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

Nom Estudiant: Gerard Farràs i Ballabriga
Enginyeria en Informàtica

Nom Consultor: Jordi Castellà-Roca

Data Lliurament: 7 de Gener del 2007

Dedicatòria i agraïments.

A la Sandra Escolies i Caballol i al Dr. Casserras i Aynés per a haver-me explicat pacientment els conceptes d'un sistema d'informació d'àmbit sanitari.

Resum del projecte

El projecte consisteix en la implementació d'un esquema criptogràfic per a poder accedir de forma segura a historials mèdics a través d'una xarxa de telecomunicacions.

Les noves tecnologies permeten accedir de forma remota a grans volums d'informació utilitzant xarxes de telecomunicacions. Una aplicació molt interessant d'aquest accés remot a dades és el de la informació de tipus sanitari: La història clínica d'un determinat pacient. D'aquesta manera, s'obren noves possibilitats en aquest terreny, com pot ser la telemedicina, amb la possibilitat de realitzar diagnòstics de forma remota (per exemple, que un laboratori pogués introduir els resultats d'una analítica o un especialista radiòleg informés del seu diagnòstic visualitzant només una radiografia) o la compartició d'historials clínics entre centres sanitaris, entre d'altres aplicacions.

S'ha de tenir molt en compte però que la història clínica d'un pacient és informació altament sensible, protegida per Llei i que s'ha d'assegurar especialment la seva protecció, i més tenint en compte que aquest informació viatge per xarxes no segures. Entre d'altres, el sistema hauria d'assegurar les propietats de *confidencialitat*, *autenticitat*, *integritat* i *no-repudi*.

El projecte ha de generar un seguit d'aplicacions per a que els usuaris i els metges puguin accedir a través d'una xarxa de comunicacions als historials clínics mantenint sempre els requisits de seguretat comentats anteriorment.

Les tecnologies emprades per a la implementació d'aquest PFC seran les següents:

- Java com a llenguatge de programació.
- Llibreria en Java IAIK per a la implementació dels protocols criptogràfics.
- MySQL com a servidor de base de dades.
- XML com a format per a transmetre les dades.
- RMI (Java Remote Method Interface) com a mètode per a realitzar les comunicacions.

Paraules clau.

Sistemes d'informació sanitari.
Hospital Information System (HIS).
Historials clínics.
Criptografia.
Seguretat informàtica.
Java.
IAIK.
Public Key Infrastructure.
XML.
MySQL.
RMI.

Nom de l'àrea de PFC.

Seguretat informàtica.

Índex de continguts

Capítol 1. Introducció.

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

Les noves tecnologies permeten accedir de forma remota a grans volums d'informació utilitzant xarxes de telecomunicacions. Una aplicació molt interessant d'aquest accés remot a dades és el de la informació de tipus sanitari: La història clínica d'un determinat pacient. D'aquesta manera, s'obren noves possibilitats en aquest terreny, com pot ser la telemedicina, amb la possibilitat de realitzar diagnòstics de forma remota (per exemple, que un laboratori pogués introduir els resultats d'una analítica o un metge radiòleg informés del seu diagnòstic visualitzant una radiografia) o la compartició d'historials clínics entre centres sanitaris, entre d'altres aplicacions.

D'altra banda, comentar que és la Llei Orgànica de Protecció de Dades de Caràcter Personal (en endavant, LOPD), qui regula la protecció de les dades de caràcter personal i el posterior tractament que pugui fer-se de les mateixes. Aquesta llei defineix el concepte de dada de caràcter personal com a qualsevol informació que concerneixi a una persona física identificada o identificable. És ben clar que les dades sanitàries son dades de caràcter personal i que, per tant, el tractament d'aquestes queda sota la responsabilitat de protecció de la LOPD.

La referència a aquest tipus de dades apareix en l'article 7 de la LOPD, dins l'apartat per a tractament de dades de caràcter personal especialment protegits i també en l'article 8.

Tot i això, no solament la LOPD regula el tractament de dades de tipus sanitari. La Llei 14/1986 General de Sanitat estableix en el seu article 10.3 el dret del ciutadà a la confidencialitat de les seves dades i en l'article 23 recull la potestat de l'administració sanitària per a crear registres. Aquests registres hauran d'adoptar les mesures tècniques i organitzatives que estableix el Real Decret 994/1999.

Definit el concepte de dada de tipus sanitari com a dada relativa a la salut de l'interessat (pacient/persona física titular de les dades) és necessari tenir en compte que aquest tipus de dades només pot ser obtingut quan expressament ho determini la Llei o l'afectat ho consenteixi de manera expressa (article 7 LOPD). Tot i això, les institucions i els centres sanitaris públics i privats podran procedir al tractament d'aquest tipus de dades subjectant-se sempre a la legislació estatal o autonòmica sanitària específica.

En quant a la legislació autonòmica existent relativa a les dades sanitàries, destaca la Llei del Parlament Català del 21 de desembre del 2000, sobre els “drets d'informació relatius a la salut i a la autonomia del pacient i a la documentació clínica”. En aquesta llei es recull el dret del pacient a conèixer tota la informació sobre la seva salut i que aquesta informació sigui verídica i comprensible i el dret de tota persona al fet que es respecti la confidencialitat de les dades referents a la seva salut.

En el moment de la recollida de les dades, el metge ha de complir amb determinades formalitats informant de forma expressa, precisa i inequívoca al pacient de:

- L'existència d'un fitxer, de la finalitat de la recollida de dades i dels destinataris de dita informació.
- La identitat i direcció del responsable del fitxer.
- Del caràcter obligatori o facultatiu de les respostes a les preguntes.

- I de la possibilitat d'exercitar els drets d'accés, rectificació, cancel·lació i oposició. És a dir, el pacient té dret a accedir a la història clínica i a obtenir una còpia de les dades. Té també dret de rectificació i cancel·lació en el cas que el pacient desitgi que les seves dades siguin rectificades o cancel·lades.

Qualsevol fitxer que contingui dades de salut ha de complir amb les mesures de seguretat de nivell alt (article 20 i ss del Real Decret94/1999), és a dir, a més de complir les mesures de seguretat de nivell bàsic i mitjà (document de seguretat, registre d'incidències, identificació/autenticació, controls d'accés, còpies de seguretat i recuperació, responsable de seguretat, controls d'accés físic), ha de comptar amb un registre d'accessos, és a dir, no tot el personal ha de tenir accés a tot el fitxer, ha de comptar amb un xifrat en les telecomunicacions i efectuar-se auditories, com a mínim, de forma bianual.

La Llei ja dona doncs les indicacions que s'haurien d'implementar. A més, en aquest projecte, es tindrà cura de les següents característiques:

- La informació dels historials clínics ha de romandre xifrada i s'ha de transportar també de forma xifrada per a assegurar-nos que es mantindrà sempre la seva “confidencialitat”.

- La informació que es desa en cada historial clínic ha estat “escrita” per professionals autoritzats, assegurant-nos sempre que aquesta és autèntica.

- La informació que es desa en cada historial clínic no pot haver estat “modificada” per a cap usuari malintencionat, amb la qual cosa el sistema ha d'assegurar que aquesta roman íntegra.
- Una vegada un determinat professional ha escrit informació en la història clínica, aquest no ha de poder dir més endavant que no ha estat ell qui l'ha realitzada, amb la qual cosa el sistema garanteix el no-repudi.

Comentar també que actualment ja hi ha en el mercat gran quantitat de programari que implementa sistemes d'informació hospitalària (HIS) però creiem que l'aportació més important d'aquest PFC és el seu intens ús d'eines criptogràfiques.

Objectius del PFC.

Com ja s'ha comentat anteriorment, el principal objectiu d'aquest projecte de final de carrera és desenvolupar un programari que implementi un sistema de gestió sanitària senzill on els usuaris es puguin connectar de forma remota a un gestor central que contingui tota la informació de forma centralitzada però emprant criptografia de clau pública per a totes les operacions.

A continuació, una llista més concreta dels objectius:

- Implementació de l'esquema criptogràfic amb les operacions següents.
  - Identificació d'un usuari en el sistema.
  - Autenticació dels usuaris.

    - Consulta dels historials clínics per part dels usuaris amb permisos.
    - Consulta dels pacients assignats a un metge.
    - Inserció de dades a l'historial mèdic d'un determinat pacient.
- La informació hauria de residir de forma centralitzada, en un servidor de base de dades relacional, com per exemple, MySQL.
- La comunicació entre els diferents agents es realitzarà a través de missatges en format XML (eXtensible Markup Language).
- La comunicació entre els diferents agents es realitzarà a través del protocol RMI (Java Remote Method Implementation).
- S'hauria de de generar almenys 3 interfícies gràfiques per als diferents perfils d'usuari:
    - Aplicatiu metge: Per a permetre que un metge pugui consultar i modificar l'historial d'un pacient de forma segura.
    - Aplicatiu pacient: Per a permetre que un usuari pugui consultar el seu (i només el seu) historial.
    - Gestor central: Aplicatiu amb el repositori amb tots els historials. Des d'aquest aplicatiu, s'haurien de poder donar d'alta en el sistema els pacients i els metges.

Enfocament i mètode seguit.

Aquest sistema necessita la interacció dels següents mòduls:

- L'esquema criptogràfic és la base del projecte. Les classes s'han dissenyat de manera que poguessin ser reutilitzades en qualsevol dels aplicatius.

- El format de la informació que es transmet es desa en documents XML per tal de poder efectuar una manipulació senzilla de les dades per a les diferents parts.

- Els usuaris del sistema s'han de poder comunicar per xarxa, així que serà necessària una plataforma de comunicació que permeti la transmissió de la documentació clínica en format XML entre el servidor central i les altres parts del sistema, ja siguin, pacients o metges.

- La informació que es gestiona s'ha de desar en una base de dades relacional.

- Finalment, per a cada perfil d'usuari del sistema es requereix una interfície gràfica per a dur a terme les funcionalitats del sistema.

En resum doncs, el sistema compta amb els mòduls següents:

- L'esquema criptogràfic.

- La representació de les dades en format XML.

- La comunicació dels components emprant RMI.

- La base de dades MySQL.

- La interfície gràfica.

- La documentació.

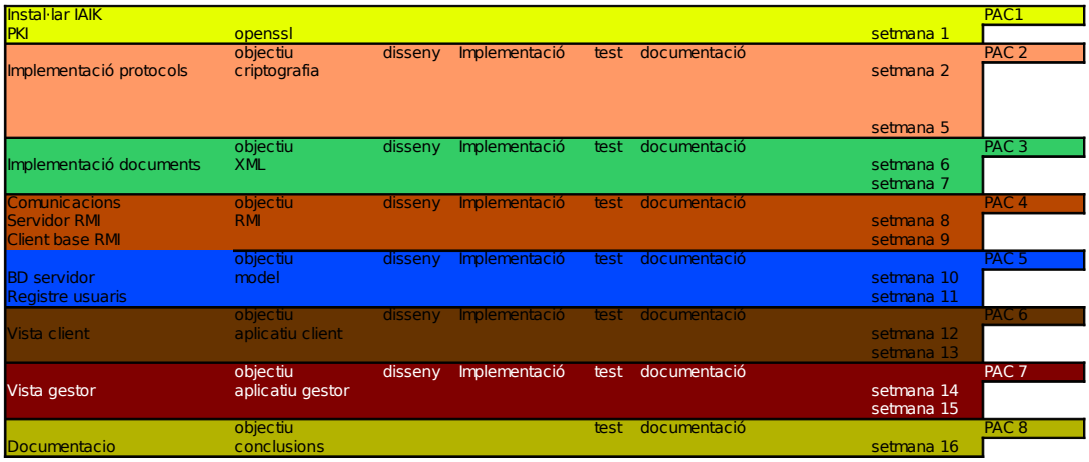
El mètode que s'ha seguit per a implementar el sistema ha estat el d'anar construint cadascun dels mòduls successivament l'un darrera l'altre efectuant proves unitàries amb cadascun d'ells. Cada mòdul s'integra al següent i es realitzen proves unitàries de nou. Així, el que s'ha fet, pas a pas, és el següent:

- Definició del protocol criptogràfic.
- Implementació dels diferents protocols en un entorn local, incloent l'esquema criptogràfic.
- Integració de XML en el mòdul.
- Implementació de la comunicació entre components. En aquest pas s'envien documents XML entre els aplicatius de metges i pacients al servidor central a través de tecnologia RMI.
- Integració de la base de dades.
- Finalment, creació de les interfícies gràfiques per als clients (pacients i metges) i gestors per a poder realitzar amb comoditat totes les operacions implementades.

Comentar que, malgrat que en la interfície gràfica és possible esmerçar-s'hi amb multitud de detalls i que, de fet, és el que s'acaba visualitzant, no ha estat l'objectiu prioritari.

Planificació del projecte.

El projecte va començar durant el setembre del 2007. La planificació que es va fer és detalla a continuació:



Calendari

Del 19 al 23 de setembre:

* Instal·lació de la llibreria IAIK i generació de PKI amb OpenSSL.

Del 24 de setembre fins al 21 d'octubre:

* Implementació dels protocols criptogràfics. Fases de disseny, implementació, test i documentació.

Del 22 d'octubre fins al 4 de novembre:

* Implementació dels documents en XML. Fases de disseny, implementació, test i documentació.

Del 5 al 18 de novembre:

* Implementació dels documents en RMI. Fases de disseny, implementació, test i documentació.

Del 19 de novembre al 2 de desembre:

* Implementació de la base de dades del servidor. Fases de disseny, implementació, test i documentació.

Del 3 al 16 de desembre:

* Implementació de la interfície gràfica d'usuari. Fases de disseny, implementació, test i documentació.

Del 17 al 30 de desembre:

* Implementació de la interfície gràfica per al gestor. Fases de disseny, implementació, test i documentació.

Del 31 al 6 de gener:

* Redacció final de la documentació.

En resum:

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

	Dia festiu
	Lliurament PFC
	Trobada inici

Productes obtinguts

- Programari per als pacients i professionals administratius.

Producte desenvolupat en Java que disposa d'una interfície gràfica per a que tant pacients com professionals sanitaris pugui accedir de forma remota als seus historials clínics seguint els protocols criptogràfics dissenyats. Els professionals sanitaris, a més, han de poder afegir nous apunts en les històries clíniques dels seus pacients així com visualitzar un llistat dels pacients que tenen assignats.

- Programari per al gestor del sistema.

Producte desenvolupat en Java que actua com a gestor del sistema, fent de pont entre les dades i els clients i validant i efectuant totes les operacions d'aquests. Ha faltat per a desenvolupar en el programari del gestor la possibilitat d'afegir nous pacients/metges en el sistema de forma automatitzada.

Breu descripció dels altres capítols de la memòria

- Capítol 2. PKI i IAIK.

  Explicació del per què és necessari IAIK, conjuntament amb el seu procés d'instal·lació. També perquè és necessari una PKI i mètode de generació dels certificats via OpenSSL.

- Capítol 3. Protocols criptogràfics.

  Descripció detallada dels protocols criptogràfics implementats.

- Capítol 4. Representació de les dades: XML.

  Descripció dels documents XML que s'han emprat per a efectuar les comunicacions de dades durant l'execució dels protocols criptogràfics.

- Capítol 5. Comunicació entre agents: RMI.

  Descripció de la comunicació dels diferents components del sistema a través de RMI.

- Capítol 6. Gestor de base de dades.

  Disseny de la base de dades que s'ha emprat per a emmagatzemar les dades del sistema.

- Capítol 7. Interfície gràfica.

  Descripció de les interfícies gràfiques que s'han generat per als aplicatius.

- Capítol 8: Jocs de proves.

  Descripció dels jocs de proves que s'han efectuat per a provar les aplicacions desenvolupades.

- Capítol 9. Valoració econòmica.

  Valoració econòmica aproximada del que s'ha desenvolupat

- Capítol 10. Conclusions.

- Capítol 11. Possible treball futur.

- Glossari de termes.

- Bibliografia.

- Annexos.

Capítol 2. Public Key Infrastructure (PKI) i IAIK.

Introducció a PKI.

L'esquema criptogràfic que s'implementa en aquest projecte es basa en criptografia de clau pública i requereix, per als diferents actors (metges, pacients i gestor) un certificat digital i una parella de claus (una de pública i una de privada).

Per a la gestió d'aquests certificats, amb les corresponents operacions de generació, revocació i validació dels diferents certificats s'empra el que s'anomena una PKI, que correspon a l'acrònim en anglès de Public Key Infrastructure.

Una PKI consta d'una autoritat de certificació (CA o Certification Authority, en anglès) i també d'autoritats de registre (RA o Registry Authority, en anglès).

Els passos següents indiquen quin és el procediment habitual per a la generació d'un certificat:

- L'usuari en qüestió genera una parella de claus i realitza una petició de certificat a la RA.
- La RA valida la identitat de l'usuari i envia la petició a la CA.
- La CA rep la petició de la RA i emet el certificat corresponent.

Si un usuari veu que la clau privada corresponent al seu certificat ha estat compromesa ha de comunicar-ho a la CA. La CA revoca aquell certificat incloent-lo en una llista de certificats revocats. La llista de certificats revocats la notem amb les sigles CRL del terme anglès Certificate Revocation List.

