Projecte fi de carrera:

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

Casimir Solbes Montaner
Enginyeria en Informàtica

Jordi Castellà Roca

Data Lliurament
07 de gener de 2008

Agraïments i dedicatòria

Al llarg dels darrers dos anys i mig que he estat cursant el segon cicle de l'enginyeria informàtica, han segut moltes les persones que d'alguna forma han tingut alguna relació, així aprofite aquesta oportunitat per a agrair-los la seva col·laboració:

Als companys, consultors i professors que he tingut al llarg de totes les assignatures que he cursat, per la col·laboració mitjançant els fòrums. A la tutora Carme Tallón per tota l'atenció prestada i en especial per resoldre les qüestions que li vaig plantejar respecte a l'avaluació d'estudis previs.

Al consultor d'aquest projecte Jordi Castellà, per l 'atenció i el suport ofereix durant el projecte.

A l 'empresa "Faus Group" en la que treballe des de fa més de sis anys, que m'ha facilitat entre altres coses la distribució de les vacances per tal de tindre més temps disponible per als estudis.

Finalment agrair a la Universitat Oberta de Catalunya la oportunitat que ens ofereix d'ampliar la formació acadèmica.

Dedique el projecte a la meua família, i especialment a la meva esposa Laura per haver suportat tot aquest període des que vaig començar els estudis a la UOC, el meu aïllament davant l'ordinador durant tantes nits i caps de setmana.

Resum

El PFC l'objectiu es dissenyar i implementar un esquema criptogràfic que gestione de forma segura els historials mèdics dels pacients a través d'una xarxa de comunicacions.

Per a implementar l'esquema criptogràfic, s'han dissenyat un conjunt de protocols criptogràfics per a cada acció o servei que ofereix el sistema:

- Consulta d'un historial.
- Consulta dels pacients assignats a un metge.
- Inserció de dades a l'historial mèdic.
- Protocol d'autentificació (protocol de Needham-Shroeder).

El sistema serà utilitzat bàsicament pels metges, pacients i els gestors del sistema, que disposaran segons cadascú dels actors les accions de registre d'usuaris, autentificació dels usuaris, i de manteniment (inserció, modificació, eliminació) de les dades l'historial mèdic.

Així cada servei tindrà definides les propietats de seguretat que li siguen definides, com la confidencialitat de les dades, l'autenticitat de la informació, la integritat de la informació i el no-repudi (que un usuari no puga negar l'execució d'una acció que l'ha realitzada).

Per a realitzar els objectius definits hem utilitzat les següents eines:

- El llenguatge de programació Java, utilitzant el SDK de Sun. Com que l'API del SDK no disposa dels criptosistemes necessaris s'utilitza la llibreria criptogràfica Apache XML-Security 1.2.1, llibreries que a més implementen la signatura digital XMLDSig i el sobre digital XMLEnc.
  El programari Eclipse es l'entorn de desenvolupament IDE utilitzat.

- Per a representar el disseny dels programes en UML, utilitzem l'eina MagicDraw.

- Utilitzem una infraestructura de clau pública PKI per a implementar els esquemes criptogràfics.

- La comunicació entre les aplicacions es realitzarà amb la tecnologia Remote Method Invocation (RMI) de Java.

- Les dades que s'intercanvien les diferents aplicacions estaran estructurades en XML, i utilitzarem la llibreria org.w3c.dom per a manipular les estructures XML amb Java.

- Per a gestionar les dades que necessiten persistència (registre de pacients i metges, historials mèdics, etc...), utilitzem la base de dades MySQL Server 5.0.

Paraules Clau

Certificat
Criptografia
Eclipse
Java
JDOM
MySQL
PKI
RMI
Seguretat
SQL
UML
XML
DTD

Nom de l’àrea de PFC

Seguretat informàtica

ÍNDEX

ÍNDEX DE FIGURES

1. INTRODUCCIÓ

1.1 JUSTIFICACIÓ DEL PFC I CONTEXT DE DESENVOLUPAMENT: PUNT DE PARTIDA I APORTACIO DEL PFC.

Les xarxes de comunicacions ens permeten accedir a un gran volum d'informació molt ràpidament sense la necessitat de desplaçar-nos, i amb independència de l'instant de temps. Aquests avantatges aporten un valor afegit més gran encara quan la persona que accedeix a la informació és un metge que consulta l'historial mèdic d'un pacient. Les dades de l'historial poden ajudar al metge a prendre una decisió correcta en la diagnosi i tractament que ha de rebre el pacient.
L'historial mèdic d'un pacient es una informació de gran valor, i per tant s'ha de protegir. Cal garantir que només serà modificada pel personal qualificat. Un altre punt important és la confidencialitat. Les dades mèdiques d'una persona són altament confidencials, i només poden ser accedides pel pacient o per personal mèdic.

Aquest projecte s'emmarca en un context on les comunicacions globals que suposa la xarxa d'Internet que són de bastant qualitat i rapidesa, i estan molt extenses en el seu ús sobre tot en l'àmbit professional. Així com respecte al maquinari disponible, que compta amb una suficient capacitat de procés per a dur a termes les operacions complexes en un temps raonable que requereixen els càlculs criptogràfics.

1.2 OBJECTIUS DEL PFC.

L'objectiu d'aquest PFC es implementar un esquema criptogràfic que garanteixi les necessitats de seguretat d'un historial mèdic que pot ser gestionat a través d'una xarxa de comunicacions.

Així les propietats de seguretat que cal complir són les següents:

- **Confidencialitat** : s'ha de preservar la confidencialitat de les dades de l'historial mèdic dels pacients.

- **Autenticitat** : la informació que es guarda en el sistema ha de disposar d'una prova de la seva autenticitat.

- **Integritat** : un cop la informació ha estat generada s'ha de garantir en tot moment la seva integritat.

- **No-repudi** : si un usuari del sistema fa una certa acció més tard no pot negar que l'hagi realitzada.

1.3 ENFOCAMENT I MÈTODE SEGUIT.

El mètode seguit per arribar a aconseguir els objectius del PFC pretén reduir la complexitat global i facilitar les tasques d'implementació, per això és realitzarà una implementació incremental del sistema en diferents fases, on per cadascuna de les fases que s'implementarà, és realitzaran proves unitàries i d'integració amb les fases anteriors ja implementades.

Així s'han identificat les següents fases:

1. Instal·lació Apache XML-Security 1.2.1 i PKI: llibreria criptogràfica Apache XML-Security 1.2.1 i creació de la infraestructura de clau pública PKI.

2. Esquema criptogràfic i representació de dades en XML: Definició i implementació del conjunt de protocols de cada tipus d'acció o servei, que amb la llibreria Apache XML-Security 1.2.1 implementen la signatura digital XMLDSig i el sobre digital XMLEnc.

3. Comunicació dels components: RMI.

4. Gestió de la informació: BD.

5. Interfície del pacient.

6. Interfície del metge.

7. Interfície del gestor del sistema.

8. Documentació.

1.4 PLANIFICACIÓ DEL PROJECTE.

El PFC s'ha desenvolupat amb la planificació que a continuació es detalla:

- Del 19 al 23 de setembre:
  - Instal·lació i proves de la llibreria criptogràfica Apache XML-SECURITY 1.2.1
  - Creació dels certificats genèrics per a l'autoritat de Certificació, el pacient, el metge i el gestor del sistema.

- Del 24 de setembre al 4 de novembre:
  - Disseny, implementació, tests i documentació de l'esquema criptogràfic i de la representació de les dades amb XML.

- Del 5 al 18 de novembre:
    - Disseny, implementació, tests i documentació de la comunicació dels components amb la tecnologia RMI.

- Del 19 de novembre al 2 de desembre:
    - Disseny, implementació, tests i documentació de la gestió de la informació: Base de dades.

- Del 3 al 16 de desembre:
    - Disseny, implementació, tests i documentació de la interfície del pacient i del metge.

- Del 17 al 30 de desembre:
    - Disseny, implementació, tests i documentació de la interfície del gestor del sistema.

- Del 31 de desembre al 6 de gener:
    - Integració de la documentació de la memòria i la presentació del projecte.

En la següent imatge es presenta de manera gràfica la planificació del projecte. Presentació gràfica realitzada amb l'eina GanttProject.

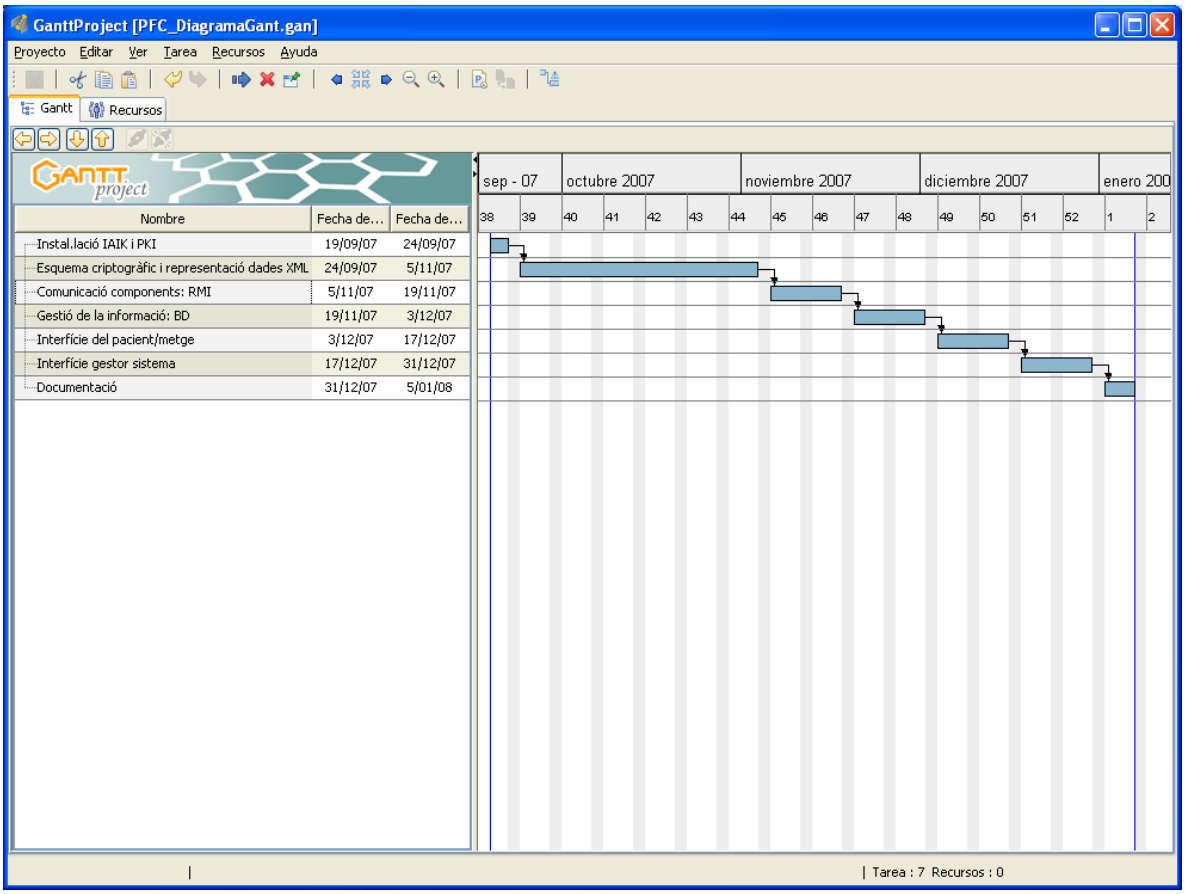


Figura 1.1: Planificació del projecte

1.5 PRODUCTES OBTINGUTS.

Com a resultat del PFC s'obtindrà un sistema amb els components de programari següents:

- **Aplicació metge**: permet que un metge pugui consultar i fer el manteniment de l'historial d'un pacient de forma segura.
- **Aplicació pacient**: el pacient utilitza aquest aplicació per consultar les dades del seu historial.
- **Gestor central**: el gestor central és qui té el repositori amb tots els historials i en controla la seva gestió. A partir d'aquest gestor es poden donar d'alta al sistema els pacients i els metges.

1.6 BREU DESCRIPCIÓ DELS ALTRES CAPÍTOLS DE LA MEMÒRIA.

- Capítol 2: Instal·lació APACHE XML-SECURITY 1.2.1 i PKI.

Descripció del procés d'instal·lació de la llibreria criptogràfica Apache XML-SECURITY 1.2.1 i la generació dels certificats de PKI amb l'eina OpenSSL.

- Capítol 3. Esquema criptogràfic i Representació de les dades XML.

Descripció del disseny del conjunt de protocols que conformen l'esquema criptogràfic i la descripció dels documents XML utilitzats per fer les transferències de dades que s'envien durant l'execució dels protocols criptogràfics.

- Capítol 4: Comunicació dels components: RMI.

Descripció de la comunicació dels diferents components del sistema mitjançant la tecnologia RMI.

- Capítol 5: Gestió de la informació: Base de dades.

Disseny de la base de dades utilitzada per emmagatzemar els historials mèdics.

- Capítol 6: Interfície del pacient.

Descripció i anàlisi de la interfície que permet al pacient interactuar amb el sistema.

- Capítol 7: Interfície del metge.

Descripció i anàlisi de la interfície que permet al metge interactuar amb el sistema.

- Capítol 8. Interfície gestor del sistema.

Descripció i anàlisi de la interfície que permet utilitzar les funcionalitats del gestor.

- Conclusions

Descripció de l'acompliment i consecució dels objectius del PFC, destacant les consideracions més importants experimentades durant la implementació.

2. INSTAL·LACIÓ APACHE XML-SECURITY 1.2.1 I PKI

2.1 XML-SECURITY 1.2.1

2.1.1 INTRODUCCIÓ

Per a la implementació de l'esquema criptogràfic utilitzarem el SDK de Sun, que com no disposa de la majoria de criptosistemes necessaris, aleshores hem triat la llibreria criptogràfica Apache XML-Security 1.2.1 degut a que implementen la signatura XMLDSig i el sobre digital XMLEnc.

La llibreria Apache XML-Security 1.2.1 esta feta per a ser utilitzada en aplicacions Java, com a proveïdor de funcions criptogràfiques. Implementa diversos algoritmes criptogràfics de generació de claus, simètriques i asimètriques, codificació i descodificació de dades, funcions hash, gestió de claus i certificats,...

2.1.2 PASOS PER A INSTAL·LAR XML-SECURITY 1.2.1

Els passos per instal·lar la llibreria criptogràfica XML-SECURITY 1.2.1 són els següents:

1. descarregar la versió última del JDK de SUN i instal·lar-lo. Cal vigilar que no s'utilitzi la màquina virtual de MS al compilar o executar.

2. descarregar la llibreria versió de XML-SECURITY 1.2.

3. descarregar les polítiques de seguretat de Java que permeten emprar qualsevol longitud de clau, Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 5.0 RC

4. (Windows) Copiar els arxius commons-logging.jar, commons-logging-api.jar, xalan.jar, xercesImpl.jar, xml-apis.jar, xmlsec-1.2.1.jar, xmlsecSamples-1.2.1.jar i xmlsecTests-1.2.1.jar als directoris:

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

5. (Linux) Copiar l'arxiu commons-logging.jar, commons-logging-api.jar, xalan.jar, xercesImpl.jar, xml-apis.jar, xmlsec-1.2.1.jar, xmlsecSamples-1.2.1.jar i xmlsecTests-1.2.1.jar al directori:

- $JAVA_HOME/jre/lib/ext

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

   - local_policy.jar
   - US_export_policy.jar

   Copiar-los a:

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

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

   - local_policy.jar
   - US_export_policy.jar

   Copiar-los a:

   - $JAVA_HOME/jre/lib/security

2.2 PKI

2.2.1 INTRODUCCIÓ.

Cada protocol criptogràfic implementat en aquest projecte necessitarà que els actors del sistema disposen d'una parella de claus i el seu corresponent certificat.

Per tal de gestionar els certificats (emissió, revocació, etc) d'un grup d'usuaris s'empra una infraestructura de clau pública. Típicament per fer referència a una infraestructura s'utilitzen les sigles PKI, que corresponen al terme en anglès Public Key Infrastructure.

Una PKI consta d'una autoritat de certificació notada amb CA, aquestes sigles corresponen al terme en anglès Certification Authority. Un altre component de la PKI són les autoritats de registre, notades amb les sigles RA Registry Authority. Quan un usuari vol obtenir un certificat normalment realitza els passos següents:

- Crea una parella de claus i realitza una petició de certificat mitjançant una RA.
- La RA valida la identitat de l'usuari que ha demanat el certificat i envia la petició a la CA.
- La CA rep les peticions de la RA i emet els certificats.

Per a construir la infraestructura de clau publica PKI, que anem a utilitzar en el PFC, utilitzarem l'eina OpenSSL, que es un paquet d' eines d'administració i llibreries de codi obert, que implementa els protocols SSL v2/v3 i TLS v1.

2.2.2 PASSOS A SEGUIR PER GENERAR ELS ARXIUS NECESSARIS.

Seguint aquests passos crearem un parella de claus i un certificat pel Gestor, un Metge, i un Pacient. Finalment veurem com encapsular una parella de claus, el certificat corresponent, el certificat de la CA en un fitxer PKCS#12.

Crear una parella de claus un certificat de CA

En aquesta primera fase crearem una parella de claus del criptosistema RSA, que correspondran a la CA que utilitzarem al PFC. Els passos són els següents:

1. Crear una parella de claus del criptosistema RSA:

    - La comanda per crear la parella de claus és:

openssl genrsa -des3 -out CA.key 2048

    - La longitud de la parella de claus es de 2048 bits.
    - La parella de claus estarà xifrada amb Triple DES.
    - Com password poseu-hi uoc0506. Si hem de revisar algun codi serà més fàcil que utilitzem el mateix password.

2. Crear un certificat autosignat:

Openssl req -new -sha1 -x509 -key CA.key -out CA.crt -days 360

    - Poseu les dades al certificat de la CA:
    - Country Name (2 letter code) [AU]:ES
    - State or Province Name (full name) [SomeState]:Barcelona
    - Locality Name (eg, city) []:Barcelona
    - Organization Name (eg, company) [Internet Widgits Pty Ltd]:UOC
    - Organizational Unit Name (eg, section) []:PFC Seguretat
    - Common Name (eg, YOUR name) []:CA PFC Seguretat
    - Email Address []:jcastellar@uoc.edu

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

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

5. Podeu verificar que el certificat és correcte amb les comandes següents:

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

Crear una parella de claus i emetre un certificat del Gestor

Primer crearem una parella de claus RSA de 1024 bits, generarem una petició de certificat emetrem un certificat partir de la petició anterior. Finalment guardarem el certificat la clau privada en un fitxer amb el format Personal Information Exchange Syntax Standard (PKCS#12).

1. Crear una parella de claus del criptosistema RSA.

   - Utilitzeu la comanda següent:

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

   - La longitud del parell de claus és de 1024 bits.
   - La parella de claus està xifrada amb Triple DES (-des3).
   - Utilitzeu el password següent per protegir la parella de claus: uoc0506.

2. Per a crear una petició de certificat utilitzeu la comanda:

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

   - El fitxer openssl.conf s'adjunta en l'annex A.
   - Exemple dels camps omplir:
     - Country Name (2 letter code) [ES]:
     - State or Province Name (full name) [Barcelona]:
     - Locality Name (eg, city) [Barcelona]:
     - Organization Name (eg, company) [UOC]:
     - Organizational Unit Name (eg, section) [PFC Seguretat]:Gestio
     - Common Name (eg, YOUR name) []:Gestor
     - D.N.I or N.S.S. [00000000A]:
     - Email Address []:jcastellar@uoc.edu

   - La funció resum (hash) és la sha1.

- El nom del fitxer de sortida és: **Gestor.csr**
- Podeu verificar que la petició es correcta amb la comanda següent:

```
openssl req -in Gestor.csr -verify -text -noout
```

3. Emetem el certificat del Gestor partir de la petició de certificat utilitzant la comanda següent:

```
openssl x509 -req -in Gestor.csr -days 180 -CA CA.crt -CAkey
CA.key -CAcreateserial -extfile openssl.conf -extensions usr_cert -
out Gestor.crt
```

- El certificat serà vàlid 180 dies.
- El fitxer amb el certificat és **Gestor.crt**.
- Podeu verificar el certificat amb:

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

4. Crearem el fitxer que conté la clau privada el certificat en format PKCS#12.

- Utilitzarem la comanda següent:

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

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

Crear una parella de claus i emetre un certificat del Pacient

Primer crearem una parella de claus RSA de 1024 bits, generarem una petició de certificat emetrem un certificat a partir de la petició anterior. Finalment guardarem el certificat la clau privada en un fitxer amb el format Personal Information Exchange Syntax Standard (PKCS#12).

1. Crear una parella de claus del criptosistema RSA.

- Utilitzeu la comanda següent:

openssl genrsa -des3 -out Pacient.key 1024

- La longitud del parell de claus és de 1024 bits.
- La parella de claus està xifrada amb Triple DES (des3).
- Utilitzeu el password següent per protegir la parella de claus: uoc0506.

2. Para crear una petició de certificat utilitzeu la comanda:

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

- El fitxer openssl.conf s'adjunta en l'annex A.
- Exemple dels camps omplir:

  - Country Name (2 letter code) [ES]:
  - State or Province Name (full name) [Barcelona]:
  - Locality Name (eg, city) [Barcelona]:
  - Organization Name (eg, company) [UOC]:
  - Organizational Unit Name (eg, section) [PFC Seguretat]:Pacients
  - Common Name (eg, YOUR name) []:Nom Pacient
  - D.N.I or N.S.S. [00000000A]: 00000001B
  - Email Address []:jcastellar@uoc.edu

- La funció resum (hash) és la sha1.
- El nom del fitxer de sortida és: **Pacient.csr**
- Podeu verificar que la petició es correcta amb la comanda següent:

openssl req -in Pacient.csr -verify -text -noout

3. Emetem el certificat del Pacient partir de la petició de certificat utilitzant la comanda següent:

openssl x509 -req in Pacient.csr -days 180 -CA CA.crt -CAkey CA.key -CAcreateserial -extfile openssl.conf -extensions usr_cert -out Pacient.crt

- El certificat serà vàlid 180 dies.
- El fitxer amb el certificat és **Pacient.crt**.
- Podeu verificar el certificat amb:

openssl verify -CAfile CA.crt Pacient.crt

4. Crearem el fitxer que conté la clau privada i el certificat en format PKCS#12.

    - Utilitzarem la comanda següent:

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

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

Crear una parella de claus i emetre un certificat del Metge

Primer crearem una parella de claus RSA de 1024 bits, generarem una petició de certificat i emetrem un certificat a partir de la petició anterior. Finalment guardarem el certificat i la clau privada en un fitxer amb el format Personal Information Exchange Syntax Standard (PKCS#12).

1. Crear una parella de claus del criptosistema RSA.

    - Utilitzeu la comanda següent:

```
openssl genrsa -des3 -out Metge.key 1024
```

    - La longitud de la parella de claus és de 1024 bits.
    - La parella de claus està xifrada amb Triple DES (des3).
    - Utilitzeu el password següent per protegir la parella de claus: uoc0506.

2. Para crear una petició de certificat utilitzeu la comanda:

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

    - El fitxer openssl.conf s'adjunta en l'annex A.

    - Exemple dels camps omplir:
      - Country Name (2 letter code) [ES]:
      - State or Province Name (full name) [Barcelona]:
      - Locality Name (eg, city) [Barcelona]:
      - Organization Name (eg, company) [UOC]:
      - Organizational Unit Name (eg, section) [PFC Seguretat]: Metges

- Common Name (eg, YOUR name) []:Nom Metge
- D.N.I or N.S.S. [00000000A]:00000002C
- Email Address []:jcastellar@uoc.edu

- La funció resum (hash) és la sha1.
- El nom del fitxer de sortida és: **Metge.csr**
- Podeu verificar que la petició es correcta amb la comanda següent:

```
openssl req -in Metge.csr -verify -text -noout
```

3. Emetem el certificat del Metge partir de la petició de certificat utilitzant la comanda següent:

```
openssl x509 -req -in Metge.csr -days 180 -CA CA.crt -CAkey CA.key -CAcreateserial -extfile openssl.conf -extensions usr_cert -out Metge.crt
```

- El certificat serà vàlid 180 dies.
- El fitxer amb el certificat és **Metge.crt**.
- Podeu verificar el certificat amb:

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

4. Crear el fitxer que conté la clau privada i el certificat en format PKCS#12.

- Utilitzarem la comanda següent:

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

- El fitxer de sortida és: Metge.p12

3. ESQUEMA CRIPTOGRÀFIC

3.1 INTRODUCCIÓ

A partir de l'estudi de les necessitats del sistema amb la identificació dels actors, la definició de les accions o serveis que és realitzen i la informació que es gestiona en cada operació, s'han definit els requisits de seguretat. Així doncs per tal d'implementar els requisits de seguretat s'han dissenyat un conjunt protocols que conformen l'esquema criptogràfic que es descriu en aquest capítol.

3.2 ACTORS DEL SISTEMA

Comptarem inicialment amb tres actors imprescindibles en el sistema:

- **Pacients** : Són els propietaris de les informacions que continguen cadascun dels seus historials.
- **Metges** : Són els responsables de la introducció de les informacions mèdiques, en els historials dels pacients.
- **Gestor del sistema** :Es el responsable del correcte funcionament del sistema i del manteniment dels pacients i dels metges.

3.3 ACCIONS / SERVEIS

El sistema ofereix les següents accions / serveis, segons cadascun dels actors :

- **Registre d'usuaris**: el gestor del sistema donarà d'alta, i farà el manteniment dels pacients i els metges en el sistema.
- **Autentificació dels usuaris**: els usuaris s'han d'autentificar de forma segura davant del sistema. A més aquesta operació s'utilitzarà en l'execució de cadascuna de les accions/serveis del sistema.
- **Consulta d'un historial**: els pacients podran consultar el seu historial, així com els metges que també podran consultar els historials del pacients que tinguen assignats.
- **Inserció de dades a l'historial**: els metges podran afegir noves dades als expedients dels seus pacients.
- **Consulta dels pacients assignats a un metge**: els metges obtindran un llistat amb el identificador dels seus pacients.

