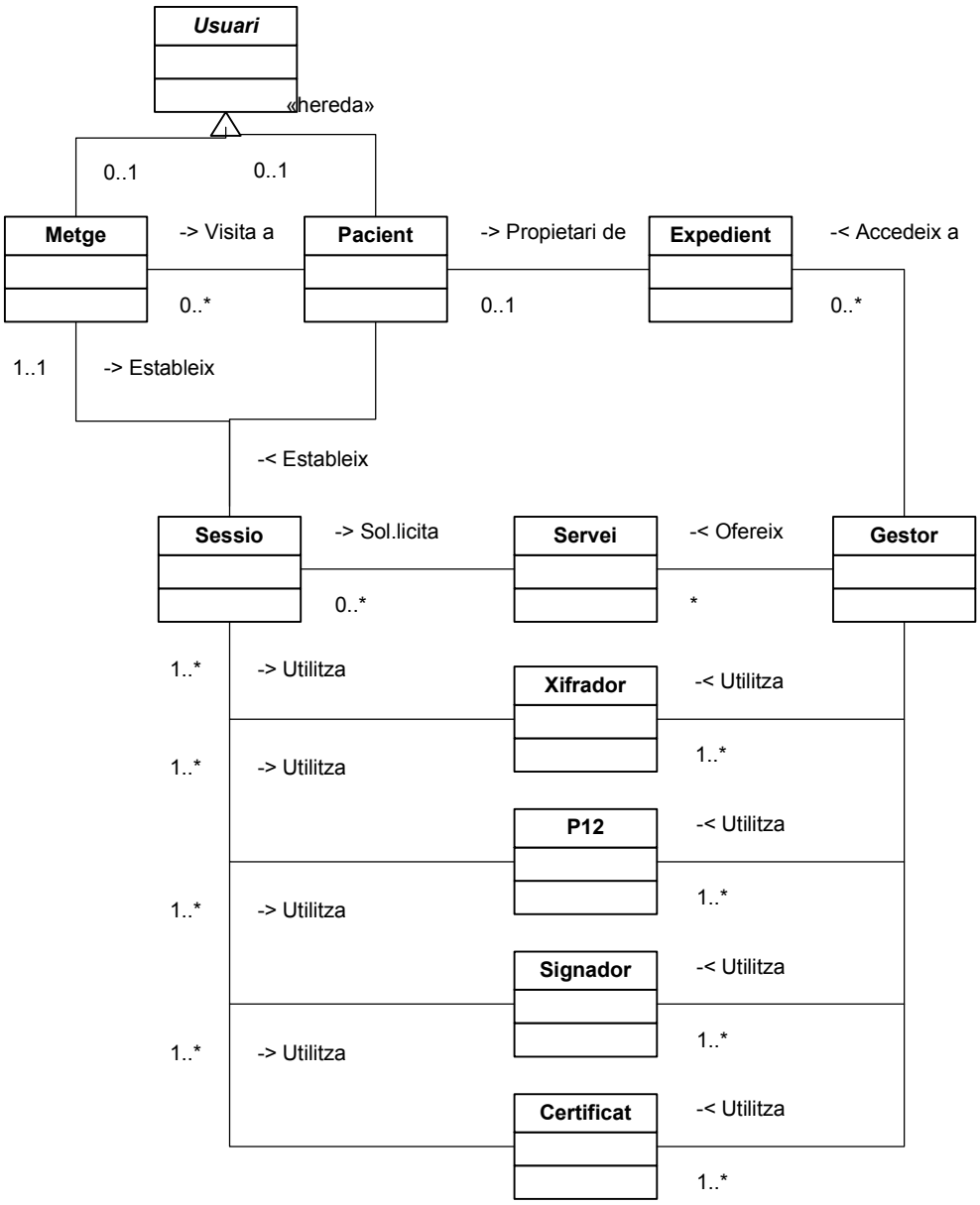


Figura 5-4 Diagrama de classes necessari per a implementar els protocols criptogràfics.

Explicació de les classes criptogràfiques de la figura 5-4:

Classe Servei

Realment no és cap classe. A l'apartat següent es veurà com queda substituïda per un paquet de classes anomenat Protocol, el qual inclou totes les classes que porten a terme els protocols: autenticació, consulta d'expedient, llistat dels pacients assignats a un metge, afegir dades a l'expedient mèdic.

Classe Xifrador

Implementa els mètodes que permeten xifrar i desxifrar informació.

Classe P12

Implementa els mètodes que permeten accedir als conjunts d'informació relatius als fitxers pkcs12: certificats, claus privades, públiques.

Classe Signador

Implementa els mètodes que possibiliten signar digitalment informació i verificar signatures.

Classe Certificat

Implementa els mètodes que permeten accedir a la informació relativa als certificats digitals: entitat emissora, nom del propietari del certificats, unitat organitzativa, etc.

5.3.3 Classes per a dur a terme els serveis i les operacions dels protocols

A l'apartat anterior s'ha emprat l'abstracció 'classe servei' per a referenciar els mètodes capaços de dur a terme els serveis i operacions objecte d'aquest PFC. Ara en aquest apartat s'explica en detall com s'han ideat aquestes classes.

S'ha pres la decisió d'implementar els serveis amb classes client i classes servidor, de manera que tot el concernent als clients quedés dins les primeres i tot el relatiu al servidor a les segones.

Així es té que la informació del certificat digital, fitxer .p12, claus, etc de cada client queda emmagatzemada en objectes d'una classe anomenada ProcolClient i la mateixa informació per al Gestor de l'aplicació queda recollida en un objecte d'una classe amb el nom ProtocolServidor.

Un cop que aquests objectes queden inicialitzats serveixen d'entrada per els constructors de les classes dels diversos protocols: autenticació, consulta d'expedient, llistat de pacients assignats a un metge, afegir dades a un historial.

A la figura 5-5 es pot observar aquesta idea:

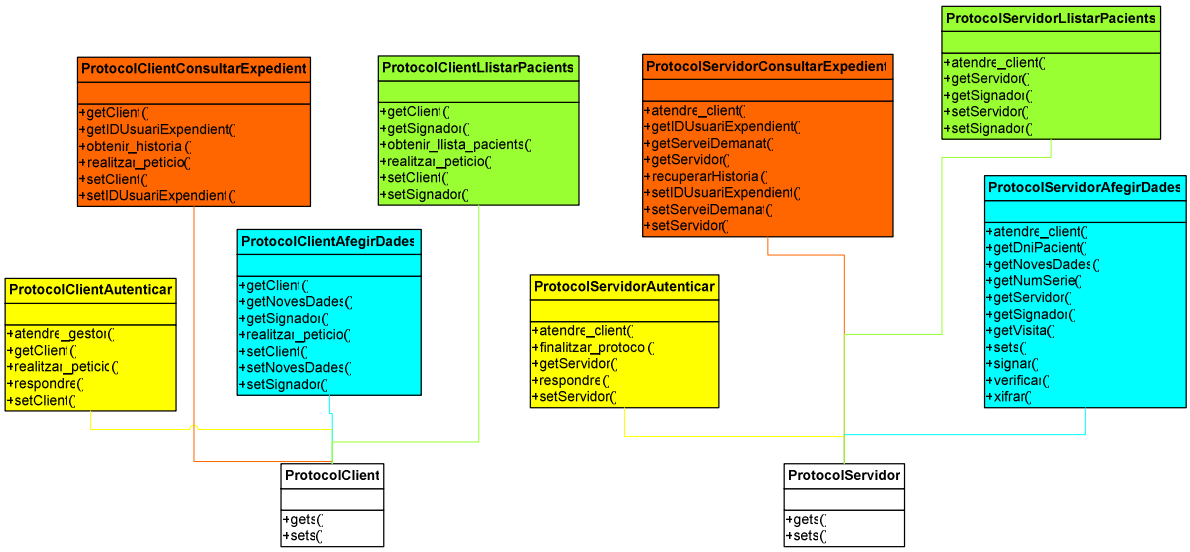


Figura 5-5. Diagrama de classes per a implementar les operacions que ofereix el servidor.

A la figura 5-5 si es pren la part client, per exemple, es té que totes les classes de servei necessiten la seva classe *ProtocolClient*, atès que és aquesta classe la que conté tota la informació bàsica relativa als criptosistemes: nom certificat, nom fitxer pcks12, clau privada, pública, contrasenyes, etc.

Per tant una classe client de servei necessitarà un objecte ProtocolClient com argument per el seu constructor.

Tot el que s'ha dit per la part client és vàlid per la part del servidor.

5.3.4 Diagrama complet: classes criptogràfiques i classes de servei

En aquest apartat s'arriba al punt en què es disposa d'un diagrama de classes plenament funcional, preparat per a ser implementat i provat amb un classe de test.

A la figura 5-6 es mostra el diagrama complet.

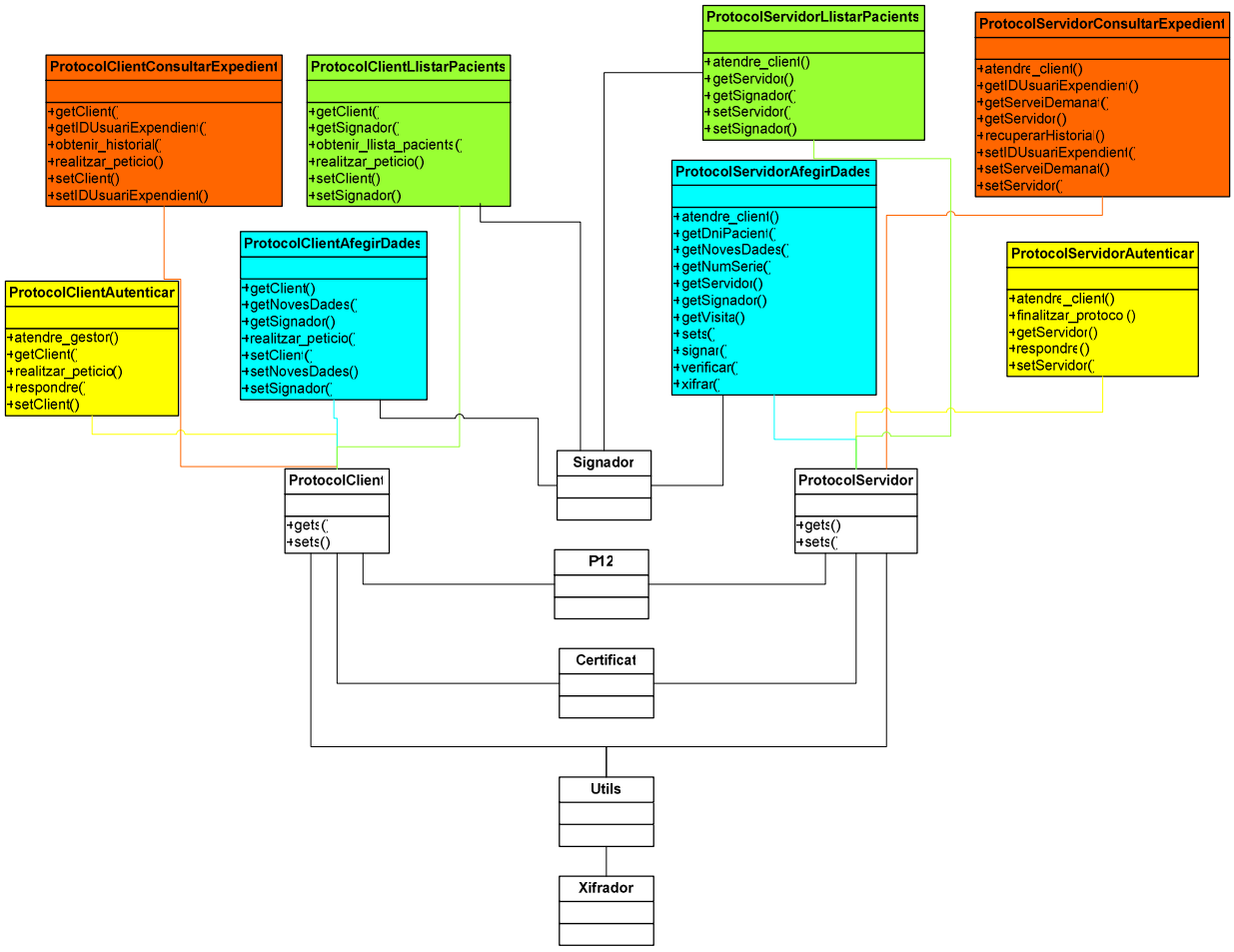


Figura 5-6. Diagrama de classes plenament funcional.

A la figura anterior s'han fusionat les classes criptogràfiques i les classes de servei per a obtenir una implementació plenament funcional.

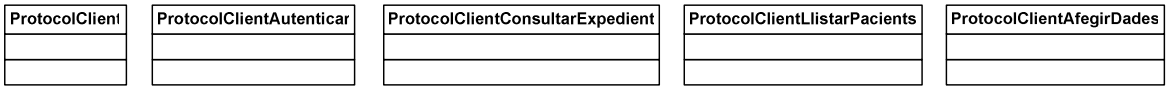
És interessant veure com una de les decisions de disseny ha estat compartir les classes criptogràfiques segons el ús: Certificat i P12 son utilitzades directament per ProtocolClient i ProtocolServidor, mentre que Xifrador és utilitzada a traves d'Utils, atès que afegeix operacions de tractament de fitxers i això dona més funcionalitat a les classes superiors.

D'una altra banda s'observa com Signador és utilitzada directament per les classes de servei relacionades amb l'obtenció del llistat de pacients i la inserció de dades en els historials mèdics; de fet son les úniques que necessiten accedir-hi.

5.3.5 Paquets de classes i subsistemes

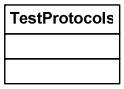
S'han agrupat les classes en paquets segons la seva funcionalitat

- Classes del paquet 'Protocols Client':



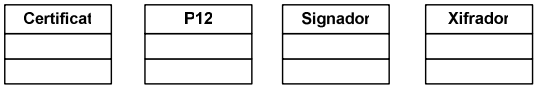
Aquest paquet engloba a totes les classes del client que implementen els mètodes per assolir els serveis i les operacions de l'aplicació.

- Classes del paquet 'Test Client':



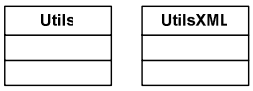
En aquest paquet només hi ha una classe amb la que es pot provar des del costat del client el correcte funcionament de l'aplicació.

- Classes del paquet 'Criptosistema':



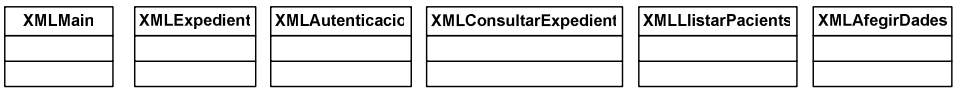
Aquest paquet agrupa les classes que permeten realitzar les operacions criptogràfiques mitjançant la llibreria IAIK.

- Classes del paquet 'Utils':



En el paquet utils es tenen dues classes, una per agrupar els mètodes auxiliars de la majoria de classes del programa i l'altra amb la mateixa finalitat però exclusivament per a classes XML.

- Classes del paquet 'XML'



Aquest paquet reuneix totes les classes que permeten crear, llegir i manipular documents XML.

- Classes del paquet 'Protocol Servidor':

ProtocolServidor	ProtocolServidorAutenticar	ProtocolServidorConsultarExpedient	ProtocolServidorLlistarPacients	ProtocolServidorAfegirDades

Aquest paquet engloba a totes les classes del servidor que implementen els mètodes (les interfícies remotes) per assolir els serveis i les operacions de l'aplicació.

- Classes del paquet 'RMI':

ServidorRM

El paquet RMI només té una classe: l'encarregada de publicar els serveis remots.

- Interficies del paquet 'Interficies Remotes':

IProtocolServidorAutenticar	IProtocolServidorConsultarExpedient	IProtocolServidorLlistarPacients	IProtocolServidorAfegirDades

Aquest paquet inclou les interficies que descriuen les operacions del servidor que seran accessibles des del costat del client.

Així doncs es té la següent representació de paquets:

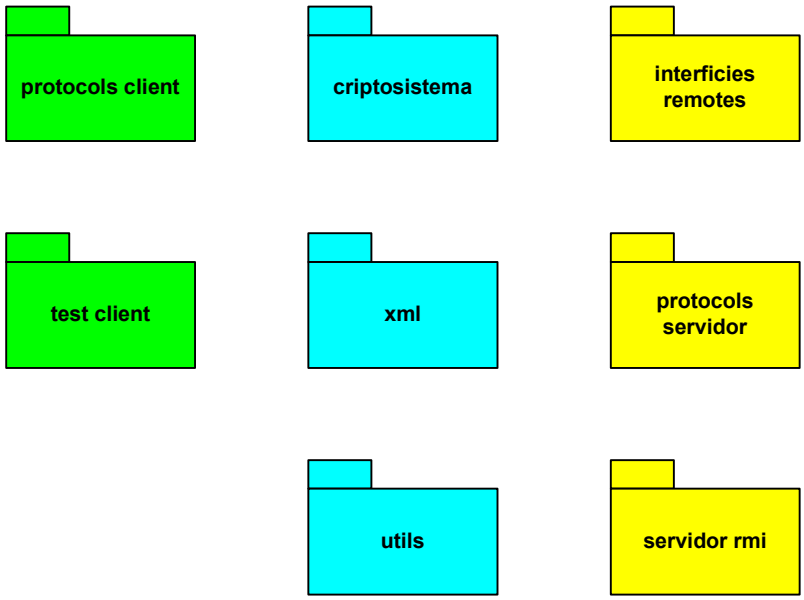


Figura 5-7. Paquets de classes.

De la mateixa manera que les diferents classes s'han agrupat en paquets, a continuació es mostra com els paquets s'han agrupat en subsistemes:

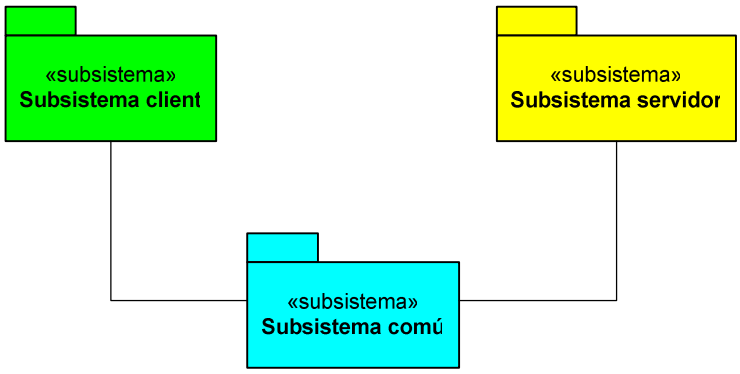


Figura 5-8. Subsistema de paquets.

Capítol 6. Representació de dades amb XML

6.1 Justificació de l'ús de la tecnologia XML

6.2 Diagrama de classes XML

6.3 Format dels documents XML

6.4 Integració amb el diagrama de classes anterior

6.1 Justificació de l'ús de la tecnologia XML

XML serveix per a estructurar, emmagatzemar i intercanviar informació de manera estandarditzada, per tant és una opció idònia per a un projecte d'aquestes característiques.

La necessitat d'intercanviar informació

Els clients i el gestor s'intercanvien una sèrie d'informació al llarg de tota l'aplicació. És tracta bàsicament de missatges per a indicar l'estat del procés i per a rebre u obtenir dades, com per exemple quan un metge demana l'expedient d'un pacient i el gestor li envia.

Tecnologia XML: API JDOM

Per a realitzar el intercanvi d'informació entre les parts s'utilitzen documents XML. Amb el API JDOM la gestió de documents XML des de Java és una tasca molt intuïtiva i eficient. Així la creació, la lectura i navegació dels documents és pot implementar de manera molt eficient.

6.2 Diagrama de classes XML

En el PFC s'ha decidit implementar unes classes específiques encarregades de la gestió dels documents XML, així les classes que porten a terme els protocols només han de crear objectes de les classes XML per a crear i llegir aquests tipus de documents.