A continuació es detallen els passos habituals que s'utilitzen per a verificar la validesa d'un certificat.

- Primer, examinar el certificat i saber quina CA l'ha generat.
- Si la CA és de confiança , seguirem amb el procés de verificació.
- El següent pas serà assegurar-nos que el certificat continua essent vàlid i no ha estat revocat. Per a esbrinar-ho, és possible fer servir un protocol tipus OCSP per a preguntar-li directament a la CA o descarregar-nos directament la CRL de la CA i cercar si hi ha el certificat en procés de validació.

OpenSSL

En aquest projecte s'ha emprat la llibreria open-source gratuïta i de lliure distribució Openssl per a la generació de la PKI. OpenSSL està disponible, a més, en múltiples plataformes (com en Windows, GNU/Linux, entre d'altres). La versió emprada és la openssl-0.9.8b-14.

Comentar també que els certificats digitals que s'emeten segueixen l'estàndard X.509. La clau privada i el corresponent certificat s'emmagatzemen en un fitxer segons l'estàndard PKCS12.

Procediments per a la generació dels fitxers de la PKI.

El projecte implementat, basat en criptografia de clau pública, requereix per a cadascun dels diferents actors del sistema una parella de claus emmagatzemades en un fitxer en format PKCS12. Aquest arxiu conté els elements següents:

- Clau pública i clau privada de l'usuari.
- Certificat de l'usuari emès per l'autoritat de certificació.
- Certificat amb l'autoritat de certificació.

El primer pas que s'ha seguit en aquest projecte ha estat l'obtenció del certificat de l'autoritat de certificació. Si el sistema es decidís d'implantar en organismes on ja es disposés d'una infraestructura de clau pública (PKI) amb una autoritat de certificació establerta, aquest pas no seria necessari.

El guió per a generar el certificat de la autoritat de Certificació és el següent:

- Pas 1. Generar la parella de claus de la CA amb una longitud de clau de 2048 bits utilitzant el guió de seqüències "generarClaus". Aquest guió generarà un fitxer anomenat CA.key.
- Pas 2. Generar un certificat autosignat a partir de la parella de claus de la CA. Aquest serà el certificat de la CA i el generarem a través del guió "generaCertificatAutosignat". El nom del certificat generat serà CA.crt .

Amb l'autoritat de certificació CA ja establerta, podem ja crear els certificats per als diferents actors.

Per a fer-ho, es seguiran els passos que es descriuen a continuació:

- Pas 1. Generem una parella de claus per a l'actor (ja sigui metge o pacient) utilitzant el guió de seqüències "generarClaus". En aquest cas, podem establir com a 1024 bits la longitud necessària per a les claus.

- Pas 2. Generem una petició de certificat a la CA a partir de les claus generades en el punt anterior utilitzant el guió "generaPeticioCertificat".

- Pas 3. L'autoritat de certificació generarà el certificat a través del guió "generaCertificat". Aquest utilitza un fitxer de configuració anomenat "openssl.cnf".

- Pas 4. Finalment, generem l'arxiu PKCS12 a partir del guió "generaPKCS12" emmagatzemant la parella de claus creades, el certificat i el certificat de la mateixa autoritat de certificació

A continuació, el codi dels guions anteriors, executables en un GNU/Linux sense problemes:

generarClaus.sh

```
#!/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
```

generaPeticioCertificat.sh

```
#!/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
```

generaCertificat.sh

```
#!/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
```

generaPKCS12.sh

```
#!/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
```

En el capítol 8, corresponent al “Joc de Proves”, en el primer apartat, s'explica amb detall com utilitzar aquests guions de seqüències per a generar les parelles de claus, els certificats i els fitxers P12 per a cadascun dels actors i usuaris del sistema.

Introducció a IAIK

Per a implementar tots els protocols criptogràfics emprarem la biblioteca criptogràfica IAIK.

IAIK Java Cryptography Extension (IAIK-JCE) és un conjunt d'APIS (Application Programming Interfaces) i implementacions de funcionalitats criptogràfiques que complementen les funcionalitats de seguretats bàsiques del JDK de Java. Conté, per exemple, funcions de has, sistemes per a xifrat simètric, asimètric, així com gestió de claus i certificats.

Passos per a instal·lar IAIK.

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

- Descarregar la versió última del JDK de SUN i instal·lar-lo. Cal vigilar que no s'utilitzi la màquina virtual de MS al compilar o executar.
- Descarregar l'última versió de IAIK. Encara que cal registrar-se, no suposa cap cost si es tracta per a projectes de recerca i/o de caire educatiu. Obtenim el fitxer iaik_jec_full.jar que és la versió completa signada, enllaç iaik-jce.
- Descarregar les polítiques de seguretat de Java que permeten emprar qualsevol longitud de clau, Java Cryptography Extension (JCE) Unlimited Strength Jurisdiccion Policy Files 5.0 RC.
- (Windows) Copiar l'arxiu iaik_jce_full.jar als directoris.
  - c:\Archivos de Programa\Java\jdk1.5.0\jre\lib\ext.
  - c:\Archivos de Programa\Java\jre1.5.0\lib\ext.
- (Linux) Copiar l'arxiu iaik_jce_full.jar al directori:
  - $JAVA_HOME/jre/lib/ext.
- (Windows) dins de l'arxiu jce_policy-1.5.0-beta2.zip hi ha els arxius:

  - local_policy.jar.
  - US_export_policy.jar

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

- (Linux) Dins de l'arxiu jce_policy-1.5.0-beta2.zip hi ha els arxius:
  - local_policy.jar.
  - US_export_policy.jar.
- Copiar-los a: $JAVA_HOME/jre/lib/security.

- Compilar i executar.

Capítol 3. Protocols criptogràfics.

Requisits de seguretat.

A continuació es descriurà l'esquema criptogràfic implementat per a la gestió segura dels historials clínics dels pacients.

El sistema permet els següents casos d'ús:

- Autenticació per part dels usuaris (metges i/o pacients) en el sistema.
- Consulta d'una història clínica per part de metges o pacients.
- Consulta del llistat de pacients per part d'un metge.
- Inserció d'un nou apunt per part d'un metge en la història clínica d'un pacient.

Notació emprada en els protocols.

La notació que farem servir per als protocols criptogràfics serà la següent:

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

Autenticació en el sistema.

Aquesta operació permet a un metge o a un pacient autenticar-se en el sistema. A continuació, presentem el protocol d'autenticació (protocol de Needham-Schroeder) per al cas d'un usuari Pi i el gestor del sistema G.

1. *Pi realitza les operacions següents:*
   1. *Obtenir un valor de forma aleatòria N_i.*
   2. *Xifrar N_i i Ipi amb la clau pública G, E_G(Ni, Ipi). Ipi és l'identificador de P_i.*
   3. *Enviar $E_G(N_i, Ip_i)$ a G;*
2. *G realitza les operacions següents:*
   1. *Desxifrar $E_G(N_i, Ip_i)$ amb Sg i obtenir Ni i Ipi.*
   2. *Obtenir el certificat de Pi amb Ipi. A partir del certificat obtindrà Ppi.*
   3. *Obtenir un valor de forma aleatòria, Ng.*
   4. *Xifrar Ni, Ng, Ig amb la clau pública Ppi de Pi, Epi(Ni, Ng, Ig).*

```
            5. Enviar Epi (Ni, Ng, Ig) a Pi.

        3. Pi realitza les operacions següents:
            1. Desxifrar Epi(Ni, Ng, Ig) amb la clau privada Spi i obtenir Ng, Ni i
                Ig.
            2. Xifrar Ng amb la clau pública Pg de G, Eg(Ng).
            3. Enviar Eg(Ng) a G;

        4. G realitza les operacions següents:
            1. Desxifrar Eg(Ng) amb la clau privada Sg i obtenir N'g.
            2. Si N'g == Ng, G i Pi, estan autenticats bilateralment.
```

Consulta d'un historial.

Aquesta operació permet a un metge obtenir l'historial clínic d'un determinat pacient o un pacient obtenir el seu propi historial.

Per a la consulta d'un historial s'executen els passos següents:

- Pas 1. L'usuari (metge o pacient) s'autentica amb el sistema.
- Pas 2. L'usuari envia al gestor l'operació desitjada(“ConsultaHistorial”).
- Pas 3. El gestor, si l'usuari s'autentica correctament, obté l'historial clínic del pacient demanat, el desxifra amb la seva clau pública, el xifra amb la clau pública de l'usuari i li envia.
- Pas 4. L'usuari desxifrarà l'historial clínic rebut i validarà, per a cada entrada, els seus números de seqüència, les signatures digitals del Gestor i també del metge.

```
       Protocol 1

   1.  U realitza les operacions següents:

       (a) Executar el Procedure 1 amb la clau pública Pg i obtenir PG[Ni, Id_usuariu];
       (b) Enviar Pg[Ni, Id_usuariu] a G;

   2.  G realitza les operacions següents:

       (a) Executar el Procedure 2 amb PG[Ni, Id_usuariu] i obtenir PU[Ni, NG,
Id_usuarig];
       (b) Enviar Pu[Ni, Ng, Id_usuarig] a U;

   3.  U realitza les operacions següents:

       (a) Desxifrar Pu[Ni, Ng, Id_usuarig] amb la clau privada Su, i obtenir Ng, N'i i
       Id_usuariG;

       (b) Si Ni' == Ni fer:

           i. Xifrar NG, Consulta, Id_usuari amb la clau pública PG de G, PG[NG,
           Consulta, Id_usuari]. Consulta indica que es vol consultar l'historial de
           l'usuari identificat amb Id_usuari;

           ii. Enviar PG[NG, Consulta, Id_usuari] a G;


       (c) Sinó, retornar error;


   4.  G realitza les operacions següents:
```

(a) Desxifrar PG[NG, Consulta, Id_usuari] amb la clau privada SG, i obtenir N'G, Consulta, Id_usuari;

(b) Recuperar GN de la BBDD. En el pas 4 del Procedure 2, NG i Ni han estat desats a la BBDD;

(c) Si N'G == NG fer:

i. Si (Id_usuariu == Id_usuari) o (Id_usuariu és metge i Id_usuari és un pacient de Id_usuariu) fer:

A. Executar el Procedure 3 amb Id_usuari i Pu, i obtenir Pu[H];

B. Enviar Pu[H] a U.

ii. Sinó, retornar error.

(d) Sinó, retornar error.

(e) Borrar NG i Ni de la BBDD:

5. U realitza les operacions següents:

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

(b) Mostrar H.

Procedure 1 (Pu)

1. Obtenir un valor de forma aleatòria, Ni;
2. Xifrar Ni i Id_usuariu amb la clau pública de G, Pg[Ni, Id_usuariu];
3. Enviar Pg[Ni, Id_usuariu] a G;

Procedure 2 (Pg(Ni, Id_usuariu))

1. Desxifrar Pg[Ni, Id_usuariu] amb SG, i obtenir; Ni i Idusuariu;
2. Obtenir el certificat U a partir de Id_usuariu. Suposem que el sistema disposa d'una Base de dades (BD) on per cada Id_usuari trobem el seu certificat corresponent. A partir del certificat es pot obtenir la clau pública Pu;
3. Obtenir un valor de forma aleatòria, Ng;
4. Guardar a la BD els valors Ni i Ng associats amb U;
5. Xifrar Ni, Ng, Id_usuariG, amb la clau pública Pu de U, Pu[Ni, Ng, Id_usuariG].

Procedure 3 (id_usuari, Pu)

1. Buscar l'historial H corresponent a id_usuari;
2. Desxifrar la part de H que està xifrada utilitzant la clau privada SG de G.
3. Xifrar H amb la clau pública Pu, Pu[H].
4. Retornar Pu[H].

Procedure 4

1. Desxifrar Pu[H] amb la clau privada Su de U, Su[Pu[H]];
2. Per cada entrada de l'historia H que està signada fer:
   (a) Verificar la signatura digital de M;
   (b) Verificar la signatura digital de G;
   (c) Verificar la seqüència;
3. Retornar H.

Consulta dels pacients assignats a un metge.

Aquest protocol implementa l'opció, per als metges, d'obtenir un llistat dels pacients que té assignats.

Els passos seran els següents:

- Pas 1. El metge s'autentica amb el sistema.
- Pas 2. El metge envia al gestor l'operació desitjada (“LlistatPacients”).
- Pas 3. El gestor, si el metge s'ha autenticat correctament, generarà un llistat dels pacients assignats al metge, signant cadascun d'ells amb la seva clau privada i xifrant després les dades amb la clau pública del metge.
- Pas 4. El metge rep el llistat que li tramet el gestor en el pas anterior, es desxifra utilitzant la seva clau privada i valida les signatures.

Protocol 2

1. U realitza les operacions següents:

   (a) Executar el procedure 1 amb la clau pública Pu, i obtenir Pg[Ni, Id_usuariu];
   (b) Enviar Pg[Ni, Id_usuariu] a G;

2. G realitza les operacions següents:

   (a) Executar el Procedure 2 amb PG[Ni, Id_usuariu], i obtenir Pu[Ni, Ng, Id_usuariG];

   (b) Enviar Pu[Ni, NG, Id_usuariG] a U;

3. U realitza les operacions següents:

   (a) Desxifrar Pu[Ni, Ng, Id_usuariG] amb la clau privada Su, i obtenir Ng, N'i i Id_usuariG;
   (b) Si Ni' = Ni fer:

   i. Xifrar NG i Llista_pacients amb la clau pública PG de G, PG[NG, Llista_pacients]. Llista_pacients indica que es vol un llistat dels identificadors d'usuari corresponents als pacients del metge identificat amb Id_usuariu;
   ii. Enviar PG[NG, Llista_pacients] a G;

   (c) Sinó, retornar error;

4. G realitza les operacions següents:

   (a) Desxifrar PG[NG, Llista_pacients] amb la clau privada SG, i obtenir N'g i Llista_pacients;
   (b) Recuperar NG de la BBDD. En el pas 4 del Procedure 4 NG i Ni han estat guardats a la BBDD;

   (c) Si N'G == Ng fer:

   i. Si Id_usuariu és metge fer:

   A. Executar el Procedure 5 amb Id_usuariu i Pu, i obtenir Pu[{Id_usuariu1,..., Id_usuarin} , SG[{Id_usuariu1, ... , Id_usuarin}]];
   B. Enviar a U Pu[{Id_usuari1, ... , Id_usuarin}, Sg[{Id_usuari1, ... , Id_usuarin}]].

(d) Sinó, retornar error;

(e) Borrar Ng i Ni de la BBDD.

5. U realitza les operacions següents:

(a) Executar el Procedure 6 amb Pu[{Id_usuari1, ..., Id_usuarin}, Sg[{Id_usuari1, ..., Id_usuarin}]], i obtenir {Id_usuari1, ..., Id_usuarin};

(b) Mostrar {Id_usuari1, ..., Id_usuarin}.

Procedure 5 (Id_usuari, Pu)

1. Cercar a la BBDD els pacients assignats al metge Id_usuari, obtenint {Id_usuari1, ..., Id_usuarin};
2. Signar {Id_usuari1,..., Id_usuarin} amb la clau privada SG de G, SG[{Id_usuari1,...., Id_usuarin}];
3. Xifrar {Id_usuari1, ... , Id_usuarin} i SG[{Id_usuari1, ... , Id_usuarin}] amb la clau pública de Id_usuari, Pu, Pu[{Id_usuari1,...., Id_usuarin} , SG[{Id_usuari1,...., Id_usuarin}]].
4. Retornar Pu [{Id_usuari1,..., Id_usuarin}, SG[{Id_usuari1,....,Id_usuarin}]].

Procedure 6 (Pu({Id_usuari1, ..., Id_usuarin}, SG({Id_usuari1,...., Id_usuarin})))

1. Desxifrar Pu[Sg[{Id_usuari1, ... , Id_usuarin}]] amb la clau privada Su de U, Su[Pu[SG[{Id_usuari1, ... , Id_usuarin}]]] i obtenir SG[{Id_usuari1,..., Id_usuarin}];
2. Verificar la signatura digital SG[{Id_usuari1, ... , Id_usuarin}] amb la clau pública PG de G;
3. Si la verificació anterior és correcta retornar {Id_usuari1, ... , Id_usuarin}.

Inserció de dades a l'historial mèdic.

Aquest protocol implementa l'opció, per als metges, d'introduir un nou apunt en l'historial clínic d'un determinat pacient.

Els passos per a fer-ho son els següents:

- Pas 1. El metge s'autentica amb el sistema.
- Pas 2. El metge signa l'apunt de la visita, que conté, almenys, l'identificador del pacient, el codi CIE9 amb el qual es relaciona el nou apunt i el seu apunt pròpiament dit.
- Pas 3. El metge envia al gestor l'apunt signat en el pas anterior.
- Pas 4. El gestor rep la informació que el metge li ha tramés en el punt anterior, la desxifra i s'assegura que qui li ha enviat la informació és realment un metge i que aquest apunt està relacionat amb un pacient que realment pertany a aquest metge.
- Pas 5. Si el pas anterior s'efectua correctament, el gestor xifrarà la informació i l'introduirà a la base de dades.