3.4 REQUISITS DE SEGURETAT

La gestió de les dades d'un historial mèdic, ha de complir les següents propietats:

- **Confidencialitat**: la informació mèdica continguda en un historial sols ha de ser accessible pel pacient, i pel metge que el tinga assignat.

- **Autenticitat**:: el sistema ha de garantir que la informació que conté un historial mèdic és autèntica i no ha segut modificada per una persona no autoritzada.

- **Integritat**: el sistema ha de garantir que la informació que conté un historial mèdic és completa i no s'ha eliminat cap informació.

- **No-repudi**: el sistema ha de garantir que quan un usuari (metge) realitze una inserció de dades a l'historial, aquest usuari no podrà negar que ell no ho ha fet.

3.5 NOTACIÓ UTILITZADA EN ELS PROTOCOLS

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

- **K** : clau d'un criptosistema simètric.

- **EK(M)** : xifratge simètric d'un missatge M amb la clau K.

- **DK(C)** : desxifratge simètric del criptograma C amb la clau K.

- **(PEntitat, SEntitat)** : parella de claus asimètriques propietat d'Entitat, on P correspon a la clau pública, i S a la privada.

- **SEntitat[M]** : signatura digital del missatge M amb la clau privada S d'Entitat.

- **PEntitat[M]** : 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.

En els protocols cada usuari U s'identifica amb Id_usuariU i disposa d'una parella de claus (Pu, Su) amb el corresponent certificat CertU. En el cas del gestor G el seu identificador d'usuari es representa com id_usuariG, i disposa de la parella a de claus (Pg, Sg).

3.6 AUTENTIFICACIÓ DELS USUARIS

El protocol d'autentificació representa el protocol de Needham-Shroeder. Aquest protocol s'utilitzarà en l'execució de cadascuna de les accions/serveis del sistema, amb una adaptació per tal que en el mateix procés d'autentificació l'usuari sol·licite l'acció que vol realitzar.

1. U realitza les operacions següents:

    a. Obtenir un valor de forma aleatòria, Ni;

    b. Xifrar Ni i Id_usuariU amb la clau pública de G, Pg[Autentificació, Ni, Id_usuariU].

    c. Enviar Pg[Ni, Id_usuariU] a G.

2. G realitza les operacions següents:

    a. Desxifrar Pg[Ni, Id_usuariU] amb Sg, i obtenir Ni i Id_usuariU.

    b. Obtenir el certificat de U a partir de Id_usuariU. A partir del certificat obtindrà la clau pública de l'usuari Pu.

    c. Obtenir un valor de forma aleatòria, Ng;

    d. Xifrar Ni, Ng, Id_usuariG, amb la clau pública Pu de U, Pu[Ni, Ng, Id_usuariG];

    e. 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 N' i = Ni fer:

        i. Xifrar Consulta, Ng, Id_usuari amb la clau pública Pg de G, Pg[Consulta, Ng, Id_usuari]. Consulta indica l'acció que es vol realitzar, i Id_Usuari l'usuari que requereix l'operació;

ii. Enviar Pg[Consulta, Ng, Id_ usuari] a G;

c. Sinó retornar error.

4. G realitza les operacions següents:

a. Desxifrar Pg[Consulta, Ng, Id_usuari] amb la clau privada Sg, i obtenim Consulta, N'g, Id usuari.

c. Si N'g = Ng el gestor G i l'usuari U estan autentificats bilateralment.

Diagrama cas d'us

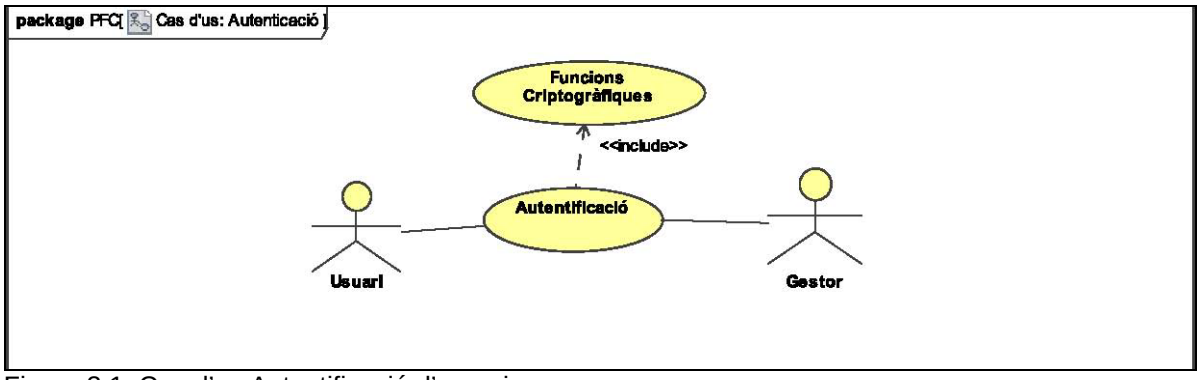


Figura 3.1: Cas d'us Autentificació d'usuaris

Diagrama de seqüència

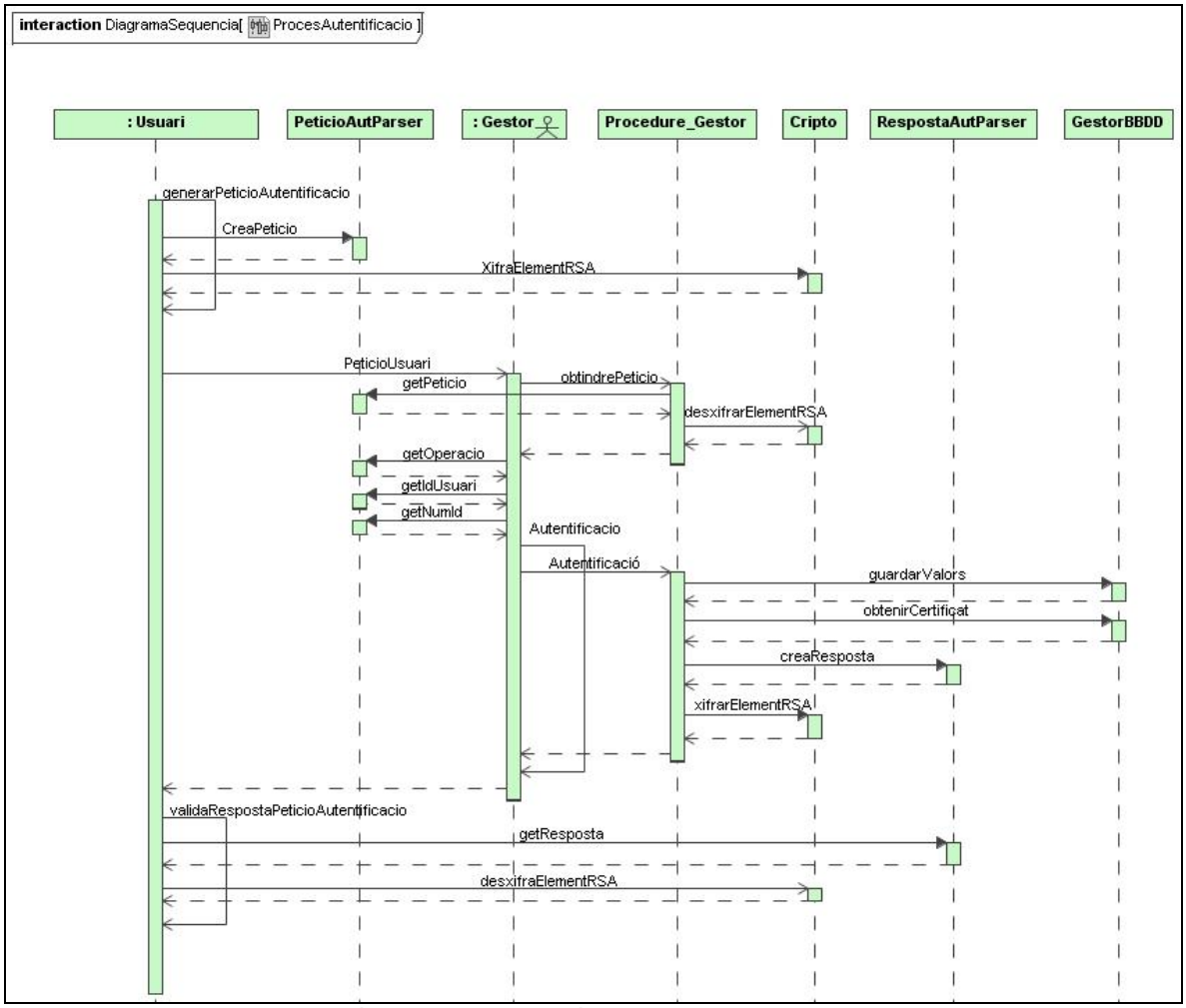


Figura 3.2: Diagrama de seqüència Autentificació d'usuaris

3.7 CONSULTA D'UN HISTORIAL

Aquest protocol pot ser utilitzat pels usuaris U metge o pacient per a consultar un historial mèdic. El gestor G verifica en cada cas el tipus d'usuari i només facilita l'historial si l'usuari hi té accés.

Protocol Consulta d'un historial

1. U realitza les operacions següents:

    a. Obtenir un valor de forma aleatòria, Ni;

    b. Xifrar Autentificació, Ni i Id_usuariU amb la clau pública de G, Pg[Autentificació, Ni, Id_usuariU]. Autentificació indica que es vol realitzar l'operació d'autentificació davant el sistema del l'usuari Id_usuariU.

c. Enviar Pg[Autentificació, Ni, Id_usuariU] a G.

2. G realitza les operacions següents:

a. Desxifrar Pg[Autentificació, Ni, Id_usuariU] amb Sg, i obtenir Autentificació, Ni i Id_usuariU.

b. Obtenir el certificat de U a partir de Id_usuariU. El sistema disposa d'una Base de Dades (BD) on per cada Id_usuariU trobem el seu certificat corresponent.

c. Obtenir un valor de forma aleatòria, Ng;

d. Guardar a la BD els valors Ni i NG associats amb U;

e. Xifrar Ni, Ng, Id_usuariG, amb la clau pública Pu de U, Pu[Ni, Ng, Id_usuariG];

f. 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 N' i = Ni fer:

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

ii. Enviar Pg[Consulta, Ng, Id_ usuari] a G;

c. Sinó retornar error.

4. G realitza les operacions següents:

a. Desxifrar Pg[Consulta, Ng, Id_usuari] amb la clau privada Sg, i obtenim Consulta, N'g, Id usuari.

b. Recuperar Ng de la BD i esborrar el registre amb Ng i Ni de la BD.

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)) aleshores fer:

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. Enviar Pu[H] a U.

ii. Sinó retornar error;

d. Sinó retornar error;

5. U realitza les operacions següents:

a. Desxifrar Pu[H] amb la clau privada Su de U, Su[Pu[H]];

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

i. Verificar la signatura digital de M;

ii. Verificar la signatura digital de G;

iii. Verificar la seqüència;

c. Mostrar H.

Cas d'ús Consulta Historial

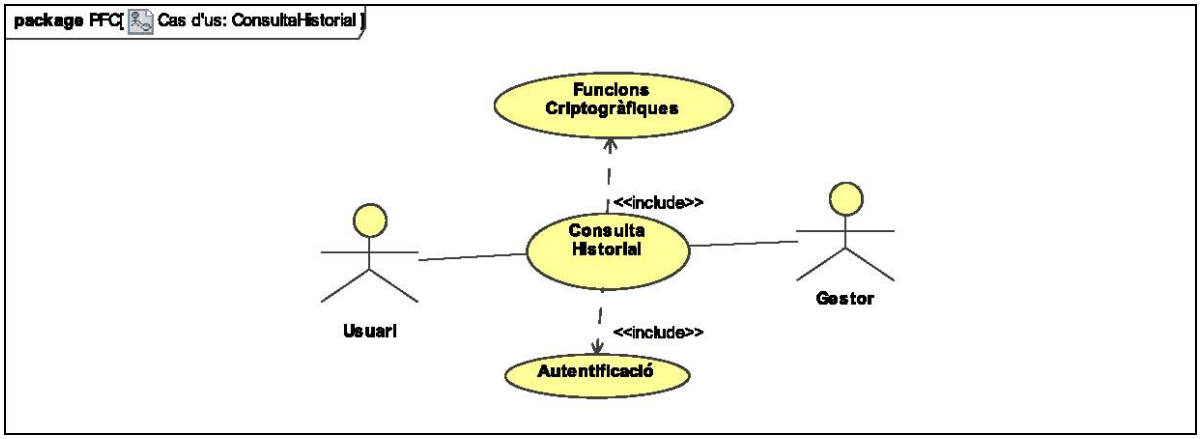


Figura 3.3: Cas d'us Consulta Historial

Diagrama de seqüència

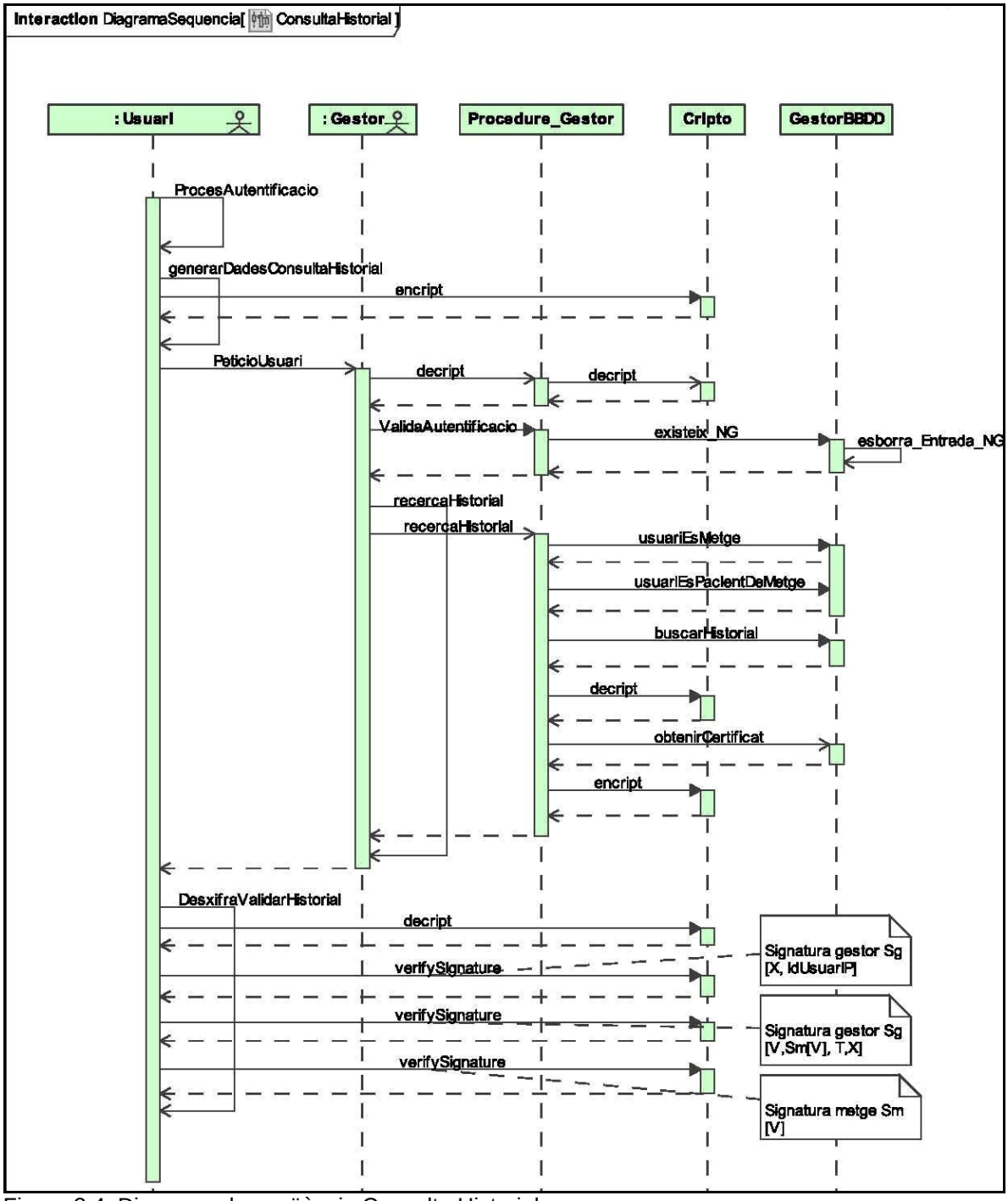


Figura 3.4: Diagrama de seqüència Consulta Historial

3.8 CONSULTA DELS PACIENTS ASSIGNATS A UN METGE

Una operació típica d'un metge es buscar l'historial d'un dels seus pacients. Amb el següent protocol un metge pot obtenir el llistat del seus pacients.

Protocol Consulta dels pacients assignats a un metge:

1. U realitza les operacions següents:

    a. Obtenir un valor de forma aleatòria, Ni;

    b. Xifrar Autentificació, Ni i Id_usuariU amb la clau pública de G, Pg[Autentificació, Ni, Id_usuariU]. Autentificació indica que es vol realitzar l'operació d'autentificació davant el sistema del l'usuari Id_usuariU.

    c. Enviar Pg[Autentificació, Ni, Id_usuariU] a G.

2. G realitza les operacions següents:

    a. Desxifrar Pg[Autentificació, Ni, Id_usuariU] amb Sg, i obtenir Autentificació, Ni i Id_usuariU.

    b. Obtenir el certificat de U a partir de Id_usuariU. El sistema disposa d'una Base de Dades (BD) on per cada Id_usuariU trobem el seu certificat corresponent.

    c. Obtenir un valor de forma aleatòria, Ng;

    d. Guardar a la BD els valors Ni i NG associats amb U;

    e. Xifrar Ni, Ng, Id_usuariG, amb la clau pública Pu de U, Pu[Ni, Ng, Id_usuariG];

    f. 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 N' i = Ni fer:

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

        ii. Enviar Pg[LlistatPacients, Ng] a G;

    c. Sinó retornar error.

4. G realitza les operacions següents:

a. Desxifrar Pg[LlistatPacients, Ng] amb la clau privada Sg, i obtenim LlistatPacients, N'g.

b. Recuperar Ng de la BD i esborrar el registre amb Ng i Ni de la BD.

c. Si N'g = Ng fer:

    i. Si (Id_usuariU és metge) aleshores fer:

        1. Cercar a la BD els pacients assignats al metge Id_usuariU, 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_usuariU PU, PU[{Id_usuari1,...,Id_usuarin}, SG[{Id_usuari1,...,Id_usuarin}]]

        4. Enviar a U PU[{Id_usuari1,...,Id_usuarin}, SG[{Id_usuari1,...,Id_usuarin}]]

    ii. Sinó retornar error;

d. Sinó retornar error;

5. U realitza les operacions següents:

a. 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}]

b. Verificar la signatura digital SG[{Id_usuari1,...,Id_usuarin}] amb la clau pública Pg de G.

c. Si la verificació anterior es correcta mostrar {Id usuari1, . . . , Id usuarin}.

Cas d'us

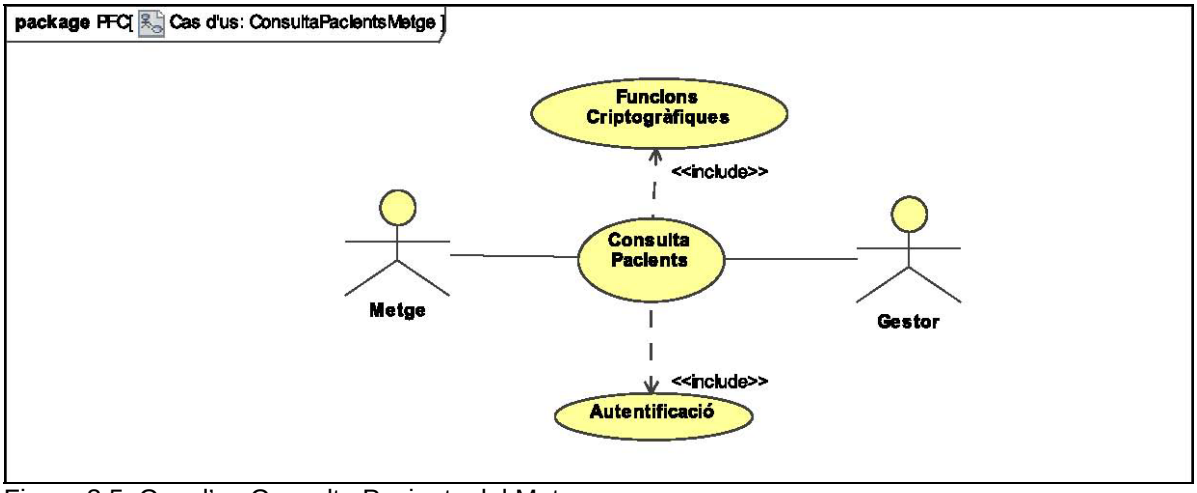


Figura 3.5: Cas d'us Consulta Pacients del Metge

Diagrama de seqüència

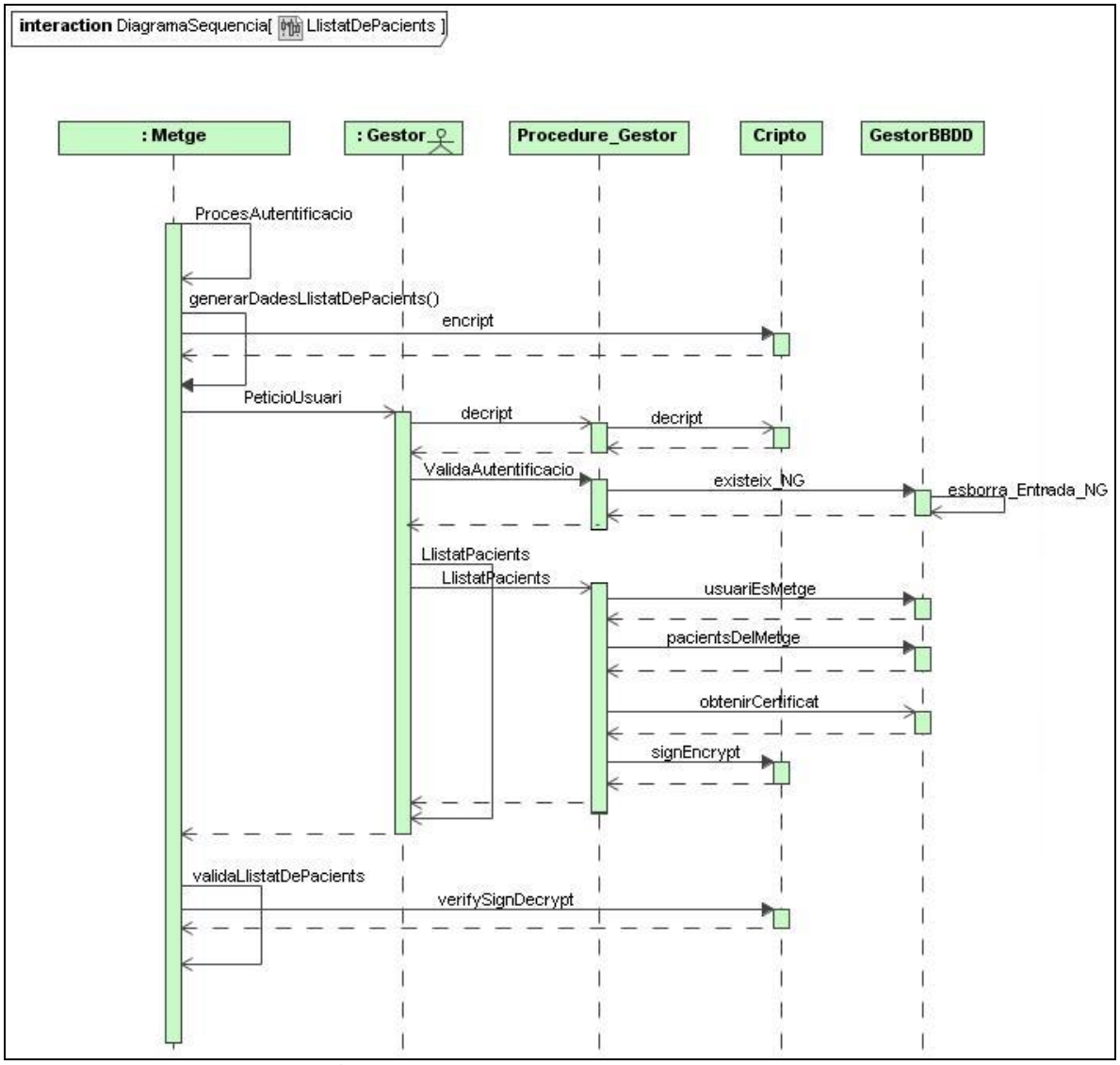


Figura 3.6: Diagrama de seqüència Consulta Pacients del Metge

3.9 INSERCIÓ DE DADES A L'HISTORIAL MÈDIC

En aquest protocol es suposa que prèviament a la inserció de les dades el metge M ha consultat l'historial del pacient P i per tant coneix el Id_usuariP. El gestor un cop rep un visita V d'un pacient P verifica que ha estat signada pel metge M assignat al pacient. A continuació afegeix la visita a l'historial H. Per garantir que l'ordre de visites no es modifica s'afegeix a cada visita una marca temporal T i un número de sèrie X. Amb aquestes dades es pot saber l'instant de la visita i l'ordre que han seguit. La visita V, el temps T i el número de sèrie son signats pel gestor G. Si un atacant elimina un registre al mig de l'historial es detectarà perquè hi haurà un salt d'un número en la sèrie de les visites. L'atacant no podrà refer la seqüència sense la clau privada del gestor G. Suposem que aquesta clau està ben protegida. A continuació el gestor G xifra les dades de la visita amb la seva clau

pública i ho guarda a la BD. Si un atacant accedeix a la BD no pot veure les dades confidencials. Finalment afegeix a l'historial una signatura digital de quin és l´ultim número de sèrie X de l'historial H. Si un atacant elimina la última visita es detectarà perquè hi haurà un salt entre la última visita i el número de sèrie X signat.