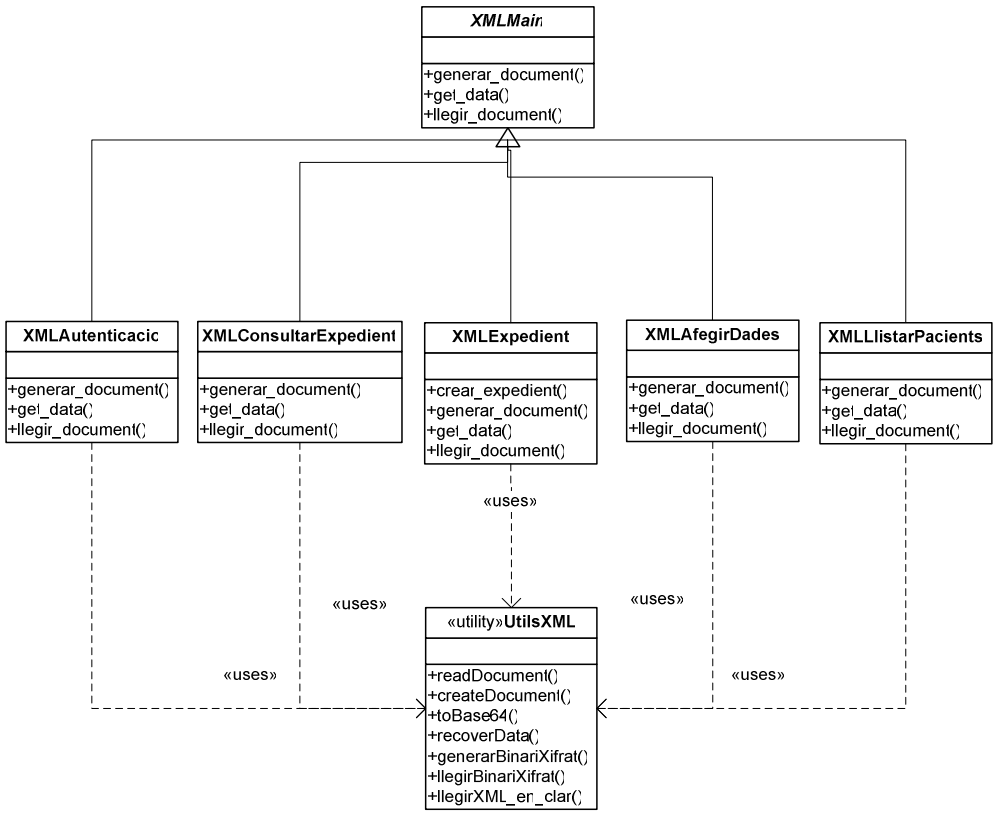


Figura 6-1. Classes per al tractament dels documents XML.

Per tant, es tenen les següents cinc classes especialitzades en la gestió de documents XML:

- XMLExpedient: Serveix per crear i llegir documents XML de la classe Expedient.
- XMLAutenticacio: Permet crear i llegir documents XML de les classes dels protocols d'autenticació.
- XMLConsultarExpedient: Utilitzada per a crear i llegir documents XML de les classes dels protocols de consulta d'expedients.
- XMLAfegirDades: Permet crear i llegir documents XML de les classes dels protocols que serveixen per a afegir dades als expedients
- XMLLlistarPacients: Serveix per a crear i llegir documents XML de les classes dels protocols que permeten obtenir la llista de pacients relativa a un metge.

Totes elles deriven de la classe abstracta XMLMain.

També hi ha una classe UtilsXML que implementa un conjunt de mètodes per a objectes de tipus XMLMain. Tots els objectes de les classes XML utilitzen aquests mètodes, per tant seria redundat que cada classe els incorpores dins seu.

Mètodes de UtilsXML:

- readDocument: Lectura d'un fitxer XML per tal de carregar-lo a memòria.
- createDocument: Guardar el document XML que tenim en memòria en un fitxer físic.
- toBase64: Es codifica la informació rebuda en Base64 i es retorna en forma de String
- recoverData: Deixa tot el document XML en un byte stream de sortida. Després amb el Parser i un byte stream d'entrada apuntant al anterior stream de sortida, obté el document XML ho converteix en un String i el retorna.
- generarBinariXifrat: Amb el document a la memòria escriu un fitxer a disc, xifra tot el contingut i genera un fitxer binari (.enc).
- llegirBinariXifrat: Llegeix un fitxer binari (.enc), el desxifra, el passa a xml, crea un objecte que sàpiga interpretar el document, el llegeix (el carrega a memòria) i retorna l'objecte.
- llegirXML_en_clar: Crea un objecte que sàpiga interpretar el document, el llegeix (el carrega a memòria) i retorna l'objecte.

6.3 Format dels documents XML

autenticacio.xml

Aquest fitxer serveix perquè client i gestor puguin efectuar el protocol d'autenticació, la correcta finalització del qual és necessària per tal que el gestor realitzi qualsevol operació demana pel client. Per a obtenir una explicació completa del protocol consultar el protocol d'autenticació al capítol 4 i el diagrama de seqüències al capítol 5.

Tal i com es pot apreciar a la figura 6-2 el primer tag és <GestioSeguraHistorialsClinics>, que serveix per a identificar el fitxer com a propi de l'aplicació. Després es té el tag <Autenticacio> amb el qual s'identifica el protocol on aquest fitxer porta a terme la seva funció.
En un tercer nivell hi ha els tags <Client> i <Gestor>, els quals contenen informació generada pel client i pel gestor, respectivament.

Quant el client:
<Id_client> on queda emmagatzemat el dni del client.
<Id_sequencia_client> on el client hi deixa un nombre aleatori que el gestor haurà de llegir.
<Id_sequencia_gestor_retornat> on el client hi deixa un nombre aleatori el qual originalment haurà estat generat pel gestor.

Quant al gestor:
<Id_gestor> on queda emmagatzemat el hash del certificat del gestor.
<Id_sequencia_gestor> on el gestor hi deixa un nombre aleatori que el client haurà de llegir.
<Id_sequencia_client_retornat> on el gestor hi deixa un nombre aleatori el qual originalment haurà estat generat pel client.

```
<?xml version="1.0" encoding="UTF-8"?>
<GestioSeguraHistorialsClinics>
  <Autenticacio>
    <Client>
      <Id_client />
      <Id_sequencia_client />
      <Id_sequencia_gestor_retornat />
    </Client>
    <Gestor>
      <Id_gestor />
      <Id_sequencia_gestor />
      <Id_sequencia_client_retornat />
    </Gestor>
  </Autenticacio>
</GestioSeguraHistorialsClinics>
```

Figura 6-2. Format del document autenticacio.xml

consulta_expedient.xml

Amb aquest fitxer les parts poden realitzar el protocol de consulta d'expedient, la finalitat del qual és que el client obtingui el historial clínic del pacient solicitat.
Per a obtenir una explicació completa del protocol consultar el capítol 4 i el diagrama de seqüències al capítol 5.

A la figura 6-3 es pot observar com el primer tag és <GestioSeguraHistorialsClinics>, que serveix per a identificar el fitxer com a propi de l'aplicació. Després es té el tag <ConsultarExpedient> amb el qual s'identifica el protocol on aquest fitxer porta a terme la seva funció.
En un tercer nivell hi ha els tags <Client> i <Gestor>, els quals contenen informació generada pel client i pel gestor, respectivament.

Quant el client:
<Id_usuari_expedient> on el client escriu el dni del historial que vol consultar.
<Id_sequencia_gestor_retornat> relatiu al protocol d'autenticació, on el client hi deixa un nombre aleatori el qual originalment haurà estat generat pel gestor.
<Servei_demanat> on el client hi deixa el nom de l'operació sol·licitada. De moment és redundant atès que amb el tag <ConsultarExpedient> n'hi hauria suficient. Està pensat per futures ampliacions.

Quant el gestor:
<Historial> on queda emmagatzemat el historial mèdic del pacient.

```
<?xml version="1.0" encoding="UTF-8"?>
<GestioSeguraHistorialsClinics>
  <ConsultarExpedient>
    <Client>
      <Id_usuari_expedient />
      <Id_sequencia_gestor_retornat />
      <Servei_demanat />
    </Client>
    <Gestor>
      <Historial />
    </Gestor>
  </ConsultarExpedient>
</GestioSeguraHistorialsClinics>
```

Figura 6-3. Format del document consultar_expedient.xml

llistar_pacients.xml

Aquest fitxer serveix perquè el client que sigui un metge pugui obtenir una relació dels seus pacients.
Per a obtenir una explicació completa del protocol consultar el capítol 4 i el diagrama de seqüències al capítol 5.

Tal i com es pot apreciar a la figura 6-4 el primer tag és <GestioSeguraHistorialsClinics>, que serveix per a identificar el fitxer com a propi de l'aplicació. Després es té el tag <LlistarPacients> amb el qual s'identifica el protocol on aquest fitxer porta a terme la seva funció.
En un tercer nivell hi ha els tags <Client> i <Gestor>, els quals contenen informació generada pel client i pel gestor, respectivament.

Quant el client:
<Id_sequencia_gestor_retornat> relatiu al protocol d'autenticació, on el client hi deixa un nombre aleatori el qual originalment haurà estat generat pel gestor.
<Servei_demanat> on el client hi deixa el nom de l'operació sol·licitada. De moment és redundant atès que amb el tag <LlistarPacients> n'hi hauria suficient. Està pensat per futures ampliacions.

Quant al gestor:
<Llista_pacients> un seguit de <Id_pacients>
<Id_pacients> on el gestor deixa els dni dels pacients que estan a càrrec del metge.
<Signatura_gestor> on el gestor escriu la seva signatura digital relativa a la llista de pacients.

```
<?xml version="1.0" encoding="UTF-8"?>
<GestioSeguraHistorialsClinics>
  <LlistarPacients>
    <Client>
      <Id_sequencia_gestor_retornat />
      <Servei_demanat />
    </Client>
    <Gestor>
      <Llista_pacients>
        <Id_pacient />
      </Llista_pacients>
      <Signatura_gestor />
    </Gestor>
  </LlistarPacients>
</GestioSeguraHistorialsClinics>
```

Figura 6-4. Format del document llistar_pacients.xml

afegir_dades.xml

Amb aquest fitxer el client que sigui un metge autoritzat pot afegir una nova visita dins l'expedient d'un dels seus pacients. Per a obtenir una explicació completa del protocol consultar el capítol 4 i el diagrama de seqüències al capítol 5.

A la figura 6-5 es pot apreciar com el primer tag és <GestioSeguraHistorialsClinics>, que serveix per a identificar el fitxer com a propi de l'aplicació. Després es té el tag <AfegirDades> amb el qual s'identifica el protocol on aquest fitxer porta a terme la seva funció. En un tercer nivell hi ha els tags <Client> i <Gestor>, els quals contenen informació generada pel client i pel gestor, respectivament.

Quant el client:

<Id_sequencia_gestor_retornat> relatiu al protocol d'autenticació, on el client hi deixa un nombre aleatori el qual originalment haurà estat generat pel gestor.

<Servei_demanat> on el client hi deixa el nom de l'operació sol·licitada. De moment és redundant atès que amb el tag <LlistarPacients> n'hi hauria suficient. Està pensat per futures ampliacions.

<Visita> format al seu torn per els següents elements:

- <Num_expedient> el dni del pacient al qual pertany el historial mèdic.
- <Data_visita> la data del dia en que s'ha realitzat la visita.
- <Servei_visita> el departament mèdic que ha intervingut.
- <Metge_visita> el dni del metge que ha realitzat la visita.
- <Descripcio_visita> explicació del motius de la visita, símptomes, etc.
- <Diagnostic> explicació de la resolució adoptada pel metge.
- <Signatura_metge> signatura digital del metge relativa a les dades de la visita.

Quant al gestor, només es té el tag <Resposta> on el gestor indica si la visita s'ha afegit correctament a l'expedient del pacient.

```
<?xml version="1.0" encoding="UTF-8"?>
<GestioSeguraHistorialsClinics>
  <AfegirDades>
    <Client>
      <Id_sequencia_gestor_retornat />
      <Servei_demanat />
      <Visita>
        <Num_expedient />
        <Data_visita />
        <Servei_visita />
        <Metge_visita />
        <Descripcio_visita />
        <Diagnostic_visita />
        <Signatura_metge_visita />
      </Visita>
    </Client>
    <Gestor>
      <Resposta />
    </Gestor>
  </AfegirDades>
</GestioSeguraHistorialsClinics>
```

Figura 6-5. Format del document afegir_dades.xml

6.4 Integració amb el diagrama de classes anterior

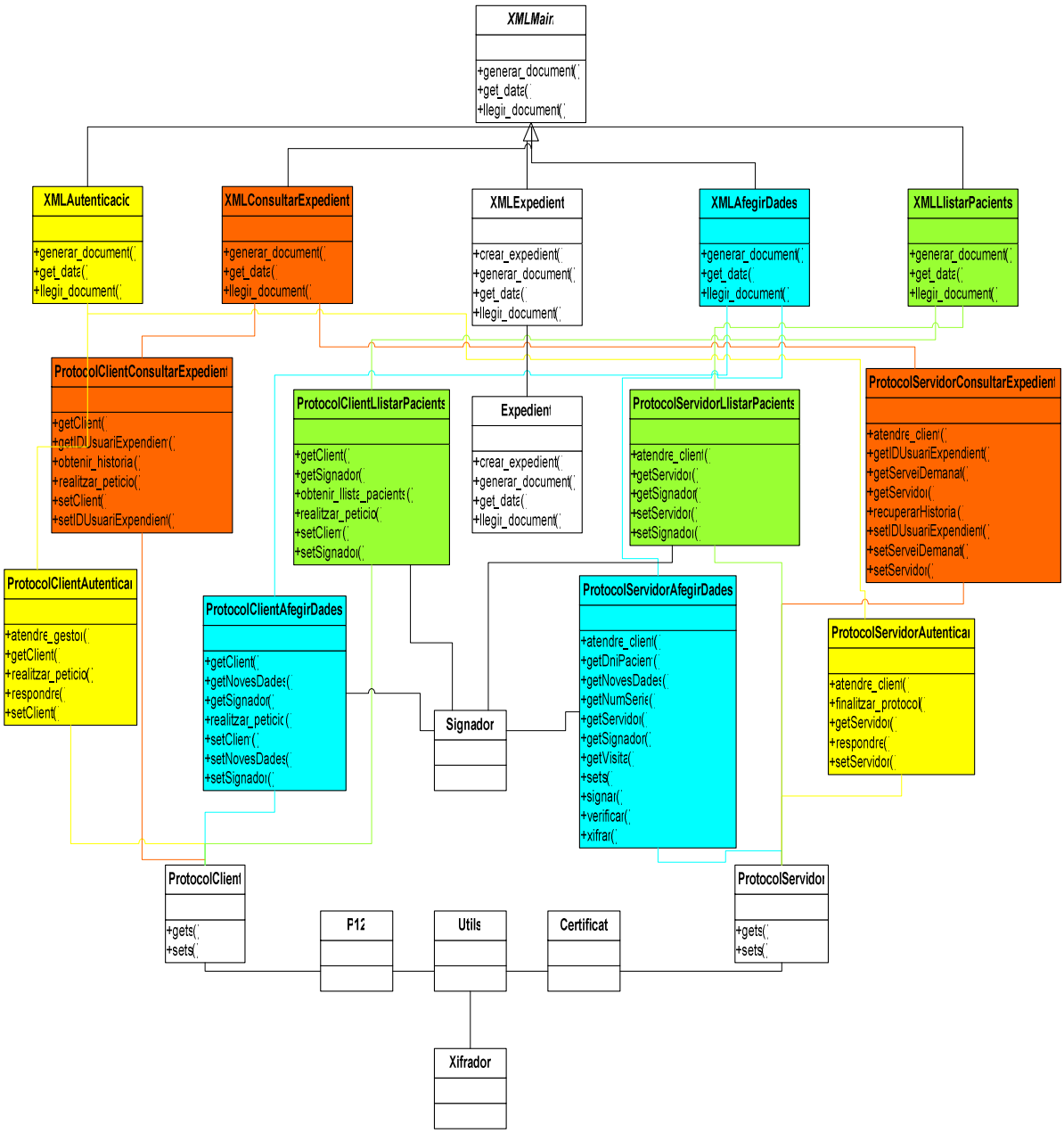


Figura 6-6. Diagrama de classes incloent les classes XML.

Capítol 7. Comunicació dels components amb RMI

7.1 Aplicació distribuïda

7.2 Adaptant l'aplicació a RMI

7.3 Provar el funcionament de RMI

7.4 Diagrama de classes RMI

7.1 Aplicació distribuïda

Una de les característiques més potents d'aquest PFC rau en el fet que l'aplicació desenvolupada ha de ser distribuïda, això és que les parts que la conformen han de tenir suficient autonomia per a poder executar-se en diferents ordinadors visibles entre ells mitjançant una xarxa de comunicacions. Aquest fet ha de ser totalment transparent a l'usuari i per tant l'aplicació ha de respondre amb la mateixa coherència que ho faria un programa monolític.

De les diferents tecnologies que fan possible desenvolupar aplicacions distribuïdes s'ha optat, donada la seva simplicitat i eficiència, per RMI (Java Remote Method Invocation).

Algunes característiques de RMI:

- És un mecanisme inclòs dins del estàndard del entorn d'execució Java per tal de poder invocar mètodes remotament.
- És idoni si les diferents parts de l'aplicació son totes Java, com és el cas d'aquest PFC.
- Permet executar l'aplicació sobre qualsevol sistema operatiu que pugui executar la màquina virtual de Java.

7.2 Adaptant l'aplicació a RMI

Fins el moment d'implementar RMI el programa ha hagut de simular el funcionament d'una aplicació distribuïda. Així per a dur a terme l'execució de cada protocol, la invocació dels mètodes dels objectes clients i de l'objecte servidor de les classes dels protocols s'han anat alternant des d'una mateixa classe de prova.

Per a adaptar l'aplicació a RMI amb el mínim impacte sobre el codi ja existent, s'ha optat perquè la part client del programa incorpori totes les classes client i en el moment de dur a terme els protocols, executi el seu propi codi client (local) i vagi invocant tal com els necessiti els mètodes servidors a partir de referències a objectes remots obtingudes mitjançant RMI.

Així el procés client és qui inicia, articula i finalitza cada protocol, emprant el seu propi codi implementat localment i executant el codi remot del servidor.

7.3 Provar el funcionament de RMI

A continuació es detalla, a tall d'exemple, els passos necessaris per a executar dins d'una intranet el protocol d'autenticació. L'escenari està format per dos ordinadors amb sistema operatiu Windows XP i amb l'entorn d'execució de Java SE 1.6.0_2:

Prerequisits

És condició indispensable que els dos ordinadors es comuniquin correctament. Cal provar de fer un ping en ambdues maquines atès que si el ping no respon el programa no funcionarà. En cas negatiu comprovar que l'antivirus no bloquegi el protocol ICMP.

Per tal de poder obtenir les referències remotes cal assegurar que el programa client apunta a l'adreça IP del host que executa el programa servidor.

Les carpetes 'certificats' i 'missatges' han d'estar a dins del directori 'bin'. A la primera hi ha de ser els fitxers de client generats amb el openssl. A la segona hi ha d'haver l'estructura de directoris que el programa espera per a poder deixar el arxius xml (en aquesta prova és suficient amb l'existència del directori 'missatges/autenticar/').

Els binaris que necessita el client

A la carpeta bin s'esperen els fitxers de classes propis del client dins de cada subdirectori:

criptosistema/

Certificat.class
P12.class
Signador.class
Xifrador.class

entitats/

Expedient.class
Metge.class
Pacient.class
Visita.class

protocols/

ProtocolClient.class
ProtocolClientAutenticar.class

test/

TestProtocolsRMI.class

utils/

Utils.class

UtilsXML.class

xml/

XMLAfegirDades.class

XMLAutenticacio.class

XMLConsultarExpedient.class

XMLExpedient.class

XMLLlistarPacients.class

XMLMain.class

D'una altra banda es necessita el stub del protocol d'autenticació de la part servidora:

protocols/

ProtocolServidorAutenticar_Stub.class

Així com la interficie remota:

interficies_remotes/

IProtocolServidorAutenticar.class

Respecte al programari de base

- Instal.lar el Java (amb el JRE n'hi ha suficient doncs no hem de compilar res)

- Per a poder emprar en Java qualsevol longitud de clau s'utilitzen les polítiques de seguretat de Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 6. Descomprimir el fitxer "jce_policy-6.zip" i copiar els fitxers "local_policy.jar" i "US_export_policy.jar" al directori […]\lib\security.