Protocol 3

1. M realitza les operacions següents:

(a) Executar el Procedure 1 amb la clau pública P_M, i obtenir $PG[N_i, Id\_usuariM]$;

(b) Enviar $PG[Ni, Id\_usuariM]$ a G;

2. G realitza les operacions següents:

(a) Executar el Procedure 2 amb $PG[Ni, Id\_usuariM]$, i obtenir $P_u[N_i,NG, Id\_usuariG]$;
(b) Enviar $PM[N_i,NG, Id\_usuari_G]$ a M;

3. M realitza les operacions següents:

(a) Desxifrar PM[Ni, Ng, Id_usuariG] amb la clau privada SM, i obtenir NG, N'i i Id_usuariG;
(b) Si Ni' == Ni fer:

i. Obtenir les dades de la visita V. La visita hauria d'incloure com a mínim Id_usuarip;
ii. Signar V amb la clau privada SM de M, SM[V];
iii. Xifrar NG, V i SM[V] amb la clau pública PG de G, PG[NG, Inserir_visita, V, SM[V]]. Inserir_visita indica que es vol afegir V a l'historial del pacient P;
iv. Enviar PG[NG, Inserir_visita, V, SM[V]] a G;

(c) Sinó, retornar error.

4. G realitza les operacions següents:

(a) Desxifrar PG[NG, Inserir_visita, V, SM[V]] amb la clau privada SG, i obtenir N'G, Inserir_visita, V i SM[V];
(b) Recuperar NG de la BBDD. En el pas 4 del Procedure 4 Ng i Ni han estat guardats a la BBDD.

(c) Si N'g = Ng fer:

i. Obtenir Id_usuarip a partir de V;

ii. Verificar que ld_usuariM és metge;

iii. Verificar que Id_usuarip és un pacient assignat a Id_usuariM;

iv. Si les verificacions anteriors són correctes fer:
A. Verificar la signatura digital SM[V] amb la clau pública PM;

B. Obtenir l'instant de temps actual T;

C. Obtenir el número de sèrie X de la última visita de l'historial H del pacient Id_usuarip;

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

E. Signar V, SM[V], T, X+1, amb la clau privada SG de G, SG[V,SM[V],

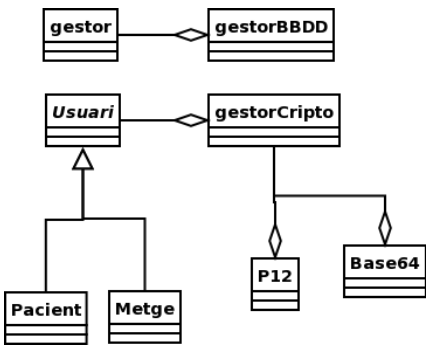
```
                                    T,                                          X+1];
                    F. Xifrar V i SM[V] amb la clau pública SG de G, SG[X+1,
Id_usuarip];

                    H. Guardar a la BBDD PG[V, SM[V]], X+1, T, SG[V,SM[V], T, X+1] i
SG[X+1, Id_usuarip];

              v. Sinó, retornar error.


       (d) Sinó, retornar error.
```

Diagrama de classes.

Per a la implementació dels casos d'ús anteriors s'han generat les classes següents:

- Usuari.java: Classe abstracta amb mètodes comuns a metges i pacients.
- Metge.java: Classe que hereda de Usuari i implementa les accions que pot efectuar un metge.
- Pacient.java: Classe que hereda de Usuari i implementa les acciones que pot efectuar un metge.
- gestor.java: Classe que implementa les accions de l'usuari del sistema.
- gestorBBDD.java: Classe que implementa procediments comuns per a interactuar amb la base de dades.
- gestorCripto.java: Classe que implementa procediments comuns per a efectuar les operacions criptogràfiques (xifrar, desxifrar, signar, verificar signatures digitals, entre d'altres).
- P12.java: Classe que representa un PKCS i l'empra la classe gestorCripto.java.
- Base64.java: Classe amb mètodes per a codificar i descodificar en base64. S'ha extret aquesta classe des de http://iharder.net/base64 i té una llicència de programari lliure.

A continuació, el diagrama de classes en format UML:



Capítol 4. Representació de les dades: XML.

Introducció

Per a traspassar informació entre els diferents agents del sistema (metges, pacients i gestor) s'utilitzaran documents en format XML.

XML és l'acrònim, en anglès, de eXtensible Markup Language (“llenguatge de marques extensible”) i és un metallenguatge extensible d'etiquetes desenvolupat per al World Wide Web Consortium (W3C). Es tracta d'una simplificació i adaptació del SGML i permet definir la gramàtica de llenguatges específics (de la mateixa manera que l'HTML és també un llenguatge definit per l'SGML). Per tant, l'XML no és realment un llenguatge en particular, sinó una manera de definir llenguatges per a diferents necessitats. Alguns d'aquests llenguatges que usen XML per a la seva definició son l'XHTML, SVG, MathML. L'XML no ha nascut solament per a la seva aplicació a Internet, sinó que es proposa com un estàndard per a l'intercanvi d'informació estructurada entre diferents plataformes. Es pot utilitzar en bases de dades, editors de text, fulles de càlcul i gairebé qualsevol altra cosa.

Per a treballar en documents XML s'utilitzarà la llibreria pública JDOM.

Definició dels documents en XML.

A continuació, un llistat amb les especificacions de tots els documents en XML que s'han emprat per a transmetre informació entre els diferents actors.

Procedure 1

```
<?xml version="1.0" encoding="UTF-8"?>
       <Pfchistorials>
              <Procedure1></Procedure1>
       </Pfchistorials>
```

L'element “Procedure1” conté la cadena xifrada amb la clau pública del Gestor G següent:

Ni-Id_usuariu

On Ni és un valor obtingut de forma aleatòria i Id_usuariu és l'identificador de l'usuari que inicia l'autentificació.

El DTD d'aquest document serà el següent:

```
<?xml version="1.0"?>
<!DOCTYPE Procedure1 [
  <!ELEMENT Pfchistorials        (Procedure1)>
  <!ELEMENT Procedure1            (#PCDATA)>
]>
```

Procedure 2

```
<?xml version="1.0" encoding="UTF-8"?>
       <Pfchistorials>
              <Procedure2></Procedure2>
       </Pfchistorials>
```

L'element “Procedure2” conté la cadena xifrada amb la clau pública de l'usuari U següent:

Ni-Ng-Id_usuarig

On Ng és també un valor obtingut de forma aleatòria.

El DTD d'aquest document serà el següent:

```
<?xml version="1.0"?>
<!DOCTYPE Procedure2 [
  <!ELEMENT Pfchistorials        (Procedure1)>
  <!ELEMENT Procedure1            (#PCDATA)>
]>
```

Pas 3

```
<?xml version="1.0" encoding="UTF-8"?>
               <Pfchistorials>
                      <Pas3>
                      </Pas3>
               </Pfchistorials>
```

L'element “Pas3” conté la cadena xifrada amb la clau pública de l'usuari U següent:

Ng-Operació-Id_usuari.

El DTD d'aquest document serà el següent:

```
<?xml version="1.0"?>
<!DOCTYPE Pas3 [
  <!ELEMENT Pfchistorials        (Pas3)>
  <!ELEMENT Pas3            (#PCDATA)>
    ]>
```

HistoriaClínica

El següent esquema de document en XML representa la història clínica d'un pacient:

```
<?xml version="1.0" encoding="UTF-8"?>
          <Pfchistorials>
                 <HistoriaClinica>
                        <Diagnostic>
                               <Pgvsmv></Pgvsmv>
                               <Sgvsmvtx></Sgvsmvtx>
                               <Sgxidusuarip></Sgxidusuarip>
                               <X></X>
                               <T></T>
                        </Diagnostic>
                 </HistoriaClinica>
                 <Pacient></Pacient>
        </Pfchistorials>
```

L'element Pacient conté l'identificador del pacient del qual s'està trametent la història clínica.

L'element “HistoriaClinica” conté la història clínica del pacient. Aquest element pot contenir diversos elements “Diagnostic” amb els elements següents:

- Pgvsmv: Conté la cadena següent: “codiCie-apunt-signatura_de_l'apunt” xifrada amb la clau pública del destinatari.
- Sgvsmvtx: Conté la cadena següent: “nhc_pacient-codiCie-apunt-signatura-tActual-numSerieX” xifrada amb la clau pública del destinatari.
- Sgxidusuarip: Conté la cadena “identificador_pacient-X”, signada amb la clau privada del Gestor i xifrada amb la clau pública del destinatari.
- X: Número de sèrie de l'apunt.
- T: Conté una cadena que representa el temps actual.

Llistat de malalties segons el codi CIE-9

A continuació, l'esquema XML que s'emprarà per a que el gestor retorni un llistat de les malalties i de les seves codificacions segons la codificació del CIE-9.

```
    <?xml version="1.0" encoding="UTF-8"?>
<Pfchistorials>
        <LlistatCIE9>
                <CodiCIE>
                        <Codi>521.2</Codi>
                        <Descripcio>
                        ABRASIO DENTAL; ABRASIO: PER DENTIFRIC,H
                        </Descripcio>
                </CodiCIE>

                <CodiCIE>
                        <Codi>910.1</Codi>
                        <Descripcio>
                        ABRASIO O CREMADA PER FRIC.CARA[ULL NO],
                        </Descripcio>
                </CodiCIE>
```

Document per a les dades administratives d'un pacient

Aquest document en XML conté les dades administratives d'un determinat pacient identificat per el seu número d'història clínica (NHC).

```
<?xml version="1.0" encoding="UTF-8"?>
<Pfchistorials>
    <Dades_Admin_Pacient>
            <nhc></nhc>
            <dni></dni>
            <tis></tis>
            <nss></nss>
            <nom></nom>
            <cognom1></cognom1>
            <cognom2></cognom2>
            <correue></correue>
    </Dades_Admin_Pacient>
</Pfchistorials>
```

El contingut de tots els nodes va xifrat amb la clau pública del destinatari, de manera que solament aquest hauria de poder llegir el contingut d'aquest document.

Llistat de Pacients

A continuació, l'esquema XML que s'emprarà per a que el gestor retorni als metges un llistat amb els seus pacients assignats:

```
<?xml version="1.0" encoding="UTF-8"?>
           <Pfchistorials>
                  <Llista_pacients>
                         <Pacient>
                         </Pacient>
                         <Signatura>
                         </Signatura>
                  </Llista_pacients>
           </Pfchistorials>
```

On l'element Pacient contindrà el nom del pacient xifrat amb la clau pública del destinatari del document.

L'element Signatura contindrà la signatura del nom del pacient creada a partir de la clau privada del Gestor.

Capítol 5. Comunicació entre agents: RMI.

Introducció.

La comunicació dels diferents actors i components del sistema és una part essencial del projecte. Tants els pacients com els professionals mèdics (metges) han de poder accedir de forma remota als historials clínics de forma remota sobre una xarxa de comunicacions. A més, els professionals mèdics han de poder visualitzar un llistat dels pacients que tenen assignats i també afegir nous apunts en les històries clíniques dels seus pacients. Evidentment, la base de dades tant de pacients, metges, i dels seus historials clínics està centralitzada i protegida mitjançant xifrat.

La comunicació dels diferents components del sistema tradicionalment suposaria el disseny d'un protocol o mecanisme de comunicació. Per evitar la sobrecàrrega de feina, i donat que la part essencial és l'esquema criptogràfic es va optar per la utilització de la tecnologia RMI.

RMI correspon a l'acrònim de *Remote Method Invocation*. Java incorpora aquesta tecnologia en la seva API estàndard. RMI consta d'un servidor on s'executen diferents instàncies de les classes servidores que es necessiten. Les aplicacions que volen emprar els mètodes remots únicament necessiten saber la interfície del servidor, és a dir, els mètodes que ofereix la classe que està en servidor. La implementació d'aquesta interfície restarà oculta i el client no arriba mai a saber que és el que s'està executant. En el projecte s'ha utilitzat RMI per a comunicar els aplicatius de pacients i metges amb el gestor del sistema. Tota la informació que s'envia entre aquests diferents actors es realitza a través d'aquesta tecnologia RMI. La utilització de RMI ha reduït notablement el temps de desenvolupament, en contraposició a haver implementat un protocol des de zero emprant sockets o webservices.

Mètodes accessibles públicament al gestor.

Els mètodes als quals es poden accedir de forma remota en el gestor del sistema es defineixen a través d'una interfície, continguda en el fitxer GestorInterRemot.java.

En aquest cas s'han definit els mètodes següents per a que siguin públicament accessibles:

```
Document_XML procedure2 ( Document_XML)
booleà pas 4 (Document_XML)
booleà verPacMetge (Pacient, metge)
Document_XML procedure3 ( Pacient, Metge)
Document_XML procedure5 (id_usuari)
booleà pas4InserirVisita (Apunt)
Document_XML retornaLlistatCIE()
Document_XML getDadesAdminPacient( us_desti, us_origen)
```

A continuació, una explicació més detallada sobre què implementa cada mètode:

- Document procedure2(org.jdom.Document a)

  Aquest procediment, que s'executa en el gestor, implementa la part del sistema d'autenticació del protocol de Needham-Schroeder entre aquesta entitat i els pacients i/o metges.

- boolean pas4(org.jdom.Document c)

  Aquest procediment implementa en el gestor el pas 4 del protocol 2.

- boolean verPacMetge(String pacient , String metge)

  Aquest procediment determina si el pacient “pacient” està assignat, o no, a un determinat metge. El mètode retornarà “cert” si és el cas o “fals” en cas contrari.

- Document procedure3 (String pacient , String metge)

  El procedure3 és l'encarregat de retornar la història clínica d'un determinat pacient a partir de l'identificador d'usuari “Pacient” i l'identificador del “Metge” (o propi pacient) que demana l'historial.

- Document procedure5 (String id_usuari)

  Aquest procediment implementat en el servidor és l'encarregat de retornar un llistat de pacients assignats al metge en qüestió, identificat pel paràmetre “id_usuari”.

- boolean pas4InserirVisita (String a)

  Aquest procediment implementa el pas 4 del protocol 3.

- Document retornaLlistatCIE ()

  Aquest mètode retornarà un document en format XML amb un llistat de malalties amb el nom o descripció de la malaltia i també amb el codi CIE-9 associat a la malaltia.

  Aquest document no viatjarà xifrat i és que no conté informació sensible, sinó pública.

- Document getDadesAdminPacient (String id_usuari_vull , String id_usuari_peticio)

  El mètode getDadesAdministratives s'encarrega de retornar les dades administratives d'un determinat pacient a partir del seu identificador.

Capítol 6. Gestor de base de dades.

Introducció

Per a que aquest sistema descrit funcioni correctament és necessari un sistema gestor de base de dades per a emmagatzemar tota la informació. En concret, els certificats d'usuari, les històries clíniques dels pacients, les dades administratives dels pacients, entre d'altres.

Les dades del sistema s'emmagatzemaran en un sistema gestor de base de dades. El servidor escollir serà MySQL, disponible com a programari lliure.

Comentar que, malgrat a la documentació es proposa realitzar la identificació dels usuaris a través del seu DNI (Document Nacional d'identitat) o a través del seu número de seguretat social (N.S.S), en aquest projecte, per a identificar els pacients emprarem un número, que denominarem NHC (Número d'història Clínica).

S'ha escollit aquest sistema com a mètode per a identificar pacients a causa del fet que en un centre sanitari poden passar-hi persones que no disposen de cap d'aquests dos números, però que, en canvi, s'han de visitar. Pot ser el cas, per exemple d'immigrants o persones desplaçades (que, malgrat tot, poden tenir NIE o Passaport o no tenir res) .

Com a identificador dels professionals sanitaris s'ha decidit emprar el seu número de col·legiat.

Comentar finalment que el fitxer complet amb totes les taules i dades d'inicialització està en la carpeta src/pfchistorials.sql.

Descripció de les taules

A continuació, la descripció de cada una de les taules. Cada camp que actua com a clau primària, apareix de forma subratllada.

Taula dtcertificats: Conté els certificats dels usuaris.

idusuari: Conté l'identificador de l'usuari
certificat: Conté el certificat de l'usuari

Taula dtcodCIE9: Conté una relació de les malalties codificades segons el CIE-9.

codicie: Conté el codi CIE
descrip: Conté la descripció del codi CIE.

Taula dtdiagnostics: Aquesta taula conté els diagnòstics per a cada pacient.

Camp nhcpacient: Conté l'identificador de l'usuari.
numseriex: Conté el número de sèrie X.
instanttemps: Conté l'instant de temps.
pgvsmv:
Sgvsmvtx:
sgxidusuarip:

Taula dtmetges: Aquesta taula conté els metges disponibles en el sistema.

numcolegiat: Conté el número de col·legiat del metge.

nom: Conté el nom del metge.
cognom1: Conté el primer cognom del metge.
cognom2: Conté el segon cognom del metge.
correue: Conté el correu electrònic del metge.
telf: Conté el telèfon de contacte del metge.