Protocol Inserció de dades a l'historial mèdic:

1. M realitza les operacions següents:

    a. Obtenir un valor de forma aleatòria, Ni;

    b. Xifrar Autentificació, Ni i Id_usuariM amb la clau pública de G, Pg[Autentificació, Ni, Id_usuariM]. Autentificació indica que es vol realitzar l'operació d'autentificació davant el sistema del l'usuari Id_usuariM.

    c. Enviar Pg[Autentificació, Ni, Id_usuariM] a G.

2. G realitza les operacions següents:

    a. Desxifrar Pg[Autentificació, Ni, Id_usuariM] amb Sg, i obtenir Autentificació, Ni i Id_usuariM.

    b. Obtenir el certificat de M a partir de Id_usuariM. El sistema disposa d'una Base de Dades (BD) on per cada Id_usuariM trobem el seu certificat corresponent.

    c. Obtenir un valor de forma aleatòria, Ng;

    d. Guardar a la BD els valors Ni i NG associats amb U;

    e. Xifrar Ni, Ng, Id_usuariG, amb la clau pública Pm de U, Pm[Ni, Ng, Id_usuariG];

    f. Enviar Pm[Ni, NG, Id_usuariG] 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 N' i = 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[InserirVisita, Ng, V, Sm[V]]. Inserir visita indica que es vol afegir V a l'historial del pacient P.

ii. Enviar Pg[InserirVisita, Ng, V, Sm[V]] a G;

c. Sinó retornar error.

4. G realitza les operacions següents:

a. Desxifrar Pg[InserirVisita, Ng, V, Sm[V]] amb la clau privada Sg, i obtenim InserirVisita, N'g, V, Sm[V].

b. Recuperar Ng de la BD i esborrar el registre amb Ng i Ni de la BD.

c. Si N'g = Ng fer:

i. Obtenir Id_usuariP a partir de V.

ii. Si (Id_usuariM és metge i Id_usuariP és un pacient de Id_usuariM)) aleshores fer:

1. Verificar la signatura digital Sm[V] amb la clau pública Pm

2. Obtenir l'instant de temps actual T

3. Obtenir el número de sèrie X de la última visita de l'historial H del pacient Id_usuariP.

4. Incrementar en una unitat X, X + 1

5. Signar V, Sm[V], T, X +1, amb la clau privada Sg de G, Sg[V, Sm[V], T, X+1]

6. Xifrar V i Sm[V] amb la clau pública Sg de G, Pg[V, Sm[V]].

7. Signar Id_usuariP i X+1 amb la clau privada SG de G, SG[X + 1, Id_usuariP]

8. Guardar a la BD Pg[V, Sm[V]], X + 1, T, Sg[V, Sm[V], T, X + 1] i SG[X + 1, Id_usuariP]

9. Retornem V amb la marca de temps T i el número de sèrie X, xifrada amb la clau pública del metge.

iii. Sinó retornar error;

d. Sinó retornar error;

Cas d'us

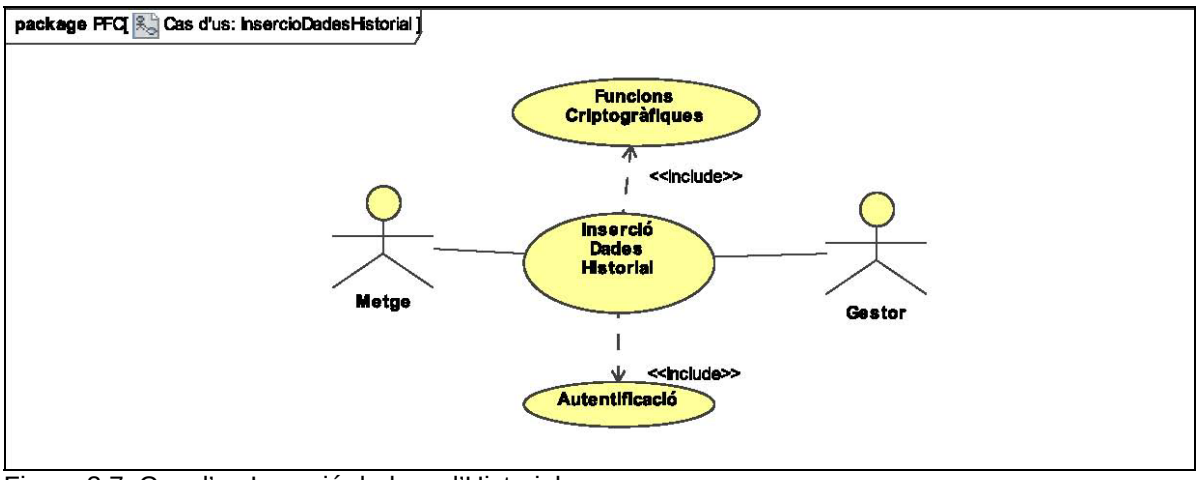


Figura 3.7: Cas d'us Inserció dades a l'Historial

Diagrama de seqüència

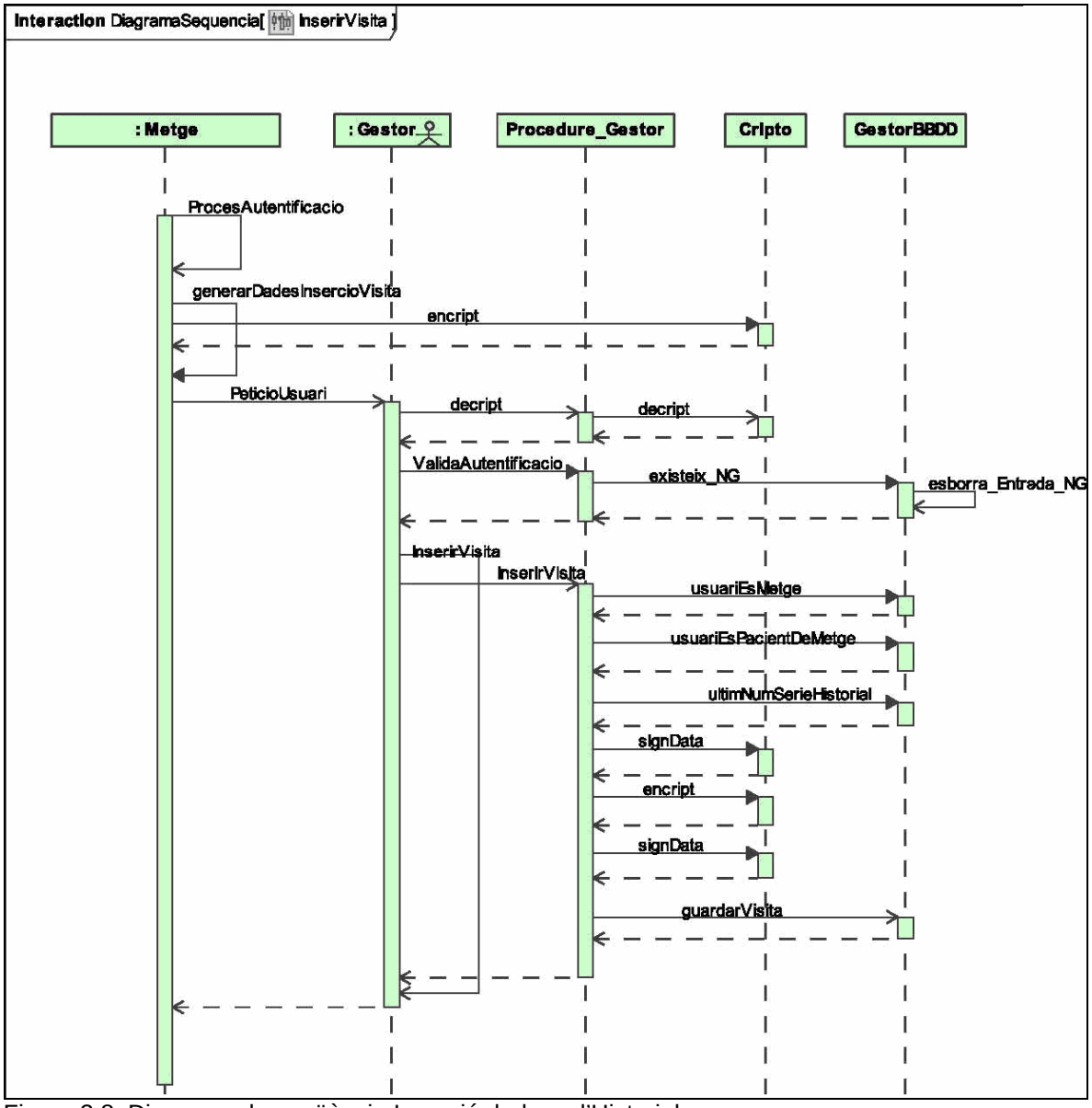
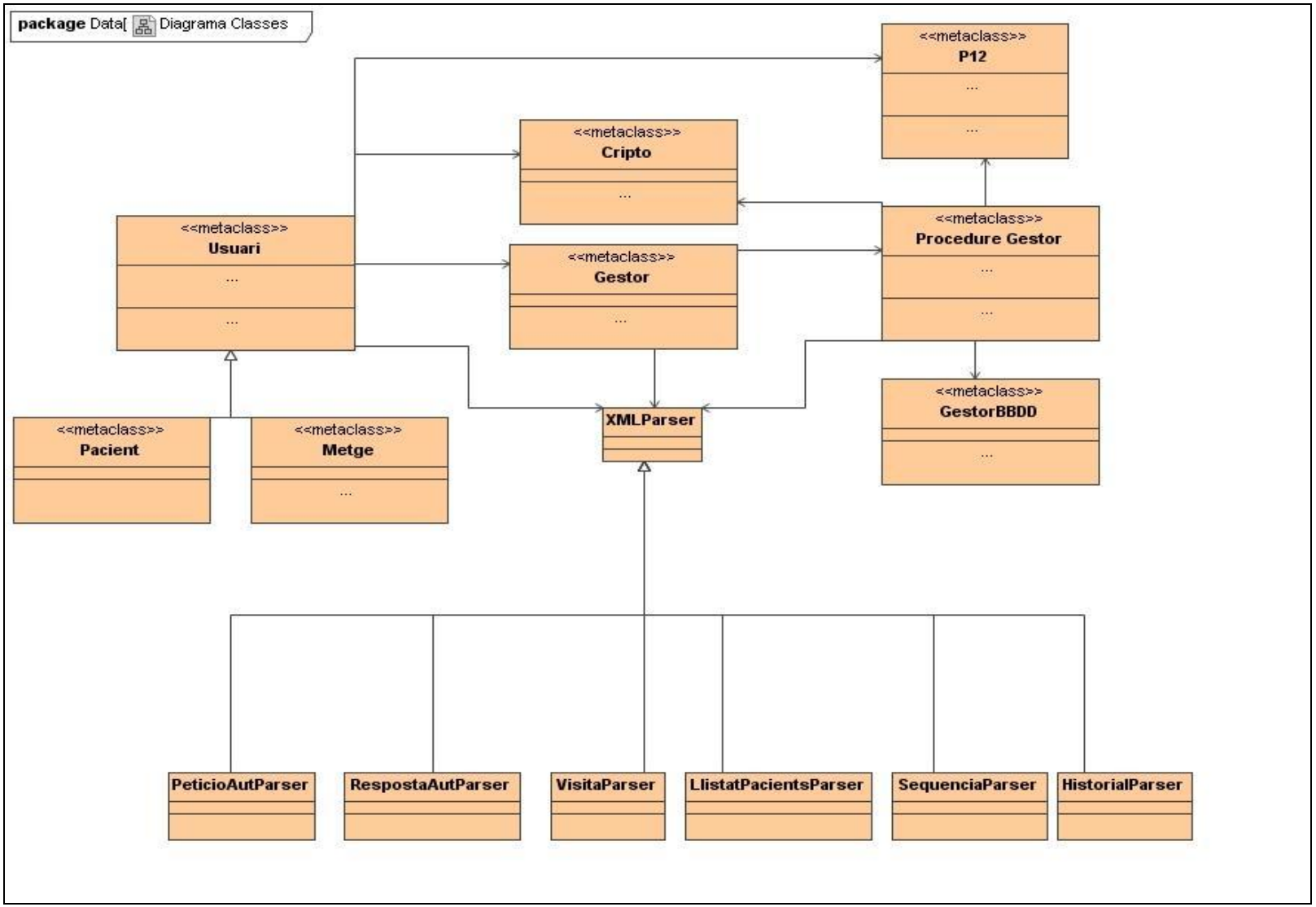


Figura 3.8: Diagrama de seqüència Inserció dades a l'Historial

3.10 DIAGRAMA DE CLASSES (Figura 3.9: Diagrama de Classes)



4. REPRESENTACIÓ DE LES DADES: XML

4.1 INTRODUCCIÓ

XML, *e**X**tensible **M**arkup **L**anguage* (llenguatge de marques extensible), es un metallenguatge extensible d'etiquetes desenvolupat per World Wide Web Consortium (W3C). Es una simplificació i adaptació del SGML (***S**tandard **G**eneralized **M**arkup **L**anguage*, Llenguatge de Marcació Generalitzat) .

XML s'utilitzarà com a estàndard per al intercanvi de informació estructurada entre diferents plataformes. En la actualitat té un paper molt important ja que permet la compatibilitat entre sistemes per a compartir la informació de una manera segura, fiable y fàcil. Així com per a la representar dades estructurades per a guardar en una base de dades.

Per aconseguir aquestos objectius, la utilització de XML ens ofereix els següents avantatges:

- Es extensible, es possible afegir noves etiquetes de forma que els consumidor dels documents pugen seguir entenent el nou format.
- L'analitzador del document es un component estàndard, en el nostre cas utilitzarem la llibreria org.w3c.dom.
- Els documents XML tenen una representació textual, una estructura senzilla d'entendre i llavors de processar-lo, fet que facilita la compatibilitat entre aplicacions.

4.2. SIGNATURA DIGITAL: XMLDSig

Per a signar els documents XML s'ha utilitzat la signatura digital XMLDSig, que és una especificació del W3C (http://www.w3.org/TR/xmldsig-core/). IETF l'ha adoptat com a estàndard d'Internet, RFC 3275 (http://www.ietf.org/rfc/rfc3275.txt)

Així doncs, Les signatures digitals que es generen en el projecte compleixen l'estàndard XMLDSig de W3C, i utilitzen l'algorisme de signatura digital RSA amb una funció de hash sha-1 (http://www.w3.org/2000/09/xmldsig#rsa-sha1).

Signatura digital XML

Les signatures digitals XML estan representades per una estructura XML que té com a node inicial l'element *Signature.*

Element *Signature*

L'element *Signature* conté 4 elements principals:

- L'element *SignedInfo*, que conté les referències als diferents recursos que estem signant, i informació relacionada amb la signatura que farem servir. De cada recurs que signem en tenim el seu *digest* que, al ser únic per a cada fitxer, ens permetrà verificar que aquest element no ha estat modificat.

- L'element *SignatureValue*, que conté la signatura de l'element *SignedInfo*.

- L'element *KeyInfo*, que conté la referència a les claus que s'han emprat per a realitzar la signatura de l'element *SignedInfo*.

- L'element opcional *Object* que conté dades que es volen signar. En aquest cas hi haurà una entrada dins les referències de l'element *SignedInfo* que farà referència a aquest element i en contindrà el seu *digest.*

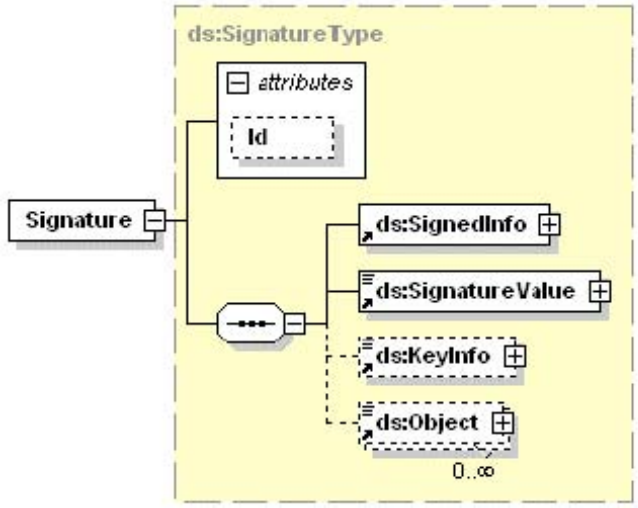
diagrama	
tipus	ds:SignatureType
atributs	Nom: **Id** Tipus: **ID** Us: **opcional**
font	<element name="Signature" type"ds:SignatureType"/>

Figura 4.1: Estructura element Signature

Detallem a continuació cadascú dels elements de l'element *Signature*.

Element *SignatureValue*

Aquest element conté el valor de la signatura, codificat en base64, de l'element *SignedInfo.* L'esquema que segueix aquest element és el següent:

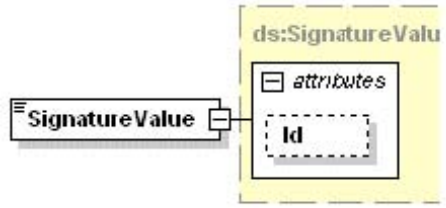
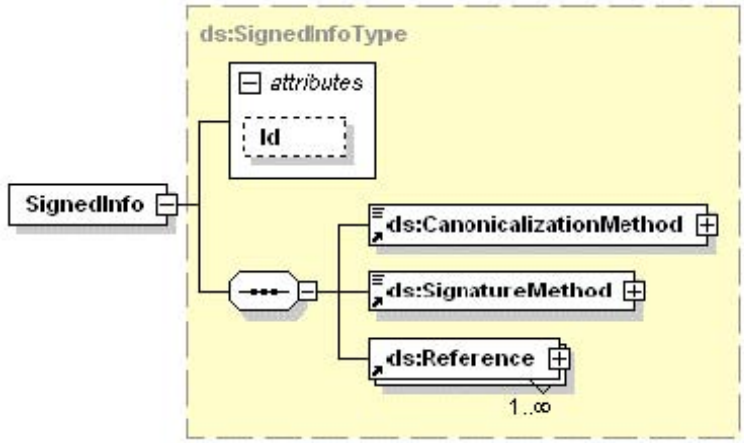
diagrama	
tipus	ds:SignatureValueType
atributs	Nom: **Id** Tipus: **ID** Us: **opcional**
font	<element name="SignatureValue" type="ds:SignatureValueType"/> <complexType name="SignatureValueType"> <simpleContent> <extension base=”base64Binary”/> <atribute name=”Id” type=”ID” user=”optional” /> </extension> </ simpleContent > </complexType>

Figura 4.2: Estructura element SignatureValue

Element *SignedInfo*

Conté el tipus de serialització que s'ha dut a terme sobre aquest element, l'algorisme que s'ha dut a terme per a signar aquest mateix element i un conjunt de referències als diferents recursos signats.

diagrama	
tipus	ds:SignedInfoType
atributs	Nom: **Id** Tipus: **ID** Us: **opcional**
font	<element name="SignedInfo" type="ds:SignedInfoType"/> <complexType name="SignedInfoType"> <sequence> <element ret=”ds:CanonicalizationMethod”/> <element ret=”ds:SignatureMethod”/>

	<element ret=”ds:Reference” maxOccurs=”unbounded”/> </sequence> <atribute name=”Id” type=”ID” user=”optional” /> </complexType>

Figura 4.3: Estructura element SignedInfo

Element *CanonalizationMethod*

L’element *CanonalitzationMethod* especifica el procés de serialització que s’ha emprat a l’element *SignedInfo.* Aquest permet passar l’element XML en qüestió a la seva forma canònica.

Hi ha diferents algorismes de canonicalització. En aquest projecte utilitzem la canonicalització exclusiva amb comentaris, és a dir, quan es serialitza l’element XML s’hi inclouen només les definicions dels espais de noms que s’utilitzen dins de l’element, i si hi ha comentaris en el codi XML aquests no s’eliminen.

```
<ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#WithComments"/>
```

Element *SignatureMethod*

Aquest element indica l’algorisme emprat per la generació i validació de la signatura. Identifica totes les funcions criptogràfiques emprades.

En aquesta projecte s’utilitza la signatura rsa-sha1, que es descriu de la següent manera:

```
<ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
```

Element *Reference*

Aquest element pot aparèixer una o més vegades i permet validar els resums hash que s’han generat sobre els diferents recursos.

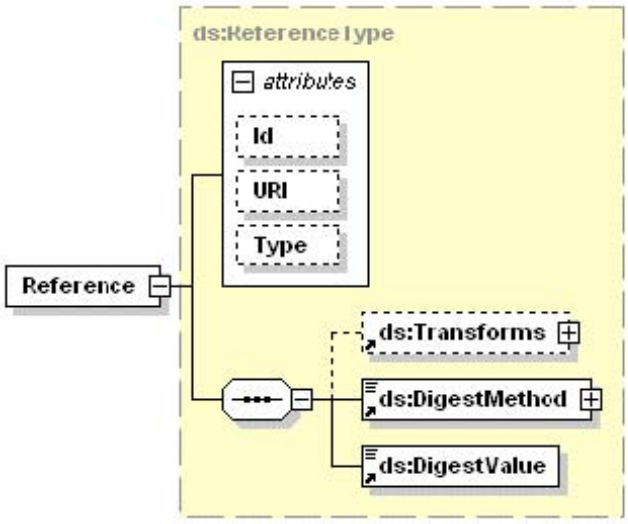
diagrama	
tipus	ds:ReferenceType
atributs	Nom: **Id** Tipus: **ID** Us: **opcional** Nom: **URI** Tipus: **anyURI** Us: **opcional** Nom: **Type** Tipus: **anyURI** Us: **opcional**
font	<element name="Reference" type="ds:ReferenceType"/> <complexType name="ReferenceType"> <sequence> <element ref="ds:Transforms" minOccurs="0"/> <element ref="ds:DigestMethod"/> <element ref="ds:DigestValue"/> </sequence> <attribute name="Id" type="ID" use="optional"/> <attribute name="URI" type="anyURI" use="optional"/> <attribute name="Type" type="anyURI" use="optional"/> </complexType>

Figura 4.4: Estructura element Refence

La informació que conté és la següent:

- Element ***DigestMethod****:* algorisme emprat pel càlcul del resum-hash. Per l'algorisme sha1:
  <ds:DigestMethod Algorithm=http://www.w3.org/2000/09/xmldsig#sha1/>

- Element ***DigestValue***: valor en base64 del resum-hash del recurs.

- Element ***Transforms***: conjunt d'elements de tipus *Transform* que descriuen com el signador del document ha obtingut les dades que després s'han passat a l'algorisme de hash, és a dir el conjunt de transformacions que s'han aplicat a un document XML abans de signar-lo. La sortida de cada transformació serveix com a entrada de la següent transformació, i la sortida del darrer element de transformació és l'entrada de l'algorisme de hash especificat a l'element *DigestMethod*. Cada transformació conté l'algorisme que s'ha emprat per a realitzar-la i els seus paràmetres.

Exemples de transformacions possibles poden ser: canonització del document, una expressió XPath per a filtrar el document o una XSLT. Les signatures “enveloped” també requereixen una transformació especial ja que abans de signar i verificar l’element objecte de la signatura, s’ha d’extreure de l’element Signature que s’està generat/verificant

Element *KeyInfo*

Element opcional que permet que els destinatari del document XML obtingui la clau necessària per a validar la signatura. Aquest element pot contenir claus, noms, certificats o informació destinada al maneig de claus públiques.

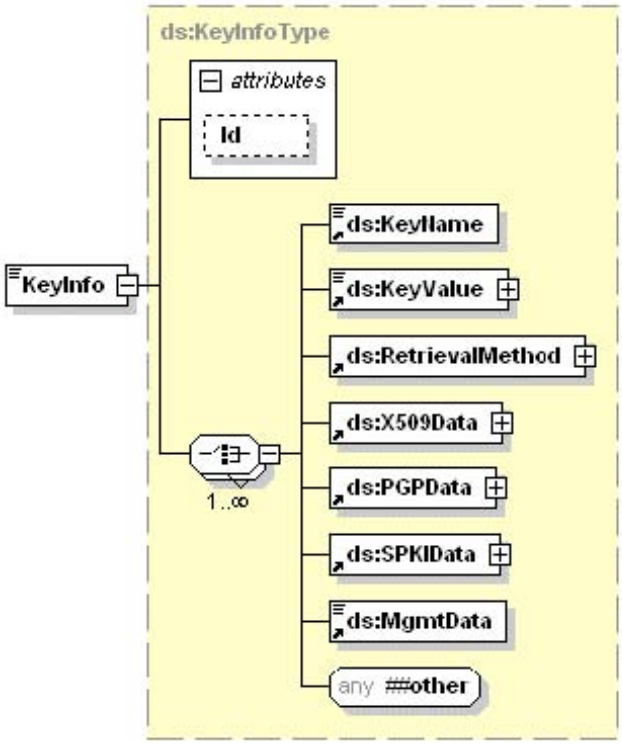
diagrama	
tipus	ds:KeyInfoType
atributs	Nom: **Id** Tipus: **ID** Us: **opcional**
font	<element name="KeyInfo" type="ds:KeyInfoType"/>

Figura 4.5: Estructura element KeyInfo

Element *Object*

Element opcional que permet incloure dades addicionals a la signatura. Els elements fills que pot portar un *Object* són de qualsevol tipus.

4.3 SOBRES DIGITALS: XMLEnc

Les dades xifrades que s'intercanvien les dues entitats tenen l'estructura de sobre digital i compleixen l'estàndard XMLEnc, que és una especificació del W3C (http://www.w3.org/TR/xmlenc-core/).