- Copiar les llibreries 'iaik_jce_full.jar' i 'jdom.jar' al mateix directori on executarem el programa client.

De la banda del servidor

Comprovar que s'estigui executant el RMI registry i el programa Java servidor responsable de publicar els serveis remots.

7.4 Diagrama de classes RMI

Utilitzar RMI comporta fer uns canvis en l'aplicació, atès que s'afegeixen noves classes a ambdues bandes (client/servidor) i s'estableixen un seguit de relacions entre elles.

A la figura 7-1 es pot observar com les classes ProtocolServidorX ofereixen tota una sèrie de mètodes mitjançant les interficies remotes. Alhora es té la classe ServidorRMI que a partir de les classes ProtocolServidorX i les interficies remotes publica els serveis perquè estiguin disponibles als clients.

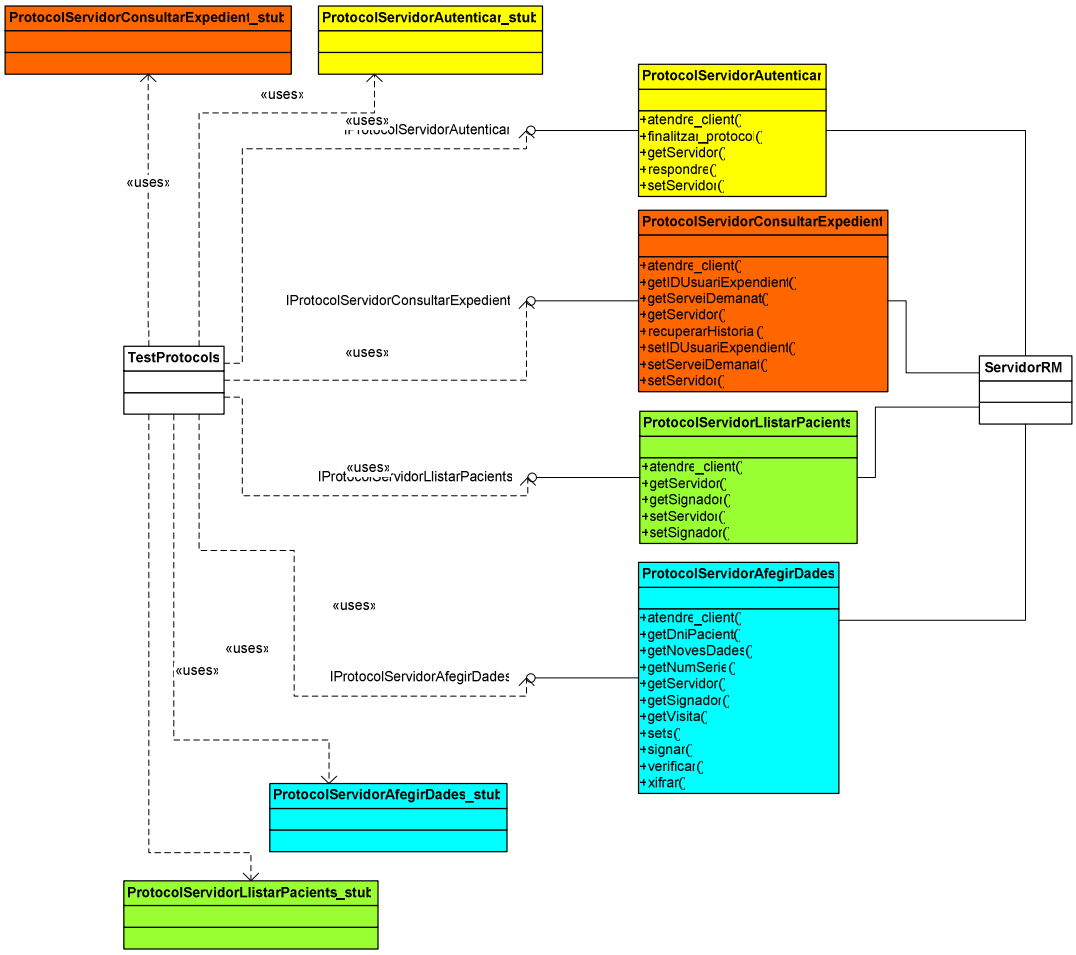


Figura 7-1. Diagrama de classes RMI.

Respecte el client, es pot apreciar com utilitza les interficies i els fitxers stub (succedanis dels objectes remots) generats pel servidor.

Capítol 8. Gestió de la persistència amb MySQL

8.1 Necessitat d'un SGBD

8.2 El SGBD MySQL

8.3 El model de dades persistents del projecte

8.3.1 Conjunt 1: informació pròpia del Gestor.

8.3.2 Conjunt 2: informació sobre els certificats dels usuaris del sistema.

8.3.3 Conjunt 3: informació relativa als metges, als pacients i a la relació entre ambdós grups.

8.3.4 Conjunt 4: informació sobre els historials clínics dels pacients.

8.3.5 Conjunt 5: informació relativa a les sessions autenticades que el Gestor manté amb els clients.

8.4 La classe DBManager

8.1 Necessitat d'un SGBD

L'aplicació necessita de tot un conjunt d'informació estructurada que cal ser processada pels diferents actors del sistema: gestor, metges i pacients. Les dades flueixen entre ells segons els protocols d'autenticació, consulta d'expedient, inserció d'informació i obtenció de llista de pacients, sempre assegurant les propietats de seguretat de la informació.

Part d'aquesta informació, com els expedients mèdics dels pacients, canvia amb certa freqüència mentre que altra, com els certificats digitals dels actors, és pràcticament invariant. Per tant, son necessaris mecanismes àgils i robusts per a poder recuperar i emmagatzemar consistentment la informació.

Per tant, és clar que l'ús d'un SGBD és una manera òptima de resoldre el problema de la persistència de les dades en una aplicació de gestió.

8.2 El SGBD MySQL

Dels diferents SGBD client/servidor relacionals existents s'ha optat per MySQL, per ser un producte de bona qualitat, portable i per disposar de llicència GPL.

Consultar l'annex B per a veure les instruccions d'instal·lació i la creació de la base de dades del projecte.

8.3 El model de dades persistents del projecte

El Gestor de l'aplicació és l'únic que accedeix al SGBD. Per tal de portar a terme les seves funcions aquest actor necessita els següents conjunts d'informació:

Nota prèvia: la part de l'aplicació del costat del servidor disposa de la classe 'Inicialitzar' en el package 'test' que permet inserir automàticament els conjunts de dades que s'expliquen tot seguit.

8.3.1 Conjunt 1: informació pròpia del Gestor.

El fitxer del certificat digital, el fitxer PKCS12 i el password d'aquest últim.

El Gestor necessita obtenir aquesta informació que li és pròpia per tal de complir amb els protocols de l'aplicació. La idea és que l'administrador de l'aplicació carregui aquests valors

una sola vegada abans d'iniciar per primera vegada l'aplicació (es tracta d'una informació que rarament canviarà).
Aquests valors s'emmagatzemen a la taula 'cripto_gestor', la qual tindrà un sol registre.

A continuació es presenta la definició de la taula 'cripto_gestor':

Nom camp	Tipus de dades	Descripció
Id	Varchar(64)	Identificador del registre
certificat	BLOB	Certificat del gestor del sistema
P12	BLOB	Fitxer pkcs12 del gestor del sistema
password_p12	Varchar(64)	Password del pkcs12

8.3.2 Conjunt 2: informació sobre els certificats dels usuaris del sistema.

Identificador d'usuari i el certificat associat.

Per tal de dur a terme els protocols el gestor necessita contínuament accedir a una taula on per a cada usuari del sistema existeixi el seu certificat digital.

La taula 'certificat_usuari' expressa aquesta necessitat:

Nom camp	Tipus de dades	Descripció
Id_usuari	Varchar(10)	Identificador del client (metge o pacient)
certificat	BLOB	Certificat del client (metge o pacient)

De manera semblant a l'apartat anterior, l'administrador de l'aplicació carregarà aquesta taula quan un nou usuari hagi d'utilitzar el sistema.

8.3.3 Conjunt 3: informació relativa als metges, als pacients i a la relació entre ambdós grups.

Identificador de metge, nom, cognoms, departament.
Identificador de pacient, nom, cognoms, identificador del metge associat.

Amb les taules 'metge' i 'pacient' se soluciona el conjunt 3:

Taula Metge		
Nom camp	**Tipus de dades**	**Descripció**
nif_metge	Varchar(10)	Identificador del metge
nom_metge	Varchar(64)	Nom del metge
cognom1_metge	Varchar(64)	Primer cognom
cognom2_metge	Varchar(64)	Segon cognom
departament	Varchar(20)	Departament on treballa

Taula Pacient		
Nom camp	**Tipus de dades**	**Descripció**
nif_pacient	Varchar(10)	Identificador del pacient
nom_pacient	Varchar(64)	Nom del pacient
cognom1_pacient	Varchar(64)	Primer cognom
cognom2_pacient	Varchar(64)	Segon cognom
id_metge	Varchar(10)	Nif del metge que el visita

També en aquest cas l'administrador de l'aplicació carregarà aquestes taules quan calgui enregistrar nous metges o pacients.

8.3.4 Conjunt 4: informació sobre els historials clínics dels pacients.

El historial mèdic, atès que conté tota la vida clínica dels pacients, és una entitat d'importància cabdal que tots els actors han de processar, ja sigui per a escriure-hi o per consultar.
El formen dos blocs d'informació: el historial i les visites.

Respecte al historial:

- Identificador del historial
- Dades personals del pacient
- Numero de visites que conformen el historial
- Signatura digital del Gestor respecte el numero de visites del historial

Quant a les visites del historial, per a cada visita és necessari:

- Numero de visita
- Marca de temps
- Informació xifrada pel Gestor (les dades enviades pel metge relatives a la visita i la signatura digital del metge sobre aquestes dades)

- Informació signada pel Gestor (la signatura digital del Gestor sobre la visita enviada pel metge, la signatura d'aquest respecte la visita, el numero i la marca de temps que corresponen a la visita)

Els metges son els únics que poden crear expedients i afegir visites, sempre i quan es tracti de d'expedients dels seus respectius pacients.
Els pacients només poden consultar els seus propis expedients. Els metges poden consultar els expedients dels seus respectius pacients, sempre i quan la informació de les visites no tingui l'accés restringit.

El Gestor del sistema implementa el mecanismes que garanteixen la seguretat per tal de recuperar de manera correcta als expedients i per a emmagatzemar-los de manera consistent a la base de dades.

Amb les taules 'expedient' i 'visita' s'expressa el conjunt 4:

Taula Expedient		
Nom camp	**Tipus de dades**	**Descripció**
num_exp	Varchar(10)	Identificador de l'expedient
nom_pacient	Varchar(64)	Nom del pacient
cognom1_pacient	Varchar(64)	Primer cognom
cognom2_pacient	Varchar(64)	Segon cognom
telefon_pacient	Varchar(15)	Telèfon del pacient
num_ultima_visita	Char(3)	Nombre de visites que té l'expedient
signatura_num_ultima_visita	BLOB	Signatura digital del Gestor respecte el numero de visites del historial

Taula Visita		
Nom camp	**Tipus de dades**	**Descripció**
num_exp	Varchar(10)	Identificador de l'expedient
num_visita	Char(3)	Número de visita
marca_temps	Varchar(23)	Data i hora de creació de la visita Exemple: 2007-12-01 17:28:58.984
info_xifrada_gestor	BLOB	Informació xifrada pel Gestor (les dades enviades pel metge relatives a la visita i la signatura digital del metge sobre aquestes dades)

info_signada_gestor	BLOB	Informació signada pel Gestor (la signatura digital del Gestor sobre la visita enviada pel metge, la signatura d'aquest respecte la visita, el numero i la marca de temps que corresponen a la visita)

En aquest cas son els metges els que creen expedients i afegeixen les visites, així doncs l'administrador de l'aplicació no ha de fer res.

8.3.5 Conjunt 5: informació relativa a les sessions autenticades que el Gestor manté amb els clients.

Número de seqüència del gestor, número de seqüència del client, identificador del client, marca de temps del inici de la connexió.

El model d'autenticació implementat no contempla l'estat de la connexió, així cada cop que un usuari demana un servei al Gestor necessita autenticar-se prèviament.

La idea és que en el moment que el Gestor autentica al client porta a terme la creació d'una sessió a la taula 'sessio_autenticada', llavors quan el client li demana un servei mira a la taula a veure si existeix una sessió d'aquestes característiques i en cas afirmatiu realitza el servei demanat, esborrant finalment la sessió de la taula. Abans d'esborrar la sessió la copia a la taula 'historic_sessions' on queda enregistrat a més a més el moment de tancament de la sessió.

Les taules següent mostren aquest concepte:

Taula Sessio_autenticada		
Nom camp	**Tipus de dades**	**Descripció**
sn_gestor	Varchar(64)	Número de seqüència generat aleatòriament pel gestor un cop autenticat el client.
sn_client	Varchar(64)	Número de seqüència generat aleatòriament pel client durant la sol·licitud d'autenticació.
id_client	Varchar(10)	Id. del client que sol·licita autenticar-se.
logon	TimeStamp	Data i hora de la creació de la sessió.

Taula Historic_sessions		
Nom camp	**Tipus de dades**	**Descripció**
sequencia	Integer	Numero auto incremental
sn_gestor	Varchar(64)	Número de seqüència generat aleatòriament pel gestor un cop autenticat el client.
sn_client	Varchar(64)	Número de seqüència generat aleatòriament pel client durant la sol·licitud d'autenticació.
id_client	Varchar(10)	Id. del client que sol·licita autenticar-se.
logon	TimeStamp	Data i hora de la creació de la sessió.
logout	TimeStamp	Data i hora de la finalització de la sessió.

En aquest cas és el Gestor de l'aplicació qui escriu i esborra la informació d'aquestes taules.

8.4 La classe DBManager

Com ja s'ha dit anteriorment l'únic actor que pot interactuar amb el SGBD és el Gestor de l'aplicació. Aquest disposa de la classe DBManager, la qual implementa tot un conjunt de mètodes estàtics per a poder connectar amb el SGBD, recuperar, esborrar, inserir i actualitzar informació.

Mètodes de DBManager:

Privats:

- carregarDriver(): carregar el driver JDBC.
- connectarSGBD(...): estableix una connexió amb la base de dades del projecte.

Públics:

- init(...): Crida els mètodes privats.

Per a afegir, actualitzar o esborrar informació:

- actualitzarExpedientMedic(...): actualitza el número de visites que conté un expedient, així com la signatura digital del Gestor respecte aquest número.

- afegirCertificat(...): insereix un nou certificat a la taula 'certificat_usuari'.
- afegirExpedientMedic(...): insereix un nou expedient a la taula 'expedient_medic'.
- afegirInfoGestor(...): insereix les dades pròpies del Gestor a la taula 'cripto_gestor', la qual conté un sol registre.
- afegirMetge(...): insereix un nou metge a la taula 'metge'.
- afegirPacient(...): insereix un nou pacient a la taula 'pacient'.
- afegirVisita(...): insereix una nova visita a un expedient existent de la taula 'visita'.
- crearSessioClient(...): insereix un registre a la taula 'sessio_autenticada' quan el Gestor ha autenticat al client que demana l'autenticació.
- esborrarSessio(...): esborra la sessió existent de la taula 'sessio_autenticada' un cop ha finalitzat el servei demanat pel client. Abans d'eliminar copia el registre a la taula 'historic_sessions'.

Per a obtenir informació:

- getCertificatGestor(...): recupera el certificat del Gestor de la taula 'cripto_gestor' a partir d'un identificador de registre (encara que només hi ha un registre en futures ampliacions en podrien haver mes)
- getCertificatUsuari(...): a partir del identificador d'usuari es retorna el certificat d'aquest emmagatzemat a la taula 'certificat_usuari'.
- getLlistaPacients(...): es retorna una matriu de String amb tots els pacients de la taula 'pacient' que visiten al metge del que s'informa.
- getP12Gestor(...): es recupera el p12 del Gestor des de la taula 'cripto_gestor' a partir d'un número de registre.
- getPartSignadaExpedient(...): es recupera la informació signada pel gestor del expedient mèdic que s'informa.
- getPartSignadaVisita(...): es recupera la informació signada pel gestor de la visita del expedient mèdic que s'informa.
- getPartXifradaVisita(...): es recupera la informació xifrada pel gestor de la visita del expedient mèdic que s'informa.
- getValueFromTaula(...): mètode genèric per a recuperar el valor d'un camp de qualsevol taula donada un certa condició en un camp qualsevol.
- getValueFromTaulaID(...): mètode genèric per a recuperar el valor d'un camp de qualsevol taula donada un certa condició en el camp 'ID'.
- getValueFromVisita(...): mètode per a recuperar el valor de qualsevol camp de text de la taula 'visita' segons una determinada condició.

Capítol 9. Interfícies d’usuari

9.1 Interficies d’usuari

9.2 Interfície del programa client

9.2.1 Els fitxers de configuració del client

9.2.1.1 El fitxer remote_host.config

9.2.1.2 El fitxer profile.config

9.2.2 El fitxer de log del client

9.2.3 El programa del pacient

9.2.4 El programa del metge

9.3 Interfície del programa servidor

9.3.1 El fitxer database.config

9.3.2 El fitxer de log del servidor

9.3.3 El programa del servidor

9.4 Diagrama de classes incloent la IU

9.1 Interficies d'usuari

La interficie d'usuari (User Interface - IU) és la part de l'aplicació que permet a l'usuari interactuar amb el programa. En el cas de les aplicacions de gestió aquesta és una part fonamental, atès que una inadequada o complexa usabilitat impedirà treure tot el rendiment de l'aplicació o inclús frustrar la seva utilització.

Les interfícies poden presentar-se de moltes de maneres: línia de comandaments, formularis més o menys sofisticats, navegadors, etc. Quan son d'aquests últims tipus s'anomenen Interficies Gràfiques d'Usuari (Graphic User Interfaces - GUI).

Per aquest PFC s'ha optat per l'ús de llibreria gràfica basada en Java 'Standard Widget Toolkit' (SWT) desenvolupada per IBM amb l'objectiu d'accedir a elements GUI natius de cada sistema operatiu. Aquesta llibreria és l'evolució de 'Abstract Window Toolkit' (AWT) desenvolupada per Sun, i és caracteritza per la seva independència de la plataforma d'execució, la riquesa dels components que la formen (taules, arbres, imatges, etc) i pel seu bon rendiment.

A tall d'exemple de l'ús de SWT, a la figura 9-1 es pot observar la pantalla d'inici que apareix quan s'executa qualsevol dels programes clients que conformen l'aplicació (metge o pacient).

Figura 9-1. Pantalla d'inici dels programes.

9.2 Interfície del programa client

El programa client permet a l'usuari realitzar les operacions i serveis que aquest necessita dur a terme. A través d'una interfície gràfica el programa client recull de l'usuari les dades necessàries per tal de fer la petició al programa gestor (també anomenat servidor) que

normalment resideix en una ubicació remota i que és qui realment implementa les operacions que sol•liciten els clients i qui pot accedir al sistema gestor de bades de dades per tal d'emmagatzemar i recuperar informació de la aplicació.

Realment el programa client es divideix en dos parts: la part del pacient i la part del metge. Més endavant s'explicarà en què consisteix cada programa.

9.2.1 Els fitxers de configuració del client

En el cas del programa client hi ha dos tipus de fitxers de configuració: remote_host.config i profile.config. Tots dos han de ser-hi al directori './conf' i son llegits sempre que s'inicia l'aplicació.