Taula dtpacients: Aquesta taula conté la informació dels pacients del sistema.

nhc: Conté l'identificador del pacient (acrònim de número d'història clínica).
Dni: Conté un document d'identificació nacional.
Tis: Conté el número de la targeta sanitària.
Nss: Conté el número de la seguretat social.
Nom: Conté el nom del pacient.
Cognom1: Conté el primer cognom del pacient.
Cognom2: Conté el segon cognom del pacient.
Correue: Conté el correu electrònic del pacient.
Telf: Conté el telèfon de contacta amb el pacient.
Direccio: Conté l'adreça de residència habitual del pacient.
Codipostal: Conté el codi postal de la població de residència habitual del pacient.
Poblacio: Conté el nom de la població de residència habitual del pacient.
Metgeassignat: Conté el número de col·legiat del metge que té assignat.
Datanaixement: Conté la data de naixement del pacient.
Sexe: Conté el sexe del pacient.

Taula dtsessionsges: Conté els números aleatoris generats pels usuaris i el gestor durant l'autenticació.

Nhc: Identificador del pacient.
Ni: Ni.
Ng: Ng.

Diagrama entitat-relació

El diagrama entitat-relació de les taules comentades anteriorment és el següent:

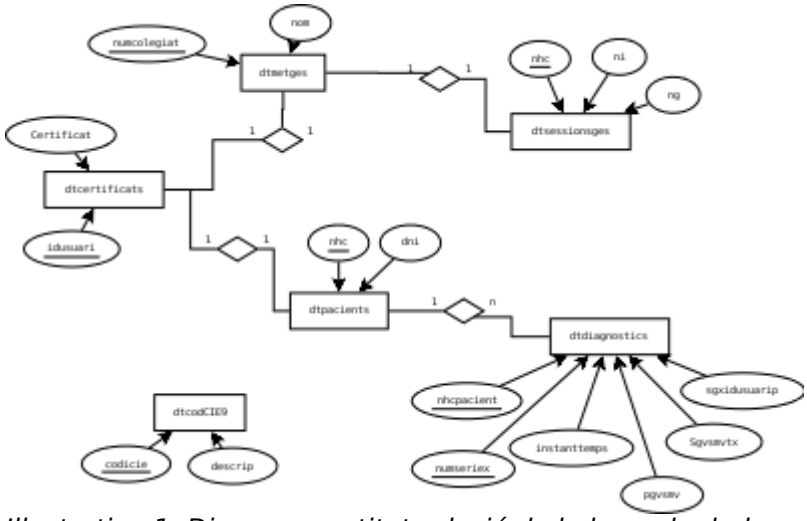


Illustration 1: Diagrama entitat-relació de la base de dades

Capítol 7. Interfície gràfica.

Introducció

La darrera part de la implementació era el disseny de les interfícies d'usuari: El sistema per a que els diferents actors del sistema puguin desenvolupar i executar totes les operacions anteriors de la forma més còmoda i intuïtiva possible.

El disseny de les interfícies d'usuari s'han realitzat amb el programari NetBeans. Malgrat es varen avaluar altres alternatives (ex. Eclipse), em semblà que estava més còmode amb NetBeans, em va semblà que era igual d'intuïtiu i a més, es tracta de programari lliure també.

NetBeans és pot descarregar gratuïtament a través de l'adreça: http://www.netbeans.org.

En la carpeta /project/ s'han deixat els fitxers en XML amb l'extensió .form que representen les interfícies d'usuari que es presentaran a continuació.

Interfície d'usuari per als clients.

La primera interfície gràfica que s'ha dissenyat és per als diferents clients del sistema: Pacients i professionals sanitaris.

La primera pantalla que apareix és per a autenticar l'usuari en el sistema i per a determinar si es tracta d'un pacient o un professional sanitari. A més del fitxer P12 i la contrasenya per a identificar l'usuari, serà necessari definir també la direcció IP de la màquina on s'executa el gestor i el seu port RMI.

Aquesta interfície està implementada en el fitxer pfcGuiLogin.form.

Illustration 2: Pantalla d'entrada

En funció de si l'usuari que s'ha introduït és un pacient o un professional sanitari, apareixerà una interfície o una altra.

A continuació, la interfície d'usuari per als pacients:

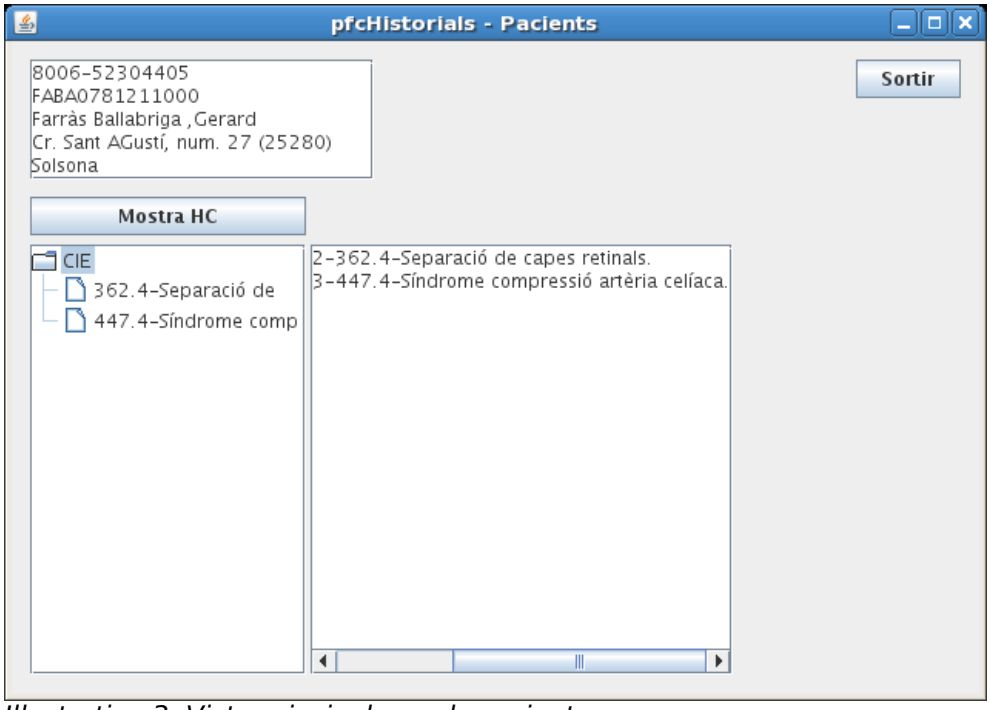


Illustration 3: Vista principal per als pacients

Com podem observar en la il·lustració anterior, aquesta pantalla mostra la informació següent:

- En la part superioresquerra un quadre amb les dades de tipus administratiu del pacient. En concret: Número d'història clínica del pacient, DNI, TIS, nom i cognoms i adreça postal.
- En la part esquerra un arbre amb el títol i codi CIE-9 de tots els episodis oberts.
- En la part dreta un quadre on mostra l'apunt en concret per a cadascun dels diferents episodis oberts.

Els dos botons que apareixen en el formulari (“Mostra HC” i “Sortir”) serveixen, respectivament, per a mostrar la història clínica i per a sortir del programa.

Aquesta interfície està implementada en el fitxer pfcHistorialGuiPacient.form.

A continuació, la interfície d'usuari per als professionals sanitaris:

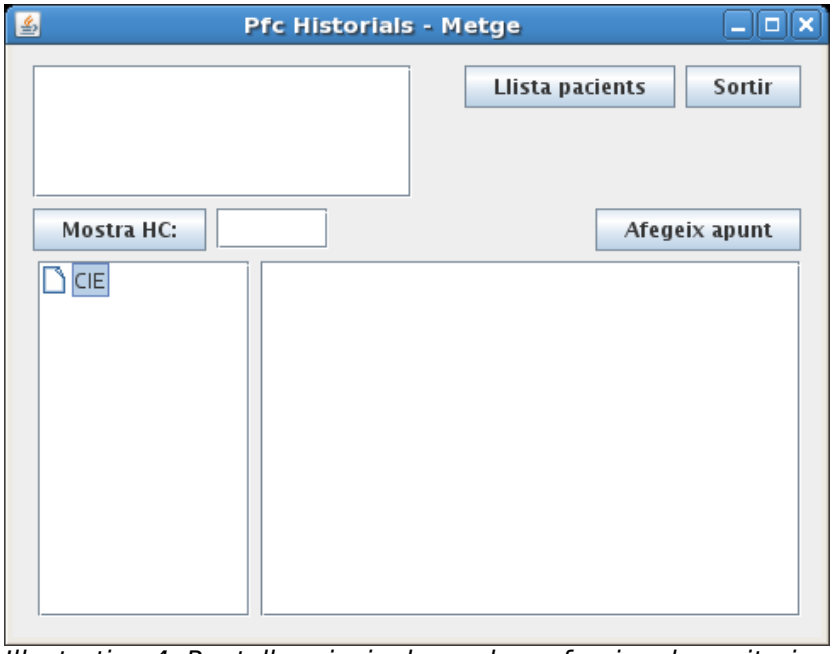


Illustration 4: Pantalla principal per als professionals sanitaris

Aquesta pantalla està implementada en el fitxer pfcHistorialGuiMetge.form.

El botó “Llista Pacients” obre un formulari on apareix un llistat amb els noms i els identificadors dels seus pacients assignats:

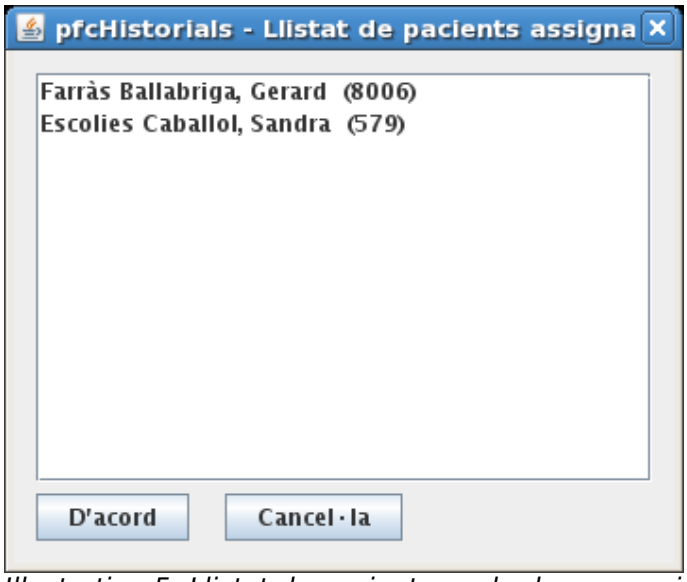


Illustration 5: Llistat de pacients, amb el seu nom i identificador

Aquí és possible seleccionar a un dels pacients per a poder visualitzar a continuació la seva història clínica.

Aquesta interfície està definida en el fitxer pfcGuiLlistaPacients.form.

També ha de ser possible afegir un nou apunt en l'historial clínic d'aquest pacient. Cada apunt s'associa a un codi CIE-9:

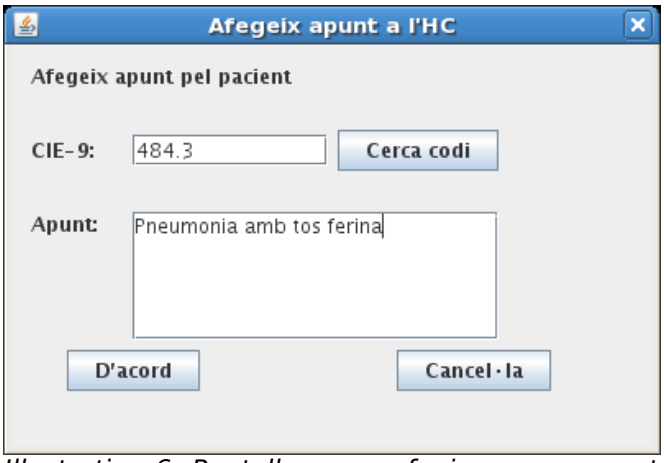


Illustration 6: Pantalla per a afegir un nou apunt en la història clínica d'un pacient.

Aquesta interfície està implementada en el fitxer pfcGuiAfegeixApunt.form.

S'ha afegit també un formulari per a cercar malalties codificades segons l'estàndard internacional CIE-9:

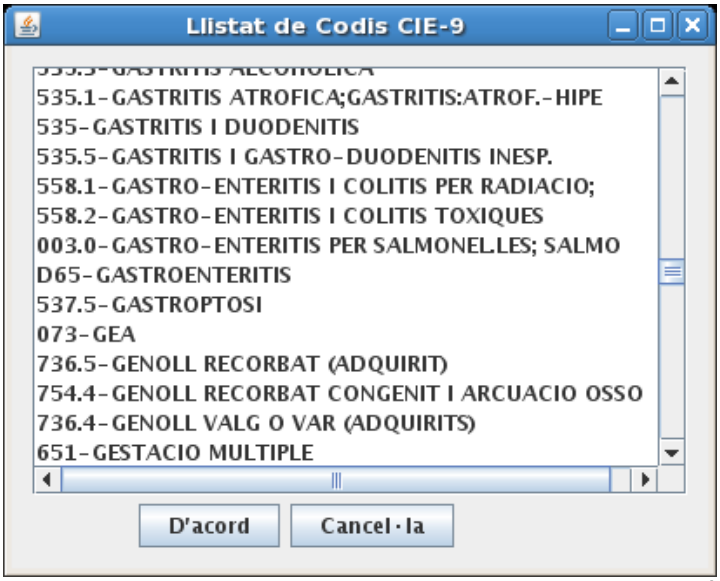


Illustration 7: Llistat de malalties segons la codificació CIE-9

Aquesta interfície està implementada en el fitxer pfcGuiCodisCIE.form.

Execució del client

Per a executar el client és necessari solament executar la instrucció:

./executaClient.sh

Que no fa més sinó executar la següent classe en Java:

$JAVA_HOME/bin/java pfcGuiLogin &

Interfície d'usuari per al gestor del sistema.

El gestor del sistema també disposa d'una interfície gràfica per a poder realitzar les seves operacions: Bàsicament per a Iniciar/aturar el servei.

També seria interessant afegir la possibilitat de crear nous usuaris (ja siguin pacients o professionals sanitaris), malgrat no s'ha disposat de més temps per a poder realitzar aquesta millora.

La interfície principal per al gestor és la següent:

Illustration 8: Pantalla principal per al gestor

A través d'aquest formulari, s'ha d'establir el port RMI on desitjarem que s'executi el servidor i també escollirem el fitxer P12 i la contrasenya per a poder identificar al Gestor.

Aquesta interfície està implementada en el fitxer pfcGuiGestor.form.

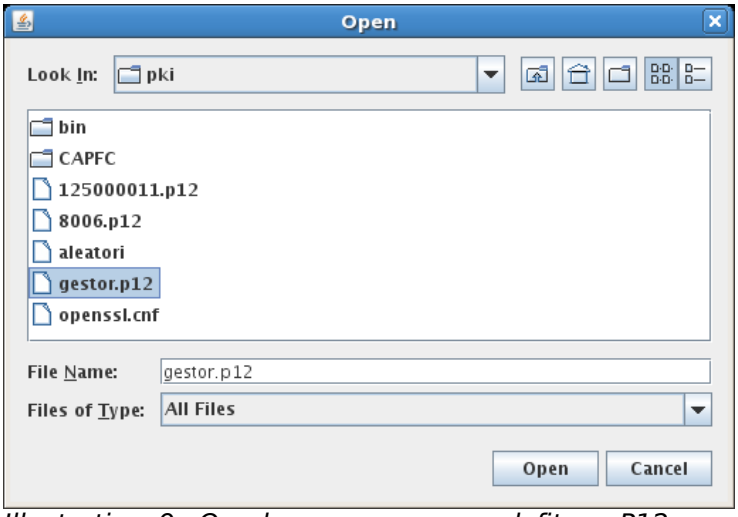


Illustration 9: Quadre per a cercar el fitxer P12 que identifica al gestor

El programa també ofereix la possibilitat d'afegir en el sistema un nou pacient. Quan es fa clic sobre el botó del formulari principal "Afegeix pacient" apareix un formulari similar al següent:

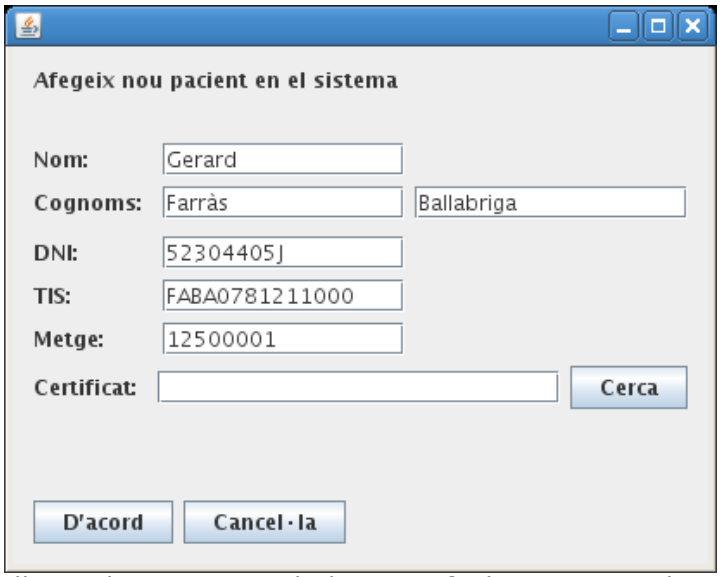


Illustration 10: Formulari per a afegir un nou pacient en el sistema.