El format habitual d'un sobre digital utilitza xifres de clau simètrica i asimètrica. El criptosistema simètric s'utilitza per xifrar el text en clar ja que aquest pot tenir una magnitud considerable i les xifres simètriques són molt ràpides. Per altra banda, com que les claus simètriques són difícils de distribuir, s'utilitza una xifra de clau asimètrica per xifrar la clau simètrica i així poder-la transportar de forma segura entre emissor i receptor.

El sobre digital que implementarem en aquest projecte utilitza els següents criptosistemes:

- Algorisme AES amb claus de 128 bits (http://www.w3.org/2001/04/xmlenc#aes128-cbc) per xifrar les dades.

- Algorisme RSA (http://www.w3.org/2001/04/xmlenc#rsa-1_5) amb claus de mòdul 2^{1024} per xifrar la clau simètrica AES.

Sobres digitals XMLEnc

A través de l'estàndard XMLEnc es defineix un procés de xifrat de dades així com la seva representació en XML.

Partim d'un fitxer XML on s'hi apliquen una sèrie de transformacions per tal de xifrar-ne les parts que contenen dades sensibles, i obtenim un nou fitxer XML que conté la mateixa informació que la inicial, tot i que part d'aquesta informació està xifrada i s'hi ha hagut d'afegir certs camps perquè el destinatari del document sigui capaç de fer el procés invers i obtenir el fitxer original.

Els sobres digitals XMLEnc estan representats per una estructura XML que té com a node inicial l'element *EncryptedData*.

Element *EncryptedData*

L'element *EncryptedData* conté la informació que s'ha xifrat, així com informació relacionada amb el xifrat de les pròpies dades. L'estructura d'aquest elements és la següent:

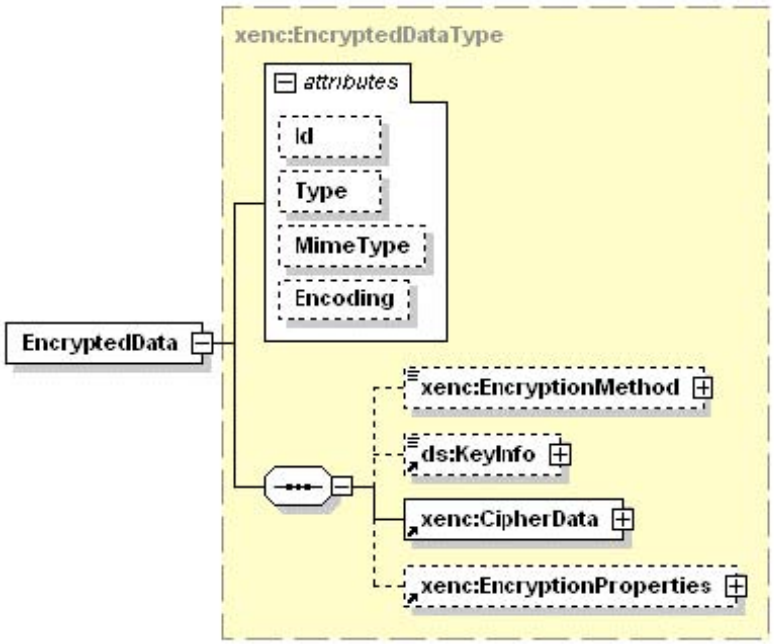
diagrama	
espai de noms	http://www.w3.org/2001/04/xmlenc#
tipus	xenc:EncryptedDataType
atributs	Nom: **Id** Tipus: **ID** Us: **opcional** Nom: **Type** Tipus: **anyURI** Us: **opcional** Nom: **MimeType** Tipus: **string** Us: **opcional** Nom: **Encoding** Tipus: **anyURI** Us: **opcional**
font	<element name="EncrytedData" type="xenc:EncrytedDataType"/> <complexType name="EncryptedDataType"> <complexContent> <extension base="xenc:EncryptedType"/> </complexContent> </complexType>

Figura 4.6: Estructura element EncryptedData

A continuació veurem cada un dels elements que composen l'element *EncryptedData*.

Element *EncryptionMethod*

Element opcional que descriu l'algorisme de xifrat que s'aplica al fragment que es vol xifrar. En el cas que aquest element no estigui definit caldrà que el destinatari del document XML xifrat conegui amb quin algorisme ha estat xifrat.

Per un xifrat AES-128:

<xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>

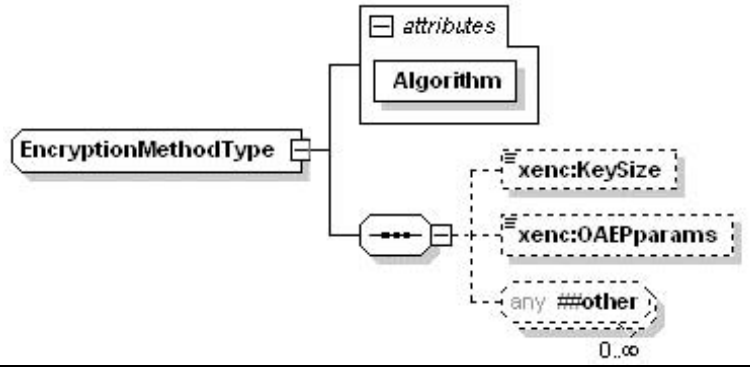
diagrama	
espai de noms	http://www.w3.org/2001/04/xmlenc#
atributs	Nom: **Algorithm** Tipus: **anyURI** Us: **opcional**
font	<complexType name="EncryptionMethodType" mixed="true"> <sequence> <element name="KeySize" type="xenc:KeySizeType" minOccurs="0"/> <element name="OAEPparams" type="base64Binary" minOccurs="0"/> <any namespace="##other" minOccurs="0" maxOccurs="unbounded"/> </sequence> <attribute name="Algorithm" type="anyURI" use="required"/> </complexType>

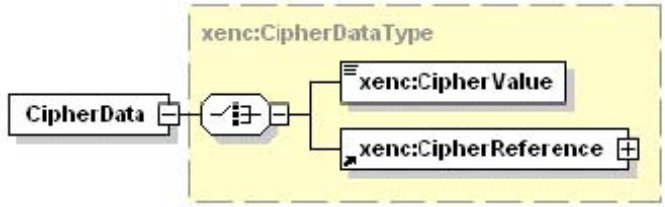
Figura 4.7: Estructura element EncryptionMethodType

Element *CipherData*

Aquest és l'element principal de ***EncryptedData*** perquè conté la informació xifrada. Pot contenir el criptograma codificat en base64 en un element fill ***ChiperValue***, o bé que aquest es trobi en un recurs extern i s'hi faci referència a través de l'element fill ***CipherReference***.

En aquest projecte utilitzarem l'element fill ***ChiperValue***.

```
<xenc:CipherData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
        <xenc:CipherValue>E..       U=</xenc:CipherValue>
</xenc:CipherData>
```

diagrama	
espai de	http://www.w3.org/2001/04/xmlenc#

noms	
tipus	xenc:CipherDataType
font	<element name="CipherData" type="xenc:CipherDataType"/> <complexType name="CipherDataType"> <choice> <element name="CipherValue" type="base64Binary"/> <element ref="xenc:CipherReference"/> </choice> </complexType>

Figura 4.8: Estructura element CipherData

Element *KeyInfo*

Aquest element opcional conté la informació de les claus usades en el procés de xifrat, per exemple pot tenir el nom del propietari de les claus, el valor de la clau pública, el certificat x509, etc.

Aquest element es comú a l'especificació XMLDSig, i en aquest apartat es pot veure l'esquema d'aquest element.

Quan es treballa amb sobres digitals, l'element *KeyInfo* ha de portar informació de dues claus:

- La clau de sessió (clau utilitzada per xifrar les dades)
- La clau de xifrat de la clau (clau utilitzada per xifrar la clau de sessió) mitjançant un element *EncryptedKey*.

Element *EncryptedKey*

L'element ***EncryptedKey*** conté la clau de sessió. Com que aquesta clau va xifrada, aquest element té una estructura molt similar a l'element *EncryptedData*. De fet, *EncryptedKey* i *EncryptedData* hereten del mateix tipus complex EncryptedType.

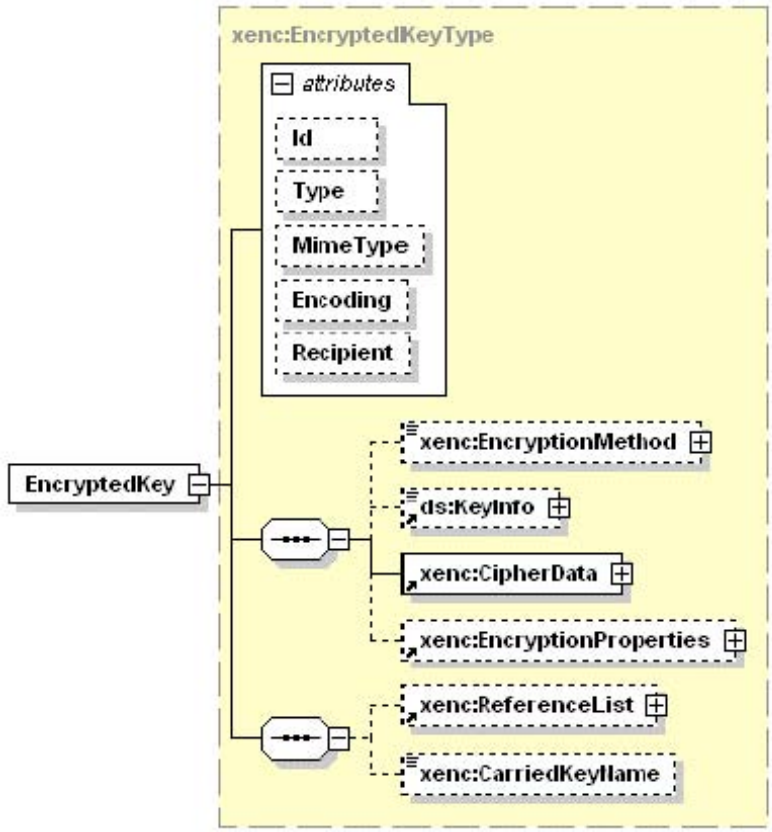
diagrama	
espai de noms	http://www.w3.org/2001/04/xmlenc#
tipus	xenc:EncryptedKeyType
atributes	Nom: **Id** Tipus: **ID** Us: **opcional** Nom: **Type** Tipus: **anyURI** Us: **opcional** Nom: **MimeType** Tipus: **string** Us: **opcional** Nom: **Encoding** Tipus: **anyURI** Us: **opcional** Nom: **Recipient** Tipus: **string** Us: **opcional**
font	<element name="EncryptedKey" type="xenc:EncryptedKeyType"/> <complexType name="EncryptedKeyType"> <complexContent> <extension base="xenc:EncryptedType"> <sequence> <element ref="xenc:ReferenceList" minOccurs="0"/> <element name="CarriedKeyName" type="string" minOccurs="0"/> </sequence> <attribute name="Recipient" type="string" use="optional"/> </extension> </complexContent> </complexType>

Figura 4.9: Estructura element EncryptedKey

Les característiques generals dels fills que usarem de EncryptedKey són les següents:

- Element *EncryptionMethod*: Indica l'algorisme utilitzat per xifrar la clau de sessió.

- Element *KeyInfo*: Porta informació de la clau utilitzada per xifrar la clau de sessió.

- Element *CipherData*: Conté la clau de sessió xifrada.

Element *EncryptionProperties*

Aquest element opcional proporciona informació addicional sobre la generació de les dades xifrades, per exemple pot indicar en quin instant de temps es va realitzar el xifrat, o bé pot portar un identificador de les dades xifrades (ens pot ser molt útil de cara a distingit duplicats) etc.

En el projecte no s'utilitzarà aquesta propietat

4.4 CLASSES UTILITZADES I ESTRUCTURA DELS DOCUMENTS XML

En la implementació de l'estructura dels documents XML utilitzats en el projecte, s'han creat les següents classes per tal de facilitar l'accés a la informació dels documents:

- **XmlParser**

  Classe de la qual hereten totes les classes utilitzades per a generar els documents XML, i que disposa dels mètodes comuns de creació del document, d'accés als elements,...

- **PeticioAutParser**

Aquesta classe implementa el document de petició d'autentificació que utilitzaran els usuaris. mostrem un exemple:

  - Definició del document DTD:

```
<!ELEMENT PeticioUsuari (NumId, Operacio, IdUsuari?)>
<!ELEMENT NumId (#PCDATA) >
<!ELEMENT Operacio (#PCDATA) >
<!ELEMENT IdUsuari (#PCDATA) >
```

  - Exemple del document:

```
<cp:PeticioUsuari xmlns:cp="http://www.uoc.edu/pfc">
        <cp:NumId>-2133636057</cp:NumId>
        <cp:Operacio>Autentificacio</cp:Operacio>
        <cp:IdUsuari>00000001-B</cp:IdUsuari>
    </cp:PeticioUsuari>
```

L'element NumId, contindrà el valor Ni en la primera part del protocol d'autentificació i el valor Ng per a la segona part. També l'element IdUsuari sols serà utilitzat per a l'operació de consulta de l'historial, i contindrà el identificador del pacient.
Quan es realitze l'operació de inserció de una visita el document de la visita serà inserit dins de la petició de l'usuari.

- **RespostaAutParser**

Aquesta classe implementa el document de resposta del Gestor a la petició d'autentificació que utilitzaran els usuaris.

  - Definició del document DTD:

```
<!ELEMENT RespostaGestor (Ng, Ni, IdUsuari)>
<!ELEMENT Ng (#PCDATA) >
<!ELEMENT Ni (#PCDATA) >
<!ELEMENT IdUsuari (#PCDATA) >
```

  - Exemple del document:

```
<cp:RespostaGestor xmlns:cp="http://www.uoc.edu/pfc">
    <cp:Ng>1342819586</cp:Ng>
    <cp:Ni>-2133636057</cp:Ni>
    <cp:IdUsuari>[C@104faf8</cp:IdUsuari>
</cp:RespostaGestor>
```

- **VisitaParser**

Aquesta classe implementa el document de la Visita enregistrada pel metge.

  - Definició del document DTD:

```
<!ELEMENT VisitaMetge (IdUsuariP, DataVisita, Sintomes, Tractament,
DataFiTractament, X?, Ti?)>
<!ELEMENT IdUsuariP (#PCDATA) >
<!ELEMENT DataVisita (#PCDATA) >
<!ELEMENT Sintomes (#PCDATA) >
<!ELEMENT Tractament (#PCDATA) >
<!ELEMENT DataFiTractament (#PCDATA) >
```

```
<!ELEMENT X (#PCDATA) >
<!ELEMENT T (#PCDATA) >
```

- Exemple del document:

```
<cp:VisitaMetge xmlns:cp="http://www.uoc.edu/pfc">
    <cp:IdUsuariP>00000001-B</cp:IdUsuariP>
    <cp:DataVisita>Fri Oct 26 17:47:04 CEST 2007</cp:DataVisita>
    <cp:Sintomes>Mal de cap</cp:Sintomes>
    <cp:Tractament>1 pastilla paracetamol</cp:Tractament>
    <cp:DataFiTractament>Fri Oct 26 17:47:04 CEST 2007
    </cp:DataFiTractament>
    <cp:X></cp:X>
    <cp:T></cp:T>
</cp:VisitaMetge>
```

Els elements X i T són utilitzats i assignats automàticament alhora de enregistrar en la base de dades la consulta.

- **SequenciaParser**

Aquesta classe implementa el document amb les dades de control del número de seqüència de les visites, que serà enregistrar en la base de dades amb la informació de la capçalera de l'historial mèdic.

- Definició del document DTD:

```
<!ELEMENT Sequencia (X, idUsuariP)>
<!ELEMENT X (#PCDATA) >
<!ELEMENT idUsuariP (#PCDATA) >
```

- Exemple del document:

```
<cp:Sequencia xmlns:cp="http://www.uoc.edu/pfc">
    <cp:X>3</cp:X>
    <cp:idUsuariP>00000001-B</cp:idUsuariP>
</cp:Sequencia>
```

- **DadesAdministrativesParser**

Aquesta classe implementa el document amb les dades administratives de l'historial mèdic dels pacients:

- Definició del document DTD:

```
<!ELEMENT DadesAdministratives (Nom, Cognoms, DataNaiximent,
LlocNaiximent, Sexe, TelefonPersonal, TelefonFamiliar, Domicili, DNI)>
```

```
<!ELEMENT Nom (#PCDATA) >
<!ELEMENT Cognoms (#PCDATA) >
<!ELEMENT DataNaiximent (#PCDATA) >
<!ELEMENT LlocNaiximent (Poblacio, Provincia, Pais) >
<!ELEMENT Poblacio (#PCDATA) >
<!ELEMENT Provincia (#PCDATA) >
<!ELEMENT Pais (#PCDATA) >
<!ELEMENT Sexe (#PCDATA) >
<!ELEMENT TelefonPersonal (#PCDATA) >
<!ELEMENT TelefonFamiliar (#PCDATA) >
<!ELEMENT Domicili (Carrer, Poblacio, Provincia, Pais) >
<!ELEMENT Carrer (#PCDATA) >
<!ELEMENT Poblacio (#PCDATA) >
<!ELEMENT Provincia (#PCDATA) >
<!ELEMENT Pais (#PCDATA) >
<!ELEMENT DNI (#PCDATA) >
```

- Exemple del document:

```
<cp:DadesAdministratives xmlns:cp="http://www.uoc.edu/pfc">
    <cp:Nom>Casimir</cp:Nom>
    <cp:CogNoms>Solbes Montaner</cp:CogNoms>
    <cp:DataNaiximent>Fri Oct 26 17:47:04 CEST 2007</cp:DataNaiximent>
    <cp:LlocNaiximent>
        <cp:Poblacio>Gandia</cp:Poblacio>
        <cp:Provincia>València</cp:Provincia>
        <cp:Pais>Espanya</cp:Pais>
    </cp:LlocNaiximent>
    <cp:Sexe>Home</cp:Sexe>
    <cp:TelefonPersonal>962877653</cp:TelefonPersonal>
    <cp:TelefonFamiliar>962877902</cp:TelefonFamiliar>
    <cp:Domicili>
        <cp:Carrer>c/ Cid Campeador, 3</cp:Carrer>
        <cp:Poblacio>Gandia</cp:Poblacio>
        <cp:Provincia>València</cp:Provincia>
        <cp:Pais>Espanya</cp:Pais>
    </cp:Domicili>
    <cp:DNI>20017083E</cp:DNI>
</cp:DadesAdministratives>
```

- **HistorialParser**

Aquesta classe implementa el document de l'historial mèdic, llavors estarà integrada pels documents generats amb les classes DadesAdministrativesParser, SequenciaParser, i el conjunt de documents de la classe VisitaParser inserits en la base de dades.

- Definició del document DTD:

```
<!ELEMENT HistorialMedic (DadesAdministratives, Sequencia?,
VisitesHistorial)>

<!ELEMENT DadesAdministratives (Nom, Cognoms, DataNaiximent,
LlocNaiximent, Sexe, TelefonPersonal, TelefonFamiliar, Domicili, DNI)>
<!ELEMENT Nom (#PCDATA) >
<!ELEMENT Cognoms (#PCDATA) >
<!ELEMENT DataNaiximent (#PCDATA) >
<!ELEMENT LlocNaiximent (Poblacio, Provincia, Pais) >
<!ELEMENT Poblacio (#PCDATA) >
<!ELEMENT Provincia (#PCDATA) >
<!ELEMENT Pais (#PCDATA) >
<!ELEMENT Sexe (#PCDATA) >
<!ELEMENT TelefonPersonal (#PCDATA) >
<!ELEMENT TelefonFamiliar (#PCDATA) >
<!ELEMENT Domicili (Carrer, Poblacio, Provincia, Pais) >
<!ELEMENT Carrer (#PCDATA) >
<!ELEMENT Poblacio (#PCDATA) >
<!ELEMENT Provincia (#PCDATA) >
<!ELEMENT Pais (#PCDATA) >
<!ELEMENT DNI (#PCDATA) >

<!ELEMENT Sequencia (X, idUsuariP)>
<!ELEMENT X (#PCDATA) >
<!ELEMENT idUsuariP (#PCDATA) >


<!ELEMENT VisitesHistorial (VisitaHist*, ) >
<!ELEMENT VisitaHist (VisitaXif, VisitaSig, X, Ti)>
<!ELEMENT VisitaXif (VisitaMetge, VisitaSig, X, Ti)>


<!ELEMENT VisitaMetge (IdUsuariP, DataVisita, Sintomes, Tractament,
DataFiTractament, X?, Ti?)>
<!ELEMENT IdUsuariP (#PCDATA) >
<!ELEMENT DataVisita (#PCDATA) >
<!ELEMENT Sintomes (#PCDATA) >
<!ELEMENT Tractament (#PCDATA) >
<!ELEMENT DataFiTractament (#PCDATA) >
<!ELEMENT X (#PCDATA) >
<!ELEMENT T (#PCDATA) >

<!ELEMENT VisitaSig (IdUsuariP, DataVisita, Sintomes, Tractament,
DataFiTractament, X?, Ti?)>
<!ELEMENT IdUsuariP (#PCDATA) >
```

```
<!ELEMENT DataVisita (#PCDATA) >
<!ELEMENT Sintomes (#PCDATA) >
<!ELEMENT Tractament (#PCDATA) >
<!ELEMENT DataFiTractament (#PCDATA) >
<!ELEMENT X (#PCDATA) >
<!ELEMENT T (#PCDATA) >

<!ELEMENT X (#PCDATA) >
<!ELEMENT T (#PCDATA) >
```

- Exemple del document:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <cp:HistorialMedic xmlns:cp="http://www.uoc.edu/pfc">
- <cp:VisitesHistorial>
- <cp:VisitaHist>
- <cp:VisitaXif>
- <cp:VisitaMetge xmlns:cp="http://www.uoc.edu/pfc">
  <cp:IdUsuariP>00000001-B</cp:IdUsuariP>
  <cp:DataVisita>Fri Nov 02 01:06:58 CET 2007</cp:DataVisita>
  <cp:Sintomes>Mal de cap</cp:Sintomes>
  <cp:Tractament>1 pastilla paracetamol</cp:Tractament>
  <cp:DataFiTractament>Fri Nov 02 01:06:58 CET 2007</cp:DataFiTractament>
    </cp:VisitaMetge>
    </cp:VisitaXif>
- <cp:VisitaSig>
- <cp:VisitaMetge xmlns:cp="http://www.uoc.edu/pfc">
  <cp:IdUsuariP>00000001-B</cp:IdUsuariP>
  <cp:DataVisita>Fri Nov 02 01:06:58 CET 2007</cp:DataVisita>
  <cp:Sintomes>Mal de cap</cp:Sintomes>
  <cp:Tractament>1 pastilla paracetamol</cp:Tractament>
  <cp:DataFiTractament>Fri Nov 02 01:06:58 CET 2007</cp:DataFiTractament>
  <cp:X>3</cp:X>
  <cp:T>1193962018765</cp:T>
    </cp:VisitaMetge>
    </cp:VisitaSig>
  <cp:X>1</cp:X>
  <cp:T>1</cp:T>
    </cp:VisitesHistorial>
- <cp:DadesAdministratives xmlns:cp="http://www.uoc.edu/pfc">
  <cp:Nom>Casimr</cp:Nom>
  <cp:CogNoms>Solbes Montaner</cp:CogNoms>
  <cp:DataNaiximent>Fri Nov 02 01:06:58 CET 2007</cp:DataNaiximent>
- <cp:LlocNaiximent>
  <cp:Poblacio>Gandia</cp:Poblacio>
  <cp:Provincia>València</cp:Provincia>
  <cp:Pais>Espanya</cp:Pais>
```

```
    </cp:LlocNaiximent>
  <cp:Sexe>Home</cp:Sexe>
  <cp:TelefonPersonal>962877653</cp:TelefonPersonal>
  <cp:TelefonFamiliar>962877902</cp:TelefonFamiliar>
- <cp:Domicili>
  <cp:Carrer>c/ Cid Campeador, 3</cp:Carrer>
  <cp:Poblacio>Gandia</cp:Poblacio>
  <cp:Provincia>València</cp:Provincia>
  <cp:Pais>Espanya</cp:Pais>
    </cp:Domicili>
  <cp:DNI>20017083E</cp:DNI>
    </cp:DadesAdministratives>
- <cp:Sequencia xmlns:cp="http://www.uoc.edu/pfc">
  <cp:X>3</cp:X>
  <cp:idUsuariP>00000001-B</cp:idUsuariP>
  </cp:Sequencia>
  </cp:HistorialMedic>
```

- **LlistatPacientsParser**

Aquesta classe implementa el document de resposta a la petició realitzada per un metge del llistat dels seus pacients:

- Definició del document DTD:

```
<!ELEMENT LlistatPacients (Pacients*)>
<!ELEMENT Pacients (idUsuariP) >
<!ELEMENT idUsuariP (#PCDATA) >
```

- Exemple del document:

```
<cp:LlistatPacients xmlns:cp="http://www.uoc.edu/pfc">
    <cp:Pacients >
    <cp:idUsuariP >00000001-B</cp:idUsuariP >
    </cp:Pacients >
</cp:LlistatPacients>
```

4.5 PROVES REALITZADES

Amb la definició de l'esquema criptogràfic del capítol anterior, i la definició dels documents XML, s'ha dut a terme un joc de proves on s'ha utilitzat una classe de prova que conté un exemple de les peticions de consulta d'un historial, consulta del llistat dels pacients del metge i de la inserció d'una visita. El codi d'aquesta classe sols inclou el codi realitzat fins aquest moment del disseny incremental del projecte.

Els documents XML resultants de les proves realitzades es poden consultar en la carpeta adjunta **“\Joc de Proves\Esquema Criptogràfic i Presentació de les Dades XML”**

Presentem a continuació el codi de la classe mencionada:

Procés Autentificació

```
public int procesAutentificacio() throws Exception
{
	String p12Metge = "F:\\UOC\\PFC\\PKI\\Certificats\\Metge.p12";
	String p12Gestor = "F:\\UOC\\PFC\\PKI\\Certificats\\Gestor.p12";