9.2.1.1 El fitxer remote_host.config

Conté la adreça ip de l'ordinador on resideix la part servidora de l'aplicació. Només hi conté una línia amb el literal 'remote_host=' i l'adreça IP.

Un exemple seria el següent:
remote_host=192.168.1.11

9.2.1.2 El fitxer profile.config

En el cas del programa del metge conté les següents línies d'informació:

dni_client=numero de dni de l'usuari metge del programa
servei_medic=Departament que correspon al metge
alies_metge=Nom comú del metge
nom_cert_client=Nom del fitxer a disc del certificat digital de metge
nom_p12=Nom del fitxer pkcs12 a disc del metge
password_p12=Contrasenya del fitxer pkcs12
nom_certificat_gestor=Nom del fitxer a disc del certificat digital del programa servidor

Un exemple en el cas del metge seria el següent:

dni_client=00000001-C
servei_medic=Traumatologia
alies_metge=Dr. Casals
nom_cert_client=00000001-C_DER.crt

nom_p12=00000001-C.p12
password_p12=uoc0506
nom_certificat_gestor=Gestor_DER.crt

Un exemple en el cas del pacient:

dni_client=00000001-B
nom_cert_client=00000001-B_DER.crt
nom_p12=00000001-B.p12
password_p12=uoc0506
nom_certificat_gestor=Gestor_DER.crt

Es pot apreciar com la diferència entre el cas del metge i el pacient rau en que aquest últim no necessita informació sobre el servei mèdic ni cap alies.

Tot i que hi ha dos fitxers de configuració per a cada tipus de programa, la informació de tots dos es presenta en un únic formulari dins de les respectives aplicacions mitjançant el botó 'Configuració'.

A continuació un exemple del formulari 'Configuració':

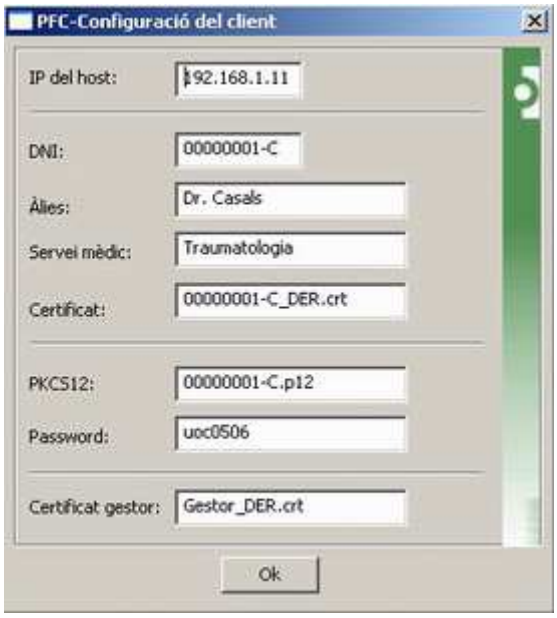


Figura 9-2. Formulari de configuració del programa client de tipus metge.

9.2.2 Els fitxer de log del client

Els programes client generen un fitxer de log en el seu directori './missatges/' on queda enregistrat el detall de la comunicació entre cada client i el servidor.

Aquest fitxer rep el nom de 'log_client.txt' i es pot consultar directament des de fora de l'aplicació tot i que els programes presenten una pestanya anomenada 'log' per a poder consultar aquesta informació d'una manera més còmoda.

9.2.3 El programa del pacient

El pacient utilitza el programa amb l'objectiu de consultar el seu expedient mèdic. Quan el pacient executa el programa pot realitzar les següents quatre accions:

- Consultar l'expedient: És l'operació principal i serveix per a obtenir totes les dades del seu historial mèdic.
- Configuració: Permet canviar l'adreça IP del programa servidor i altres dades sobre el perfil del client molt importants per el funcionament del programa.
- Crèdits: Pantalla amb informació relativa a l'autor i l'objectiu del programa.
- Sortir: Permet finalitzar el programa.

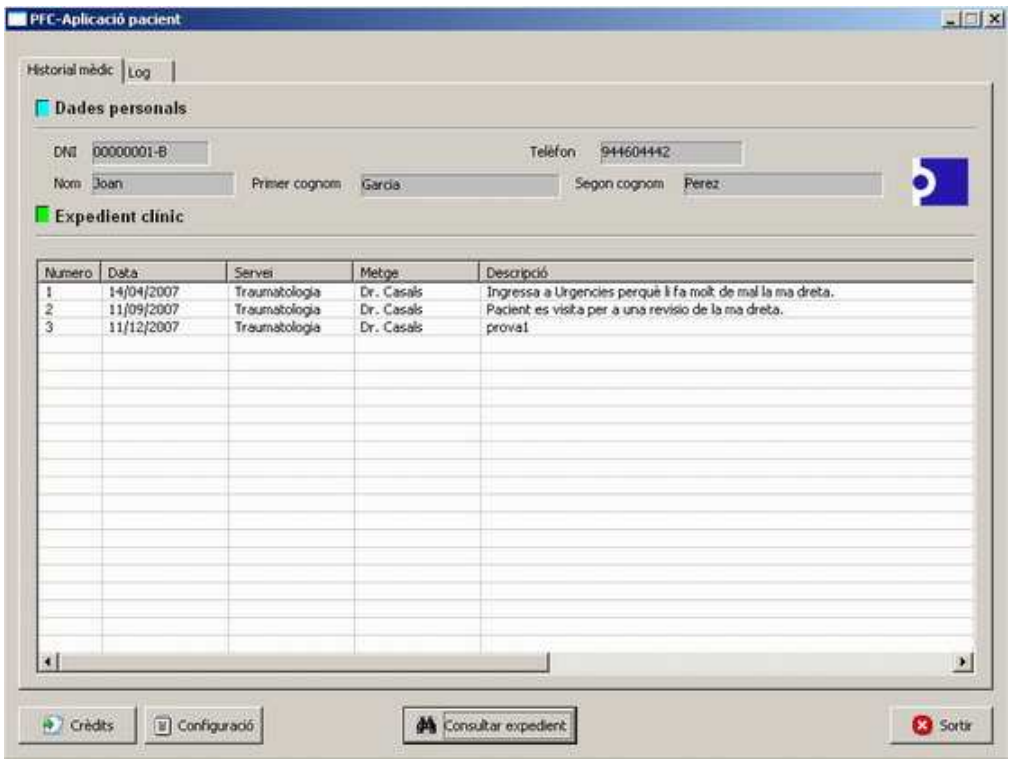


Figura 9-3. Programa del client pacient amb l'expedient recuperat.

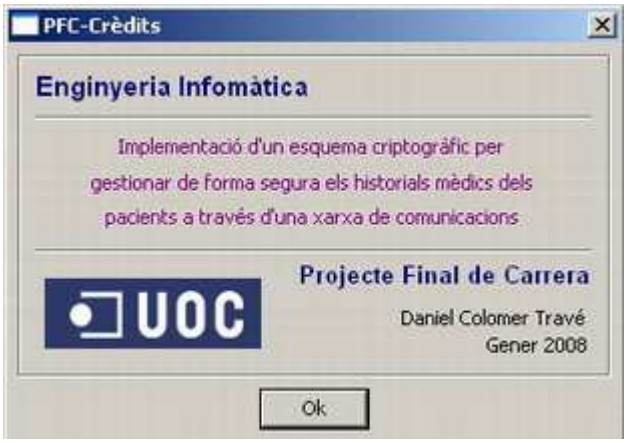


Figura 9-4. Pantalla de crèdits.

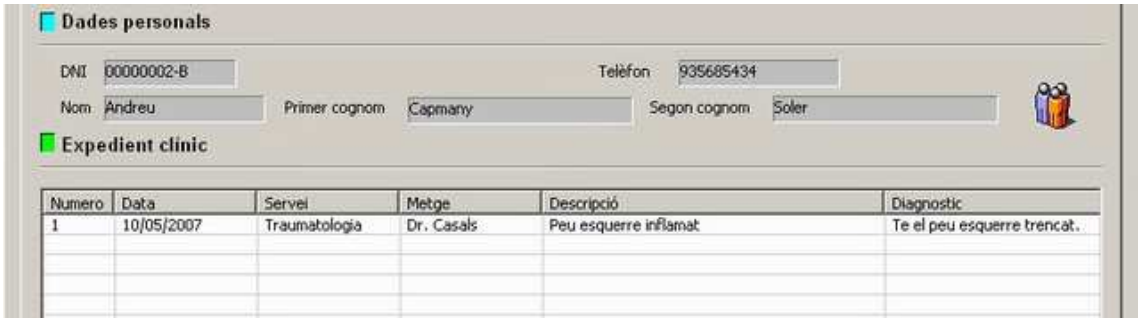
9.2.4 El programa del metge

El metge utilitza el programa per tal de consultar l'expedient mèdic d'algun dels seus pacients i per a afegir noves visites a aquests expedients.

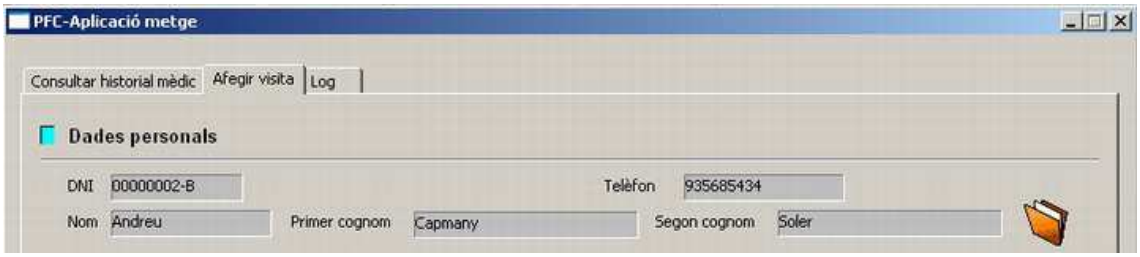
A continuació es mostra un exemple d'un cas típic d'ús d'aquest programa:

El metge ha de visitar a un pacient amb dni 00000002-B. Aleshores entra en el programa i selecciona el client:

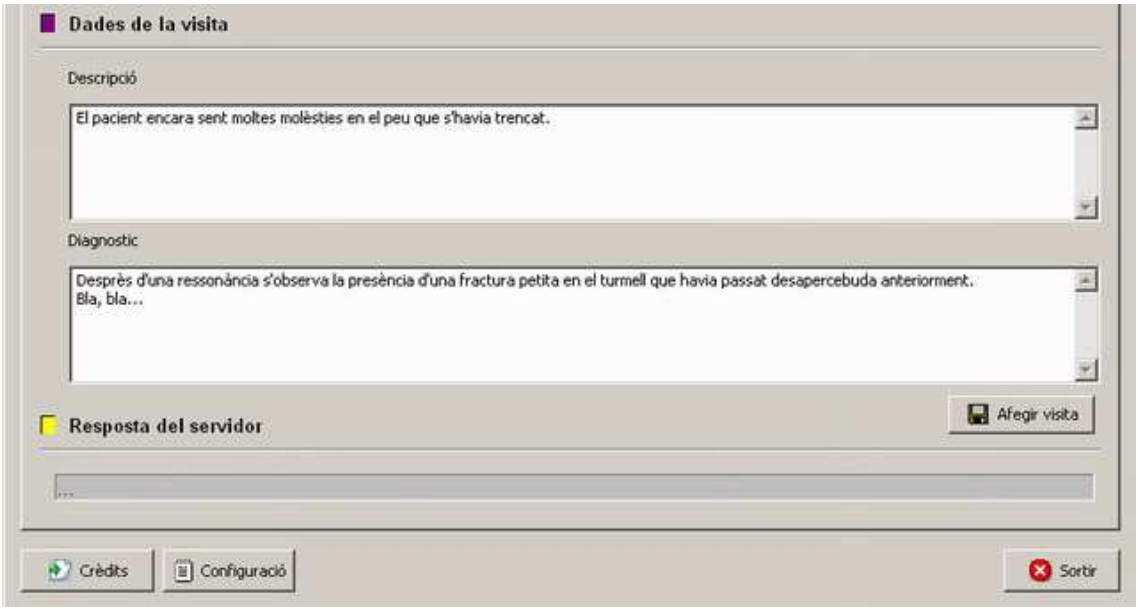
Llavors polsa sobre el botó 'Consultar expedient' i rep del servidor totes les dades de l'expedient mèdic d'aquest pacient:



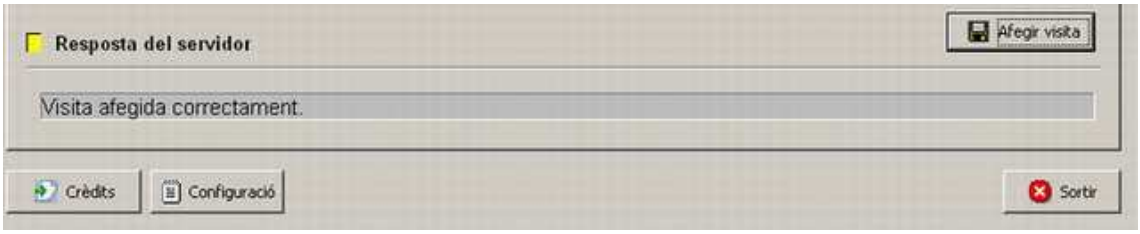
En aquests moments el metge pot llegir la informació del historial que pugui ser rellevant de cara a la visita que tindrà imminentment amb el pacient.
Un cop que la visita ha estat practicada, el metge farà servir el programa per tal d'enregistrar-la dins del historial del pacient. Per a dur a terme aquesta acció polsarà sobre la pestanya 'Afegir visita'. En aquesta nova pantalla es pot apreciar com, d'una banda es recuperen el camps relatius a les dades personals del pacient':



I d'una altra banda com apareixen dos camps tipus 'memo' per tal que el metge pugui explicar detalladament la descripció de la visita i quin és el diagnòstic:



Quan el metge ho consideri oportú polsarà sobre el botó 'Afegir visita' i amb això quedarà emmagatzemada aquesta informació en el servidor. El metge tindrà la seguretat d'aquest fet si apareix un missatge indicant que el procés ha acabat correctament:



Si ara tornem a la pestanya 'Consultar historial mèdic', seleccionem el pacient i polsem sobre el botó 'Consultar expedient' obtindrem les dades refrescades i podrem observar la darrera visita afegida:

Amb això es dona per finalitzada la operativa que seguiria un metge en l'exercici de les seves funcions.

A continuació, a efectes didàctics, passem a analitzar la pestanya 'Log':

Log generat pel client en el decurs de les operacions anteriors:

Quan el metge inicia el programa, es necessari carregar el desplegable que conté la llista de pacients assignats al metge i així aquest podrà seleccionar un pacient de tots els disponibles.

Per a dur a terme aquesta acció s'ha de realitzar primerament el protocol d'autenticació entre client i servidor i posteriorment el protocol 'llista de pacients'.

A la següent figura s'observa com s'ha produït exitosament el protocol d'autenticació:

A la figura següent es veu com un cop efectuat el protocol d'autenticació comença el protocol 'llista de pacients', el qual acaba correctament amb un total de dos pacients enviats pel servidor:

Quan el metge ha seleccionat un pacient i ha polsat sobre el botó 'Consultar Expedient' es produeix novament el protocol d'autenticació i seguidament el protocol 'Consulta expedient'. A la figura següent queda palès com el programa metge rep el historial mèdic correctament.

Un cop el metge ha introduït les dades de la visita aquesta és enviada al servidor per a poder ser emmagatzemada a la base de dades. A la imatge següent s'aprecia aquest aspecte:

Log generat pel gestor (o servidor) en el decurs de les operacions anteriors:

Lògicament el servidor s'ha d'haver iniciat abans que qualsevol client. En el fragment de log següent es pot veure com en el moment d'executar el programa servidor aquest es connecta amb la base de dades i executa els serveis que permetran als clients accedir als mètodes remots via RMI. Si tot ha anat bé el servidor indicarà aquest fet amb el missatge: Servidor preparat!:

```
2007-12-14 18:06:54.046

Connectat a la Base de Dades.
Llançat el servei Autenticar.
Llançat el servei Consultar Expedient.
Llançat el servei Llistar Pacients.
Llançat el servei Afegir Dades.
Servidor preparat!
```

A continuació s'observa com un programa client li demana de realitzar el protocol d'autenticació i aquest es porta a terme correctament:

```
2007-12-14 18:07:03.625
PROTOCOL AUTENTICACIÓ.
Servidor rep->id_usuari_u: 00000001-C.
Servidor rep->id_usuari_sequencia: 2092694770003458221.
Servidor envia->id_usuari_sequencia: 2092694770003458221.
Servidor envia->id_gestor_sequencia: 2329366078810367970.
Servidor envia->id_gestor: ed7c623cd2522d4e8de5badc1776c2fcda1d5cc0.

2007-12-14 18:07:03.812
Servidor rep->id_sequencia_gestor_prima: 2329366078810367970.
Autenticació CORRECTA per part del servidor.
Autenticació bilateral.
```

Un cop s'ha produït el protocol d'autenticació es realitza el protocol de llista de pacients:

```
2007-12-14 18:07:03.921
Servidor rep->id_gestor_sequencia_prima: 2329366078810367970.
Servidor rep->servei_demanat: LLISTAR_PACIENTS.
El client té permis per a obtenir el llistat.
S'ha enviat la llista de pacients al client.
```

A continuació es té el registre del servidor quan el metge sol·licita un historial mèdic (a partir d'ara s'obvia el registre de l'autenticació):

```
2007-12-14 18:17:30.328
Servidor rep->id_sequencia_gestor_prima: 8282337349823852246.
Servidor rep->Servei sol·licitat: CONSULTA.
Servidor rep->_id_usuari_expedient: 00000002-B.
El client té permis per a fer la consulta.
El gestor ha recuperat el historial.
S'ha enviat el historial al client.
```

Per últim es mostra el registre quan el metge sol·licita afegir una nova visita a l'expedient:

2007-12-14 18:48:07.687
Servidor rep->id_gestor_sequencia_prima: 3952624106091237627.
Servidor rep->servei_demanat: AFEGIR_DADES.
Servidor rep->Visita: .
-Numero Expedient: 00000002-B</v>.
-Visita numero: null.
-Data: 14/12/2007</v>.
-Marca de temps: null.
-Servei: Traumatologia</v>.
-Metge: Dr. Casals</v>.
-Descripcio: El pacient encara sent moltes molèsties en el peu que s'havia trencat.</v>.
-Diagnostic: Després d'una ressonància s'observa la presència d'una fractura petita en el turmell que havia passat desapercebuda anteriorment.
Bla,bla...</vs>.
-Signatura metge:
MIIKKgIBATELMAkGBSsOAwIaBQAwCwYJKoZIhvcNAQcBoIII4jCCBE4wggM2oAMC
AQICCQCRBTC9Po+DnzANBgkqhkiG9w0BAQUFADBwMQswCQYDVQQGEwJFUzESMBAG
A1UECBMJQmFyY2Vsb25hMRIwEAYDVQQHEwlCYXJjZWxvbmExDDAKBgNVBAoTA1VP
QzEWMBQGA1UECxMNUEZDIFNlZ3VyZXRhdDETMBEGA1UELhMKMDAwMDAwMDAtQTAe
Fw0wNzEyMDIxNjE5NTJaFw0wODA1MzAxNjE5NTJaMIGfMQswCQYDVQQGEwJFUzES
MBAGA1UECBMJQmFyY2Vsb25hMRIwEAYDVQQHEwlCYXJjZWxvbmExDDAKBgNVBAoT
A1VPQzEPMA0GA1UECxMGTWV0Z2VzMQ8wDQYDVQQDEwZGZXJyYW4xEzARBgNVBC4T
CjAwMDAwMDAxLUMxIzAhBgkqhkiG9w0BCQEWFGZlcnJhbi5jYXNhbHNAdW9jLmVz
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDZk0JWhQseVRICEIOiCup3OYsv
zXsIwOhiHRzcL3GYamYIKrKHfbk8ca+vszoS3T03kmpy2ExPzlANPkwD0KhdiZWo
ypIRDgkexFO+q+oGLDJyG4Cks+FYoxQccgXEBNGmM8Pj0sReaQ+zTU44EERBVeXE
9DsehHO8zUYTnxqlIQIDAQABo4IBPTCCATkwCQYDVR0TBAIwADARBglghkgBhvhC
BBYEFImzxJ+1skI/g+BWPH5i7oqzfcoaMIGiBgNVHSMEgZowgZeAFImzxJ+1skI/
g+BWPH5i7oqzfcoaoXSkcjBwMQswCQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vs
b25hMRIwEAYDVQQHEwlCYXJjZWxvbmExDDAKBgNVBAoTA1VPQzEWMBQGA1UECxMN
UEZDIFNlZ3VyZXRhdDETMBEGA1UELhMKMDAwMDAwMDAtQYIJAM6lSnumBYuwMAkG
A1UdEQQCMAAwCQYDVR0SBAIwADANBgkqhkiG9w0BAQUFAAOCAQEAEJ8+h+vJWqSc
OIfzCwYy7w3RsnXvdaQPKei7FIeM8wVHDfSgX8lrFONfDS17sL8Iw2b9f8Zs1chO
w1/aELAIOUIYiG9h2yRjR20QzTLTOXkFDsVvi28IlsQFH5+03cLEtUH56DnC4+/j
bDuR7HEtZKb6HF79LcmCq+cKDtuNsthcOW+GQJ4fE9L6HeeNzcMkDly3RXUt8Vna
Nhh6gSRPe4YNR4eH2yCtha4Nh6NGNkhxNF7aMBJYQjvRVnCXpDjw+4nziufpHptZ
BzFu/WuaK5moEd1n27wmdXJzkVfZdOhBK3N8COYjmgyEG5mcpknueYLALOP1EO2T
b6Nh4gxWxzGCASMwggEfAgEBMH0wcDELMAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJh
cmNlbG9uYTESMBAGA1UEBxMJQmFyY2Vsb25hMQwwCgYDVQQKEwNVT0MxFjAUBgNV
BAsTDVBGQyBTZWd1cmV0YXQxEzARBgNVBC4TCjAwMDAwMDAwLUECCQCRBTC9Po+D
nzAJBgUrDgMCGgUAMA0GCSqGSIb3DQEBAQUABIGAOCrZIoADMzGFFcnNheYNp38P
+0TCC+Mj2xYtQPKY7hCVVsXrnwXcM7zCnm1E6y0fjy2FC68qWcKsiorHPImGNrLX
5C4ZbHx6GRp7/qw0z3XX3oPVUmoIRq+n53T9XAg900GEO6+LGe+N8qsCH6Fq9x/c
RkgC0Mw+tclCdFTKvFM=.
El client és metge i té el pacient assignat.
La verificació de la signatura del metge és correcta.
La verificació de la signatura de la ultima visita és correcta.
El actual numero de sèrie de la visita es: 1.
El nou numero de sèrie de la visita es: 2.
La marca de temps és 2007-12-14 18:48:07.703.
El servidor ha signat la visita, la marca de temps i el num sèrie.
El Servidor ha xifrat amb la seva clau pública la visita.
El Servidor ha guardat la informació a la base de dades.
Servidor envia: Visita afegida correctament.

9.3 Interfície del programa servidor

En aquest PFC es demanava implementar alguna tecnologia de servidor capaç d'oferir serveis remots als clients. Com ja s'ha comentat en capítols previs l'opció escollida ha estat RMI.

La funció principal del programa servidor és inicialitzar el servidor RMI i publicar el serveis remots en el RMI Registry. Amb això els clients ja poden emprar els mètodes remots.

Per a dur a terme aquesta tasca hagués estat suficient amb un script disponible al servidor encarregat d'inicialitzar el RMI Registry, però de seguida es va veure de molta utilitat la creació d'una interfície gràfica que dones algunes opcions pràctiques a l'administrador del servidor: inicialització i parada del programa servidor, visualització del log en temps real, consulta del historial de connexions, creació d'usuaris, etc. Més endavant es comenten aquestes característiques.

9.3.1 El fitxer database.config

El programa gestor fan servir un arxiu de configuració anomenat database.config el qual és llegit des de la ubicació './conf' sempre que s'inicia l'aplicació de la banda del servidor.

Aquest fitxer es necessita per emmagatzemar informació relativa a la base de dades MySQL:

HOST_BD=Adreça IP de l'ordinador on resideix el sistema gestor de base dades MySQL
NOM_BD=Nom de la base de dades
LOGIN_BD=Nom d'usuari amb permis per a operar a la BBDD
PASSWORD_BD=Contrasenya de l'usuari anterior

Un exemple seria el següent:
HOST_BD=192.168.1.11
NOM_BD=pfc
LOGIN_BD=daniel
PASSWORD_BD=daniel

La resta d'informació de configuració que necessita el servidor es guarda directament a la base de dades, però òbviament les dades presentades anteriorment han de ser fora de la base de dades per la seva naturalesa.

9.3.2 Els fitxer de log del servidor

El servidor genera un fitxer de log en el seu directori './missatges/' on queda enregistrat el detall de les accions del servidor, tant de la comunicació amb els clients com de la seva pròpia activitat.

Aquest fitxer rep el nom de 'log_servidor.txt'. El fitxer es pot consultar directament des de fora de l'aplicació tot i que el programa presenta una pestanya anomenada 'log' per a poder consultar aquesta informació d'una manera més còmoda.

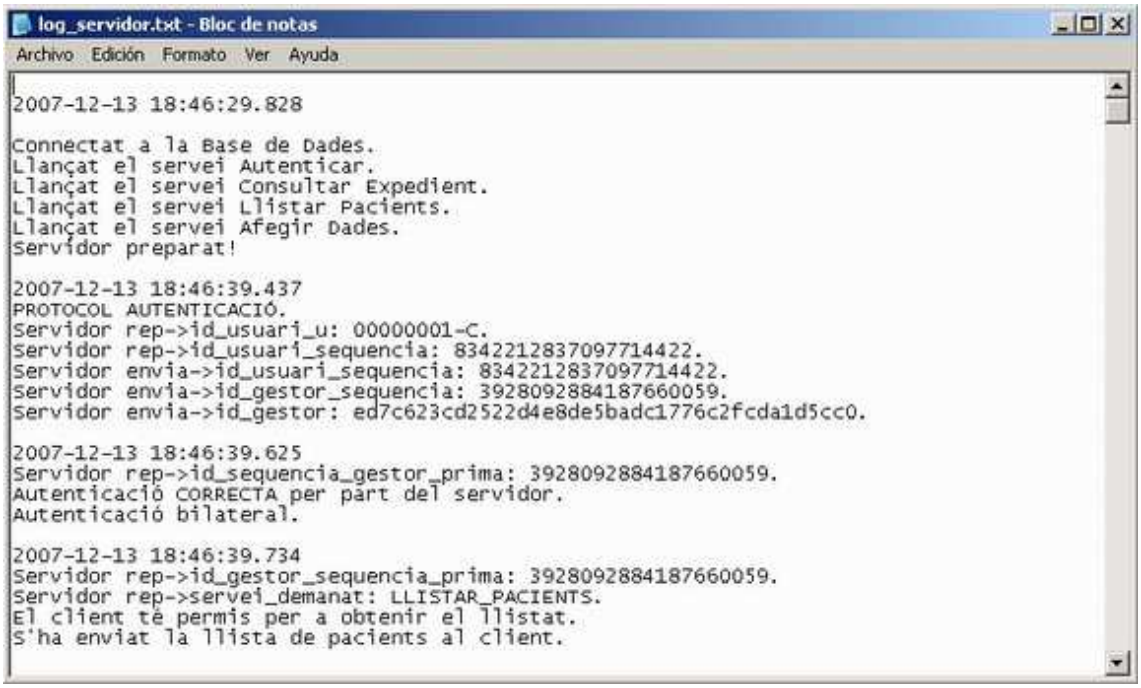


Figura 9-5. Fragment del fitxer de log del servidor consultat des del sistema operatiu.

9.3.3 El programa del servidor

Quan s'inicia el programa aquest ho fa en l'estat 'No iniciat', això és no hi ha connexió amb la base de dades ni s'està executant el RMI Registry, per tant els clients no es poden executar. De fet la única operació disponible és la consulta de l'últim log que el servidor va generar.

A la figura 9-6, on es mostra aquesta situació, es pot apreciar com el servidor no està iniciat i a la pestanya de log apareix informació que cal destacar:

A l'execució anterior el programa va detectar que no disposava de la informació d'inicialització suficient a la base de dades. Per aquest motiu va llençar la seva pròpia bateria de dades d'inicialització, la qual s'explica al capítol de proves d'aquesta memòria.

D'una altra banda el log informa de quina va ser la data i l'hora en que es va inicialitzar, els serveis que va posar a disposició dels clients i a quina hora va finalitzar la seva execució.

Finalment, es pot observa com al costat de la pestanya 'Log' hi apareixen dues més: 'Afegir clients' i 'Historic connexions'.

'Afegir clients' permet donar d'alta usuaris (metges i pacients) a la base de dades. En el cas de tractar-se de pacients per defecte es genera un historial mèdic buit (sense visites) per aquest pacient.

'Historic connexions' recupera de la base de dades el historial de les connexions que els clients han establert amb el servidor. A més a més permet esborrar permanentment aquesta informació de la base de dades, així com generar un fitxer pla en disc amb aquesta informació.

Figura 9-6. Servidor arrencat però no iniciat.

Amb el servidor sense inicialitzar no es possible fer res mes. Si polsem sobre la pestanya 'Afegir clients' (figura 9-7) veurem com està tot desactivat.

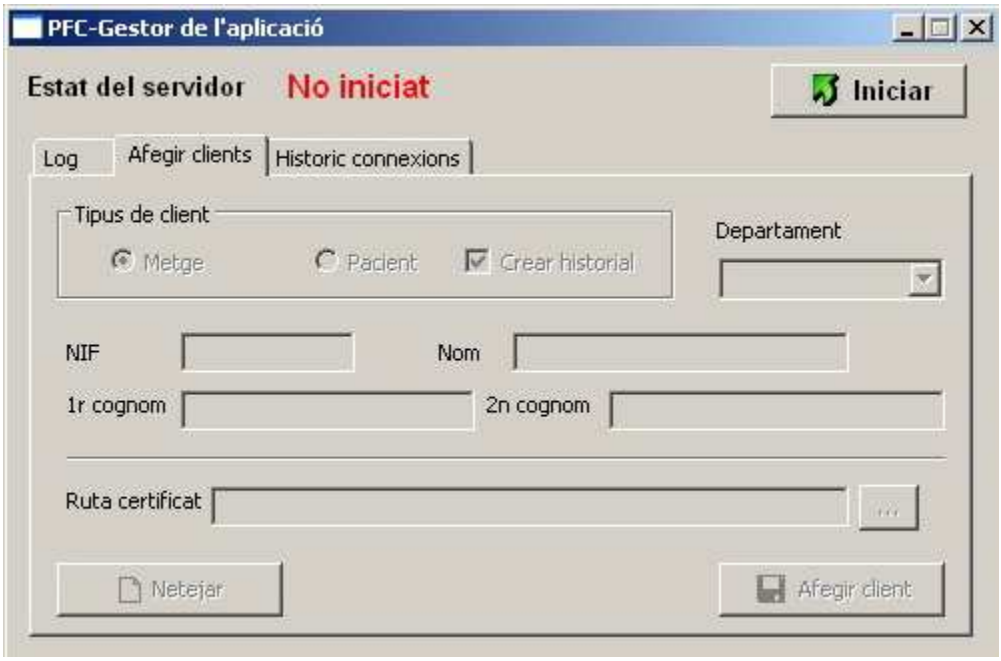


Figura 9-7. No es pot afegir cap usuari amb el Servidor detingut.

El mateix passa amb el 'Historic de connexions' (figura 9-8)

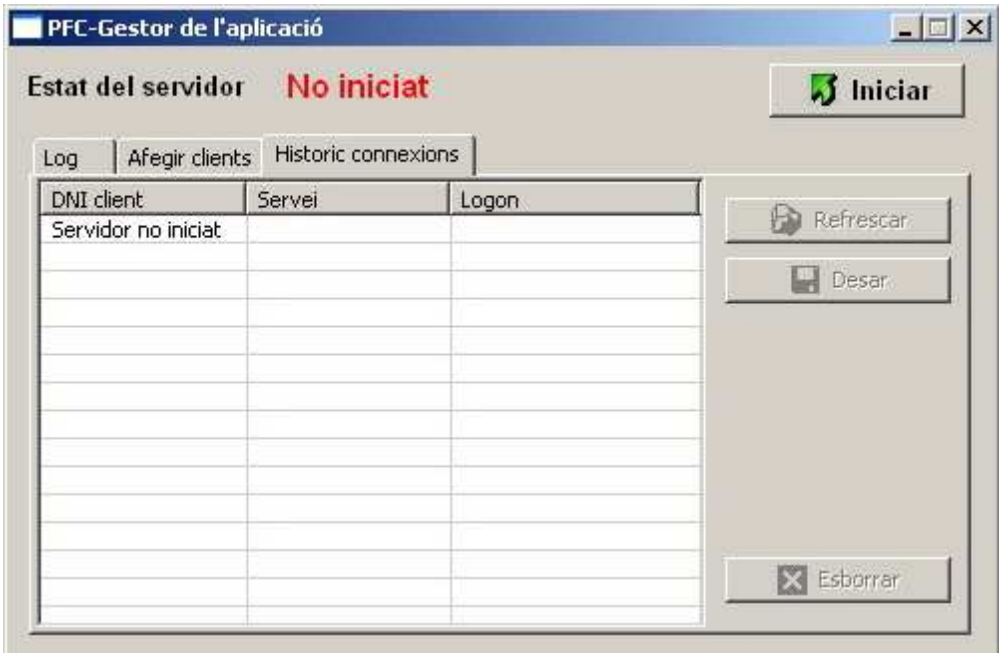


Figura 9-8. No es pot consultar el historial de connexions amb el Servidor detingut.

A la figura 9-9 es pot veure què succeeix un cop que polsem sobre el botó 'Iniciar'.
És destacable és el missatge 'Iniciat' en color verd així com el canvi en la icona i en el text del botó de la dreta, el qual ara serveix pera apagar el servidor i en canvi abans servia per iniciar-lo. També s'observa com la pestanya 'Log' ha esborrar la informació anterior i s'ha actualitzat a la situació present.

Figura 9-9. Servidor iniciat.

A la imatge 9-10 es mostra com donar d'alta un usuari de tipus metge.

Figura 9-10. Creació d'un metge.

En el moment d'indicar la ruta i el certificat digital de l'usuari disposem del botó amb el text '...' el qual obre un quadre de diàleg estàndard que permet cerca i seleccionar còmodament el fitxer amb extensió crt. Aquest fet s'il·lustra a la figura 9-11:

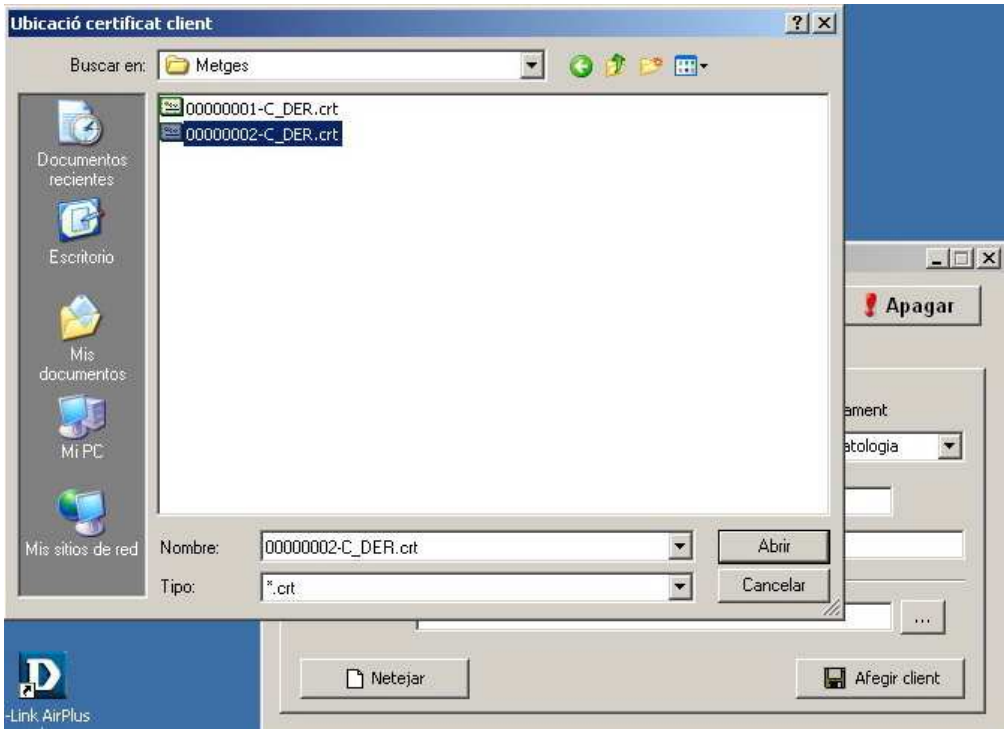


Figura 9-11. Selecció del certificat digital d'un nou usuari metge.

A la imatge 9-12 s'observa com un cop seleccionat el certificat, polsem sobre el botó 'Afegir client' per a donar d'alta el nou usuari metge:

Figura 9-12. Final de procés de creació d'un nou usuari metge.

Mentre es donen d'alta usuaris el servidor RMI s'està executant i donant el servei als clients. Si anem a la pestanya 'Historic connexions' i polsem sobre el botó 'Refrescar' podrem veure qui s'ha connectat i que ha fet.

A la figura 9-13 es mostra aquest fet, on queda pales que s'ha connectat un client de tipus metge (atès que els de tipus pacient no poden obtenir una llista de pacients) ha recuperat del servidor la llista dels pacients que té assignats i ha demanat l'expedient mèdic d'un d'ells.

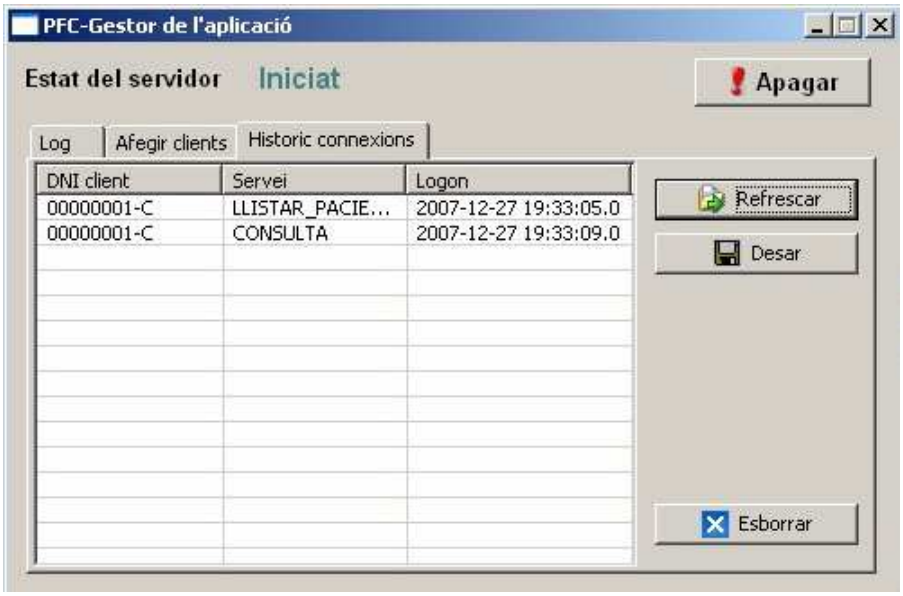


Figura 9-13. Històric de connexions.