Comentar que aquesta funcionalitat NO ha estat implementada però que es deixa indicat l'aspecte que hauria de tenir.

Aquesta interfície està implementada en el fitxer pfcGuiGestorNewPacient.form.

De manera similar als pacients, hauria d'existir un formulari per a afegir nous professionals en el sistema, però que tampoc NO ha estat implementada.

Execució del gestor

Per a executar el gestor del sistema, executem les comandes següents:

cd bin/
./executaGestor.sh port-rmi

A continuació, el guió de seqüències executaGestor.sh:

```
#!/bin/bash
echo "PFC - Historials - Execució del Gestor..."

if [ ! -n "$1" ]
then
  echo "Us: `basename $0` port-RMI (ex. `basename $0` 2099)"
  exit 0
fi

if [ ! -n "$JAVA_HOME" ]
then
  echo "Definiu la variable JAVA_HOME per a poder executar el programa gestor"
  exit 0
fi

echo "Executem Gestor en el port RMI " $1
APP_PATH="/home/gerard/pfc-historials/entrega/bin/"
$JAVA_HOME/bin/rmiregistry $1 &
$JAVA_HOME/bin/java -classpath lib/iaik_jce_full.jar:lib/jdom.jar:lib/mysql-conn
ector-java-5.0.7-bin.jar:$APP_PATH pfcGuiGestor $1
```

Capítol 8. Jocs de proves

Introducció

L'objectiu del joc de proves és donar un exemple de la configuració i l'execució completa del sistema presentat. En aquest capítol es mostra com crear els certificats dels usuaris, la configuració de la base de dades, l'execució del servidor RMI i un exemple concret d'autenticació, accés a la història clínica d'una pacient per part d'un professional sanitari, l'accés al llistat dels seus pacients i la introducció d'un nou apunt.

Generació dels certificats

El primer pas és la preparació dels arxius necessaris pels usuaris de l'aplicatiu. El primer que s'ha de preparar és el certificats autosignat de l'entitat certificadora. Per a dur a terme aquesta tasca s'empraran els arxius de comandes que s'han comentat en el capítol corresponent. Totes les contrasenyes que introduïm serà sempre la mateixa: “uoc07”.

Adjunt a aquesta memòria s'ha subministrat un directori anomenat PKI que conté l'estructura de carpetes preparada per a dur a terme la tasca següent. Es recomana ubicar-se en la carpeta “pki/bin” per a executar els guions de següencies de forma més còmoda.

Per a crear les claus per a la autoritat certificadora amb una longitud de 2048 bits i després generar un certificat autosignat farem el següent:

./generarClaus CA.key 2048

Obtindrem un fitxer anomenat CA.key que conté la parella de claus de la CA. A continuació, generarem un certificat autosignat per a la CA. Executem la comanda següent:

./generaCertificatAutosignat CA.key CA.crt 365

El paràmetre 365 es refereix al total de dies que aquest certificat serà vàlid. Durant l'execució d'aquesta comanda es demanarà, a més, que s'introdueixin totes les dades per a poder identificar la CA.

S'han utilitzat els següents:

- Country Name: ES
- State or Province Name: Lleida.
- Locality Name: Solsona.
- Organization Name: UOC.
- Organization Unit Name: UOC.
- Common Name: Entitat Certificadora.
- Email: gfarrasb@uoc.edu

Per pantalla, apareix quelcom similar al següent:

```
[gerard@Agripina bin]$ ./generaCertificatAutosignat CA.key CA.crt 365
Enter pass phrase for CA.key:
unable to load Private Key
4190:error:06065064:digital envelope routines:EVP_DecryptFinal_ex:bad
decrypt:evp_enc.c:461:
4190:error:0906A065:PEM routines:PEM_do_header:bad decrypt:pem_lib.c:425:
[gerard@Agripina bin]$ ./generaCertificatAutosignat CA.key CA.crt 365
Enter pass phrase for CA.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:ES
State or Province Name (full name) [Berkshire]:Lleida
Locality Name (eg, city) [Newbury]:Solsona
Organization Name (eg, company) [My Company Ltd]:UOC
Organizational Unit Name (eg, section) []:UOC
Common Name (eg, your name or your server's hostname) []:Entitat
Certificadora
Email Address []:gfarrasb@uoc.edu
```

Obtenim el certificat autosignat per a l'autoritat certificadora. A continuació, procedirem a generar els arxius per als usuaris. El primer serà per al gestor del sistema. Per a fer-ho, farem el següent:

Generem les claus del gestor: ./generarClaus gestor.key 1024

Seguidament, generem una petició de certificat per la CA que s'ha creat en l'apartat anterior. Emprarem la comanda següent:

./generaPeticioCertificat gestor.key gestor.csr ../openssl.cnf

Les dades que s'han emprat en aquest exemple son les següents:

- Country Name: ES
- State or Province Name: Lleida.
- Locality Name: Solsona
- Organization Name: UOC
- Organizational Unit Name: Gestors
- Common Name: Gestor

Per a no crear problemes de localització, mourem el fitxer CA.key en la carpeta pki/CAPFC/private i CA.crt a PKI/CAPFC.

Per a generar el certificat per al gestor executarem la següent comanda a través de l'arrel de la carpeta PKI:

```
[gerard@Agripina pki]$ ./bin/generaCertificat ./bin/gestor.csr ./bin/gestor.crt openssl.cnf
Using configuration from openssl.cnf
Enter pass phrase for ./CAPFC/private/CA.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'ES'
stateOrProvinceName   :PRINTABLE:'Lleida'
localityName          :PRINTABLE:'Solsona'
organizationName      :PRINTABLE:'UOC'
organizationalUnitName:PRINTABLE:'Gestors'
commonName            :PRINTABLE:'Gestor'
Certificate is to be certified until Jan  2 17:35:31 2009 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

L'últim pas és ja generar un arxiu .p12 que és el que s'emprarà des de l'aplicatiu. Per a generar-lo executarem la comanda següent:

```
./bin/generaPKCS12 ./bin/gestor.key ./bin/gestor.crt ./CAPFC/CA.crt gestor.p12
```

A continuació, generarem els mateixos fitxers per a un metge i també per a un pacient.

Per a generar el certificat per al pacient Gerard Farràs i Ballabriga, amb número d'història clínica 8006 executarem les comandes següents:

```
[gerard@Agripina bin]$ ./generarClaus 8006.key 1024
Generating RSA private key, 1024 bit long modulus
.............++++++
.............++++++
e is 65537 (0x10001)
Enter pass phrase for 579.key:
Verifying - Enter pass phrase for 579.key:

[gerard@Agripina bin]$ ./generaPeticioCertificat 8006.key 8006.csr
../openssl.cnf
Enter pass phrase for 8006.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [ES]:
State or Province Name (full name) [Catalunya]:Lleida
Locality Name (eg, city) [Barcelona]:Solsona
Organization Name (eg, company) [Universitat Oberta de Catalunya]:UOC
Organizational Unit Name (eg, section) [Consultors]:Pacients
Common Name (eg, YOUR name) []:Gerard Farràs i Ballabriga
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:


[gerard@Agripina pki]$ ./bin/generaCertificat ./bin/8006.csr ./bin/8006.crt
openssl.cnf
Using configuration from openssl.cnf
Enter pass phrase for ./CAPFC/private/CA.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'ES'
stateOrProvinceName   :PRINTABLE:'Lleida'
localityName          :PRINTABLE:'Solsona'
organizationName      :PRINTABLE:'UOC'
organizationalUnitName:PRINTABLE:'Pacients'
commonName            :T61STRING:'Gerard Farràs i Ballabriga'
Certificate is to be certified until Jan  2 18:05:42 2009 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated

[gerard@Agripina pki]$ ./bin/generaPKCS12 ./bin/8006.key ./bin/8006.crt
./CAPFC/CA.crt 8006.p12
```

Per a generar el certificat per al Dr. Josep Vilaseca, amb número de col·legiat 125000011 executarem les comandes següents:

```
[gerard@Agripina bin]$ ./generarClaus 125000011.key 1024
Generating RSA private key, 1024 bit long modulus
.....++++++
.....++++++
e is 65537 (0x10001)
Enter pass phrase for 125000011.key:
Verifying - Enter pass phrase for 125000011.key:
[gerard@Agripina bin]$ ./generaPeticioCertificat 125000011.key 125000011.csr
../openssl.cnf
Enter pass phrase for 125000011.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [ES]:
State or Province Name (full name) [Catalunya]:Lleida
Locality Name (eg, city) [Barcelona]:Solsona
Organization Name (eg, company) [Universitat Oberta de Catalunya]:UOC
Organizational Unit Name (eg, section) [Consultors]:Metges
Common Name (eg, YOUR name) []:Josep Vilaseca
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
[gerard@Agripina bin]$ ./generaPeticioCertificat 125000011.key
125000011.csr ../openssl.cnf
[gerard@Agripina bin]$ cd ..
[gerard@Agripina pki]$ ./bin/generaCertificat ./bin/125000011.csr
./bin/125000011.crt openssl.cnf
Using configuration from openssl.cnf
Enter pass phrase for ./CAPFC/private/CA.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'ES'
stateOrProvinceName   :PRINTABLE:'Lleida'
localityName          :PRINTABLE:'Solsona'
organizationName      :PRINTABLE:'UOC'
organizationalUnitName:PRINTABLE:'Metges'
commonName            :PRINTABLE:'Josep Vilaseca'
Certificate is to be certified until Jan  2 18:12:13 2009 GMT (365 days)
Sign the certificate? [y/n]:y
1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated

[gerard@Agripina pki]$ ./bin/generaPKCS12 ./bin/125000011.key
./bin/125000011.crt ./CAPFC/CA.crt 125000011.p12
```

Instal·lació i configuració de la base de dades

Per a instal·lar la base de dades es requereix un servidor de base de dades MySQL ja funcionant correctament. En aquest cas s'ha utilitzat el “MySQL-Server-5.0.27-1” sobre una màquina Fedora Core 6.

Creem la base de dades amb la comanda següent:

mysqladmin -u root -p create pfchistorials

A continuació, hi afegim les taules i el contingut d'inicialització:

mysql -u root -p pfchistorials < src/pfchistorials.sql

El contingut del fitxer pfchistorials.sql està en l'ànnex B.

D'altra banda, serà necessari un usuari en el servidor de MySQL per a connectar-se a aquesta base de dades.

És possible crear-lo a través de la comanda següent:

mysql -u root -p

mysql> GRANT ALL PRIVILEGES ON pfchistorials.* TO 'nomusuari'@'localhost' IDENTIFIED BY 'contrasenya' WITH GRANT OPTION;

Ara, simplement s'ha de configurar el fitxer cfgBBDD.txt amb les dades de connexió a a la base de dades. En la primera línia s'ha d'indicar l'anfitrió del servidor de base de dades, en la segona el nom de la base de dades, en la tercera el nom d'usuari per a connectar-se i, en la quarta, la contrasenya d'aquest usuari.

A continuació, un exemple:

```
[gerard@Agripina mysrc]$ more cfgBBDD.txt
127.0.0.1
pfchistorials
root
contrasenya
```

Inserció dels usuaris en la base de dades

Per a que els usuaris del sistema siguin funcionals és necessari que en la taula de certificats de la base de dades hi hagin els certificats dels pacients i metges del sistema.

Per a executar aquesta operació l'ideal seria fer-ho a través de la interfície gràfica del gestor, encara que no s'ha tingut temps de desenvolupar-ho.

Per a realitzar aquest pas el millor serà introduir els usuaris directament en les taules de la base de dades a través d'instruccions SQL.

Per a introduir les dades administratives d'un nou usuari:

```
INSERT INTO `dtpacients` ( `nhc` , `dni` , `tis` , `nss` , `nom` , `cognom1` , `cognom2` , `correue` , `telf` , `direccio` , `codipostal` , `poblacio` , `metgeassignat` , `datanaixement` , `sexe` )
VALUES (
'579', '78124367A', 'ESCA1770121000', NULL , 'Eva', 'Escarré', 'Cavallada', 'gfarrasb@uoc.edu', '620875013', 'Cr. Sant Agustí, num 27 2on', '25280', 'Solsona', '12500001', '21/01/1978', 'F');
```

A continuació, s'hauria d'introduir el certificat:

```
INSERT INTO `dtcertificats` ( `idusuari` , `certificat` )
VALUES ('579',
'MIIEOTCCAyGgAwIBAgIBAjANBgkqhkiG9w0BAQUFADCBjTELMAkGA1UEBhMCRVMx
DzANBgNVBAgTBkxsZWlkYTEQMA4GA1UEBxMHU29sc29uYTEMMAoGA1UEChMDVU9D
MQwwCgYDVQQLEwNVT0MxHjAcBgNVBAMTFUVudGl0YXQgQ2VydGlmaWNhZG9yYTEf
MB0GCSqGSIb3DQEJARYQZ2ZhcnJhc2JAdW9jLmVkdTAeFw0wODAxMDMxODEyMTNa
Fw0wOTAxMDIxODEyMTNaMFYxCzAJBgNVBAYTAkVTMQ8wDQYDVQQIEwZMbGVpZGEx
DDAKBgNVBAoTA1VPQzEPMA0GA1UECxMGTWV0Z2VzMRcwFQYDVQQDEw5Kb3NlcCBW aWxhc2VjYTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAuJW+ufGa0VLXWjfo lDDE6Eesw6NlkcPWI+vUwFux827gGNFmHwoGoVRrjarQsOdsHvZ1keB9l8O5azcq JFWnLwkRLGa0/x0RxK+/Y88ck3pdPNK2E+Gl6gXhECy4ZpN/QHL6E/a1ojJ+OSNJ /aKw1YJko9S8C4EWPcbIJ452rM8CAwEAAaOCAVwwggFYMAkGA1UdEwQCMAAwEQYJ YIZIAYb4QgEBBAQDAgWgMAsGA1UdDwQEAwIF4DAxBglghkgBhvhCAQ0EJBYiU2Vn dXJldGF0IGVuIFhhcnhlcyBkZSBDb21wdXRhZG9yczAdBgNVHQ4EFgQUs1TZ5Do4 FLoatnuzli3E6GNG24AwgcIGA1UdIwSBujCBt4AUPFQSdsWjUvP/PNKuXHd6DnXc p0ShgZOkgZAwgY0xCzAJBgNVBAYTAkVTMQ8wDQYDVQQIEwZMbGVpZGExEDAOBgNV BAcTB1NvbHNvbmExDDAKBgNVBAoTA1VPQzEMMAoGA1UECxMDVU9DMR4wHAYDVQQ QD ExVFbnRpdGF0IENlcnRpZmljYWRvcmExHzAdBgkqhkiG9w0BCQEWEGdmYXJyYXNi QHVvYy5lZHWCCQDtaSvp+/cSJzAJBgNVHREEAjAAMAkGA1UdEgQCMAAwDQYJKoZI hvcNAQEFBQADggEBAABXb+FZNnf2yfS2oyeBWIHbKerZFm+bzgdnWJWKFkbYyN9H wzkpza0L+tt399pXyikDjMSeP3suAwzRcUsE7CTaZON/rojlj6GvV3Ym5iETDz93 +qeLQwfhcT3MMsX//1PaKNamhROSxT6de+Ei1xN1B/l44/7wuZi9aE7N+T49QdVO lCvkW0zuhHcbCd6DnclHsyzmK5O+9BlMWhQPGLWYfXK5whdJQ6jkistX/tvvU/oF ae3T+n6la/on09X10wGPql+hiDoSrHg87r5MKEysDpS4ELvIPsIqYlBbllI5m3RU TanoLGB2pWJvO6zYCuEVGCc/GT16XMhFICfQ2Rw=');
```

Configuració per a l'execució a través de la màquina virtual de Java.

El programari s'ha desenvolupat a través del llenguatge de programació Java, així que és necessari disposar d'un Java Runtime Environment.

A part de la llibreria IAIK, que s'ha explicat anteriorment com configurar-la, és necessari, a més disposar de les següents llibreries:

- Jdom.jar: Que conté les classes per al tractament dels fitxers en XML.
- mysql-connector-java-5.0.7-bin.jar: Que conté les classes per a connectar-se a través de Java al servidor de base de dades MySQL.

Aquestes llibreries es poden establir a través de la variable d'entorn CLASSPATH.

Jocs de proves per als protocols criptogràfics

Com que aquest projecte es desenvolupava paulativa i gradualment era necessari assegurar-se, abans de saltar a etapes posteriors que les que es deixaven enrera funcionaven correctament.

El cor més important resideix en la implementació dels protocols criptogràfics. Per tant doncs, per a comprovar que el sistema funcionava correctament, es van escriure un seguit de programes en Java que permetien comprovar si la comunicació entre els actors funcionava correctament “abans” però d'implementar les comunicacions a través de XML, RMI i l'entorn gràfic.

En l'annex C s'ha deixat el codi font alguns d'aquests primers programes:

- Codi d'un programa en Java per a que un metge obtingui el llistat dels seus pacient “sense” la utilització de comunicació via RMI i “sense” interfície gràfica (emprant solament mode de comandes).

- Codi d'un programa en Java per a que un metge introdueixi un diagnòstic i un apunt per a un determinat pacient utilitzant comunicació amb el Gestor directament, “sense” RMI i “sense” interfície gràfica (emprant solament mode de comandes).

- Codi d'un programa en Java per a que un metge obtingui el llistat dels seus pacients utilitzant comunicació amb el Gestor via RMI però “sense” interfície gràfica (emprant solament mode de comandes).

- Codi d'un programa en Java per a que un metge pugui veure la història clínica d'un determinat pacient utilitzant comunicació amb el Gestor via RMI però “sense” interfície gràfica (emprant solament mode de comandes).

Jocs de proves definitiu

Finalment analitzarem un joc de proves amb el projecte desenvolupat fins el moment. Per a executar-lo és necessari disposar de tot l'entorn preparat., en concret, l'entorn de Java, la llibreria IAIK instal·lada, el servidor de MySQL en actiu, la base de dades creada, el fitxer /bin/cfgBBDD.txt configurat correctament i els certificats generats.

Respecte als certificats i fitxers P12, recordem que disposem dels següents:

- pki/gestor.12: Fitxer .p12 per al gestor. La contrasenya és “uoc07”.
- pki/8006.p12: Fitxer .p12 per al pacient amb número d'història clínica 8006. La contrasenya és “uoc07”.
- pki/125000011.p12: Fitxer .p12 per al metge amb número de col·legiat 125000011. La contrasenya és “uoc07”.

Execució del gestor

El primer pas serà arrancar el gestor, definint un port RMI:

./bin/executaGestor.sh 4567 &

Apareixerà una finestra similar a la següent:

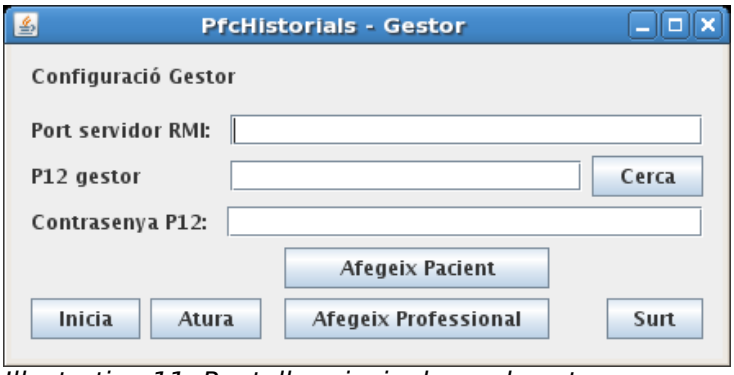


Illustration 11: Pantalla principal per al gestor

Abans de fer clic en el botó “Inicia” establirem les dades següents:

- Port servidor RMI: 4567
- P12 Gestor: Cercarem el fitxer pki/gestor.p12
- Contrasenya P12: uoc07

El servidor RMI està ja en marxa.

Execució de la vista de pacients

A continuació, executem el client:

./executaClient.sh

Apareixerà la pantalla d'autenticació següent:

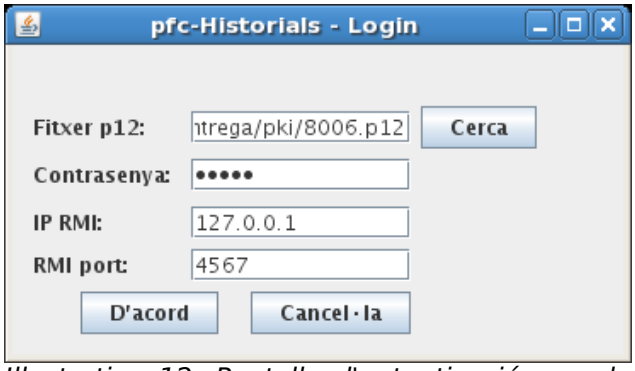


Illustration 12: Pantalla d'autenticació per als usuaris (pacients i metges).

En aquest cas, cercarem el fitxer pki/8006.p12 i posarem la contrasenya (uoc07). També la direcció IP de la màquina on està el servidor RMI (en aquest cas, en local, 127.0.0.1) i el port RMI (4567).

Si tot funciona correctament, hauria d'aparèixer la pantalla següent:

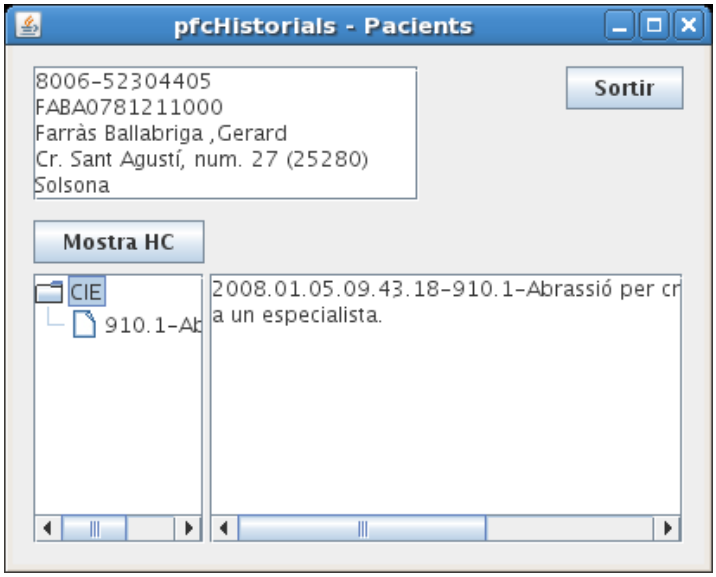


Illustration 13: Pantalla principal per als pacients

En aquesta pantalla, una vegada s'ha fet clic al botó “Mostra HC” apareix la informació següent:

- En el quadre d'adalt a l'esquerra apareix informació de tipus administratiu del pacient en concret.
- A l'esquerra un arbre amb els codis CIE i el títol de la malaltia o incidència.
- A la dreta, els apunts dels metges.

Execució de la vista de professionals sanitaris

A continuació, executarem l'aplicació validant-nos com a professionals sanitaris (en concret, a través del professional fictici “Josep Vilaseca” amb número de col·legiat 125000011).

El primer pas és executar les comandes següents:

```
cd bin/
./executaClient &
```

Apareixerà el mateix quadre d'autenticació que per als pacients. En aquest cas cercarem el fitxer pki/125000011.p12 i posarem la contrasenya (uoc07). També la direcció IP de la màquina on està el servidor RMI (en aquest cas, en local, 127.0.0.1) i el port RMI (4567).

A través del camp “organizationalUnitName” el sistema reconeixerà aquest usuari com un professional sanitari i, per tant, apareixerà la interfície següent:

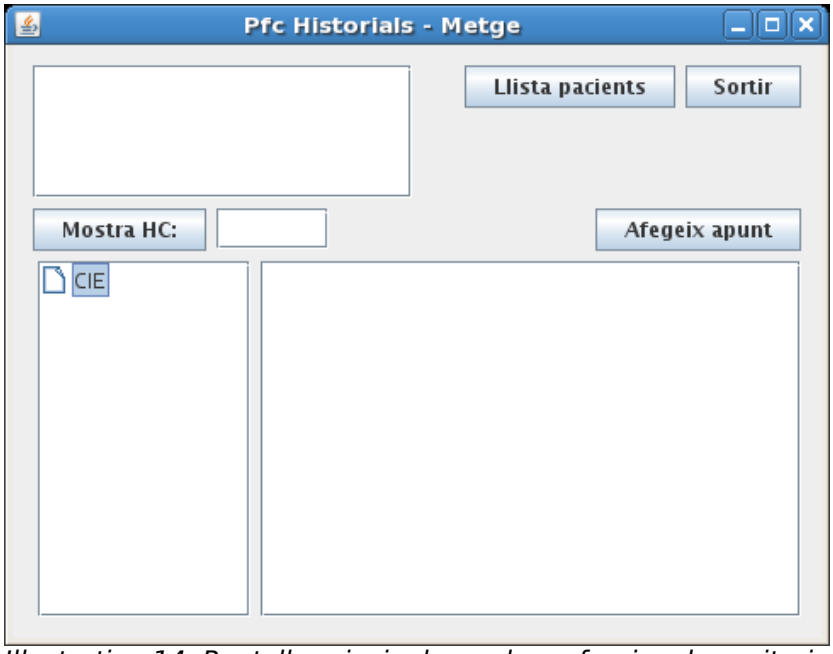


Illustration 14: Pantalla principal per als professionals sanitaris

Fent clic en el botó “Llista de pacients” és possible veure un llistat dels pacients que aquest professional té assignats, i a més, en podem escollir un per a passar visita.

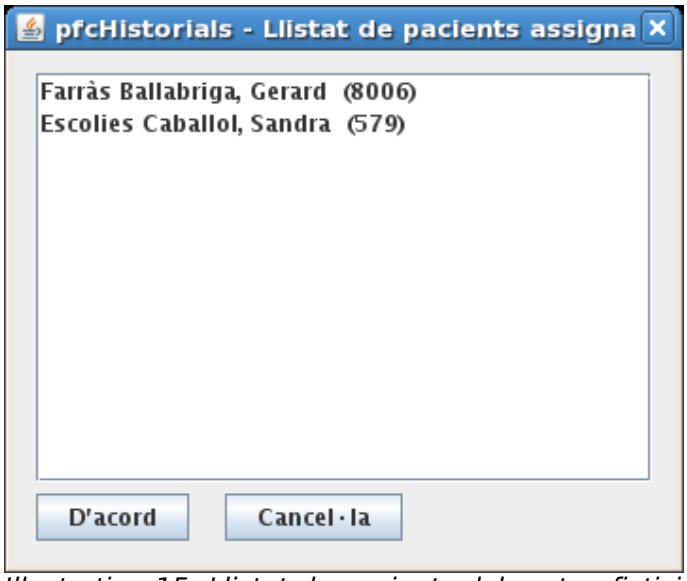


Illustration 15: Llistat de pacients del metge fictici "Josep Vilaseca", amb el seu nom i identificador (NHC). Escollint-ne un i fent clic a d'Acord, podem seleccionar-lo per a passar visita.

Si, el que ens interessa és afegir un apunt, doncs farem clic en el botó "Afegeix apunt":

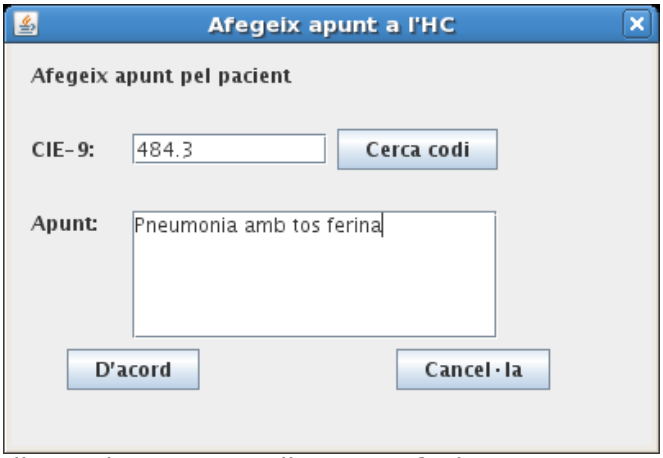


Illustration 16: Pantalla per a afegir un nou apunt en la història clínica d'un pacient.

Capítol 9. Valoració econòmica del projecte.

Tenint en compte que s'han invertit unes 320 hores en el desenvolupament del projecte, a una tarifa de 40 €/hora, podríem estimar que el preu de desenvolupament del projecte és de: 12.800 €.

D'altra banda, per a posar en marxa el sistema implementat en un entorn real de producció, s'ha de tenir en compte que l'única llicència per al programari que hem emprat seria el de la llibreria IAIK.

El preu actual per a una llicència de tipus bàsic de IAIK JCE Basic Licence (non-US version) per a un sol desenvolupador costa 500 €. 100 € més per a cada nova llicència.

La resta de tecnologies emprades son de llibre distribució sense la necessitat de pagar llicències de programari:

- GNU/Linux com a sistema operatiu.
- Java com a llenguatge de desenvolupament.
- MySQL com a servidor de base de dades.
- El connector mysql-connector-java-5.0.7-bin.jar per a accedir des de Java a un servidor de base de dades MySQL.
- La llibreria JDOM.
- NetBeans per a la creació dels GUIs d'usuari.
- OpenOffice per a la redacció de la documentació.
- Dia per a l'elaboració de les gràfiques presents en la documentació.
- GIMP per a les captures de pantalla presents en la documentació.

Capítol 10. Conclusions i opinió personal

Un cop arribats a aquest punt és el moment de fer una valoració de la feina requerida i de la feina presentada. En aquest capítol es demostra com la majoria dels requisits que es van demanar a l'enunciat del projecte han estat satisfactòriament assolits.

A continuació, s'anirà comentant l'estat final, l'opinió personal i una crítica per a cadascuna de les PACS lliurades:

- PAC 1: Instal·lar IAIK i generació de la PKI a través de OpenSSL: Personalment vaig trobar aquesta PAC força fàcil i és que estava perfectament ben documentada en el mateix enunciat i també gràcies als guions de seqüències ja lliurats.

- PAC 2: Implementació dels protocols. Aquesta fou la PAC que vaig trobar més interessant i on es troben, realment, els fonaments d'aquest projecte. Crec que, en aquest projecte, s'ha assolit prou bé, encara que es podria fer molt més eficient i i potser l'hauria d'haver sotmès a una bona bateria de proves.

  Per a traspassar la informació xifrada em va semblar que la forma més senzilla seria emprar codificació en Base 64. A més, de seguida vaig trobar una classe en Java amb llicència gratuïta que implementava aquesta codificació (mirar la classe Base64.java, extreta, en realitat de http://iharder.sourceforge.net/current/java/base64/).

- PAC 3. Implementació de documents XML: Coneixia la tecnologia XML però mai no l'havia treballada des de Java. Per a fer-ho, em va semblar bé la llibreria JDOM. A més JDOM està disponible sota una llicència open source de tipus Apache, així que no hi ha problemes per a emprar-la.

  Crec que aquesta PAC ha estat superada correctament malgrat crec que potser hagués estat una bona idea crear una classe expressament per a la lectura i creació dels diferents fitxers en XML alliberant així d'aquestes tasques a “Gestor”, “Usuari”, “Metge” i “Pacient”: Crec que m'ho pensaria millor si s'hagués de tornar-ho a fer.

- PAC 4. Comunicacions via RMI. Com ja que havia treballat recentment en una altra assignatura amb RMI em va funcionar ràpidament i sense massa dificultats. Crec que PAC superada també, sense més comentaris.

- PAC 5. BD servidor i registre d'usuaris. La base de dades en format MySQL ja l'havia estat treballant en paral·lel durant les PACs anteriors, així que vaig passar-me per alt, també, la fase de registre d'usuaris: Greu error. PAC NO superada.

- PAC 6 i PAC 7. Interfícies gràfiques: Per al disseny de les interfícies gràfiques d'usuari vaig estar cercant múltiples alternatives i és que, com que NO en coneixia cap d'apriori, doncs em tocava fer-ho ara. En aquest punt vaig perdre molt de temps: Vaig provar Eclipse, NetBeans i, fins i tot, de programar les interfícies directament (comentar que, fins aquest moment, TOT el desenvolupament el feia a través d'un simple editor de text (concretament “gedit” de la família de GNOME sobre GNU/Linux)).

Em va semblar que el programari NetBeans era més senzill d'utilitzar que Eclipse i, a més, escrivia uns fitxers amb extensió .form en format XML per a cada formulari que es traduïen directament a classes Java, així que em vaig decantar per aquesta opció.

Tot i això, per la propera vegada, si veig que per a una determinada fase d'un projecte hi ha disseny de GUIS, cercaré de bones a primeres un IDE per a pugui desenvolupar-ho TOT des d'allà i és que, després, unir el disseny de les interfícies amb el que ja estava desenvolupat em va costar força temps també.

De totes maneres, crec que les dues PACS han estat superades encara que, en comptes d'haver-me dedicat a l'arbre amb els codis CIE, al llistat de malalties en format CIE i al quadre amb les dades administratives dels pacients, doncs hagués estat millor implementar el tema del registre d'usuaris en el servidor.

- PAC 8. Documentació. Anar escrivint la documentació PAC a PAC és una gran idea.

Comentar, finalment que he gaudit força amb l'elaboració d'aquest projecte de final de carrera i en tinc doncs una opinió molt positiva.

Primer ja que conec força d'aprop la temàtica que s'ha tractat (i és que col·laboro en un Centre Sanitari d'Atenció Primària comarcal) així que crec que programar un sistema d'informació clínic amb criptografia de base és una “gran” i “interessant” innovació: I és que, que jo sàpiga, no conec cap programari d'aquest tipus amb aquesta característica.

Important també per a mi poder treballar amb tecnologies, les quals em sento força còmode: Java, MySQL, XML sobre un entorn GNU/Linux.

Aquest projecte de final de carrera m'ha servit, finalment, per a treballar en un projecte no gran, però sí de certa entitat, on posar en conjunció múltiples coneixements i tecnologies adquirides durant la carrera. M'ha ensenyat, això sí, a ser més rigorós, a planificar-me millor i a tenir en compte que un programari es construeix com una casa: Dels fonaments fins a la teulada i s'ha de tenir en compte que, si hi ha alguna columna que falla, doncs la casa se'n pot anar a terra.

Capítol 11. Possible treball futur.

Considerem que aquest treball conté un primer “esbós” experimental per a la gestió d'historials clínics, on s'han definit un conjunt de protocols criptogràfics que implementen un substrat on la seguretat de les dades i les comunicacions forma part del cor del sistema.

Un possible treball futur seria doncs convertir aquest primer treball en un veritable HIS (Hospital Information System) amb les eines típiques d'aquests sistemes d'informació però emprant sempre protocols amb base criptogràfica.

En concret, podríem afegir el següent:

* Un perfil administratiu, que tingués accés a les dades administratives dels pacients, poder gestionar les agendes dels metges, però sense accés a la documentació clínica.

Per al perfil mèdic:

- Gestió d'agendes.
- Possibilitat d'efectuar ordres mèdiques per als pacients (analítiques, radiografies, i derivacions o interconsultes a d'altres professionals.).
- Possibilitat d'efectuar receptes.
- Possibilitat d'afegir un quadre amb condicionants i problemes: Ex. Si al pacient se li ha detectat hipertensió, diabetis, si és fumador, si es tracta d'una persona gran que viu sola i/o aïllada.
- Protocols: Possibilitat d'afegir un determinat tipus de proves per a determinades malalties o condicionants. Per exemple, que el sistema avisi cada mig any que un pacient amb diabetis s'ha de fer un seguit de proves de forma sistemàtica.

En quant a comunicacions:

- Generar interfícies via web, amb applets en Java, per exemple, per a l'accés al sistema per a que no fos necessari instal·lar cap tipus de programari en les estacions de treball dels usuaris.
- Implementar en els clients per a comunicar-se a webservices externs oficials de la Generalitat de Catalunya:
  - Registre Central d'Assegurats (RCA): Base de dades que conté els pacients que posseeixen targeta sanitària TIS a Catalunya i on s'indica dades de tipus administratiu, Unitat Primària a la que estan assignats, règim laboral, entre d'altres.
  - SIGIT: Sistema Integrat de Gestió d'Incapacitats Temporals). Webservices que s'utilitzen per a comunicar en temps real a l'ICAM una determinada baixa laboral.
  - I més endavant, integrar-se en el projecte de “Recepta electrònica” i també el el sistema d'història clínica compartida de Catalunya.

Glossari de termes

API: Sigles de Application Programming Interface. Interfície a través de la qual un programa accedeix als serveis del sistema operatiu i d'altres. Una API proveeix un nivell d'abstracció entre l'aplicació i el kernel per tal d'assegurar la portabilitat del codi.

Autoritat de Certificació: Entitat que emet certificats digitals d'usuaris o companyies, de manera que aquests es poden identificar davant d'un tercer. És de vital importància que l'Autoritat de Certificació comprovi que la part que demana un certificat és realment qui diu ser.

Base 64: Codificació que empra només 6 bits per caràcter. D'aquesta manera tenim 64 possibles valors. En el nostre cas permet transmetre cadenes que contenen les signatures com a cadena de caràcters sense fer malbé el contingut de les mateixes. Base64 també s'utilitza molt en comunicacions amb dades binàries provinents de text, com ara el correu electrònic.

Base de dades: Un o més conjunts estructurats de dades persistents, normalment associades a programaris que les consulten o actualitzen. Una base de dades és un component d'un sistema Gestor de base de dades.

Certificat: Arxiu que conté les dades que donen fe de l'autenticitat de la persona o entitat que presenta.

CIE: Acrònim de Classificació Internacional d'Enfermetats. Actualment s'utilitza la versió CIE-10.

Document Type Definition o DTD: Definició d'un document XML o SGML. Consisteix en unes regles per interpretar els documents i per establir les regles de construcció dels mateixos.

Fingerprint: Funció de hash que s'aplica sobre un certificat d'un usuari per a obtenir un identificador únic i més còmode d'utilitzar. En el projecte s'utilitzem fingerprints per identificar de manera única als actors del sistema a partir dels seus certificats.

Funció de hash: Resum amb pèrdua que dona lloc a una seqüència de longitud fixa a partir d'unes dades sense importar la longitud d'aquestes. Les seves propietats permeten la seva utilització alhora de verificar la integritat de les dades. Les funcions de hash també son utilitzades per assignar posicions en temes de cerca i ordenació.

IAIK: Acrònim de “Institute for Applied Information Processing and Communication”. Aquests són els desenvolupadors de la llibreria criptogràfica amb el mateix nom.

Interfície: Punt d'interacció i/o comunicació entre un ordinador i una altra entitat, ja sigui persona o un altre equip.

Java: Llenguatge de programació multiplataforma, robust, interpretat, distribuït, orientat a objectes, portable, desenvolupat per Sun Microsystems a mitjans dels 90.

JDOM: Acrònim de Java Document Object Model. Solució completa basada en Java per a accedir i modificar documents XML des de codi en llenguatge Java.

LOPD: La Llei Orgànica 15/1999 del 13 de desembre de Protecció de Dades de Caràcter Personal, abreviada com a LOPD, és una Llei Orgànica espanyola que té per objecte garantir i protegir, en el que concerneix al tractament de les dades personals, les llibertats públiques i els drets fonamentals de les persones físiques i especialment el del seu honor, intimitats i privacitats personal i familiar.

MySQL: Sistema gestor de base de dades de lliure distribució.

NHC: Acrònim de número d'història clínica. Número identificador d'un pacient en un HIS.

PKCS: Acrònim de Public-Key Cryptography Standards. Són un conjunts d'estàndards definits pels laboratoris RSA que especifiquen els estàndards de clau pública.

PKI: Acrònim de Public Key Infrastructure corresponent a infraestructura de clau pública.

Port: Camí lògic o extrem d'un canal en un sistema de comunicació. Els protocols TCP i UDP utilitzats en Ethernet utilitzen els ports per a desmultiplexar canals lògics en la mateixa interfície de xarxa d'una computadora.

Protocol d'autenticació: Conjunt d'operacions que duen a terme dues o més parts de manera que, al final, almenys una de les parts queda autenticada davant de la resta.

RCA: El Registre Central d'Assegurats (RCA) permet la identificació única dels assegurats del CatSalut, mitjançant el Codi d'Identificació Personal (CIP), la gestió i consulta de les seves dades i l'actualització d'aquestes les unitats proveïdores de serveis sanitaris. També permet gestionar l'emissió de targetes sanitàries, la localització dels assegurats per àrees bàsiques de salut i l'assignació de la unitat proveïdora d'atenció primària.

RMI: Sigles de Remote Method Invocation. API propietària de Java que permet a les aplicacions locals d'executar codi que es troba allotjat en una altra màquina remota. Aquesta última posa a disposició uns mètodes públics que seran accessibles a través d'una interfície.

SIGIT: El Sistema Integrat de Gestió de la Incapacitat Temporal son un conjunt de webservices de l'Institut Català d'Avaluacions Mèdiques que s'empren per a informar en temps real al centre de les incapacitats temporals per a treballar en un determinat pacient.

Signatura: Document generat a partir d'un missatge i la clau privada d'un usuari. Al xifrar les dades del missatge amb la clau privada es genera un missatge xifrat que una tercera persona pot desxifrar amb la clau pública i comprovar que és el mateix que les dades originals. D'aquesta manera s'assegura que l'origen de les dades no ha estat modificat i que la persona que l'envia és qui diu ser, ja que només ella té accés a la seva clau privada.

Guió de seqüències (scripts): Conjunt d'operacions que s'agrupen en un arxiu que les executa una darrere l'altra per a facilitar l'execució de tasques repetitives. En el nostre cas utilitzem aquests guions per a la generació dels certificats. Durant el desenvolupament també els hem usat per a arrencar el servidor.

Sistema gestor de base de dades: Conjunt d'aplicacions que normalment porten control d'un conjunt de dades persistents, oferint alhora facilitat d'accés i consulta als usuaris finals.

Xifratge asimètric: Mètode emprat en criptografia que utilitza dues claus, una pública i una privada, per a dur a terme el xifrat i el posterior desxifrat d'un missatge. En general, s'utilitza la clau pública de l'usuari destí per a xifrar i aquest utilitza la clau privada per a desxifrar.

XML: Metallenguatge escrit en SGML que permet a un usuari de crear el seu propi llenguatge de tags. Freqüentment utilitzat per a facilitar l'intercanvi de documents en entorns de xarxa.

Bibliografia i referències

The J2SE Development Kit (JDK).
URL: http://java.sun.com/downloads.

Java Remote Method Invocation (Java RMI).
URL: http://java.sun.com/products/jdk/rmi/.

Java™ Platform , Standard Edition 6 API Specification
URL: http://java.sun.com/javase/6/docs/api/ Java Platform SE 6

The JDOM XML Api.
URL: http://www.jdom.orgs/apidocs/index.html

The MySQL database server.
URL: http://www.mysql.com/documentation/index.html,

Extensible Markup Language (XML).
URL: http://www.w3.org/XML/

Tutorial d'XML.
URL: http://www.w3schools.com/xml/

Openssl: The open source toolkit for SSL/TLS.
URL: http://www.openssl.org.

NetBeans.
URL: http://www.netbeans.org/

Annexos

Annex A. Fitxer de configuració del PKI.

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

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

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

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

[ new_oids ]

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

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

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

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

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

x509_extensions  = usr_cert             # The extentions to add to the cert

# Extensions to add to a CRL. Note: Netscape communicator chokes on V2 CRLs
```

```
# so this is commented out by default to leave a V1 CRL.
# crl_extensions  = crl_ext

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

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

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

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

####################################################
    ################
[ req ]
default_bits            = 1024
default_keyfile     = privkey.pem
distinguished_name      = req_distinguished_name
attributes          = req_attributes
x509_extensions  = v3_ca        # The extentions to add to the self signed cert

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

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