	// Creem una instància del Pacient
	metge = new Metge(p12Metge, "uoc0506", p12Gestor, "uoc0506");

	// PROCES AUTENTIFICACIO:  Metge: metge.generarPeticioAutentificacio();
	int Ni = new SecureRandom().nextInt();

	PeticioAutParser PeticioAut = new PeticioAutParser();
	PeticioAut.creaPeticio(Usuari.Autentificacio, Ni, metge.idUsuariU);

	PeticioAut.writeFile("F:\\UOC\\PFC\\Proves\\PeticioAut.xml");

	// Xifrem la petició d'autentificació amb la clau pública del gestor
	Cripto.xifrarElementRSA(PeticioAut.document, PeticioAut.getPeticio(),
	metge.certGestor);

	PeticioAut.writeFile("F:\\UOC\\PFC\\Proves\\PeticioAut_Xifrada.xml");

	byte[] peticioXifradaPg = PeticioAut.getDocInBytes();

	// Usuari: Envia petició Atentificació per al GESTOR
	// Creem una instància del Gestor i li pasem la petició d'autentificació
	gestor = new Gestor(p12Gestor, "uoc0506");

	byte[] respostaGestorXPu = null;

	{
	  Procedure_Gestor Procedure_Gestor = new Procedure_Gestor();
	  PeticioAutParser Peticio =
	  Procedure_Gestor.obtindrePeticio(peticioXifradaPg,gestor.privateKey);

	 // Desxifrem la petició d'autentificació Pg[Peticio, Ng,Id_usuari, ...]
	amb la clau privada del Gestor
	Peticio.writeFile("F:\\UOC\\PFC\\Proves\\PeticioAut_DesXifrada_Gestor.xm
	l");
	// Gestor opté la operació a realitzar
	String Operacio = Peticio.getOperacio().getTextContent();

	if (Operacio.equals("Autentificacio"))
	{
	 // Gestor realitaza proces autentificació
	 String _idUsuariU = Peticio.getIdUsuari().getTextContent();
	 int _Ni = Integer.valueOf(Peticio.getNumId().getTextContent());

	 int Ng = new SecureRandom().nextInt();

	 GestorBBDD.guardarValors(_Ni, Ng, _idUsuariU);

	 X509Certificate CertU =GestorBBDD.obtenirCertificat(_idUsuariU);
```

```
      // Gestor genera el missatge de resposta
       RespostaAutParser Resposta = new RespostaAutParser();
       Resposta.creaResposta(Ng, _Ni,gestor.idUsuariG);

      Resposta.writeFile("F:\\UOC\\PFC\\Proves\\RespostaAut.xml");

      // Gestor xifra missatge resposta amb la clau pública de l'usuari
      Cripto.xifrarElementRSA(Resposta.document,Resposta.getResposta(),
      CertU);

      Resposta.writeFile("F:\\UOC\\PFC\\Proves\\RespostaAut_Xifrada.xml
");

      respostaGestorXPu = Resposta.getDocInBytes();

      }
     }

      // Metge: rep la resposta a la petició d'autentificació i la
      desxifra amb la clau privada Su
      RespostaAutParser Resposta = new
      RespostaAutParser(respostaGestorXPu);

      Cripto.desXifrarElementRSA(Resposta.document,
      Resposta.getResposta(), metge.privateKey);

      Resposta.writeFile("F:\\UOC\\PFC\\Proves\\RespostaAut_DesXifrada_
      Metge.xml");

      String[] RespostaClarArray = Resposta.extrauResposta();

      int Ng = Integer.valueOf(RespostaClarArray[0]);
      int Nip = Integer.valueOf(RespostaClarArray[1]);

      try
      {
            if (Nip != Ni)
            {
                  throw new Exception("Error en el procediment
                  d'autentificació");
            }
      }
      catch(Exception e)
      {
          e.printStackTrace();
          System.exit(0);
      }

      return Ng;
}
```

Inserir dades en el historial

```
public void ProvaInserirDadesHistorial() throws Exception
{     System.out.println("PROVA INSERIR DADES HISTORIAL");

      String idUsuariP = "00000001-B";

      int Ng = this.procesAutentificacio();

      // Metge: Genera les dades de la inserció visita
      PeticioAutParser Peticio = new PeticioAutParser();
      Peticio.creaPeticio(Usuari.Inserir_visita, Ng, null);
      Peticio.InserirVisita(idUsuariP, new Date(), "Mal de cap", "1
      pastilla paracetamol", new Date());

      Peticio.writeFile("F:\\UOC\\PFC\\Proves\\PeticioAut_Visita.xml");

// Xifrem la petició d'autentificació amb la clau pública del gestor
      Cripto.signarElement(Peticio.document, Peticio.getDadesVisita(),
      metge.privateKey, metge.x509);

      Peticio.writeFile("F:\\UOC\\PFC\\Proves\\PeticioAut_Visita_Signad
      a.xml");

      // Xifra la petició d'autorització amb les dades de l'operació,
      amb la clau pública del Gestor
      Cripto.xifrarElementRSA(Peticio.document, Peticio.getPeticio(),
      metge.certGestor);

      Peticio.writeFile("F:\\UOC\\PFC\\Proves\\PeticioAut_Visita_Signad
aXifrada.xml");

      byte[] PetInsVisita =  Peticio.getDocInBytes();

      // Gestor rep la petició d'autorització amb les dades de
      l'operació i la desxifra

      Procedure_Gestor Procedure_Gestor = new Procedure_Gestor();
      PeticioAutParser PeticioInsVisita =
      Procedure_Gestor.obtindrePeticio(PetInsVisita,gestor.privateKey)

      PeticioInsVisita.writeFile("F:\\UOC\\PFC\\Proves\\PeticioAut_Visi
      ta_DesXifrada.xml");

      String Operacio= PeticioInsVisita.getOperacio().getTextContent();

      if(Operacio.equals("Inserir_visita"))
      {
       // Executa la inserció de la visita
       gestor.InserirVisita(metge.idUsuariU, PeticioInsVisita);
      }
}
```

Consulta Historial

```
public void ProvaConsultarHitorial() throws Exception
{ System.out.println("PROVA CONSULTA DADES HISTORIAL");

 DadesAdministrativesParser dadesAdm=new DadesAdministrativesParser();

 dadesAdm.CrearDadesAdministratives("Casimr", "Solbes Montaner", new
Date(),"Gandia","València","Espanya","Home","962877653","962877902",
"c/ Cid Campeador, 3", "Gandia", "València", "Espanya", "20017083E");

dadesAdm.writeFile("F:\\UOC\\PFC\\Proves\\DadesAdministratives.xml");

String idUsuariP = "00000001-B";

int Ng = this.procesAutentificacio();

// Metge Genera les dades de la petició de consulta de l'historial
// amb el valor Ng obtés, i ho xifra amb la clau pública del gestor
// metge.generarDadesConsultaHistorial(Ng, idUsuariP);

PeticioAutParser Peticio = new PeticioAutParser();
Peticio.creaPeticio(Usuari.ConsultaHistorial, Ng, idUsuariP);

Peticio.writeFile("F:\\UOC\\PFC\\Proves\\PeticioAut_Consulta.xml");

Cripto.xifrarElementRSA(Peticio.document,Peticio.getPeticio(),
metge.certGestor);

Peticio.writeFile("F:\\UOC\\PFC\\Proves\\PeticioAut_Consulta_Xifrada.x
ml");

byte[] PetConsHist = Peticio.getDocInBytes();

// Gestor rep la petició de consulta de l'historial i la desxifra
Procedure_Gestor Procedure_Gestor = new Procedure_Gestor();
PeticioAutParser PeticioConsulta =
Procedure_Gestor.obtindrePeticio(PetConsHist, gestor.privateKey);

PeticioConsulta.writeFile("F:\\UOC\\PFC\\Proves\\PeticioAut_Consulta_D
esXifrada.xml");

String Operacio = PeticioConsulta.getOperacio().getTextContent();

byte[] hist = null;
```

```
if(Operacio.equals("ConsultaHistorial"))
{
   int Ngp=
Integer.valueOf(PeticioConsulta.getNumId().getTextContent());
 // Cerquem a la BD el Ng guardat en el procedure 2.4
   String idUsuariU = Procedure_Gestor.ValidaAutentificacio(Ngp);

   if (idUsuariU != null)
   {
      // Executa la inserció de la visita
      hist = gestor.recercaHistorial(idUsuariU,
      PeticioConsulta.getIdUsuari().getTextContent());

   }
}

if (hist != null)
{
 // El metge rep l'historial
   HistorialParser Historial = new HistorialParser(hist);
   Historial.writeFile("F:\\UOC\\PFC\\Proves\\PeticioAut_Consulta_Hist
orial_Xifrat.xml");

 // Verificar signatura del Metge, del Gestor i verificar la seqüencia
  Document Document = metge.DesxifraValidarHistorial(hist);

  HistorialParser HistorialDesxifrat = new HistorialParser(Document);

   HistorialDesxifrat.writeFile("F:\\UOC\\PFC\\Proves\\PeticioAut_Cons
   ulta_Historial_DesXifrat.xml");
}

}
```

Consulta Llistat Pacients

```
public void ProvaLlistatPacients() throws Exception
{  System.out.println("PROVA CONSULTA LLISTAT PACIENTS");

 int Ng = this.procesAutentificacio();

 // Metge: Genera les dades de la inserció visita
 PeticioAutParser Peticio = new PeticioAutParser();
 Peticio.creaPeticio(Usuari.Llistat_pacients, Ng, null);

Peticio.writeFile("F:\\UOC\\PFC\\Proves\\PeticioAut_LlistatPacients.xml");

// Xifra la petició d'aut. amb les dades de l'operació, amb la clau pública del Gestor
Cripto.xifrarElementRSA(Peticio.document,Peticio.getPeticio(), metge.certGestor);

Peticio.writeFile("F:\\UOC\\PFC\\Proves\\PeticioAut_LlistatPacients_Xifrada.xml");

byte[] Petllistat =  Peticio.getDocInBytes();

//Gestor rep petició d'aut. amb les dades de l'operació i la desxifra
Procedure_Gestor Procedure_Gestor = new Procedure_Gestor();
PeticioAutParser PeticioLlistatPacients =
Procedure_Gestor.obtindrePeticio(Petllistat, gestor.privateKey);

PeticioLlistatPacients.writeFile("F:\\UOC\\PFC\\Proves\\PeticioAut_LlistatPacients_DesXifrada.xml");

String Operacio=PeticioLlistatPacients.getOperacio().getTextContent();

if(Operacio.equals("Llistat_pacients"))
{
  int Ngp =
Integer.valueOf(PeticioLlistatPacients.getNumId().getTextContent());
  // Cerquem a la BD el Ng guardat en el procedure 2.4
  String idUsuariU = Procedure_Gestor.ValidaAutentificacio(Ngp);

  if (idUsuariU != null)
  {
      llistatb = gestor.LlistatPacients(idUsuariU);
  }
}
```

```
if (llistatb != null)
{
  LlistatPacientsParser llistatPacients = new
  LlistatPacientsParser(llistatb);

  llistatPacients.writeFile("F:\\UOC\\PFC\\Proves\\PeticioAut_LlistatP
acients_XifratGestor.xml");

  Cripto.desXifrarElementRSA(llistatPacients.document,
  llistatPacients.getLlistatPacients(), metge.privateKey);

llistatPacients.writeFile("F:\\UOC\\PFC\\Proves\\PeticioAut_LlistatPac
ients_DesXifratGestor.xml");

  Element llistat = llistatPacients.getLlistatPacients();
  if (!Cripto.verifySignature(XmlParser.getSignature(llistat),
      metge.x509.getPublicKey()))
 {
    String[] llistatStr = llistatPacients.extrauLlistatPacients();
 }
 else
 {
      throw new Exception("No verificada la signatura");
 }
}

}
```

5. COMUNICACIÓ DELS COMPONENTS: RMI

5.1 INTRODUCCIÓ

RMI (Invocació Remota de Mètodes) es un mecanisme ofert per Java (inclòs en els JDK de Sun des de la versió 1.1.) que permet comunicar aplicacions remotes, localitzant els objectes remots per tal d'enviar i rebre els arguments i resultats d'una invocació.

Es tracta, per tant, d'objectes o processos que es comuniquen entre dues màquines separades per una xarxa, que intercanvien invocacions d'operacions o mètodes que porten com arguments d'entrada i/o sortida objectes, estructures de dades i referències a altres objectes.

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

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

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

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

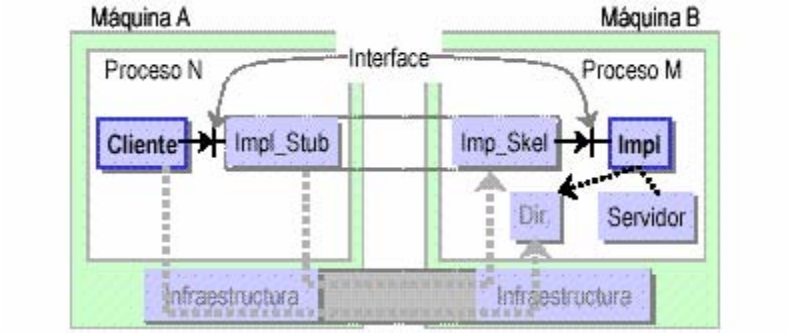


Figura 5.1: Objecte client invoca mètodes d'un objecte remot.

5.2 FUNCIONAMENT I IMPLEMENTACIÓ DE RMI AL SISTEMA

Per tal de realitzar la implementació de la comunicació amb RMI en el projecte, hem realitzat els següents passos:

1. S'ha creat una interfície Java, on definim els mètodes de la classe remota que podran ser invocats. Aquesta interfície ha d'heretar de la interfície *java.rmi.Remote* per a indicar que pot ser cridada des de qualsevol màquina virtual de Java i cada mètode ha de llançar l'excepció *java.rmi.RemoteException*.

A continuació presentem com a exemple, la interfície que hem generat en el projecte *IGestor*, en la qual definim l'únic mètode accessible remotament pels usuaris *PeticioUsuari*:

```
public interface IGestor  extends java.rmi.Remote
{

public byte[] PeticioUsuari(byte[] _Pg_xifrat) throws java.rmi.RemoteException;

}
```

2. En la classe que implementa la recepció de les peticions dels usuaris *Gestor*, s'ha modificat la seua definició declarant que implementa la interfície remota presentada en el punt anterior *IGestor* i que hereta de la classe *java.rmi.server.UnicastRemoteObject* per a indicar que pot ser utilitzada remotament. Finalment en el mètode constructor s'ha afegit l'excepció *java.rmi.RemoteException*:

```
public class Gestor extends java.rmi.server.UnicastRemoteObject
implements IGestor
{
      public Gestor(String _filePKCS12Gestor, String _password) throws
      java.rmi.RemoteException ...
```

3. S'ha generat un fitxer xml anomenat "*configuracio.xml*" que es guardarà en disc tant de l'aplicació servidor com en la del client, i que contindrà la informació del nom o IP del servidor i del número del port, on s'executarà el registre RMI. Aleshores també s'ha creat una nova classe anomenada *configuracioParse*" que s'utilitzarà per a accedir al fitxer de configuració, aquesta classe hereta de la classe *xmlParser.* A continuació podem visualitzar el contingut del fitxer de configuració:

```
<Configuracio>
        <RMI_host>localhost</ RMI_host >
        < RMI_port >1099</ RMI_port >
</Configuracio>
```

4. S'ha creat una nova classe que s'executarà en el servidor per tal que instancie i anuncie l'objecte que implementa la interfície *IGestor*. Així en aquesta classe es crear i instal·la un "security manager", crear una interfícies de l'objecte remot i el registra:

```
public class GestorServer
{// mètode constructor
 public GestorServer(String _filePKCS12Gestor, String _password){
  try
  {
     // Creem i instalem un gestor de seguretat "security manager"
     if (System.getSecurityManager() == null)
      System.setSecurityManager(new RMISecurityManager());

      // Creem una interficie de l'objecte remot
      IGestor gestor = new Gestor(_filePKCS12Gestor, _password);

      // Afegim al registre RMI l'objecte remot
     configuracioParser config = new configuracioParser();

     Naming.rebind("rmi://"+ config.getServidorValue() +":"+
     config.getPortValue() +"/Servei_Gestor", gestor);
  }
  catch (final Exception e) {
        System.out.println("Problema: " + e);
    }
  }

   public static void main(String args[]) {
         new GestorServer(args[0], args[1]);
   }
}
```

Tal i com s'ha indicat en la introducció d'aquet capítol amb el gestor de seguretat "security manager" podrem controlar quins objectes i quins mètodes pot invocar cada màquina clienta remota, així com restringir les accions pot dur a terme un objecte sobre la màquina en la qual corre i evitar que els errors d'un procés pugui afectar a la seguretat i integritat de la màquina. Per tal de poder-ho gestionar s'ha creat el fitxer de configuració de la seguretat *security.policy* :

```
grant {
      permission java.security.AllPermission;

      // Other options:
      // permission java.net.SocketPermission "127.0.0.1:1024-",
"accept, connect, listen, resolve";
      // permission java.net.SocketPermission "localhost:1024-",
"accept, connect, listen, resolve";

      // From
http://java.sun.com/docs/books/tutorial/rmi/running.html
      // Copyright 1995-2005 Sun Microsystems, Inc. Reprinted with
permission

      // permission java.net.SocketPermission "*:1024-65535",
"connect,accept";
    // permission java.net.SocketPermission "*:80", "connect";

      // permission java.net.SocketPermission "*:1024-65535",
"connect,accept";
    // permission java.io.FilePermission
"c:\\home\\ann\\public_html\\classes\\-", "read";
    // permission java.io.FilePermission
"c:\\home\\jones\\public_html\\classes\\-", "read";
};
```

5. Finalment quan un usuari vol realitzar una petició al gestor ha d'instanciar l'objecte de forma remota, substituint bàsicament la invocació local (new()) per la invocació remota (amb Naming.lookup()) :

```
protected IGestor ObtindreGestor()  {

IGestor IGestor = null;

Try {
  configuracioParser config = new configuracioParser();

  IGestor = (IGestor)java.rmi.Naming.lookup("rmi://"+
  config.getServidorValue() +":"+ config.getPortValue()
  +"/Servei_Gestor");

}catch (NotBoundException e)  {
      System.out.println("NotBoundException: " + e); }
}catch (MalformedURLException e) {
      System.out.println("MalformedURLException: " + e); }
}catch (RemoteException e) {
      System.out.println("RemoteException: " + e); }
catch(Exception e){
      System.out.println("Exception: " + e); }

return IGestor;
}
```

Per tal de realitzar la implementació, no ha fet falta generar els representat local (stub) ja que a partir del JDK 1.5 (tant del client com del servidor) de forma interna i en temps d'execució es generen un proxys genèrics que fan les funcions que feien els stubs. Tampoc ha segut necessari generar els representats remots (skeleton) ja que a partir del JDK 1.2, les seves funcions son realitzades pel suport RMI de la JVM.

5.3 DIAGRAMA DE CLASSES AMPLIACIO CLASSES RMI

Diagrama de Classe ampliat a les classes generades (*IGestor* i *GestorServer*) en la implementació del RMI.

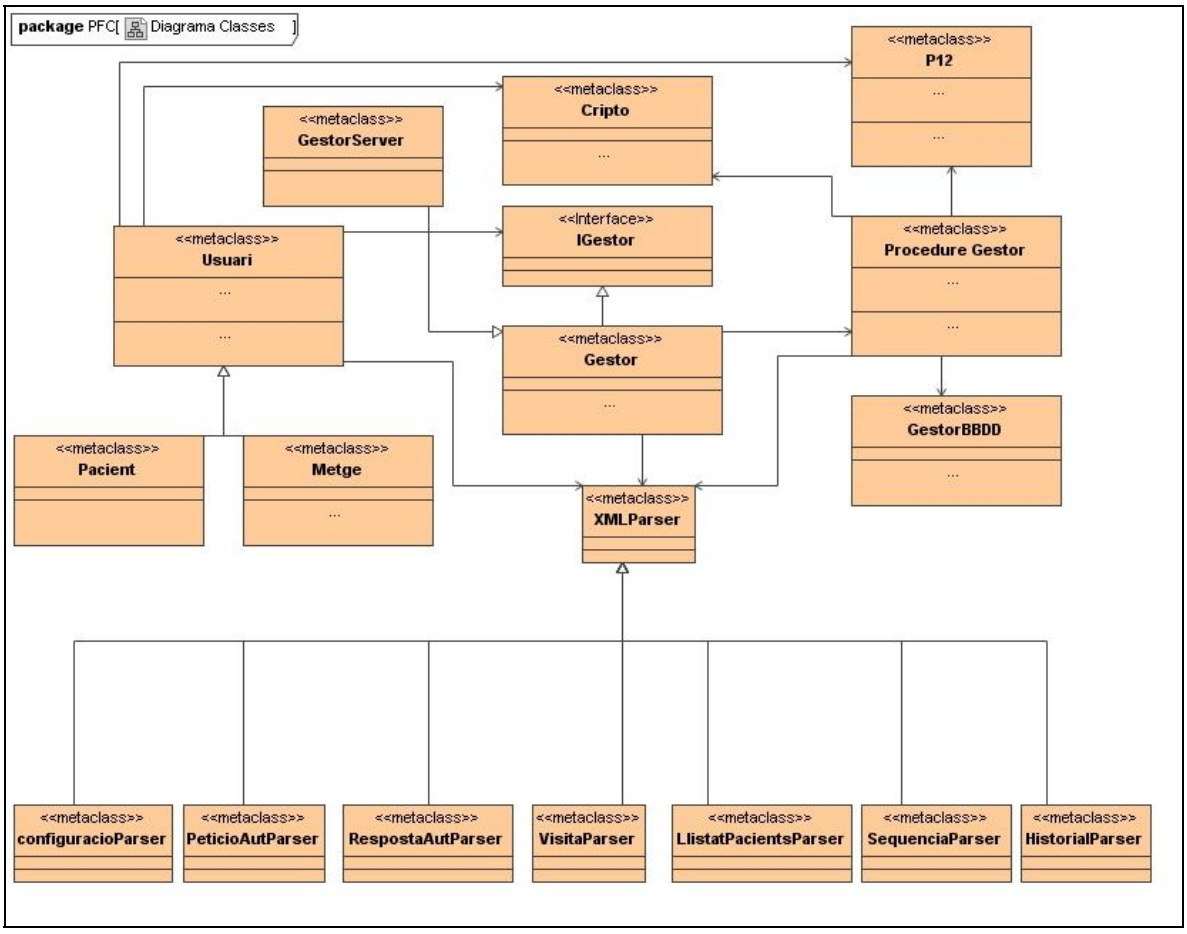


Figura 5.2 Diagrama de classes ampliat

5.4 PROVES REALITZADES

Les proves han segut realitzades dintre del mateix entorn de desenvolupament Eclipse, per a la qual cosa s'ha tingut que instal·lar el plugin RMI per a Eclipse de

www.genady.net., i llavors s'ha sol·licitat i concedit per part de l'empresa propietària del plugin una llicencia personal amb una validesa de 4 mesos.

Així s'ha creat una classe ProvesRMI on s'executa la primera part del procés d'autentificació definit en el capítol de l'esquema criptogràfic, llavors las diferència entre aquesta prova i la realitzada en el capítol de l'esquema criptogràfic és la forma d'obtindré una instancia del gestor ja que ara l'usuari executa el mètode *obtindreGestor* que el seu codi es pot veure en el segon apartat d'aquest capítol punt 4, on es fa la recerca de la interfície de l'objecte *gestor* en el servei de noms del RMI amb el codi *"`Naming.lookup()`"*.

Inicialment s'ha executat la classe GestorServer com una aplicació RMI ("RMI application"), la qual instància l'objecte del gestor i l'anuncia en el servei de noms del RMI, després ja executem la classe de proves com una aplicació Java ("Java application") i finalment comprovem els resultat: el gestor rep la petició d'autentificació enviada per l'usuari i obtenim la seva resposta, generant els documents XML resultants de la prova, que es poden consultar en la carpeta adjunta **"\Joc de Proves\RMI"**.

6. GESTIÓ DE LA INFORMACIÓ: BASE DE DADES

6.1 INTRODUCCIÓ

Per a gestionar les dades que necessiten persistència (registre de pacients i metges, historials mèdics, pacients assignats a un metge, etc...) hem d'utilitzar un sistema gestor de base de dades on definim les estructures de dades que necessitem enregistrar, amb els seus tipus de dades, restriccions, definicions de claus primàries i foranes.

Llavors en aquest projecte utilitzarem MySQL Server 5.0 com a Sistema de Gestió de Base de Dades relacional. El SGBD MySQL ha estat desenvolupat per l'empresa MySQL AB com a programari lliure, i està disponible en diverses plataformes: Linux, Windows i Mac entre altres.

6.2 MODEL DE LA BASE DE DADES

- **Model entitat relació (amb notació UML):**

En el model entitat relació hem definit tres entitats rellevants, els usuaris, el identificador de la connexió i l'historial de les visites.

L'entitat *Usuaris* contindrà la informació dels usuaris, tant dels pacients com dels metges, així doncs s'ha creat la relació entre els usuaris *PacientsMetge* per tal de poder definir els pacients d'un metge. L'entitat *HistorialVisites* representa la informació de les visites que ha realitzat l'usuari pacient al metge, per aquest fet existeix la relació *té* amb els usuaris pacients. I finalment l'entitat *IdentificadorConnexió* representa la informació persistent que es gestiona en el procés d'autentificació i que està relacionada amb l'usuari que realitza la connexió.

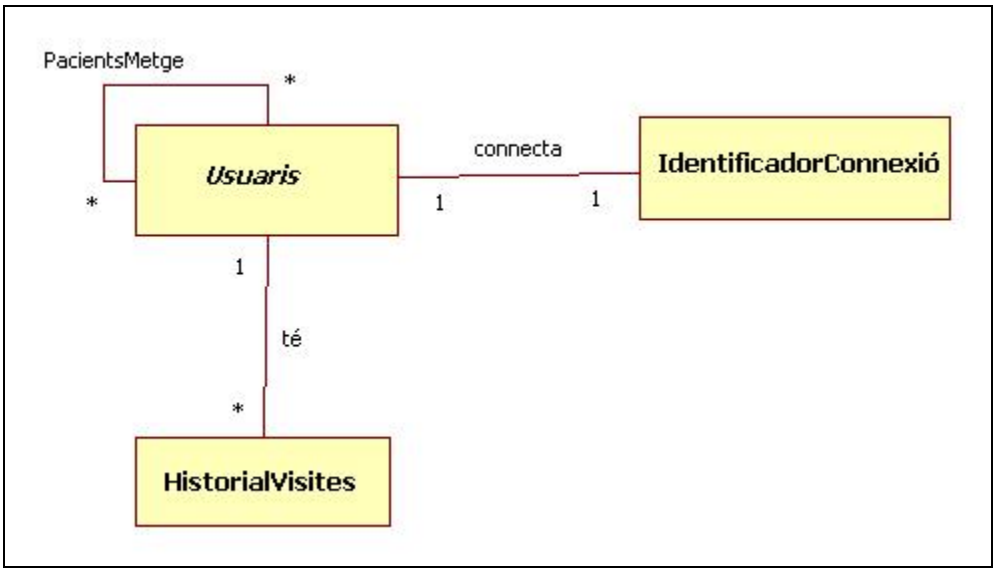


Figura 6.1 Diagrama entitat relació

- **Descripció dels camps i les taules de la base de dades**

A partir del diagrama entitat relació, i les dades que cadascuna de les entitats ha de contindré, definim el model relacional així per cadascuna de les entitats s'ha creat en el model relacional en una taula, i a més la relació *PacientsMetge* de l'entitat *Usuaris*, ha esdevingut en una altra taula.

- **Taula Usuaris**

La Taula Usuari emmagatzemarà les dades dels usuaris, metges i pacients. El camp identificador dels usuaris *IdUsuari* el la clau primària de la taula.

Nom Camp	Tipus de Dades	Descripció
IdUsuari	String	Identificador de l'usuari
Certificat	byte[]	Certificat digital X509
DadesAdministratives	byte[]	Document XML amb les dades Administratives
Sequencia	byte[]	Document XML amb el número de seqüència de les visites de l'historial, signat pel Gestor

- **Taula HistorialPacient**

La Taula HistorialPacients emmagatzemarà les dades de les visites del pacients generades pel metge. El camps identificador del pacient IdUsuari i el camp Número de sèrie *X* formaran conjuntament la clau primària de la taula. I el camp IdUsuari es una clau forana relacionada amb la taula Usuari en el seu camp IdUsuari.

Nom Camp	Tipus de Dades	Descripció
IdUsuari	String	Identificador de l'usuari Pacient
X	Integer	Número de sèrie
T	DateTime	Marca temporal
VisitaXif	byte[]	Document XML xifrat pel Gestor amb les dades de la visita signades pel metge.
VisitaSig	byte[]	Document XML signat pel Gestor amb les dades de la visita signades pel metge.

- **Taula IdentificacioConnexio**

La Taula IdentificacioConnexio emmagatzemarà les dades que s'utilitzaran en el procés d'autentificació. El camp Número identificador connexió del gestor *Ng* serà la clau primària de la taula.

Nom Camp	Tipus de Dades	Descripció
IdUsuari	String	Identificador de l'usuari Pacient
Ni	Integer	Número identificador connexió del l'usuari
Ng	**Integer**	Número identificador connexió del gestor

- **Taula PacientsMetge**

La Taula PacientsMetge emmagatzemarà els identificadors dels metges amb els identificadors els pacients que tinga assignats. El camp Identificador de l'usuari Metge *IdUsuariM* serà la clau primària de la taula.

Nom Camp	Tipus de Dades	Descripció
IdUsuariM	String	Identificador de l'usuari Metge
IdUsuariP	String	Identificador de l'usuari Pacient

6.3 IMPLEMENTACIO BASE DE DADES: MySQL.

En la creació de les taules de la base de dades s'ha utilitzat la propietat *ENGINE=InnoDB* per a que el sistema gestor de base de dades *MySQL* done suport transaccional i proporcione un complet compliment de les propietats transaccionals ACID (Atomicitat, Consistència, Isolament, Durabilitat).

En la definició dels tipus de dades, per a aquells camps que han d'emmagatzemar dades binaries (byte[]), com per exemple els certificats dels usuaris, i els camps que contenen documents XML, la base de dades MySQL ens facilita els tipus de dades BLOB, TINYBLOB, MEDIUMBOLB i LONGBLOB, que és diferencien per la capacitat d'emmagatzemament, així s'ha utilitzat el tipus de dades LONGBLOB que té la màxima capacitat amb 2^{32} bytes.

Per als camps que han requerit emmagatzemar la data i l'hora s'ha utilitzat el tipus de dades Datetime que ens proporciona MySQL. Datetime és una combinació d'una data i d'una hora. El seu format és 'AAAA-MM-DD HH:MM:SS', i té un rang que van des de '1001-01-01 00:00:00' fins a '9999-12-31 23:59:59'.

L'script utilitzat en la creació de la base de dades i les taules, amb la definició dels camps utilitzats, es pot visualitzar en *l'annex b.* ***Script de creació de la Base de Dades***.

6.4 IMPLEMENTACIO GESTIO BASE DE DADES: JAVA.

Per a la implementació de l'accés i gestió de la base de dades amb el llenguatge Java, s'ha utilitzat l'API JDBC proporcionat per l'estàndard de Java.

Així totes les operacions de connexió i gestió de la base de dades s'ha realitzat mitjançant una classe especifica anomenada GestorBBDD. Aquesta classe amb els seus mètodes ja havia segut creada en la fase d'implementació de l'esquema criptogràfic, i en la present fase s'ha completat amb la correcta codificació de totes les operacions que es realitzen sobre la base de dades.

Per tal de realitzar la connexió a la base de dades, s'ha ampliat el fitxer que conté els paràmetres de configuració "*configuracio.xml*" (i llavors la classe el gestiona *configuracioParser* també ha segut actualitzada) afegint les dades bàsiques per a realitzar la connexió:

- Adreça IP del servidor on està instal·lada la base de dades.
- Port del servidor per el qual es realitza la connexió
- Nom de la base de dades que es vol connectar
- Usuari de la base de dades
- Contrasenya de l'usuari

Aquestes dades sols caldrà que estiguen informades en el fitxer de configuració de l'aplicació del Gestor del sistema, ja que aquest és l'únic actor que realitzarà operacions directament sobre la base de dades.

Presentem a continuació el fitxer de configuració amb els paràmetres utilitzats en les proves realitzades :