Si volem més detall, a la pestanya 'Log' podem veure la direcció IP del programa client, quin expedient ha demanat i quin ha estat el resultat final. Això queda il·lustrat a la figura 9-14:

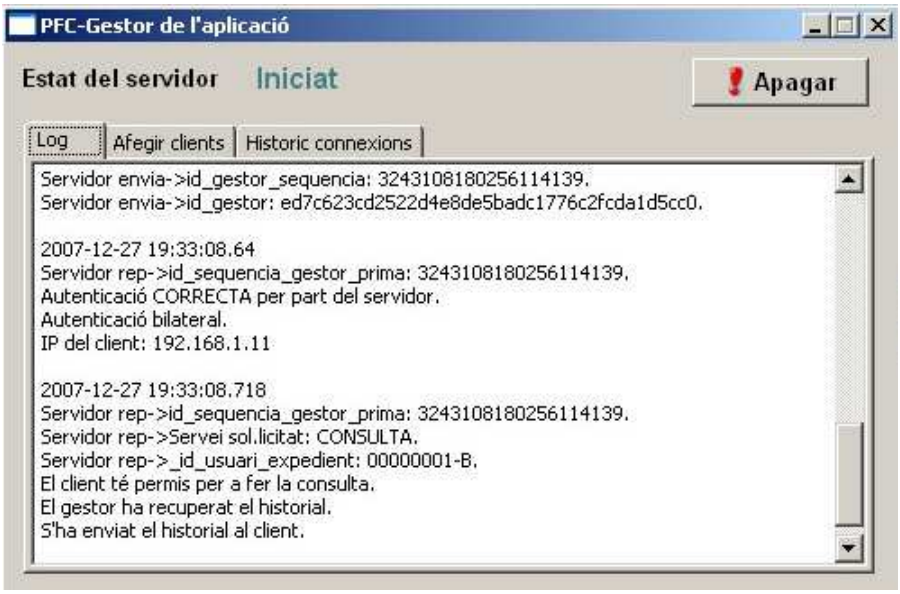


Figura 9-14. Consulta del log del servidor amb el programa iniciat.

Per últim, a la figura 9-15 es mostra el cas de parada del servidor. Amb el botó 'Apagar' se'ns demana confirmació i un cop acceptada s'atura el servidor RMI, es desconnecta de la base de dades i finalitza el programa:

Figura 9-15. Finalització del servidor.

9.4 Classes incloent les interficies d'usuari

Respecte el programes client: S'explica el cas del programa del Metge, atès que el del pacient és força semblant:

Les classes de test (i el corresponent package) s'han transformat en una classe anomenada RunMetge dins del package runable, el constructor de la qual rep els paràmetres necessaris des de la classe GUI_Metge: remote_host, dni_client, nom_cert_client, nom_p12, password_p12, nom_certificat_gestor.

D'una altra banda la classe RunMetge implementa els mètodes runProtocolAutenticacio, runProtocolConsulta, runProtocolLlistarPacients i runProtocolAfegirDades, els quals son cridats des de la classe GUI_Metge.
Per tant es veu com GUI_Metge és una interfície d'usuari encarregada de recollir les dades de l'usuari i passar-les com a paràmetres als diferents mètodes de la classe RunMetge.

Respecte el programa servidor: La classe GUI_Gestor instancia un objecte de la classe ServidorRMI, que és qui estableix la connexió amb la base de dades i qui inicia el servidor RMI.

Capítol 10. Joc de proves

10.1 Introducció

10.2 Algorismes, programes de prova i altres conceptes

10.2.1 Expedient mèdic

10.2.1.1 Cicle de vida d'un Expedient

10.2.1.2 Exemple de l'expedient mèdic d'un pacient

10.2.1.3 Signatura d'un expedient

10.2.1.4 Xifratge d'un expedient

10.2.1.5 Lliurament de l'expedient sol•licitat al metge

10.2.2 Fitxers de log

10.3 Bateria de dades per a fer operativa l'aplicació

10.1 Introducció

Al llarg de cadascuna de les etapes del projecte ha estat fonamental provar acuradament el seu funcionament, atès que les noves etapes es recolzen en la correctesa de les fases anteriors. El present capítol recull els algorismes i entitats més importants que han servit per a comprovar l'eficiència i l'efectivitat de cada desenvolupament parcial i que han ajudat a conceptualitzar les parts més complexes de l'aplicació.

10.2 Algorismes, programes de prova i altres conceptes

10.2.1 Expedient mèdic

En aquesta secció es destaquen les entitats i conceptes que redunden en els historials clínics, punt fonamental d'aquest treball. Es treballen les idees de:

- Cicle de vida d'un historial
- Signatura d'un historial
- Xifratge d'un historial
- Lliurament d'un historial solicitat per un metge

10.2.1.1 Cicle de vida d'un Expedient

Necessitem d'una banda (quan encara no es disposava de base de dades) d'una classe *Expedient* que ens permeti crear un expedient:

```
Expedient exp=new Expedient("00000001-B", "Joan", "Garcia", "Pérez", "944604442");
```

Amb aquesta classe ja tenim una representació a memòria d'un expedient mèdic. D'una altra banda d'una classe *Historia* per tal de poder crear les diferents visites que conformen l'expedient d'un pacient (més endavant aquesta classe es va batejar com 'Visita'):

```
Historia h1=new Historia("14/04/2007","Traumatologia","Dr. Ferran",
"Ingressa a Urgencies perquè li fa molt de mal la ma dreta.","Té la ma trencada.","");
Historia h2=new Historia("11/09/2007","Toxicologia","Dra. Sanchez",     "Pacient es visita
perquè fa una setmana que te mal de panxa.",   "té gastroenteritis.","");

exp.afegirHistoria(h1);
exp.afegirHistoria(h2);
```

Ara afegint dues histories a l'expedient es té una representació a memòria d'un expedient mèdic complet.

Amb aquestes dades ja es pot anar més enllà i crear un document XML que representi un Expedient mèdic digital. La classe encarregada d'això es *XMLExpedient*, la qual disposa de mètodes per a crear i llegir documents XML de tipus *Expedient*.

Així doncs, per a crear un historial fem servir una classe de prova anomenada *CrearHistorialTest*, la qual a partir d'un expedient (expedient+histories) invoca a la classe encarregada de la creació del document:

```
XMLExpedient xmlExp=new XMLExpedient(NOM_FITXER);

// Creem el document XML a partir d'un objecte de la classe Expedient
xmlExp.crear_expedient(exp);

System.out.println("Historial creat.");
```

Representació gràfica de les explicacions anteriors:

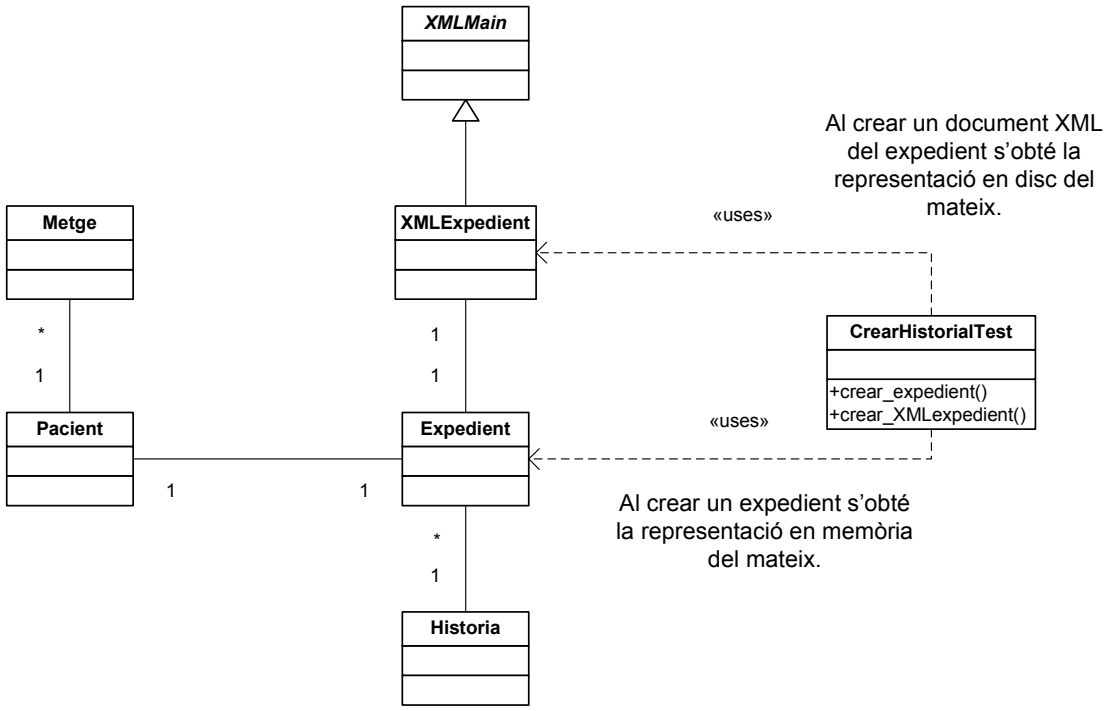


Figura 10-1. Cicle de vida d'un historial mèdic.

El resultat obtingut és un document XML el nom del qual és el dni del pacient (per exemple 00000001-B.xml) i on les dades de cada camp apareixen codificades amb base 64. D'aquesta manera s'evita el problema de que el document XML no sigui vàlid pel fet de trobar caràcters no permesos amb la codificació utilitzada (ISO-8859-1, utf-16 -unicode-, utf-8, etc.).

Però ha de quedar clar que el document no està ni signat ni xifrat, solament codificat amb base64

10.2.1.2 Exemple de l'expedient mèdic d'un pacient

```
<?xml version="1.0" encoding="UTF-8"?>
<GestioSeguraHistorialsClinics>
  <Expedient Num_exp="MQ==">
    <Dades_personals>
      <DNI>MDAwMDAwMDEtQg==</DNI>
      <Nom>Sm9hbg==</Nom>
      <Cognom1>R2FyY2lh</Cognom1>
      <Cognom2>UOlyZXo=</Cognom2>
      <Telefon>OTQ0NjA0NDQy</Telefon>
    </Dades_personals>
    <Historia>
      <Visita Num_visita="MQ==">
        <Data_hist>MTQvMDQvMjAwNw==</Data_hist>
        <Servei_hist>VHJhdW1hdG9sb2dpYQ==</Servei_hist>
        <Metge_hist>RHIuIEZlcnJhbg==</Metge_hist>

<Descripcio_hist>SW5ncmVzYSBhIFVyZ2VuY2llcyBwZXJxdeggbGkgZmEgbW9sdCBkZ
SBtYWwgbGEg&#xD;
bWEgZHJldGEu</Descripcio_hist>
        <Diagnostic_hist>VOkgbGEgbWEgdHJlbmNhZGEu</Diagnostic_hist>
        <Signatura_metge_hist />
      </Visita>
      <Visita Num_visita="Mg==">
        <Data_hist>MTEvMDkvMjAwNw==</Data_hist>
        <Servei_hist>VG94aWNvbG9naWE=</Servei_hist>
        <Metge_hist>RHJhLiBTYW5jaGV6</Metge_hist>

<Descripcio_hist>UGFjaWVudCBlcyB2aXNpdGEgcGVycXXoIGZhIHVuYSBzZXRtYW5hI
HF1ZSB0ZSBt&#xD;
YWwgZGUgcGFueGEu</Descripcio_hist>
        <Diagnostic_hist>dOkgZ2FzdHJvZW50ZXJpdGlzLg==</Diagnostic_hist>
        <Signatura_metge_hist />
      </Visita>
    </Historia>
  </Expedient>
</GestioSeguraHistorialsClinics>
```

Figura 10-2. Exemple d'historial mèdic.

Els expedients els creen i els mantenen els metges. Quan es crea un expedient el següent és signar-lo amb la clau privada del metge i finalment el Gestor (un automatisme) el xifra amb la seva clau pública abans d'emmagatzemar-lo a la base de dades a la qual ell només hi té accés.

10.2.1.3 Signatura d'un expedient

Els metges signen les dades que han introduït a l'expedient. Aquesta informació està compresa entre la sèrie de tags <Visita Num_visita></Visita> dins del tag <Historia></Historia>
Per a signar un expedient fem servir la classe de prova *SignarHistorialTest*, la qual a partir de la clau privada d'un metge obre el document XML existent (l'expedient), signa les dades corresponents a la visita del pacient que ell ha introduït i posteriorment estableix la signatura de les dades entre els tags <Signatura_metge_hist ></Signatura_metge_hist >.

```
XMLMain xmlMain=new XMLExpedient();// Llegir el XML

XMLExpedient xmlE=(XMLExpedient)UtilsXML.llegirXML_en_clar
        (xmlMain, nom_fitxer, sh.getP12Metge());

String dades_a_signar=
        xmlE.get_atribut_from_historia("Num_visita", "2", "Diagnostic_hist");

// Signem les dades i obtenim un byte[]
byte[] dades_signades=sh.getSignador().signData(dades_a_signar);

xmlE.set_atribut_from_historia("Num_visita", "2",
        "Signatura_metge_hist", new String(dades_signades));

xmlE.generar_document();
System.out.println("Historial signat pel metge.");
```

10.2.1.4 Xifratge d'un expedient

Una de les funcions del Gestor dels expedients és la d'emmagatzemar-los segurament. Per aquest motiu ha de xifrar-los amb la seva clau pública.
Amb la classe de test XifrarHistorialTest es llegeix un expedient (que hauria d'estar signat) i es genera un fitxer (binari) amb extensió .enc amb el mateix nom que el document original. Aquest fitxer és el que finalment s'emmagatzema a la base de dades. Óbviament, únicament el Gestor amb la seva clau privada podrà recuperar el expedient en format XML.

```
XifrarHistorialTest xh=new XifrarHistorialTest();

// Xifrar el text del historial
byte[] historial_xifrat=
Utils.xifrar(f,xh.getPublicKeyGestor(),xh.getCertificatGestor());

// Generar un fitxer amb la resposta del gestor
Utils.generarFitxerXifrat(historial_xifrat, "./historials/00000001-B.enc");
System.out.println("Fitxer xifrat.");
```

10.2.1.5 Lliurament de l'expedient sol·licitat al metge

Què fa el Gestor

El Gestor rep en format binari un arxiu .enc (consulta_expedient.enc) que li ha enviat el Metge. Llavors amb la seva clau privada el desxifra i obté el document XML de intercanvi (consulta_expedient.xml).

El Gestor llegeix el document XML de intercanvi i ja sap quin és el expedient que li està demanant. Aleshores, després de les comprovacions d'accés pertinents, agafa del repositori de historials l'expedient del pacient sol·licitat, el qual està xifrat amb la clau pública del Gestor i emmagatzemat en format binari en un arxiu .enc.

Seguidament desxifra l'expedient amb la seva clau privada i obté el expedient en clar. Llavors el xifra amb la clau pública del Metge, per tal de que només aquest el pugui desxifrar.

Després agafa el contingut i l'emmagatzema en un arxiu .enc, el torna a obrir i obté un byte array amb el qual crea un String que ubicarà en el tag <Historial> del document XML de intercanvi amb el Metge.

Posteriorment crea un nou document XML de intercanvi que inclou en el tag <Historial> el text xifrat que representa l'expedient del pacient demanat. Finalment xifra aquest document XML amb la clau pública del Gestor obtenint un arxiu en format binari amb extensió .enc que és lliurat al Metge.

Què fa el metge

El Metge rep en format binari un arxiu .enc (consulta_expedient.enc) que li ha enviat el Gestor. Llavors amb la seva clau privada el desxifra i obté el document XML de intercanvi (consulta_expedient.xml).

Després extreu del tag <Historial> del document XML de intercanvi el text que representa l'expedient del pacient que havia solicitat. Amb aquest text crea un arxiu amb extensió .enc (dni_pacient.enc).

Després llegeix el fitxer .enc i obté un byte[] que serveix d'entrada a la funció de desxifratge, la qual després d'aplicar la clau privada del Metge retorna un String que representa l'expedient del pacient demanat.

Finalment amb el String anterior crea directament el document XML (dni_pacient.xml) que era l'objecte de la consulta del Metge.

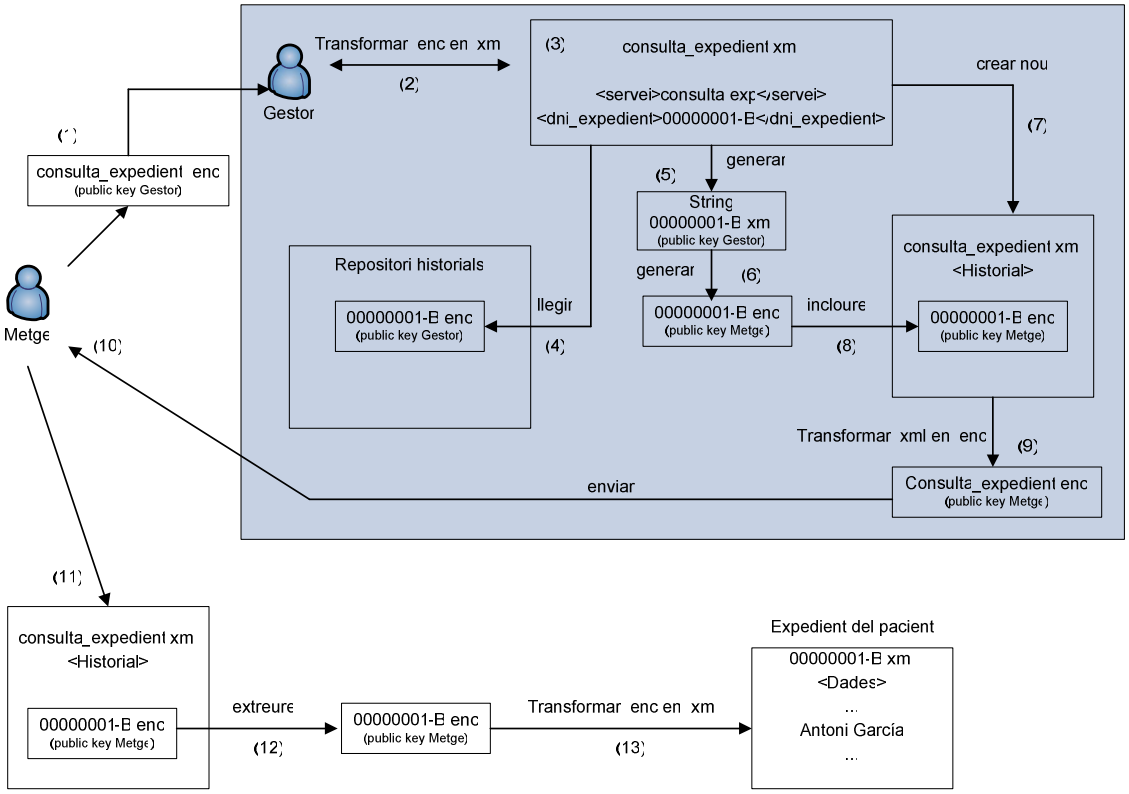


Figura 10-3. Exemple de lliurament d'un historial.

10.2.2 Fitxers de log

En comptes de mostrar missatges per pantalla tant els clients com el servidor van anotant en els seus respectius arxius de log (log_client.txt i log_servidor.txt) els resultats de les operacions realitzades durant l'execució del programa.

Aquests fitxers es creen en el directori 'missatges' i s'esborra el contingut en cada nova execució del programa.