```
string_mask = nombstr

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

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

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

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

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

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

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

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

emailAddress                    = Email Address
emailAddress_max                = 40

# SET-ex3                       = SET extension number 3

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

unstructuredName                = An optional company name

[ usr_cert ]

# These extensions are added when 'ca' signs a request.

# This goes against PKIX guidelines but some CAs do it and some software
# requires this to avoid interpreting an end user certificate as a CA.

basicConstraints=CA:FALSE

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

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

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

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

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

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

# This will be displayed in Netscape's comment listbox.
nsComment                       = "Seguretat en Xarxes de Computadors"

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

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

# Copy subject details
issuerAltName=issuer:copy

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

[ v3_req ]

# Extensions to add to a certificate request

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

[ v3_ca ]


# Extensions for a typical CA


# PKIX recommendation.

subjectKeyIdentifier=hash

authorityKeyIdentifier=keyid:always,issuer:always

# This is what PKIX recommends but some broken software chokes on critical
# extensions.
```

```
#basicConstraints = critical,CA:true
# So we do this instead.
basicConstraints = CA:true

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

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

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

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

[ crl_ext ]

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

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

Annex B. Fitxer de configuració de la base de dades.

A continuació, el fitxer amb les instruccions necessàries per a crear les taules base, sense dades, de la base de dades MySQL. (Per raons d'espai no hi ha afegit el llistat de codificacions CIE-9, així que el millor és accedir al fitxer complet situat a src/pfchistorials.sql).

```
CREATE TABLE IF NOT EXISTS `dtcertificats` (
  `idusuari` varchar(10) NOT NULL,
  `certificat` text NOT NULL,
  PRIMARY KEY  (`idusuari`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Estructura de la taula `dtcodCIE9`
--

CREATE TABLE IF NOT EXISTS `dtcodCIE9` (
  `codicie` varchar(10) NOT NULL,
  `descrip` varchar(40) NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Estructura de la taula `dtdiagnostics`
--

CREATE TABLE IF NOT EXISTS `dtdiagnostics` (
  `nhcpacient` int(11) NOT NULL,
  `numseriex` int(11) NOT NULL,
  `instanttemps` varchar(50) NOT NULL,
  `pgvsmv` text NOT NULL,
  `Sgvsmvtx` text NOT NULL,
  `sgxidusuarip` text NOT NULL,
  PRIMARY KEY  (`nhcpacient`,`numseriex`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;



--
-- Estructura de la taula `dtmetges`
--

CREATE TABLE IF NOT EXISTS `dtmetges` (
  `numcolegiat` int(11) NOT NULL,
  `nom` varchar(40) NOT NULL,
  `cognom1` varchar(40) NOT NULL,
  `cognom2` varchar(40) NOT NULL,
  `correue` varchar(40) NOT NULL,
  `telf` varchar(40) NOT NULL,
  PRIMARY KEY  (`numcolegiat`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;



--
-- Estructura de la taula `dtpacients`
--

CREATE TABLE IF NOT EXISTS `dtpacients` (
  `nhc` int(11) NOT NULL auto_increment,
```

```
  `dni` varchar(40) NOT NULL,
  `tis` varchar(40) default NULL,
  `nss` varchar(14) default NULL,
  `nom` varchar(12) NOT NULL,
  `cognom1` varchar(40) NOT NULL,
  `cognom2` varchar(40) default NULL,
  `correue` varchar(40) default NULL,
  `telf` varchar(40) default NULL,
  `direccio` varchar(40) default NULL,
  `codipostal` varchar(40) default NULL,
  `poblacio` varchar(40) default NULL,
  `metgeassignat` varchar(40) NOT NULL,
  `datanaixement` varchar(40) default NULL,
  `sexe` varchar(40) NOT NULL,
  PRIMARY KEY  (`nhc`)
) ENGINE=MyISAM AUTO_INCREMENT=8008 DEFAULT CHARSET=latin1
   AUTO_INCREMENT=8008;



--
-- Estructura de la taula `dtsessionsges`
--

CREATE TABLE IF NOT EXISTS `dtsessionsges` (
  `nhc` int(11) NOT NULL,
  `ni` int(11) NOT NULL,
  `ng` int(11) NOT NULL,
  PRIMARY KEY  (`nhc`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Volcant dades de la taula `dtsessionsges`
--
```

Annex C. Codi Java per a proves de fases no finals.

A continuació, en llenguatge Java, un conjunt de programes emprats com a jocs de proves en fases de desenvolupament.