```
<Configuracio>
        ...
        <MySQL_host >localhost </MySQL_host >
        <MySQL_port >3306</MySQL_port >
        <MySQL_database>pfc_csm</MySQL_database>
        <MySQL_user>casimir</MySQL_user>
        <MySQL_password>casimir</MySQL_password>
        ...
</Configuracio>
```

6.5 PORVES REALITZADES

Per tal realitzar les proves s'ha creat una nova classe *ProvesMySQL* amb els mètodes que implementen les operacions bàsiques que els usuaris poden realitzar en el sistema:Inserció de dades a l'historial d'un pacient, Consulta de l'historial d'un pacient i Consulta de llista de pacients del metge. On cadascuna d'aquestes operacions ja implementa el procés d'autentificació davant del l'aplicació *gestor*.

Així primerament s'ha creat un mètode per tal de enregistrar les dades inicials del sistema, registre d'un metge i un pacient amb les seves dades administratives i els seus certificats, així com l'assignació del pacient a la llista del pacients del metge.

Els documents XML resultants de les proves realitzades (MySQL_Consulta_Historial.xml i MySQL_Consulta_LlistatPacients.xml) es poden consultar en la carpeta adjunta **"\Joc de Proves\BBDD".**

Presentem a continuació el codi de la classe mencionada:

Operació d'inserció de dades a l'historial d'un pacient

```
public void ProvaInserirDadesHistorial() throws Exception
{
   if (metge == null)
      metge = new Metge(p12Metge, "uoc0506", p12Gestor, "uoc0506");

    System.out.println("PROVA INSERIR DADES HISTORIAL");

    String idUsuariP = "00000001-B";

    metge.InserirVisita(idUsuariP, new Date(), "Mal de cap", "1
    pastilla paracetamol", new Date());
}
```

Consulta de l'historial d'un pacient

```
public void ProvaConsultarHitorial() throws Exception
{
	if (metge == null)
	metge = new Metge(p12Metge, "uoc0506", p12Gestor, "uoc0506");

	System.out.println("PROVA CONSULTA DADES HISTORIAL");

	String idUsuariP = "00000001-B";

	Document Document = metge.ConsultaHistorial(idUsuariP);

	HistorialParser HistorialDesxifrat = new
	HistorialParser(Document);

	HistorialDesxifrat.writeFile("F:\\UOC\\PFC\\Proves\\MySQL_Consult
	a_Historial.xml");
}
```

Consulta del llista de pacients del metge

```
public void ProvaLlistatPacients() throws Exception
{
	if (metge == null)
	metge = new Metge(p12Metge, "uoc0506", p12Gestor, "uoc0506");

	System.out.println("PROVA CONSULTA LLISTAT PACIENTS");

	Document Document = metge.LlistatDePacients();

	LlistatPacientsParser llistatPacients = new
	LlistatPacientsParser(Document);

	llistatPacients.writeFile("F:\\UOC\\PFC\\Proves\\MySQL_Consulta_L
	listatPacients.xml");
}
```

Inserció de les dades inicials

```
public void Inserir_Dades_Prova() throws Exception
{
  // Creem una instància del Metge
  metge = new Metge(p12Metge, "uoc0506", p12Gestor, "uoc0506");

  GestorBBDD GestorBBDD = new GestorBBDD();

  // Inserim el Usuari Metge
  DadesAdministrativesParser dadesAdm = new
      DadesAdministrativesParser();

  dadesAdm.CrearDadesAdministratives("Casimr", "Solbes Montaner", new
  Date(), "Gandia","València", "Espanya", "Home", "962877653",
  "962877902","c/ Cid Campeador, 3", "Gandia", "València", "Espanya",
  "20017083E");

   GestorBBDD.InserirUsuari(metge.idUsuariU, metge.x509.getEncoded(),
   dadesAdm.getDocInBytes());

   // Inserim un Usuari Pacient
   dadesAdm = new DadesAdministrativesParser();
      dadesAdm.CrearDadesAdministratives("Casimr", "Solbes Montaner",
      new Date(), "Gandia","València", "Espanya", "Home", "962877653",
      "962877902", "c/ Cid Campeador, 3", "Gandia", "València",
      "Espanya", "20017083E");

    Pacient pacient = new Pacient(p12Pacient, "uoc0506", p12Gestor,
    "uoc0506");
    GestorBBDD.InserirUsuari(pacient.idUsuariU,
    pacient.x509.getEncoded(), dadesAdm.getDocInBytes());

    // Inserim el pacient a la llista de pacients del metge
      GestorBBDD.InserirPacientsMetge(metge.idUsuariU,
      pacient.idUsuariU);
}
```

7. INTERFÍCIE DE L'APLICACIÓ

7.1 INTRODUCCIÓ

La interfície de l'aplicació, es la forma en que els usuaris es comuniques amb l'aplicació i els permet realitzar les accions que el sistema dispose. Així el que es pretén en la implementació de la interfície gràfica de l'aplicació, és que els usuaris pugen interactuar amb el sistema de forma ràpida, simple i intuïtiva.

Continuant amb la implementació del patró arquitectònic Model Vista Controlador (MVC) utilitzat al llarg del PFC, la interfície de l'aplicació correspon llavors amb el component Vista, que es un component diferenciat del component Controlador que correspon a la lògica de l'aplicació i el Model que correspon al sistema gestor de la base de dades.

La interfície gràfica de l'aplicació ha està dividida en dues part la dels clients pacients i metges, i una segona interfície per als gestors del sistema. La implementació realitzada en les dues interfícies gràfiques, ha seguit el mateix estil de disseny dels formularis i menús utilitzats en l'aplicació.

En la interfície dels clients, que es comuna per als pacients i per als metges es personalitzarà en el moment d'iniciar la sessió, on segons el tipus de client s'activaran les opcions de menú que tinguen accés.

7.2 IMPLEMENTACIÓ I DESCRIPCIO DELS COMPONENTS DE LA INTERFICIE

Per a implementar la interfície gràfica s'ha utilitzat dintre de l'entorn de desenvolupament de l'eclipse el plugin "visual editor project".

S'han generat les següents classes següents , que son comunes a tota la interfície de l'aplicació:

- **CampData**

Aquesta classe que hereta de "JPanel", implementa un camp que conté una data que es del tipus "JFormattedTextField" amb el format de la data "dd/MM/yyy", i que disposa de la facilitat d'incrementar i disminuir els dies, mesos o anys, segons estiga posicionat el cursor, amb les tecles de fletxes de cap a dalt i cap a baix. Així doncs, en tots els formularis on s'ha utilitzat un camp de tipus data s'ha fet servir aquesta classe.

- **CampBucarFitxer i CampBuscaFitxer_Filtro**

La classe CampBuscarFitxer que hereta de ”JPanel”, implementa un camp que conté la ruta d’un fitxer amb l’opció de poder buscar la ruta del fitxer pel sistema prement el boto adjunt al camp de text. A més amb la classe CampBuscarFitxer_Filtro que hereta “FileFilter” establim quina extensió dels fitxers seran filtrats durant la recerca del fitxer. Així aquest camp s’ha utilitzat en els formularis d’inici de sessió i en els de Configuració, per tal d’introduir el fitxer del certificat digital.

Ja en els formularis que realitzen la funcionalitat de l’aplicació, disposen d’un menú amb les següents funcionalitats que són comunes en les dues interfícies existents en l’aplicació:

- El menú “Inici”, disposem en la primera l’opció “Iniciar sessió” que sols està habilitada quan no existeix un usuari autentificat. La segona opció “Finalitzar sessió” permetrà a l’usuari finalitzar la sessió iniciada, amb la qual cosa es tancaran automàticament totes les finestres que estiguen obertes en el sistema i es deshabilitaran les opcions del menú propies de l’usuari que s’ha autentificat i finament s’activarà la primera opció del menú “Iniciar sessió”. Finalment disposem de l’opció “Sortir” que directament tanca tota l’aplicació.

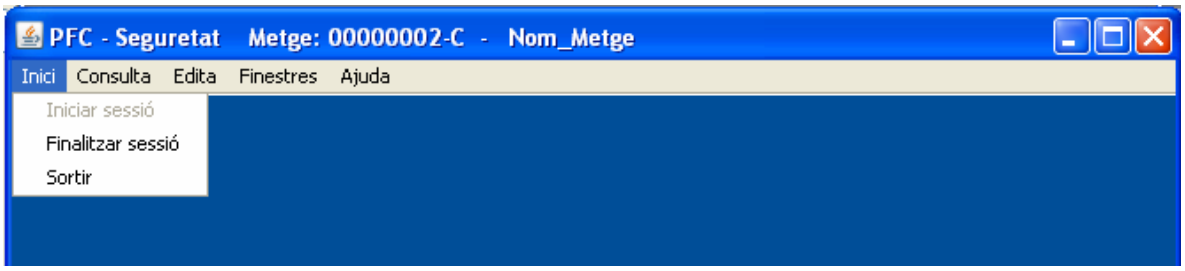


Figura 7.1 Menú “Inici”

- En el menú “Editar” disposem de l’opció “Configuració” que mostrarà un formulari per a gestionar el fitxer de configuració segons s’haja autentificat un usuari client o l’usuari gestor (en els següents apartats detallarem aquest formulari) :

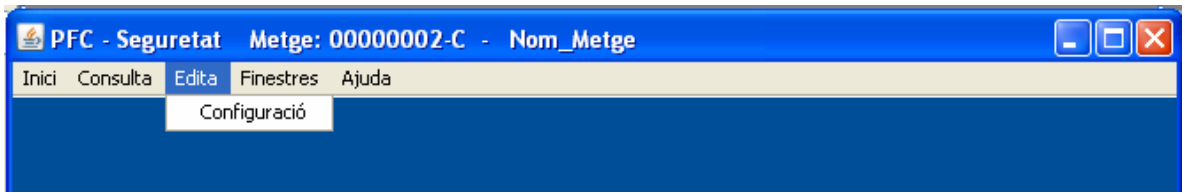


Figura 7.2 Menú “Edita”

- En el menú “Finestres”, es mostren totes les finestres que estan obertes en el sistema, per tal que l’usuari puga seleccionar-ne una per tal d’activar-la. A més es disposa d’una opció per tal de “Tancar totes les finestres”:

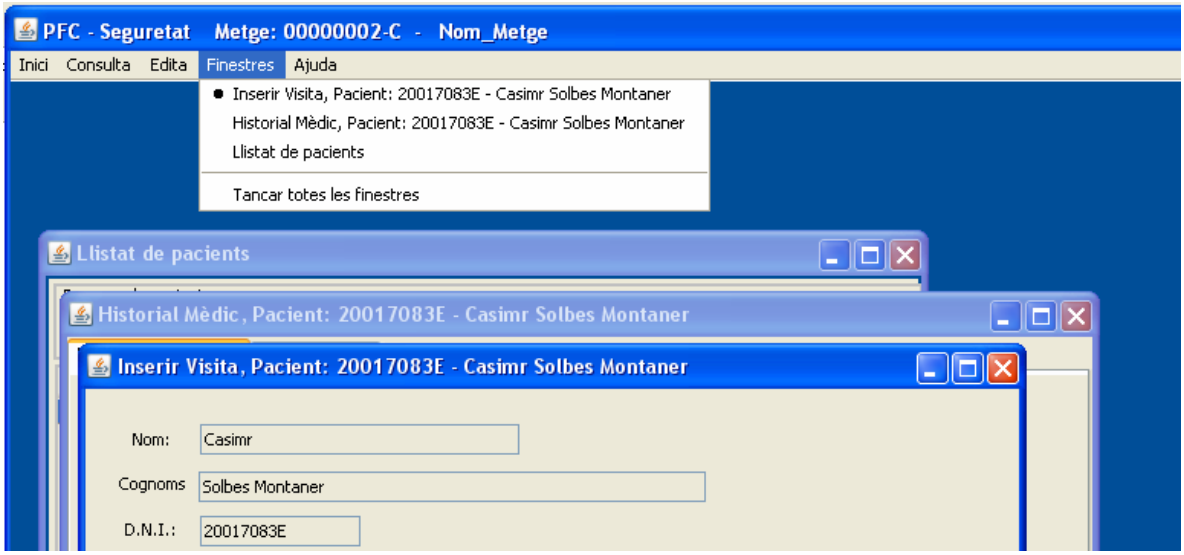


Figura 7.3 Menú “Finestres”

- En el menú “Ajuda” tenim l’opció “Sobre el PFC” on s’informa sobre el títol, l’autor i el consultor del PFC:

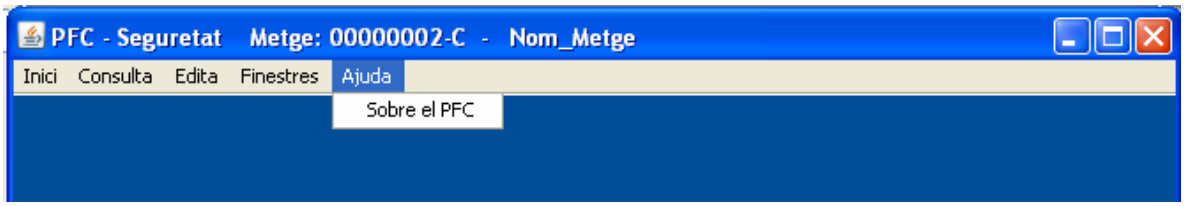


Figura 7.4 Menú “Ajuda”

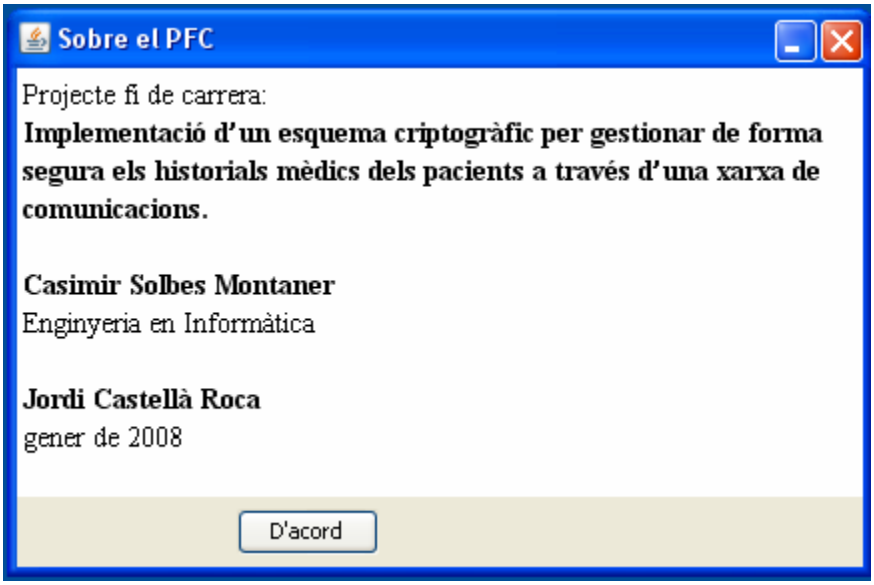


Figura 7.5 Formulari Sobre el PFC

En aquest apartat ens hem deixat per comentar el menú “Consulta”, ja que aquest menú disposa de les opcions especifiques per a cada tipus d’usuari, que comentarem en els següents apartats.

7.3 INTERFICIE DELS CLIENTS

La interfície dels clients, tant dels metges com del pacients es una única interfície, on quan s’autentifica cada usuari s’activen les opcions dels menús especifiques per a metges o per a pacients.

Passem a descriure les classes generades durant la implementació:

- **Principal**

En esta classe que hereta de JFrame, s’implementa el formulari principal de l’aplicació que conté les opcions del menú, així com el diàleg d’inici de sessió on es sol·licita la identificació del client (certificat del client i la contrasenya del certificat) i a més també inclou el formulari de informatiu “Sobre el PFC”.

La resta de formularis de la interfície que s’executen a partir d’aquest formulari principal hereten de la classe JInternalFrame, així aconseguim que aquest formulari siga el formulari principal de l’aplicació.

Llavors, quan s’inicia l’execució de l’aplicació ens mostra directament el diàleg d’inici de sessió :

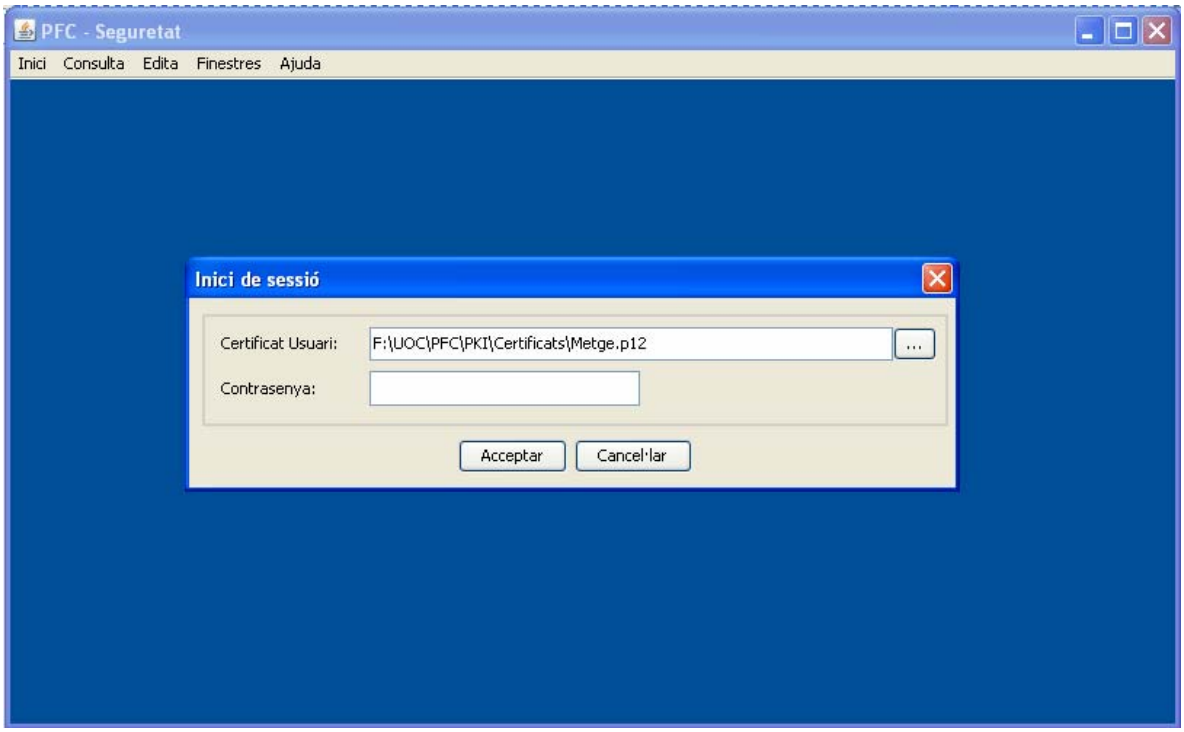


Figura 7.5 Formulari Principal – Inici sessió

Una vegada iniciada la sessió en el títol del formulari ens mostra el tipus d'usuari amb el seu identificador i el seu nom, i ja podrem accedir al menú que tindrà activades les opcions de menú actives segons el tipus d'usuari.

Per tal de facilitar a l'usuari habitual de l'aplicació, l'autentificació davant el sistema, en el fitxer de configuració de l'usuari s'ha afegit un nou paràmetre "<Path_Cert_Usuari>" (que no es mostra en el formulari del manteniment de la configuració de l'usuari que més endavant vorem), on guardarem la ruta de l'últim certificat que l'usuari ha utilitzat per a iniciar la sessió en el sistema, i així en els successius inicis de sessió el camp "Certificat Usuari" s'inicialitzarà amb el darrer valor utilitzat:

```
<Configuracio>
        ...
    <Path_Cert_Usuari >
        F:\UOC\PFC\PKI\Certificats\Metge.p12
    </Path_Cert_Usuari >
        ...
</Configuracio>
```

- **Configuració**

En el menú “Editar” disposem de l’opció “Configuració” per a gestionar el fitxer de configuració, anomenat “configuracioUsuari.xml” i que cada instal·lació de l’aplicació client ha de tindre. Mostrem l’aspecte del formulari comentat:

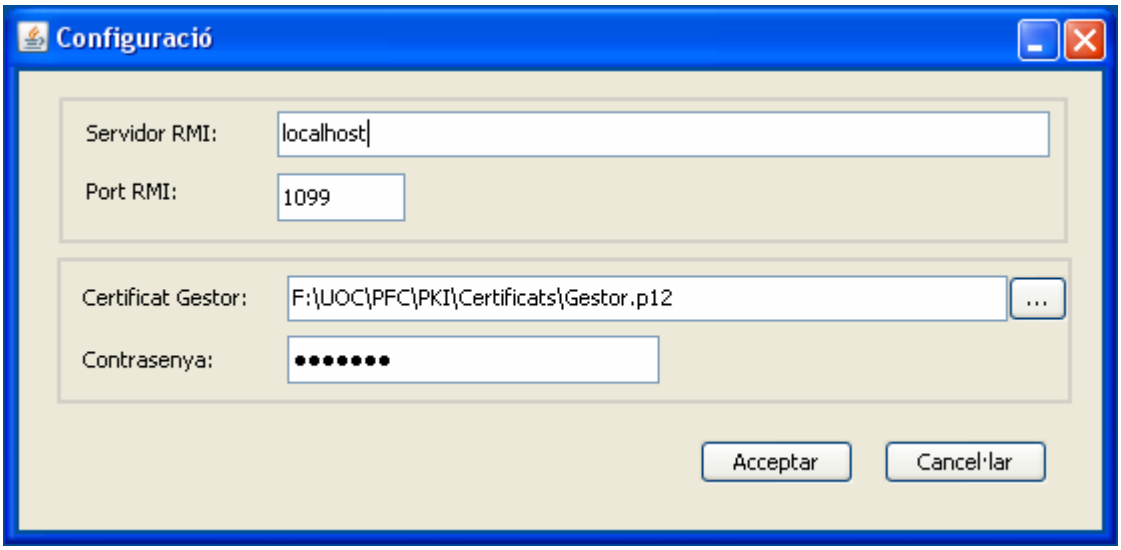


Figura 7.7 Formulari Configuració Clients

7.3.1 INTERFICIE DEL METGE

Els Metges disposaran en el menú “Consulta” les opcions especifiques que poden realitzar, així els Metges tindran l’opció “Llistat de pacients” que mostrarà els pacients que tenen assignats:

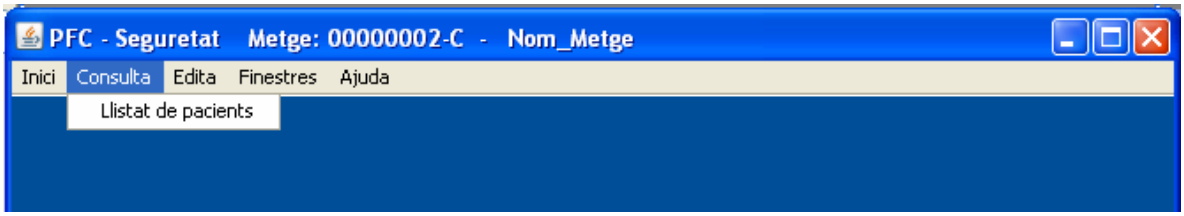


Figura 7.8 Menú Metge - Consulta

- **LlistatDePacients**

Aquesta classe representa el llistat dels pacients assignats a un metge. Disposa de dos camps per a poder filtrar les entrades de la llista, tant pel DNI com pel nom i cognoms del pacient.
Una vegada el metge a localitzat al pacient podrà accedir a l’Historial del pacients prement el botó “Consulta Historial”:

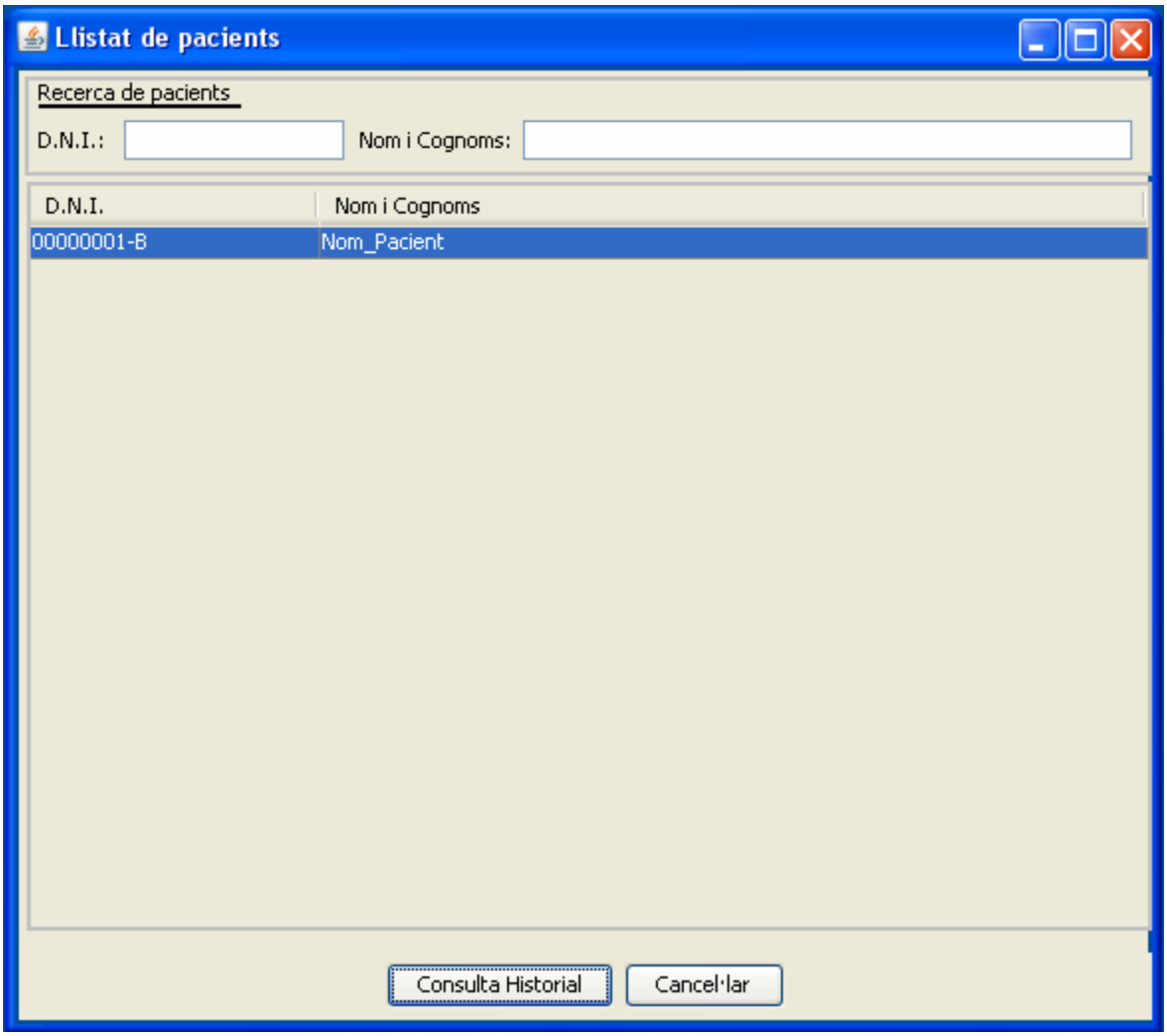


Figura 7.9 Formulari Llistat de Pacients

- **HistorialMedic**

Aquesta classe implementa el formulari de l'historial mèdic dels pacients. El formulari té dues pestanyes on en la primera es visualitzen les dades administratives del pacient:

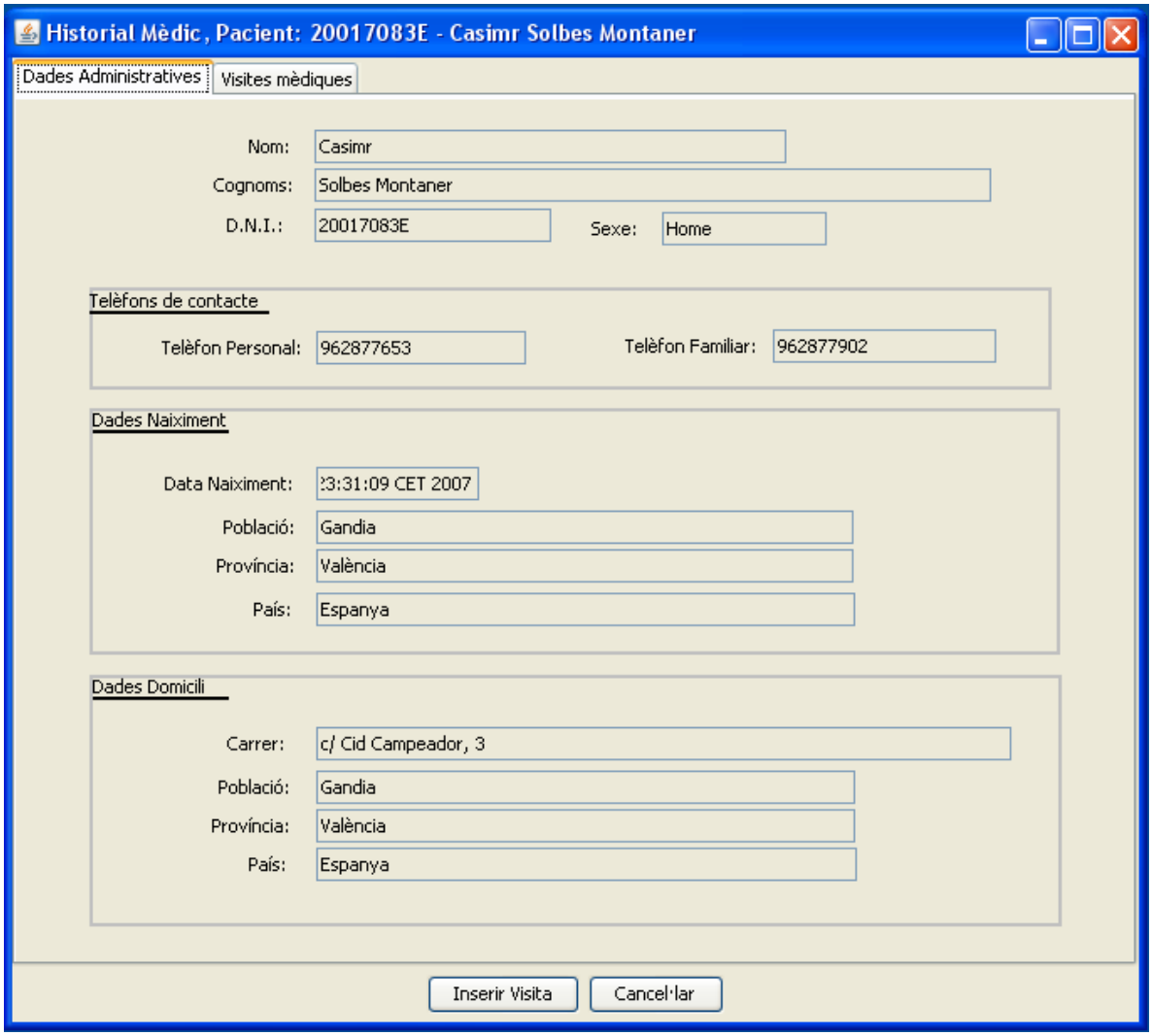


Figura 7.10 Formulari Historial Mèdic – Dades Administratives (Vista Metge)

i en la segona pestanya es poden consultar les visites que el pacient a realitzat al metge:

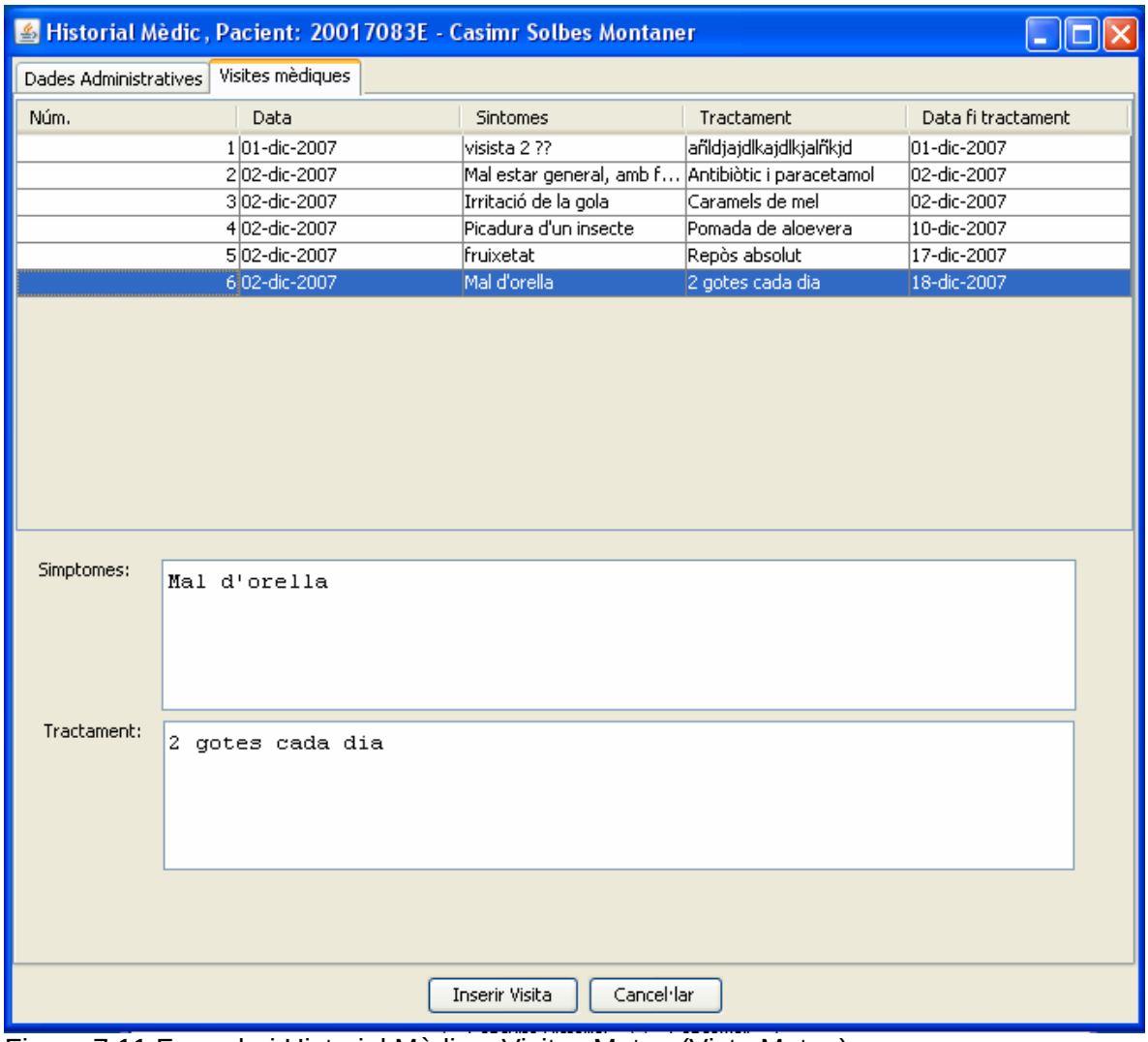


Figura 7.11 Formulari Historial Mèdic – Visites Metge (Vista Metge)

El formulari de l'historial mèdic disposa d'un botó per tal de inserir una nova visita a l'historial del pacient. Aquesta opció sols serà visible si l'usuari que ha accedit al formulari es un metge.

- **InserirVisita**

Aquesta classe representa el formulari d'introducció de les dades d'una visita al metge per part del pacient. En la capçalera del formulari es poden visualitzar les dades identificatives del pacient i en la resta del formulari es troben els camps editables que contenen la informació de les visites:

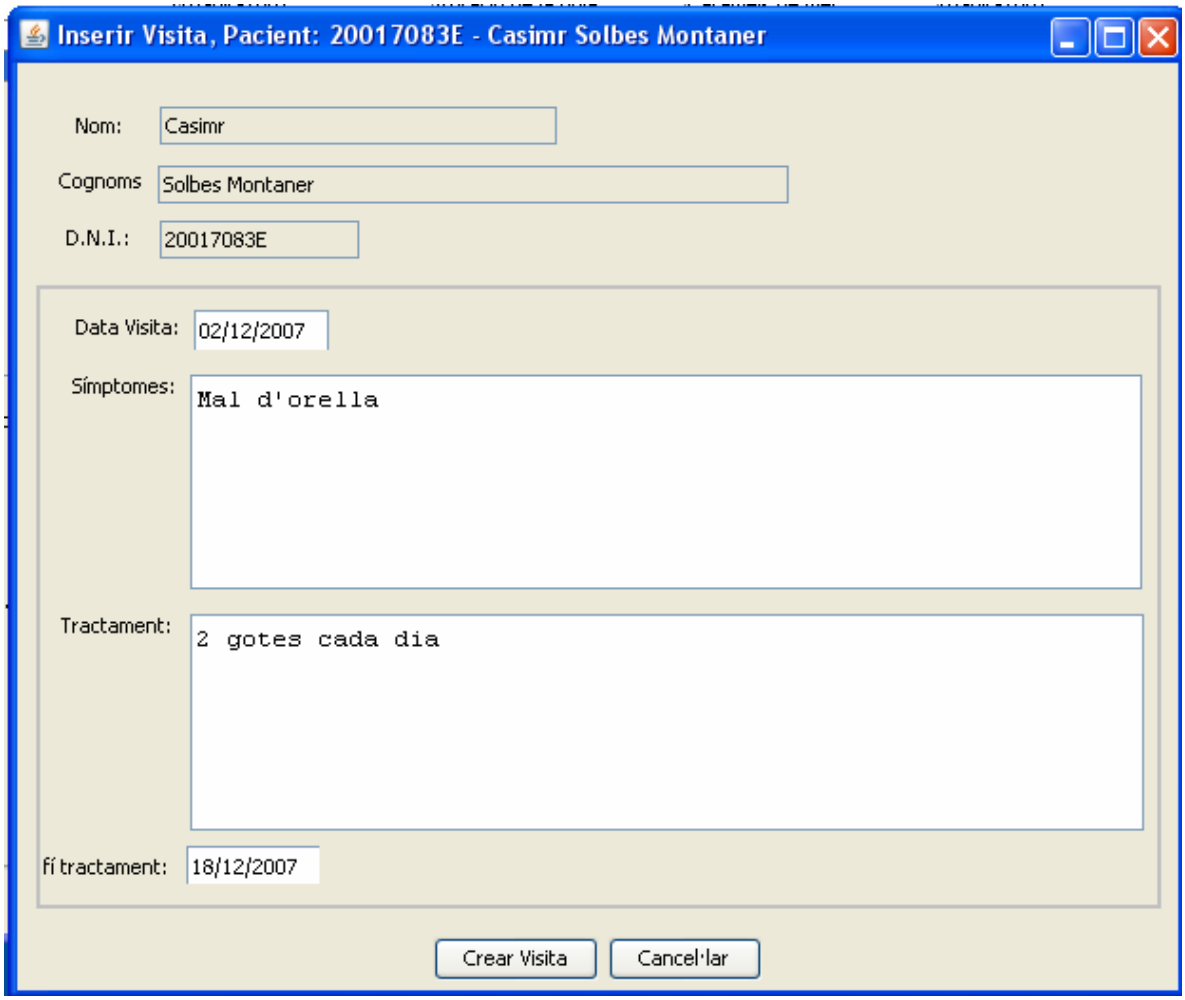


Figura 7.12 Formulari Inserir Visita

Una vegada l'usuari ha introduït les dades de la visita, amb el botó "Crear Visita" inserirà la visita en el sistema, tancant-se el formulari i tornant a visualitzar l'historial del pacient amb la nova visita ja introduïda en la pestanya "Visites mèdiques"

7.3.2 INTERFICIE DEL PACIENT

En el menú "Consulta" disposem de les opcions especifiques que els Pacients poden realitzar, en aquest cas el Pacient té disponible l'opció "Historial" per a realitzar es la consulta del seu Historial:

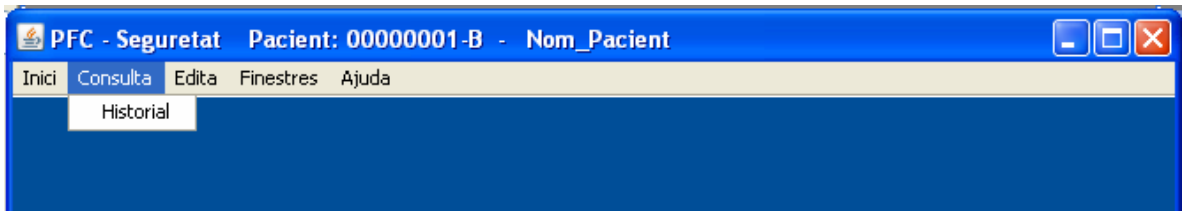


Figura 7.13 Menú Pacient - Consulta

Com a resultat de la consulta de l'historial el pacient visualitzarà el formulari de l'Historial Mèdic, que és el mateix formulari que el presentat en l'apartat anterior per al Metge, amb la única diferència que no disposa de l'opció de crear una visita:

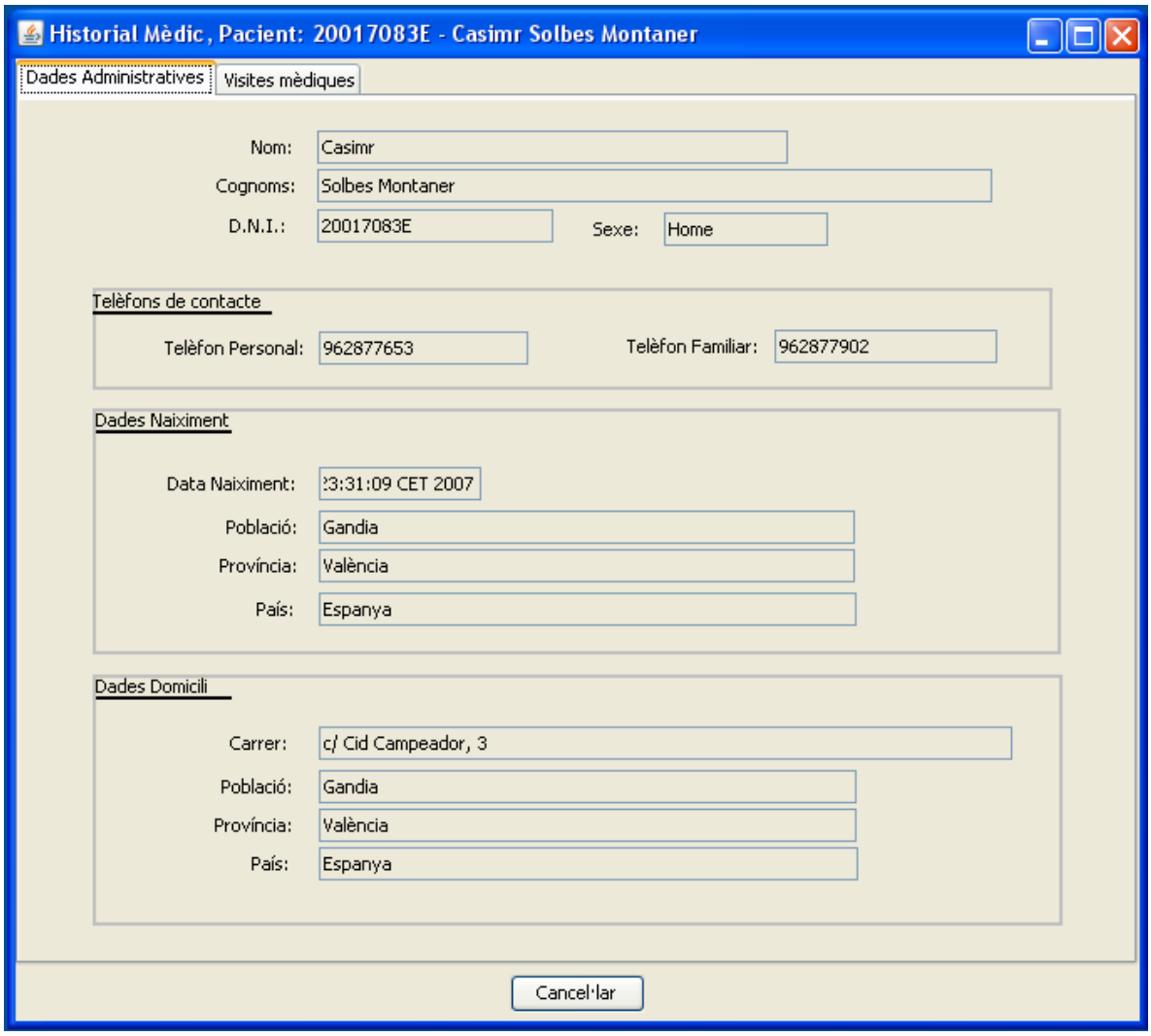


Figura 7.14 Formulari Historial Mèdic – Dades Administratives (Vista Pacient)

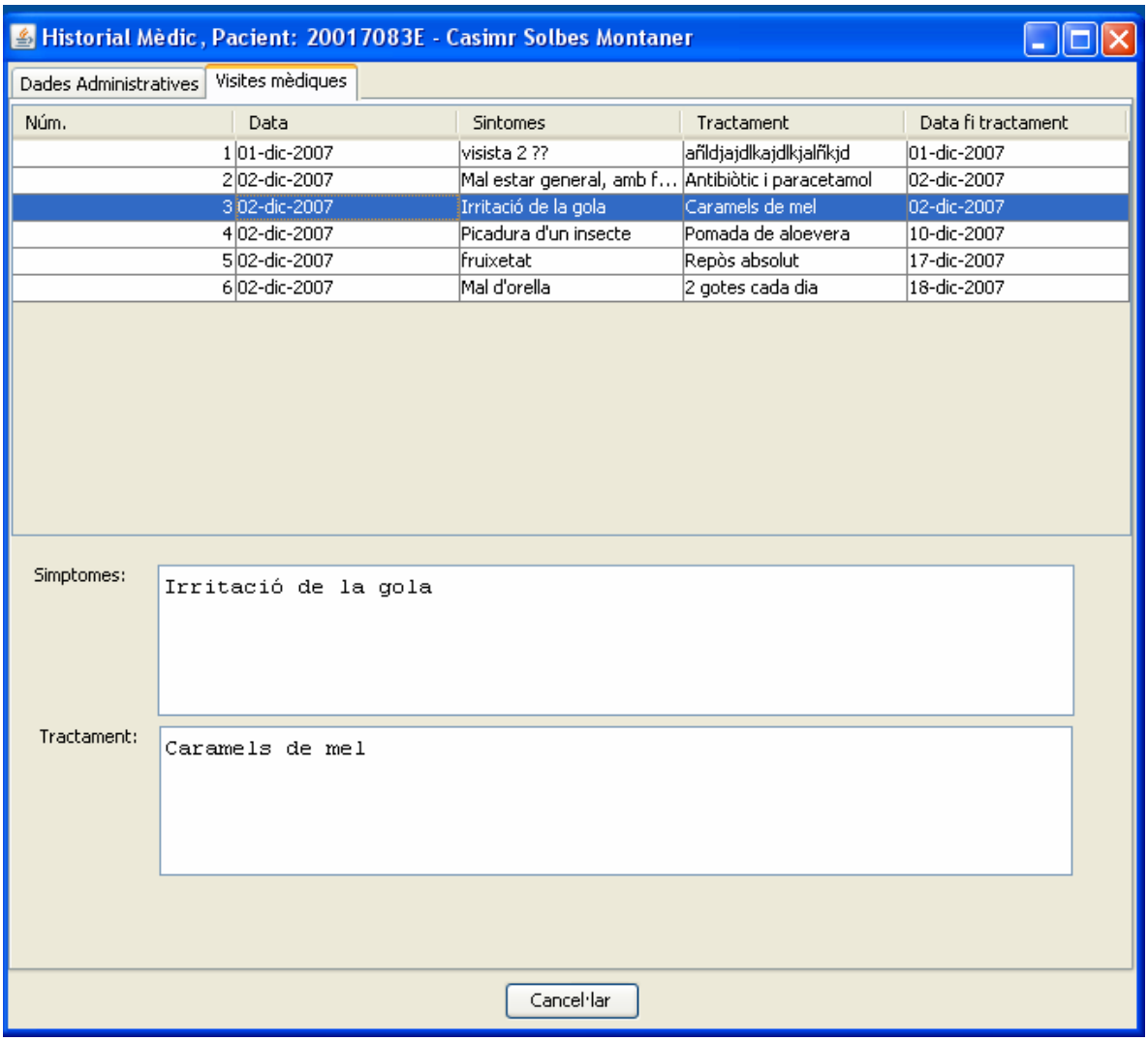


Figura 7.15 Formulari Historial Mèdic – Visites Metge (Vista Pacient)

7.4 INTERFICIE DEL GESTOR DEL SISTEMA