A la figura 10-4 es pot observar les anotacions de log en el programa client durant el protocol d'autenticació i a la figura 10-5 per al programa servidor.

```
2007-11-10 19:22:41.0
Client envia->id_usuari_u: 00000002-C.
Client envia->id_sequencia: 2580880918658238164.

2007-11-10 19:22:41.25
Client rep->id_sequencia_prima: 2580880918658238164.
Client rep->id_gestor_sequencia: 172585822804194654.
Client rep->id_gestor: 2D:21:CD:96:A1:EA:F7:65:8E:6C:0B:EE:B8:02:0F:44:3B:3B:F9:A4.
Autenticació CORRECTA per part del client.
Client envia->id_sequencia_gestor: 172585822804194654.

2007-11-10 19:22:41.312
Protocol Autenticacio realitzat correctament.
```

Figura 10-4. Fragment del fitxer client_log.txt

```
2007-11-10 19:29:23.734
Llançat el servei Autenticar.
Llançat el servei Consultar Expedient.
Llançat el servei Llistar Pacients.
Llançat el servei Afegir Dades.
Servidor preparat!

2007-11-10 19:30:51.234
Servidor rep->id_usuari_u: 00000002-C.
Servidor rep->id_usuari_sequencia: 2815379456437168424.
Servidor envia->id_usuari_sequencia: 2815379456437168424.
Servidor envia->id_gestor_sequencia: 1236364670203286570.
Servidor envia->id_gestor: 2D:21:CD:96:A1:EA:F7:65:8E:6C:0B:EE:B8:02:0F:44:3B:3B:F9:A4.

2007-11-10 19:30:51.421
Servidor rep->id_sequencia_gestor_prima: 1236364670203286570.
Autenticació CORRECTA per part del servidor.
Autenticació bilateral.
```

Figura 10-5. Fragment del fitxer servidor_log.txt

10.3 Bateria de dades per a fer operativa l'aplicació

L'aplicació necessita d'una certa informació a la base de dades per tal de no donar errors al intentar executar tant el programa servidor com els clients. Aquesta informació pot ser considera de dos tipus: informació essencial per tal de permetre l'execució dels programes i informació de prova per a poder validar les operacions d'autenticació, de consulta dels historials mèdics, recuperació de la llista de pacients d'un metge i la inserció de noves visites als historials mèdics.

Per aquest motiu sempre que s'executa el servidor aquest comprova l'existència d'aquesta informació i si detecta que no hi és executa una rutina per a omplir les taules pertinents de la base de dades.

Concretament aquesta informació és la següent:

Creació dels metges:

```
private static void afegir_metges() throws IOException
{
        // Afegim dos metges
        DBManager.afegirMetge("00000001-C","Ferran","Casals","Gil","Traumatologia")
        DBManager.afegirMetge("00000002-C","Lluis","Bosch","Gil","Dermatologia");
}
```

Creació dels pacients:

```
private static void afegir_pacients() throws IOException
{
// Afegim quatre pacients. Els dos primers assignats al primer metge
// i els dos seguents al segon metge
DBManager.afegirPacient("00000001-B","Joan","Garcia","Perez","00000001-C");
DBManager.afegirPacient("00000002-B","Andreu","Capmany","Soler","00000001-C");
DBManager.afegirPacient("00000003-B","Mireia","Casals","Ramis","00000002-C");
DBManager.afegirPacient("00000004-B","David","Lopez","Garcia","00000002-C");
}
```

Certificats digitals: del gestor, dels metges i dels pacients:

```
private static void afegir_certificats() throws IOException {
// Afegim sis certificats d'usuari: dos metges i quatre pacients
DBManager.afegirCertificat("00000001-C", RUTA_CERTS_METGES+"00000001-C_DER.crt");
DBManager.afegirCertificat("00000002-C", RUTA_CERTS_METGES+"00000002-C_DER.crt");
DBManager.afegirCertificat("00000001-B", RUTA_CERTS_PACIENTS+"00000001-B_DER.crt");
DBManager.afegirCertificat("00000002-B", RUTA_CERTS_PACIENTS+"00000002-B_DER.crt");
DBManager.afegirCertificat("00000003-B", RUTA_CERTS_PACIENTS+"00000003-B_DER.crt");
DBManager.afegirCertificat("00000004-B", RUTA_CERTS_PACIENTS+"00000004-B_DER.crt");
// Afegim certificat del gestor i el seu pkcs12
DBManager.afegirInfoGestor("1", RUTA_CERTS+"Gestor_DER.crt", RUTA_CERTS+"Gestor.p12",
"uoc0506");
}
```

Afegir historial: Amb aquest mètode fem el rol del metge i el rol del gestor per tal d'afegir un historial mèdic a la base de dades:

```
private static void afegir_historial()
throws IOException, CertificateException, SQLException, PKCSException {
// Primerament creem una visita
Visita v1=new Visita();

v1.setNumeroExpedient("00000001-B");
v1.setData("14/04/2007");
v1.setServei("Traumatologia");
v1.setMetge("Dr. Casals");
v1.setDescripcio("Ingressa a Urgencies perquè li fa molt de mal la ma dreta.");
v1.setDiagnostic("Te la ma trencada.");

// Hem d'afegir separadors de camp per tal que el client pugui
// diferenciar cada camp un cop rebi l'informació del gestor
String dades_a_signar1=v1.getNumeroExpedient()+SEPARADOR_CAMPS_VISITA+
v1.getData()+SEPARADOR_CAMPS_VISITA+v1.getServei()+SEPARADOR_CAMPS_VISITA+
v1.getMetge()+SEPARADOR_CAMPS_VISITA+v1.getDescripcio()+SEPARADOR_CAMPS_VISITA+
v1.getDiagnostic()+SEPARADOR_VISITA_SIGNATURA;

// Signem les dades amb la clau privada del metge

// Creem un signador per al metge
Signador sMetge=null;
try {
        Security.insertProviderAt(new IAIK(), 2);
        sMetge=new Signador(RUTA_CERTS_METGES+"00000001-C.p12","uoc0506");
} catch(Exception e){
        e.printStackTrace();
        System.exit(0);
}

// Afegim la signatura a la visita
v1.setSignaturaMetge(sMetge.signData(dades_a_signar1));

//------------------FINS AQUI L'ACCIO DEL METGE----------------------

// ACCIONS DEL GESTOR AL REBRE UNA VISITA QUE CAL INSERIR

// Verificar la signatura del metge

// Creem un signador per al gestor
Signador sGestor=null;
try {
        sGestor=new Signador(RUTA_CERTS+"Gestor.p12","uoc0506");
} catch(Exception e){
        e.printStackTrace();
        System.exit(0);
}

// Verificar la signatura del metge
sGestor.verifySignature(v1.getSignaturaMetge(),dades_a_signar1.getBytes());

/* El gestor ha de signar la visita, la signatura del metge de la visita, el nou numero de visita i la marca
de temps. Es tracta de la primera visita de l'expedient */

String num_visita1="1";

// Otenim la marca de temps
String marca_temps=Utils.getMarcaTemps();

String dades_a_signar1_gestor=dades_a_signar1+new String(v1.getSignaturaMetge())+
                        num_visita1+marca_temps;
```

Continuació:

```
byte[] info_signada1_gestor=sGestor.signData(dades_a_signar1_gestor);

// XIFRAR

// Obtenim el certificat del gestor
Certificat certGestor=new Certificat(DBManager.getCertificatGestor("1"));

// Obtenim un byte[] amb les dades de la visita
byte[] dadesv1=(dades_a_signar1.getBytes());

// Obtenim el byte[] de la signatura del metge de la visita
byte[] sigv1=v1.getSignaturaMetge();

// Els concatenem
byte[] visita_completa1=Utils.ConcatenarBytesArrays(dadesv1, sigv1);

// Els xifrem
byte info_xifrada1_gestor[]=Utils.xifrarByteArray(visita_completa1, certGestor);

// AFEGIR A LA BBDD

// Afegim un historial mèdic perquè no n´hi ha cap
result=DBManager.afegirExpedientMedic("00000001-B","Joan","Garcia","Perez","944604442");

// Afegim la visita
result=DBManager.afegirVisita("00000001-B", num_visita1, marca_temps, info_xifrada1_gestor,
                                        info_signada1_gestor);

// Actualitzem l'expedient mèdic ara que hem afegit una visita
byte[] signatura_num_ultima_visita=sGestor.signData(num_visita2+"00000001-B");

result=DBManager.actualitzarExpedientMedic("00000001-B",  num_visita2,
            signatura_num_ultima_visita);

}
```

Capítol 11. Conclusions

11.1 Consecució dels objectius

11.2 Possibles ampliacions

11.3 Opinió personal

11.1 Consecució dels objectius

S'han assolit en la seva pràctica totalitat els objectius establerts a l'enunciat del projecte:

Creació d'un sistema criptogràfic garantint les necessitats de seguretat d'un historial mèdic que pot ser gestionar a través d'una xarxa de comunicacions.

El sistema resultant consistirà dels tres components de programari següents:

- Aplicatiu metge: permet que un metge pugui consultar i afegir visites al historial d'un pacient de forma segura.
- Aplicatiu pacient: el pacient utilitza aquest programa per consultar les dades del seu historial.
- Aplicatiu central: Disposa de tots els historials i controla la seva gestió. Permet donar d'alta al sistema els pacients i metges. A més a més s'ha afegit la monitorització en temps real de les connexions i un resum d'històric d'aquestes connexions.

D'una altra banda, per falta de temps no ha estat possible portar a terme totes i cadascuna de les recomanacions que contemplava l'enunciat del PFC. A continuació es detallen algunes característiques què finalment s'ha deixat com a possibles ampliacions:

- No s'ha controlat la classificació de la informació dels historials segons la seva privadesa. Així si un metge té assignat a un pacient podrà visualitzar l'expedient complet.
- No s'ha implementat la possibilitat de modificació dels historials per part del metges. Només poden afegir noves dades.

11.2 Possibles ampliacions

El projecte s'ha dissenyat des d'un principi per admetre ampliacions futures. Cal pensar que la motivació bàsica d'aquest PFC ha estat la creació d'un criptosistema per a permetre el trànsit d'expedients mèdics i d'informació relacionada de forma segura dins d'una xarxa de comunicacions, així que no s'ha fet èmfasi en el conjunt de serveis i operacions que el Gestor de l'aplicació pot oferir als clients.

Recordem les operacions que s'ofereixen: consulta d'expedient, inserció de visites, recuperacions de llista de pacients, totes tres supeditades a l'autenticació bilateral. Quan un

Client demana una operació al Gestor aquesta sol·licitud apareix explícitament en el document XML que tots dos fan servir per a comunicar-se.

Seguint aquesta filosofia l'aplicació esdevé una eina fàcilment ampliable amb la creació de nous tipus de sol·licituds de servei. En altres paraules, la infraestructura ja està creada i les ampliacions consistiran en fer variacions específiques del codi ja existent.

Veiem unes quantes possibilitats:

Classificació de les entrades dels historials segons nivells de seguretat

Un metge no necessariament ha de veure tota la informació continguda en un historial, atès que aquest pot contenir informació sensible i/o irrellevant per l'especialitat del metge. Per tant cada entrada d'un historial podria disposar d'un control de nivell d'accés.

Modificació i eliminació de les dades d'un expedient per part dels metges

Aquesta operació cal implementar-la amb molta cura, atès que la modificació o eliminació del contingut d'un historial és un tema força delicat. Caldrà evitar no deixar rastre de la informació objecte de modificació o eliminació, portant una mena de control de revisions o versions o passant-la a alguna mena d'històrics que permeti que algun responsable imparcial pugui controlar en un moment determinat.

Sol·licitud de visita per part del pacients

Fora interessant que els pacients poguessin visualitzar la disponibilitat de l'agenda del seu metge i així facilitar la programació de les pròximes visites.

Un altre punt d'ampliació és el tipus d'actors que poden intervenir en el sistema. Tenim els metges, els pacients i el Gestor central, però seria possible afegir altre tipus de personal sanitari, com infermers/es, administratius/ves, etc.

Així els infermers podrien omplir dades dels expedients mèdics segons el dictat dels metges i quedant aquesta informació pendent de la revisió final dels metges.
D'una altra banda els administratius podrien validar la programació de les visites sol·licitades pels pacients.

11.3 Opinió personal

Quan vaig triar l'àrea de Seguretat com tema de Projecte de Final de Carrera el meu objectiu era conèixer en detall la implementació d'un sistema criptogràfic, atès que aquest era un aspecte que mai havia tingut l'oportunitat d'experimentar. Doncs be, l'objectiu s'ha aconseguit i en aquests moments disposo d'una base teòrica i pràctica suficient per assolir reptes més ambiciosos dins d'aquesta branca de la Informàtica.

Però a més a més he après en profunditat altres conceptes i tecnologies com XML, RMI, Interficies gràfiques d'usuari en Java, etc... que tot i tenir un paper secundari dins del marc d'aquest projecte és indiscutible la importància que tenen actualment en el desenvolupament de programari en general.

Definitivament la realització d'aquest treball ha estat una experiència enriquidora que m'ha permès integrar gran part dels coneixements adquirits tant en la carrera com en la vida professional. Tot i això l'assoliment d'aquest projecte no ha resultat una tasca gens senzilla, doncs cada fase ha presentat el seu nivell de dificultat i ha requerit un intensiu anàlisi i esforç intel·lectual.

Cal remarcar que ha hagut molta feina a "baix nivell", de fet ha costat molt poder obtenir una primera versió beta. No ha estat fins les etapes finals d'integració quan s'ha pogut apreciar el resultat d'aquests mesos de treball. Així destacaria com de important és fixar-se uns objectius realitzables per cada etapa i en seguir el compliment sistemàtic del pla de treball, doncs la realització de cada fase en el seu ordre portarà al tancament amb èxit del projecte.

Capítol 12. Bibliografia i annexos

12.1 Bibliografia

Java

Sun Microsystems
http://www.java.com/es/

Criptografia amb Java

Java SE Security
http://java.sun.com/javase/technologies/security/

Java Cryptography Extension (IAIK-JCE)
The "Institute for Applied Information Processing and Communication" (IAIK) of the Graz University of Technology
http://jce.iaik.tugraz.at/

XML

Guía Breve de Tecnologías XML
http://www.w3c.es/

W3Schools
http://www.w3schools.com/xml/default.asp

RMI

Remote Method Invocation (RMI)
http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp

SWT

SWT: The Standard Widget Toolkit
http://www.eclipse.org/swt/

Tots els vincles web son operatius a data 31/12/2007

12.2 Annex A. Scripts per gestionar la PKI

generarClaus: Crea una parella de claus del criptosistema RSA. El poden emprar tant els usuaris com altres components de la infraestructura.

```
#!/bin/bash

if [ $# -le 1 ]; then
       echo "Usage: "$0" <key_file> <key_length>"
       echo "or"
       echo "Usage: "$0" <key_file> <key_length> <random_file_length>"
       exit 1
fi

if [ $# -eq 2 ]; then
       openssl genrsa -des3  -out $1 $2
       exit 0
fi

echo getting random bytes
head -c $3 /dev/random > aleatori
echo creating key pair
openssl genrsa -des3  -rand aleatori -out $1 $2

exit 0
```

El més destacable és l'ordre `openssl genrsa -des3  -out $1 $2` on el openssl fa servir el criptosistema RSA amb xifratge triple DES de la longitud especificada en el segon paràmetre.
La sortida serà el fitxer de claus amb el nom especificat en el primer paràmetre.

generaCertificatAutosignat: S'utilitza per a crear el certificat de la CA. Així, a partir d'un parell de claus s'obté un certificat autoritzat on la CA queda legitimada per ella mateixa.

```
#!/bin/bash

if [ $# -le 2 ]; then
       echo "Usage: "$0" <key_file> <file.crt> <dies>"
       exit 1
fi

openssl req -new -sha1 -x509  -key $1 -out $2 -days $3

exit 0
```

En aquest cas és important observar com s'especifica la funció resum sha1 i el format de certificat X.509, un dels més estesos en infraestructura de clau pública.
El primer paràmetre és el fitxer amb el parell de claus generat per la CA. El segon és el nom que tindrà el certificat de la CA i el tercer el nombre de dies de vigència del certificat.

generaPeticioCertificat: Es fa servir quan un usuari ja ha generat el seu parell de claus i vol que l'Autoritat de Registre li gestioni l'obtenció d'un certificat.

```
#!/bin/bash
if [ $# -le 2 ]; then
       echo "Usage: "$0" <key_file> <file.csr> <config_file>"
       exit 1
fi
openssl req -new -sha1 -config $3 -key $1 -out $2
exit 0
```

A partir del parell de claus i de la informació introduïda al fitxer de configuració openssl.cnf el sistema té suficient per a generar la petició. El fitxer amb la sol·licitud tindrà el nom especificat en el segon paràmetre.

generaCertificat: La CA l'utilitza per a crear el certificat d'usuari demanat per la RA.

```
#!/bin/bash
if [ $# -le 2 ]; then
       echo "Usage: "$0" <file.csr> <file.crt> <config_file>"
       echo "or"
       echo "Usage: "$0" <file.csr> <file.crt> <config_file>
<extensions_section>"
       exit 1
fi
if [ $# -le 3 ]; then
       openssl ca -config $3 -out $2 -infiles $1
       exit 0
fi
openssl ca -config $3 -out $2 -extensions $4 -infiles $1
exit 0
```

El fitxer de petició de certificat (*.csr) generat prèviament, el nom del certificat que es vol obtenir i el fitxer openssl.cnf son els paràmetres necessaris per tal de generar el certificat d'usuari.

generaPKCS12: Permet obtenir un fitxer contenidor amb el parell de claus del usuari, el seu certificat i el certificat de la CA.

```
#!/bin/bash

if [ $# -le 3 ]; then
        echo "Usage: "$0" <key_file> <file.crt> <CA_certificate>
<file.p12>"
       exit 1
fi

openssl pkcs12 -export -in $2 -inkey $1  -certfile $3 -out $4

exit 0
```

12.3 Annex B. Instal·lació i configuració de MySQL

Instal·lació del SGBD

Aquest PFC s'ha provat tant amb la versió 4.1 com amb la 5.0 de MySQL Server. Les explicacions d'aquest apartat son per la 5.0.

Cal executar el setup.exe que apareix un cop descomprimit el zip. (L'assistent de la instal·lació és força senzill, triar les opcions per defecte). Credencials del administrador: usuari 'root' i password 'root.

Per a comprovar la instal·lació: obrir una finestra del DOS, ubicar-nos al directori bin del MySQL i fer:
>mysqlshow -u root -p (return)
Enter password> root

I ens mostrarà les BBDD de prova existents.

Instal·lació dels complements

Instal·lant el programa mysql-gui-tools-5.0-r12-win32.msi disposarem de les eines MySQL Administrator i MySQL Query Browser, amb les quals podrem administrar i gestionar completament el servidor SQL.

Després cal instal.lar el connector Java JDBC de MySQL; mysql-connector-java-3.1.8a.zip. No cal instal·lar-lo, només descomprimir-lo i copiar-lo a C:\. Després s'ha de modificar la variable d'entorn CLASSPATH afegint la ruta "c:\mysql-connector-java-3.1.8a\mysql-connector-java-3.1.8-bin.jar". Si no funciona això últim es pot afegir el .jar al conjunt de llibreries del projecte dins del Eclipse: java build path->libraries->add external JARs.

Creació d'usuaris

Només s'ha creat un usuari: l'usuari 'daniel' amb contrasenya 'daniel', propietari de la base de dades 'PFC'. La creació de l'usuari es fa des de l'script 'creacioBD.sql' que es mostra al final d'aquest capítol.

Creació de la base de dades

La creació de la base de dades ‘PFC’ es fa des de l’script ‘creacioBD.sql’ que es mostra al final d’aquest capítol.

Creació de les taules del projecte

La creació de les taules de la base de dades 'pfc' la farem amb un script des del 'MySQL Query Browser'. Entrem com usuari ‘root' password 'root' i ignorem la selecció de cap schema.

Un cop a dins anem a 'file'->'open script' i seleccionem el fitxer on tenim les instruccions de creació de les taules.