Codi d'un programa en Java per a que un metge obtingui el llistat dels seus pacient “sense” la utilització de comunicació via RMI i “sense” interfície gràfica (emprant solament mode de comandes).

```
import iaik.security.provider.IAIK;
import java.security.*;
import org.jdom.Document;

/**
 * La classe <code>ConsultaPacients</code> conté les instruccions per a que un
    metge pugui veure els pacients que té assignats.
 *
 * @author <a href="mailto:gfarrasb@uoc.edu">Gerard Farràs i Ballabriga</a>
 * @version 1.0
 */

public class ConsultaPacients {

  public static void usage () {

        System.out.println("Aquesta classe retorna el llistat de pacients d'un
    determinat metge.\n");
        System.out.println("Utilització:");
        System.out.println("\t--metge Amb l'identificador del metge del qual volem
    obtenir el llistat.");
        System.out.println("\t--passwd Amb la contrasenya del certificat del
    metge.");
        System.out.println("\t--debug (opcional) Per a debugar el programari.");
        System.out.println("\nEx. java ConsultaPacients --metge=25013981 --
    passwd=uoc0506.");
        System.exit(-1);

  }

  public static void main(String[] args){

        boolean debug = false;
        String metge = "";
        String contrasenya = "";

        if (args.length == 0) {
            usage ();
        }

        for (int i=0;i<args.length;i++) {

            if ( (args[i].substring(0,2)).compareTo("--") != 0) {
                usage();
```

```
                        System.exit(-1);
                } else {

                        if ( args[i].substring(2).startsWith("debug") ) {
                                debug = true;
                        }

                        if ( args[i].substring(2).startsWith("metge") ) {
                                int pos = args[i].indexOf("=");
                                if (pos <= 0) usage();
                                metge = args[i].substring( pos + 1 );
                        }

                        if ( args[i].substring(2).startsWith("passwd") ) {
                                int pos = args[i].indexOf("=");
                                if (pos <= 0) usage();
                                contrasenya = args[i].substring( pos + 1);
                        }
                }

        }

        if ((metge.compareTo("") == 0) || (contrasenya.compareTo("") == 0)) {
                usage();
        }

        if (debug) {
                System.out.println("[ConsultaPacients] Metge --> " + metge );
                System.out.println("[ConsultaPacients] Passwd --> " +
contrasenya );
        }

        Security.insertProviderAt(new IAIK(), 2);

        /*
                jcasserras és un usuari de tipus metge i demana el seu llistat de
pacients.
        */
        Metge jcasserras = new Metge( metge, contrasenya );
        jcasserras.setDebug( debug );

        /*.
                Usuari gestor
        */
        Gestor gestor = new Gestor ();
        gestor.setDebug ( debug );

        /*
                Implementem el protocol 1
        */
        //String a = jcasserras.procedure1 () ;
        Document a = jcasserras.procedure1 ();
        Document b = gestor.procedure2 ( a );

        /*
```

```
                JCasserras ha de desxifrar el missatge amb la seva clau privada Su i
    obtenir Ng, Ni i Id_Usuarig
            */
            Document c = jcasserras.pas3 ( b , "Llista_pacients" );

            /*
                JCasserras envia al gestor Pg[Ng, Consulta, Id_usuari] a G;
            */
            boolean autentificacio = gestor.pas4 ( c );

            if (autentificacio) {

                if (debug) System.out.println("[ConsultaPacients] Retornem un
    llistat dels pacients");
                Document llistatPacients = gestor.procedure5 ( metge );
                jcasserras.procedure6 ( llistatPacients );

            } else {
                System.out.println("[ConsultaPacients] Identificació
    INCORRECTA!");
                System.exit(-1);
            }

    }

}
```

Codi d'un programa en Java per a que un metge introdueixi un diagnòstic i un apunt per a un determinat pacient utilitzant comunicació amb el Gestor directament, “sense” RMI i “sense” interfície gràfica (emprant solament mode de comandes).

```
import iaik.security.provider.IAIK;
import java.security.*;

/**
 * La classe <code>InsereixHisto</code> conté les instruccions per a que un metge
 * pugui introduir diagnòstics i apunts en l'historial clínic d'un determinat pacient.
 *
 * @author <a href="mailto:gfarrasb@uoc.edu">Gerard Farràs i Ballabriga</a>
 * @version 1.0
 */

public class InsereixHisto {

  public static void usage () {

            System.out.println("Aquesta classe insereix un nou apunt en la història
    clínica d'un determinat pacient.\n");
            System.out.println("Utilització:");
            System.out.println("\t--metge Amb l'identificador del metge.");
            System.out.println("\t--passwd Amb la contrasenya del certificat del
    metge.");
            System.out.println("\t--pacient Amb l'identificador del pacient.");
```

```
        System.out.println("\t--codicie Amb el codi CIE del diagnòstic a introduir.");
        //System.out.println("\t--apunt Amb l'apunt a introduir (entre comentes).");

        System.out.println("\t--debug (opcional) Per a debugar el programari.");
        System.out.println("\nEx. java InsereixHisto --metge=25013981 --
  passwd=uoc0506 --pacient=8006 --codicie=802");
        System.exit(-1);

  }

  public static void main(String[] args){

        boolean debug = false;
        String metge = "";
        String contrasenya = "";
        String pacient = "";
        String codicie = "";

        if (args.length == 0) {
                usage ();
        }

        for (int i=0;i<args.length;i++) {

                if ( (args[i].substring(0,2)).compareTo("--") != 0) {
                        usage();
                        System.exit(-1);
                } else {

                        if ( args[i].substring(2).startsWith("debug") ) {
                                debug = true;
                        }

                        if ( args[i].substring(2).startsWith("metge") ) {
                                int pos = args[i].indexOf("=");
                                if (pos <= 0) usage();
                                metge = args[i].substring( pos + 1 );
                        }

                        if ( args[i].substring(2).startsWith("passwd") ) {
                                int pos = args[i].indexOf("=");
                                if (pos <= 0) usage();
                                contrasenya = args[i].substring( pos + 1);
                        }

                        if ( args[i].substring(2).startsWith("pacient") ) {
                                int pos = args[i].indexOf("=");
                                if (pos <= 0) usage();
                                pacient = args[i].substring ( pos + 1 );
                        }

                        if ( args[i].substring(2).startsWith("codicie") ) {
                                int pos = args[i].indexOf("=");
                                if (pos <= 0) usage();
                                codicie = args[i].substring ( pos + 1 );
```

```
                        }

                        if ( args[i].substring(2).startsWith("apunt") ) {
                                int pos = args[i].indexOf("=");
                                if (pos <= 0) usage();
                                codicie = args[i].substring ( pos + 1 );
                        }


                }

        }

        if ((metge.compareTo("") == 0) || (contrasenya.compareTo("") == 0) ||
(pacient.compareTo("") == 0)) {
                usage();
        }

        if (debug) {
                System.out.println("[InsereixHisto] Metge --> " + metge );
                System.out.println("[InsereixHisto] Passwd --> " + contrasenya );
                System.out.println("[InsereixHisto] Pacient --> " + pacient );
                System.out.println("[InsereixHisto] Codi CIE --> " + codicie );
        }

        Security.insertProviderAt(new IAIK(), 2);

        /*
                Aquest és un usuari "genèric" del sistema i tant podria ser un
metge com un pacient
        */
        Metge rafart = new Metge( metge , contrasenya );
        if (debug) rafart.setDebug(true);

        /*
                Usuari gestor
        */
        Gestor gestor = new Gestor ();
        if (debug) gestor.setDebug(true);

        /*
                Implementem el protocol 3
        */
        org.jdom.Document a = rafart.procedure1 () ;
        org.jdom.Document b = gestor.procedure2 ( a );

        /*
                rafart ha de desxifrar el missatge amb la seva clau privada Su i
obtenir Ng, Ni i Id_Usuarig
        */
        String operacio = new String ("Inserir_Visita");
        String apunt = "ABRASIO O CREMADA PER FRICCIO DIT(S) DE LA MA,SENSE
INFECCIO";

        String c = rafart.pas3InserirVisita ( b , pacient , codicie, apunt );
```

```
            /*
                    rafart envia al gestor Pg[Ng, V, Sm[V]] a G;
            */
            gestor.pas4InserirVisita( c );



    }
```

Codi d'un programa en Java per a que un metge obtingui el llistat dels seus pacients utilitzant comunicació amb el Gestor via RMI però "sense" interfície gràfica (emprant solament mode de comandes).

```
import iaik.security.provider.IAIK;
import java.security.*;
import org.jdom.Document;

/**
 * La classe <code>ConsultaPacients</code> conté les instruccions per a que un
    metge
 * pugui veure els pacients que té assignats.
 *
 * @author <a href="mailto:gfarrasb@uoc.edu">Gerard Farràs i Ballabriga</a>
 * @version 1.0
 */

public class ConsultaPacients {

  public static void usage () {

            System.out.println("Aquesta classe retorna el llistat de pacients d'un
    determinat metge.\n");
            System.out.println("Utilització:");
            System.out.println("\t--metge Amb l'identificador del metge del qual volem
    obtenir el llistat.");
            System.out.println("\t--passwd Amb la contrasenya del certificat del
    metge.");
            System.out.println("\t--rmi Amb el port RMI.");

            System.out.println("\t--debug (opcional) Per a debugar el programari.");
            System.out.println("\nEx. java ConsultaPacients --metge=25013981 --
    passwd=uoc0506.");
            System.exit(-1);

  }

  public static void main(String[] args){

            boolean debug = false;
            String metge = "";
```

```
		String contrasenya = "";
		String rmi = "";

		if (args.length == 0) {
			usage ();
		}

		for (int i=0;i<args.length;i++) {

			if ( (args[i].substring(0,2)).compareTo("--") != 0) {
				usage();
				System.exit(-1);
			} else {

				if ( args[i].substring(2).startsWith("debug") ) {
					debug = true;
				}

				if ( args[i].substring(2).startsWith("metge") ) {
					int pos = args[i].indexOf("=");
					if (pos <= 0) usage();
					metge = args[i].substring( pos + 1 );
				}

				if ( args[i].substring(2).startsWith("passwd") ) {
					int pos = args[i].indexOf("=");
					if (pos <= 0) usage();
					contrasenya = args[i].substring( pos + 1);
				}

				if ( args[i].substring(2).startsWith("rmi") ) {
					int pos = args[i].indexOf("=");
					if (pos <= 0) usage();
					rmi = args[i].substring( pos + 1);
				}
			}

		}

		if ((metge.compareTo("") == 0) || (contrasenya.compareTo("") == 0) ||
(rmi.compareTo("") == 0) ) {
			usage();
		}

		if (debug) {
			System.out.println("[ConsultaPacients] Metge --> " + metge );
			System.out.println("[ConsultaPacients] Passwd --> " +
contrasenya );
		}

		Security.insertProviderAt(new IAIK(), 2);

		/*
			jcasserras és un usuari de tipus metge i demana el seu llistat de
pacients.
```

```
        */
        Metge jcasserras = new Metge( metge, contrasenya );
        jcasserras.setDebug( debug );

        /*.
                Usuari gestor
        */
        //Gestor gestor = new Gestor ();
        //gestor.setDebug ( debug );

        /*
                Implementem el protocol 1
        */
        //String a = jcasserras.procedure1 () ;
        Document a = jcasserras.procedure1 ();

        try {

            GestorInterRemot gestor =
(GestorInterRemot)java.rmi.Naming.lookup("//127.0.0.1:" + rmi + "/GestorRMI");

            // Imprimimos miMetodo1() tantas veces como devuelva miMetodo2()
                Document b = gestor.procedure2 ( a );

                //Document b = gestor.procedure2 ( a );

                /*
                        JCasserras ha de desxifrar el missatge amb la seva clau
privada Su i obtenir Ng, Ni i Id_Usuarig
                */
                Document c = jcasserras.pas3 ( b , "Llista_pacients" );

                /*
                        JCasserras envia al gestor Pg[Ng, Consulta, Id_usuari] a G;
                */
                boolean autentificacio = gestor.pas4 ( c );

                if (autentificacio) {

                        if (debug) System.out.println("[ConsultaPacients] Retornem
un llistat dels pacients");
                        Document llistatPacients = gestor.procedure5 ( metge );
                        jcasserras.procedure6 ( llistatPacients );

                } else {
                        System.out.println("[ConsultaPacinents] Identificació
INCORRECTA!");
                        System.exit(-1);
                }

         }   catch (Exception e)     {

                    e.printStackTrace();
            }
```

```
    }

}
```

Codi d'un programa en Java per a que un metge pugui veure la història clínica d'un determinat pacient utilitzant comunicació amb el Gestor via RMI però "sense" interfície gràfica (emprant solament mode de comandes).

```
import iaik.security.provider.IAIK;
import java.security.*;
import org.jdom.Document;

/**
 * La classe <code>ConsultaHistorial</code> conté les instruccions per a que un
    metge pugui veure l'historial clínic d'un determinat pacient.
 *
 * @author <a href="mailto:gfarrasb@uoc.edu">Gerard Farràs i Ballabriga</a>
 * @version 1.0
 */

public class ConsultaHistorial {

  public static void usage () {

            System.out.println("Aquesta classe retorna l'historial clínic d'un determinat
    pacient.\n");
            System.out.println("Utilització:");
            System.out.println("\t--metge Amb l'identificador del metge.");
            System.out.println("\t--passwd Amb la contrasenya del certificat del
    metge.");
            System.out.println("\t--pacient Amb l'identificador del pacient.");
            System.out.println("\t--rmi Amb el port RMI.");

            System.out.println("\t--debug (opcional) Per a debugar el programari.");
            System.out.println("\nEx. java ConsultaHistorial --metge=25013981 --
    passwd=uoc0506 --pacient=8006");
            System.exit(-1);
  }

  public static void main(String[] args){

            boolean debug = false;
            String metge = "";
            String contrasenya = "";
            String pacient = "";
            String rmi = "";

            if (args.length == 0) {
                    usage ();
            }

            for (int i=0;i<args.length;i++) {
```

```
                if ( (args[i].substring(0,2)).compareTo("--") != 0) {
                        usage();
                        System.exit(-1);
                } else {

                        if ( args[i].substring(2).startsWith("debug") ) {
                                debug = true;
                        }

                        if ( args[i].substring(2).startsWith("metge") ) {
                                int pos = args[i].indexOf("=");
                                if (pos <= 0) usage();
                                metge = args[i].substring( pos + 1 );
                        }

                        if ( args[i].substring(2).startsWith("passwd") ) {
                                int pos = args[i].indexOf("=");
                                if (pos <= 0) usage();
                                contrasenya = args[i].substring( pos + 1);
                        }

                        if ( args[i].substring(2).startsWith("pacient") ) {
                                int pos = args[i].indexOf("=");
                                if (pos <= 0) usage();
                                pacient = args[i].substring ( pos + 1 );
                        }

                        if ( args[i].substring(2).startsWith("rmi") ) {
                                int pos = args[i].indexOf("=");
                                if (pos <= 0) usage();
                                rmi = args[i].substring( pos + 1);
                        }


                }

        }

        if ((metge.compareTo("") == 0) || (contrasenya.compareTo("") == 0) ||
(pacient.compareTo("") == 0) || (rmi.compareTo("") == 0)) {
                usage();
        }

        if (debug) {

                System.out.println("[ConsultaHistorial] Metge --> " + metge );
                System.out.println("[ConsultaHistorial] Passwd --> " +
contrasenya );
                System.out.println("[ConsultaHistorial] Pacient --> " + pacient );

        }

        Security.insertProviderAt(new IAIK(), 2);
```

```
        /*
                Aquest és un usuari "genèric" del sistema i tant podria ser un
metge com un pacient
        */
        Metge jcasserras = new Metge( metge , contrasenya );

        /*
                Usuari gestor
        */
        try {

            GestorInterRemot gestor =
(GestorInterRemot)java.rmi.Naming.lookup("//127.0.0.1:" + rmi + "/GestorRMI");


        /*
                Implementem el protocol 1
        */
        org.jdom.Document a = jcasserras.procedure1 () ;
        org.jdom.Document b = gestor.procedure2 ( a );

        /*
                JCasserras ha de desxifrar el missatge amb la seva clau privada Su i
obtenir Ng, Ni i Id_Usuarig
        */
        org.jdom.Document c = jcasserras.pas3 ( b , "Consulta" );

        /*
                Gerard envia al gestor Pg[Ng, Consulta, Id_usuari] a G;
        */
        boolean autentificacio = gestor.pas4 ( c );

        if (autentificacio) {

                if (debug) System.out.println("[ConsultaHistorial] Autentificació
correcta" + pacient );

                /* A continuació comprovem si aquest pacient pertany a aquest
metge o si pacient == metge */

                if ((pacient.compareTo (metge) == 0 ) || ( gestor.verPacMetge(
pacient, metge ) )) {

                        Document histoClinic = gestor.procedure3 ( pacient ,
metge );
                        jcasserras.procedure4 ( histoClinic );

                } else {
                        System.out.println("[Error] El pacient " + pacient + " NO
està assignat al metge " + metge + " o aquest pacient està intentant accedir a un
historial que no és el seu");
                }
```

```
            }
             }   catch (Exception e)     {
                        e.printStackTrace();
                }
    }
}
```

Annex D. Relació de fitxers font continguts en la carpeta src/

- AutenticaUsuari.java: Classe amb les instruccions necessàries per a que un usuari s'autentiqui en el sistema.
- Base64.java: Classe per a codificar i decodificar en base 64.
- gestorBBDD.java: Classe que s'encarrega de tots els tràmits amb la base de dades.
- gestorCripto.java: Classe que s'encarrega de totes les gestions criptogràfiques.
- GestorInterRemot.java: Mètodes remots accessibles via RMI al Gestor.
- Gestor.java: Classe que implementa les funcionalitats del gestor del sistema.
- Metge.java: Classe que implementa les funcionalitats d'un professional sanitari en el sistema.
- P12.java: Classe per a treballar fitxers P12.
- Pacient.java: Classe que implementa les funcionalitats d'un pacient en el sistema.
- pfcGuiAfegeixApunt.java: Classe que implementa la interfície gràfica d'usuari que s'utilitza quan els professionals sanitaris afegeixen un nou apunt en la història clínica d'un determinat pacient.
- pfcGuiCodisCIE.java: Classe que implementa la interfície gràfica d'usuari que s'utilitza quan els professionals sanitaris escullen el codi CIE d'una determinada malaltia.
- pfcGuiGestor.java: Classe que implementa la interfície gràfica d'usuari per al gestor del sistema.
- pfcGuiGestorNewPacient.java: Classe que implementa la interfície gràfica d'usuari amb el formulari per a afegir un nou pacient en el sistema. Comentar que, malgrat la interfície està implementada, NO és funcional.
- pfcGuiLlistaPacients.java: Classe que implementa la interfície gràfica d'usuari amb el formulari en el qual apareixen, en la vista de professionals sanitaris, un llistat amb els pacients que té assignats.
- pfcGuiLogin.java: Classe que implementa la interfície gràfica d'usuari amb el formulari en el qual apareix la pantalla de login al sistema.
- pfcHistorialGuiMetge.java: Classe que implementa la interfície gràfica d'usuari amb el formulari principal de la vista dels professionals sanitaris.
- pfcHistorialGuiPacient.java: Classe que implementa la interfície gràfica d'usuari amb el formulari principal de la vista dels pacients.
- Usuari.java: Classe que implementa un usuari genèric del sistema (ja siguin pacients o metges).