Script per a la creació de les taules

```
CREATE DATABASE IF NOT EXISTS PFC;

use PFC;

create user daniel identified by 'daniel';

GRANT ALL ON PFC.* TO 'daniel'@'%';

-- alter table pacient drop foreign key FK_METGE;

drop table if exists metge;

create table metge (
       nif_metge varchar(10) not null,
       nom_metge varchar(64) not null,
       cognom1_metge varchar(64) not null,
       cognom2_metge varchar(64),
       departament varchar(20) not null,
       primary key(nif_metge)
);


drop table if exists pacient;

create table pacient (
       nif_pacient varchar(10) not null,
       nom_pacient varchar(64) not null,
       cognom1_pacient varchar(64) not null,
       cognom2_pacient varchar(64),
       id_metge varchar(10) not null,
       primary key(nif_pacient)
);

alter table pacient add constraint FK_METGE foreign key (id_metge)
references metge (nif_metge);
```

Script per a la creació de les taules (continuació):

```
drop table if exists sessio_autenticada;
create table sessio_autenticada (
       sn_gestor varchar(64) not null,
       sn_client varchar(64) not null,
       id_client varchar(10) not null,
       servei varchar(20) null,
       logon TimeStamp(22) not null,
       primary key(sn_gestor,sn_client)
);


drop table if exists historic_sessions;
create table historic_sessions (
       sequencia int not null auto_increment,
       sn_gestor varchar(64) not null,
       sn_client varchar(64) not null,
       id_client varchar(10) not null,
       servei varchar(20) null,
       logon TimeStamp(22) not null,
       logout TimeStamp(22) not null,
       primary key(sequencia)
);

drop table if exists certificat_usuari;
create table certificat_usuari (
       id_usuari varchar(10) not null,
       certificat blob not null,
       primary key(id_usuari)
);

drop table if exists cripto_gestor;
create table cripto_gestor (
       id varchar(64),
       certificat blob,
       p12 blob,
       password_p12 varchar(10),
       primary key(id)
);

drop table if exists expedient_medic;
create table expedient_medic (
       num_exp varchar(10) not null,
       nom_pacient varchar(64) not null,
       cognom1_pacient varchar(64) not null,
       cognom2_pacient varchar(64),
       telefon_pacient varchar(15),
       data_creacio TimeStamp(14) not null,
       num_ultima_visita varchar(3),
       signatura_num_ultima_visita blob,
       primary key(num_exp)
);

drop table if exists visita;
create table visita (
       num_exp varchar(10) not null,
       num_visita varchar(3) not null,
       marca_temps varchar(23) not null,
       info_xifrada_gestor blob not null,
       info_signada_gestor blob not null,
       primary key(num_exp,num_visita)
);
```

12.4 Annex C. Índex de figures

12.5 Annex D. Solució a possibles problemes

12.5.1 Errors en l'execució

Errors en la part del servidor

Símptomes: Quan s'executa el servidor (run_servidor.bat) apareix un missatge d'error com el de la figura 12.1:

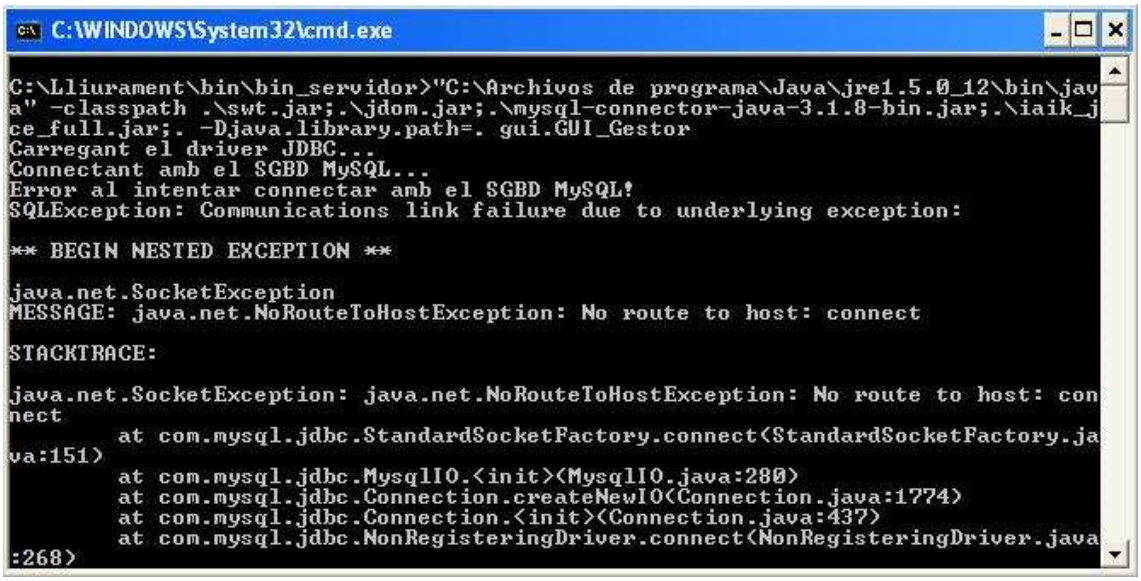


Figura 12.1. Problema amb el programa servidor relacionat amb la BBDD.

Causa: L'error està relacionat amb la base de dades MySQL.

Solució: Podem provar el següent:

- Assegurar-se que està instal.lat el programa MySQL i que el servei està iniciat.
- Si la base de dades està en un ordinador diferent al del programa servidor, comprovar des de les opcions d'administració que MySQL està preparat per escoltar per el port TCP 3306. Amb la instal·lació normal per defecte ja està preparat per permetre connexions de xarxa per aquest port.
- Comprovar que hem executar el fitxer de comandes SQL que crea l'usuari 'daniel' i la base de dades 'pfc'. Per a fer això executem el programa 'Query Browser' de MySQL, ens validem com a root i executem l'script de comandes.
- Comprovar que el fitxer 'database.config' del directori 'conf' apunta a l'adreça IP de l'ordinador que està executant la base de dades.

Símptomes: Quan s'executa el servidor (run_servidor.bat) apareix la interficie gràfica, però quan es polsa sobre el botó 'Iniciar', tarda un cert temps en respondre i finalment dona un error com el de la figura 12.2. A més a més els clients no es poden connectar al servidor:

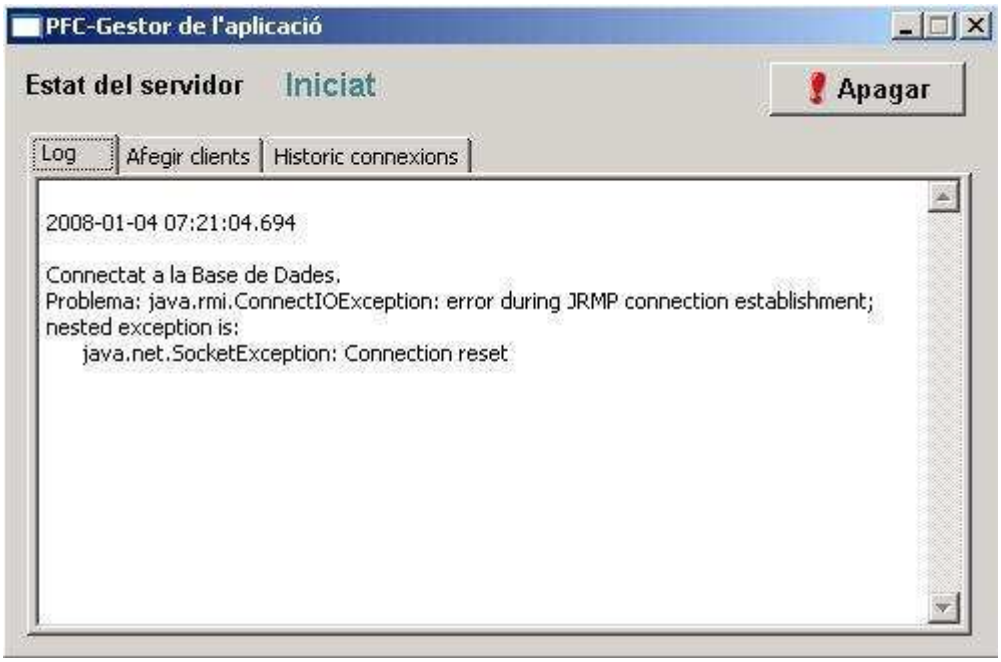


Figura 12.2. Problema amb el programa servidor.

Causa: Segurament alguna aplicació de l'ordinador on s'executa el servidor està utilitzant el port TCP 1099 i això provoca que el servidor no pugui iniciar el RMI registry.
Si no és el cas, per algun altre motiu el programa servidor no pot iniciar el RMI registry.

Solució: Alliberar el port TCP 1099 per tal que la nostra aplicació el pugui utilitzar. Hi ha molt programes gratuïts que permeten eliminar els ports de manera gràfica (per exemple CurrPorts v1.31 http://www.nirsoft.net).

Error en el programa client

Símptomes: Quan s'executa el programa client, apareix la pantalla d'inici, però després d'uns segons surt un missatge com el de la figura 12.3:

Figura 12.3. Problema amb el programa client.

Causa: Un motiu podria ser que el programa servidor no estigui en funcionament. També podria estar ocasionat perquè l'adreça IP del fitxer de configuració remote_host.config del directori 'conf' estigui apuntant a una adreça incorrecta.

Solució: Verificar que el servidor estigui iniciat i que l'adreça IP del fitxer de configuració remote_host.config del directori 'conf' estigui apuntant a l'adreça IP de l'ordinador on s'està executant el servidor.

Errors en qualsevol tipus de programa (client o servidor)

Errors diversos de Java

Símptomes: Quan s'executa el programa (client o servidor) obtenim un missatge com el de la figura 12.4:

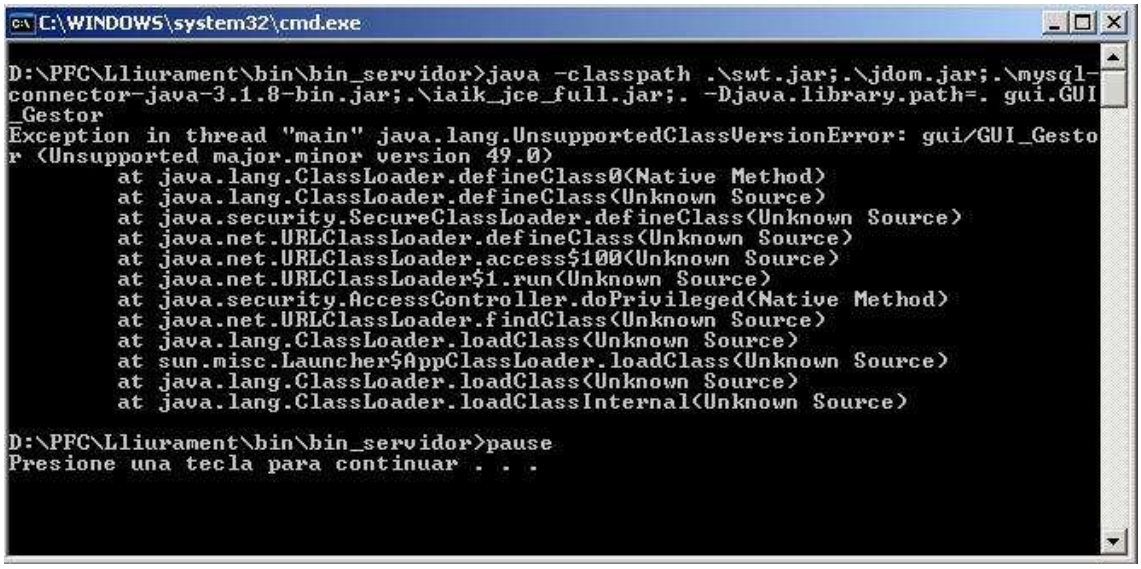


Figura 12.4. Problema per error de Java.

Causa: Un motiu podria ser que la versió de Java que tenim instal·lada sigui inferior a la 5. L'altre motiu podria ser que tenim més d'una versió de Java en el nostre sistema i la versió que s'està invocant sigui anterior a la 5.

Solució: D'una banda cal un entorn d'execució Java igual o superior a la versió 5.0. Així si comprovem que el nostre sistema disposa d'una versió inferior o el Java no està instal.lat descarreguem un jre (entorn d'execució Java) apropiat des de la web de Sun (http://www.java.com/es/).

Per exemple instal·lem el fitxer: jre-1_5_0_12-windows-i586-p.exe que correspon a J2SE Runtime Environment 5.0 Update 12

D'una altra banda ens assegurem de quina és la versió de Java que tenim al path. Si tenim diferents java.exe en el nostre ordinador cal especificar en el fitxer run_xxxx.bat la ruta del java que volem executar. Per exemple amb la sentencia següent estem especificant sense ambigüitat la versió de java que volem executar:

"C:\Archivos de programa\Java\jre1.5.0_12\bin\java" -classpath
.\swt.jar;.\jdom.jar;.\mysql-connector-java-3.1.8-bin.jar;.\iaik_jce_full.jar;.
-Djava.library.path=. gui.GUI_Gestor

Símptomes: Quan s'executa el programa (client o servidor) obtenim un missatge com el de la figura 12.5:

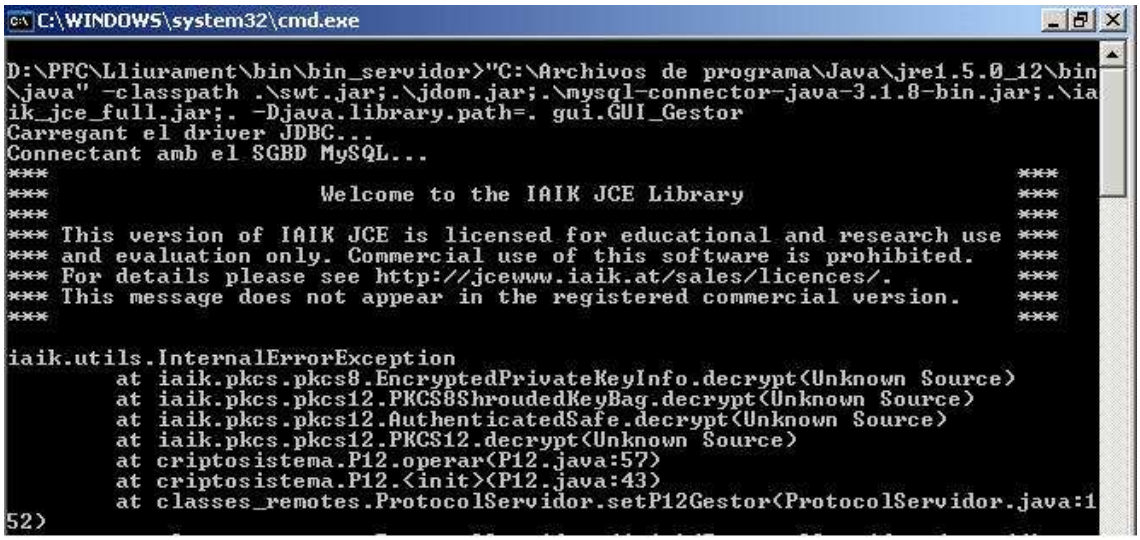


Figura 12.5. Problema per error de Java.

Causa: L'aplicació necessita longitud de claus de xifratge superior als estàndards de Java.

Solució: Sobreescriure els fitxers local_policy.jar i US_export_policy.jar del directori C:\Archivos de programa\Java\jre1.5.0_12\lib\security
per els fitxers homònims però que sí admeten longitud de claus de xifratge llargues.

Des de l'adreça:
https://sdlc1e.sun.com/ECom/EComActionServlet;jsessionid=47CDB31F935F3F7450847C13341CD0D0
de la web de Sun ens podem descarregar el fitxer jce_policy-5.zip o jce_policy-6.zip, segons la nostra versió de java.

12.5.2 Establiment de l'entorn de desenvolupament i possibles errors en la compilació

Disposar de l'entorn de desenvolupament per a compilar i executar l'aplicació no és una tasca gens senzilla. En aquest apartat s'explica pas per pas el que s'ha fet per tal de preparar l'entorn partint d'un sistema amb només el sistema operatiu Windows XP instal.lat.

Eclipse

Aquesta aplicació ha estat desenvolupada amb la versió 3.1.2 del famós IDE.
Descomprimim el fitxer 'eclipse-SDK-3.1.2-win32.zip' al directori c:\Eclipse 3.1

Longitud de claus de xifratge superior als estàndards de Java

Com ja s'ha explicat anteriorment al directori lib/security de la nostra versió de java sobreescrivim els fitxers 'local_policy.jar' i 'US_export_policy.jar' per els fitxers homònims però que sí admeten longitud de claus de xifratge llargues.

Les llibreries que necessitem (JAR)

Necessitem els següents arxius:

Llibreria	Descripció
iaik_jce_full.jar	Permet la utilització de les classes criptogràfiques
jdom.jar	Per tal de llegir i crear documents XML
mysql-connector-java-3.1.8-bin.jar	Connector Java per a MySQL
swt.jar	Permet la creació d'elements gràfics

En aquest projecte s'ha ubicat aquests arxius al directori arrel de l'Eclipse.

Configuració de l'Eclipse

Tenim els següents tres projectes a la carpeta 'project':

PFC_Client_Metge
PFC_Client_Pacient
PFC_Servidor

Des de l'Eclipse importem els tres projectes ens apareixeran molts errors:

Comencem canviant la versió de JRE si s'escau:

Window->Preferences->Java->Installed JREs

Mirem si tenim seleccionat el JRE 5.0 o un d'anterior. Si només està el anterior fem el següent:
Botó 'Add'
JRE Name: JRE 5.0
JRE Home directori: C:\Archivos de programa\Java\jre1.5.0_12
Botó 'Ok'

Desseleccionem el JRE anterior i seleccionem el JRE 5.0.

Possibles problemes de compilació

A fer en cada projecte:

PROBLEMA
Project PFC_Client_Metge is missing required library: 'C:\eclipse 3.1\swt.jar'

SOLUCIÓ
Cal copiar el fitxer 'swt.jar' al directori c:\eclipse 3.1. Un cop fet això refresquem el projecte i desapareixen els errors d'aquest tipus.

PROBLEMA
The project was not built since it depends on PFC_Servidor, which has build path errors PFC_Client_Metge

SOLUCIÓ
Hem de mirar primer els problemes que presenta el projecte PFC_Servidor

PROBLEMA
Project PFC_Servidor is missing required library: 'C:\Archivos de programa\MySQL\mysql-connector-java-3.1.8\mysql-connector-java-3.1.8-bin.jar' PFC_Servidor

Eclipse no troba la llibreria per connectar des de Java amb MySQL.

SOLUCIÓ
1.Copiem el fitxer 'mysql-connector-java-3.1.8-bin.jar' al directori que vulguem, per exemple a c:\eclipse 3.1\

2.Llavors anem a l'Eclipse fem un clic a sobre del projecte PFC_Servidor->Build Path->Configure Build Path.
3.Anem a la pestanya 'Libraries' seleccionem l'entrada que diu 'mysql-connector...' i polsem sobre el botó 'Remove'.
4.Polsem sobre el botó 'Add External JARs' i seleccionem el fitxer 'mysql-connector-java-3.1.8-bin.jar' des del directori c:\eclipse 3.1
5.Acceptem

Nota: Amb aquestes accions haurem solucionat els problemes anteriors, però ara veurem com apareixen d'altres problemes nous.

PROBLEMA
Syntax error, parameterized types are only available if source level is 5.0 SWTResourceManager.java PFC_Client_Metge/src/com/swtdesigner

SOLUCIÓ
Per a cada projecte fem el seguent:

1.Clic amb el botó dret.
2.Properties
3.Java Compiler
4.Marquem 'Enable project especific setting'
5.Seleccionem 5.0 en el desplegable 'Compiler compliance level' de l'apartar JDK Compliance.

Acceptem la pregunta sobre recompilar cada projecte.

PROBLEMA
The import iaik cannot be resolved Certificat.java PFC_Client_Metge/src/criptosistema

Eclipse no troba la llibreria per utilitzar les funcions avançades de xifratge.

SOLUCIÓ
1.Copiem el fitxer 'iaik_jce_full.jar' al directori que vulguem, per exemple a c:\eclipse 3.1\
2.Llavors anem a l'Eclipse fem un clic a sobre del projecte PFC_Servidor->Build Path->Configure Build Path.
3.Polsem sobre el botó 'Add External JARs' i seleccionem el fitxer 'iaik_jce_full.jar' des del directori c:\eclipse 3.1
4.Acceptem

PROBLEMA

The import org.jdom cannot be resolvedUtilsXML.java PFC_Client_Metge/src/utils

Eclipse no troba la llibreria per utilitzar XML.

SOLUCIÓ

1.Copiem el fitxer 'jdom.jar' al directori que vulguem, per exemple a c:\eclipse 3.1\

2.Llavors anem a l'Eclipse fem un clic a sobre del projecte PFC_Servidor->Build Path->Configure Build Path.

3.Polsem sobre el botó 'Add External JARs' i seleccionem el fitxer 'jdom.jar' des del directori c:\eclipse 3.1

4.Acceptem