En aquest apartat descriurem la interfície del gestor, pel que fa a les funcionalitats especifiques del gestor del sistema, així s'han generat durant la implementació les següents classes:

- **Principal_Gestor**

En esta classe que hereta de JFrame, s'implementa el formulari principal de la interfície del gestor que conté les opcions del menú, així com el diàleg d'inici de sessió on es sol·licita la identificació del gestor (certificat del gestor i la contrasenya del certificat.

Al igual que en el formulari “Principal” de la interfície dels Usuaris, la resta de que s’executen a partir d’aquest formulari principal del gestor hereten de la classe JInternalFrame, així aconseguim que aquest formulari siga el formulari principal de l’aplicació.

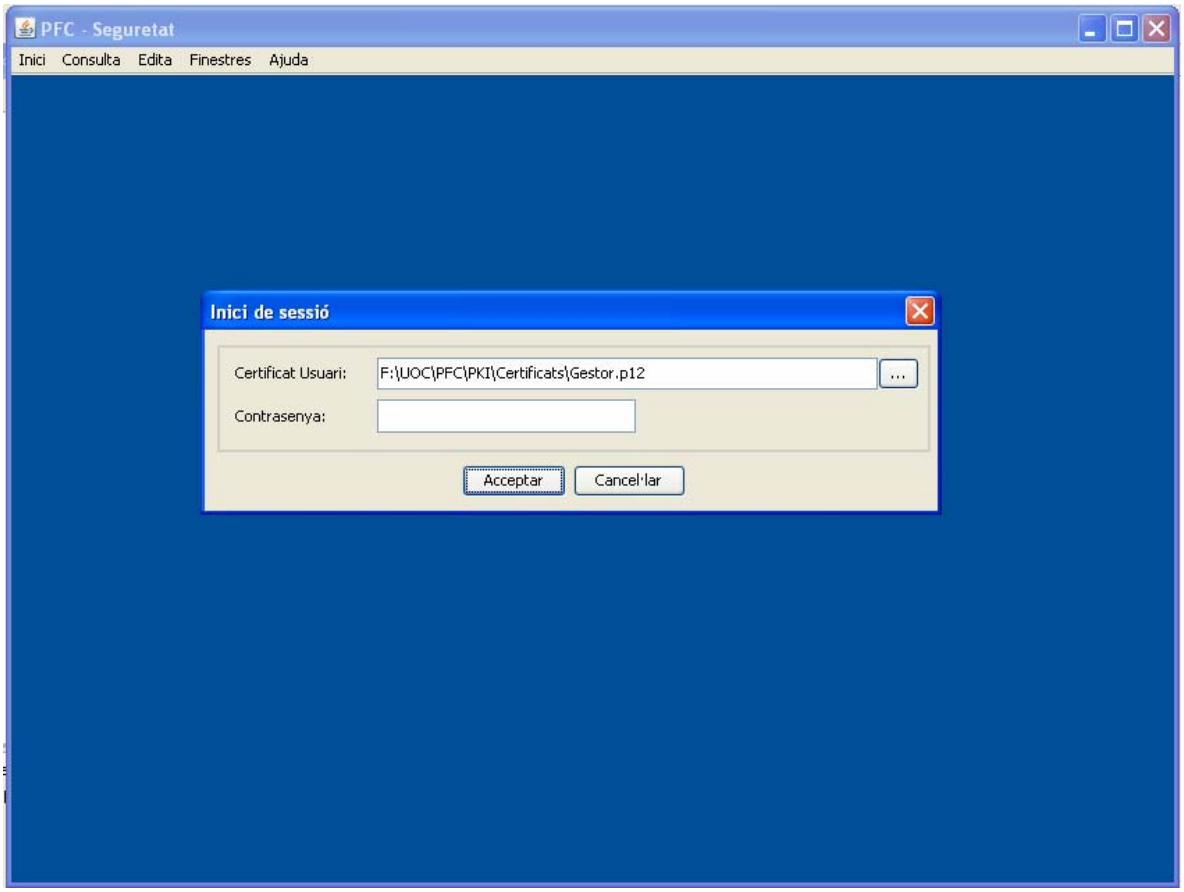


Figura 7.16 Formulari Principal Gestor – Inici sessió

Una vegada Iniciada la sessió en el menú “Consulta” l’aplicació tindrem l’opció “Llistat d’usuaris”:

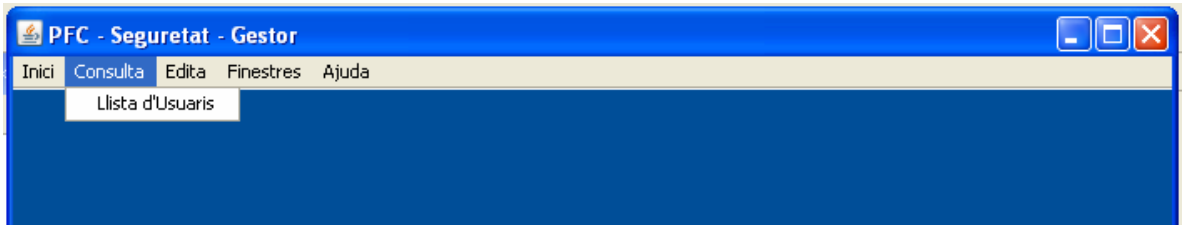


Figura 7.17 Menú Gestor – Consulta

- **Llistat d’Usuaris**

Aquesta classe representa el llistat de tots els usuaris clients registrats en el sistema. En el formulari es disposa de dos camps per a poder filtrar les entrades de la llista, tant pel DNI com pel nom i cognoms dels usuaris :

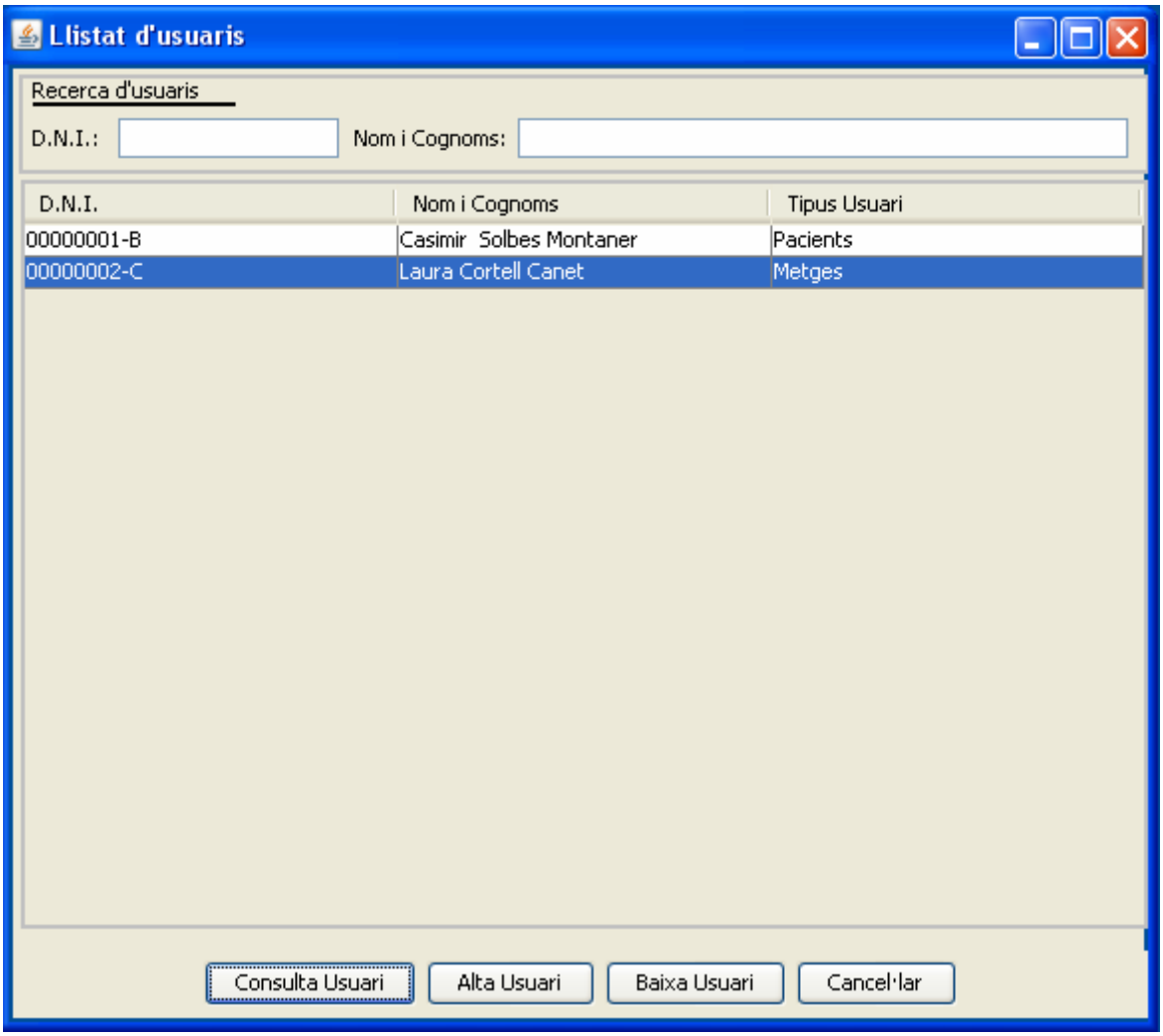


Figura 7.18 Formulari Llistat Usuaris

A partir d'aquest formulari el gestor disposa de les opcions de "Consulta Usuari", "Alta Usuari" i "Baixa Usuari", opcions que en els següents punts anem a tractar.

- **Consulta Usuaris**

Aquesta classe implementa el formulari de consulta de les dades d'un usuari amb la possibilitat d'actualitzar les dades ja que disposa de tots els camps editables exceptuant el camp identificador de l'usuari el DNI. Així tant per als usuaris metges com per als pacients el formulari contindrà una pestanya amb les dades Administratives, i per tal de poder confirmar l'actualització de les dades l'usuari ha de prémer el botó "Actualitzar dades administratives".

En la següent imatge visualitzem la consulta d'un usuari de tipus pacient:

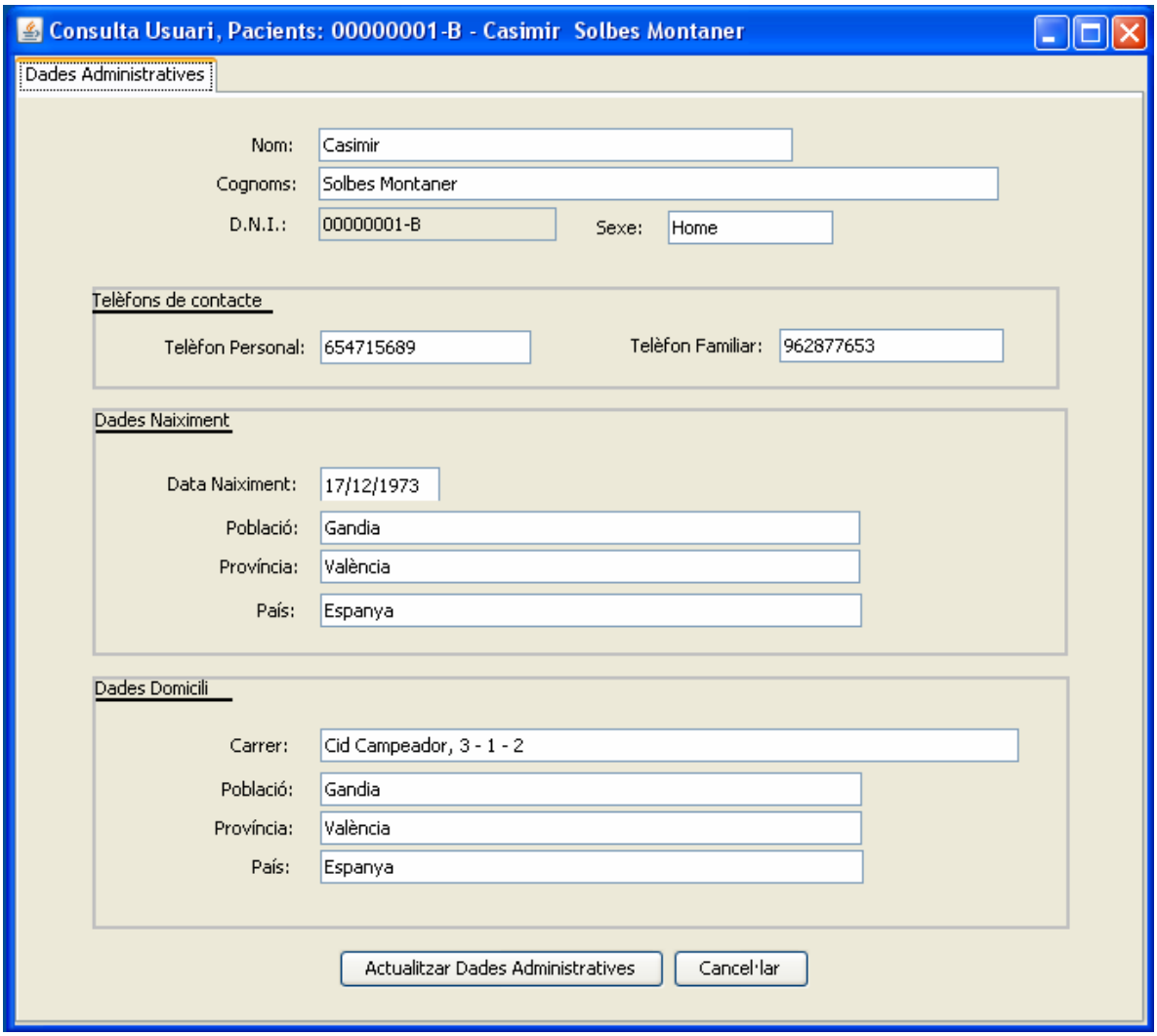


Figura 7.19 Formulari Consulta Usuari - Pacient

En la consulta d'un usuari de tipus metge, conté una pestanya més anomenada "Pacients del Metge" on es pot gestionar els pacients que estan assignats al metge. Existeixen dues columnes una amb els pacients assignats al metge i l'altra amb els pacients que no tenen cap metge assignat, així amb els botons indicats amb fletxes es podem moure els pacients de cadascuna de les columnes fet que realitzarà l'assignació / desassignació d'un pacient a un metge.

Vegem a continuació les imatges de la consulta d'un usuari de tipus metge:

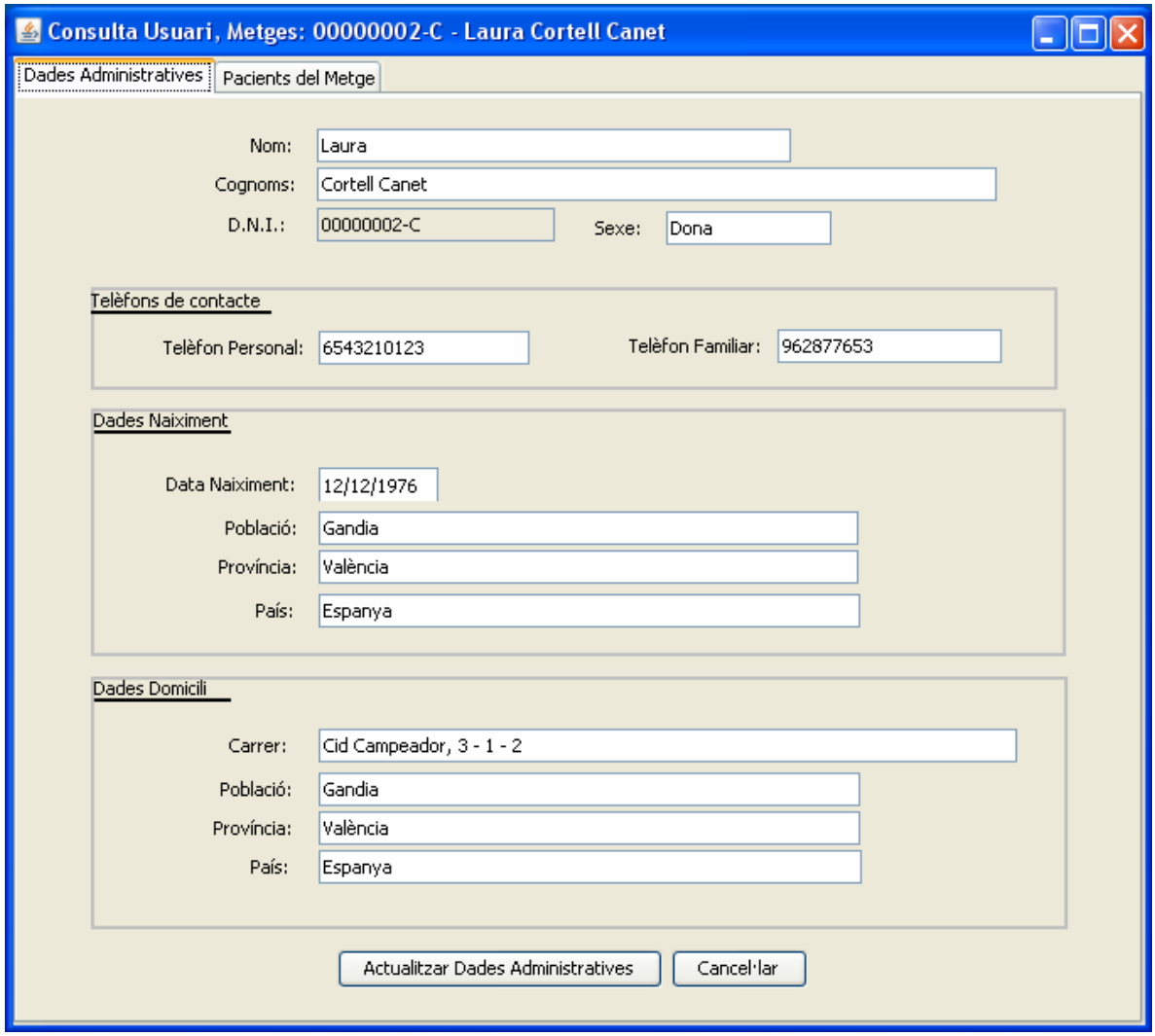


Figura 7.20 Formulari Consulta Usuari – Metge Dades Administratives

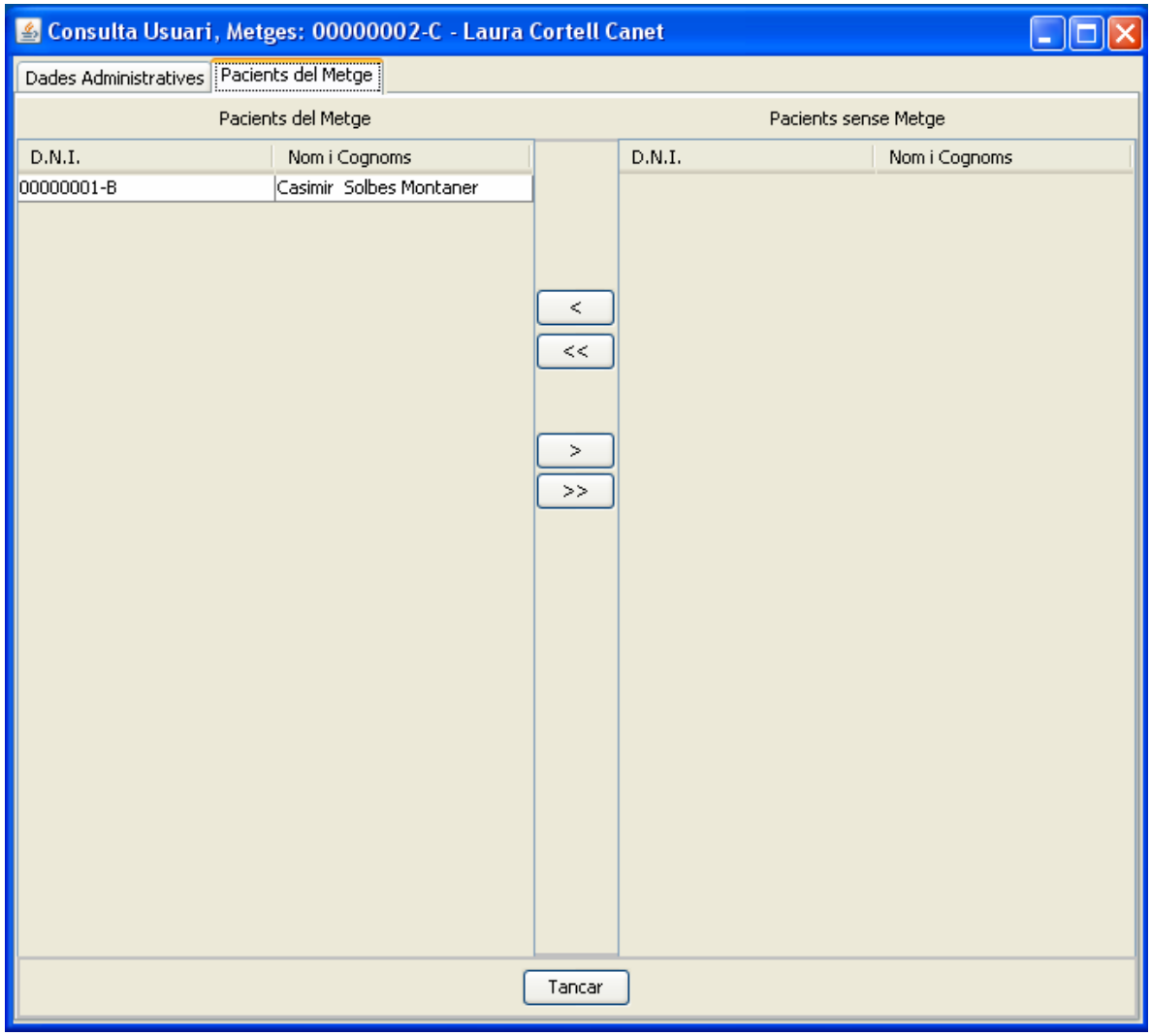


Figura 7.21 Formulari Consulta Usuari – Metge Pacients del Metge

- **Alta Usuari**

Aquesta classe representa el formulari d'alta d'un usuari on s'introdueix el certificat del l'usuari amb la seva contrasenya i les dades administratives. Com a primer pas per a donar d'alta un usuari s'ha d'introduir el certificat i la contrasenya de l'usuari, i prémer el botó "Carregar", aleshores els camps de les dades administratives s'inicialitzen amb els valors obtinguts del certificat i es permet que els camps siguen editables excepte el camp DNI ja que aquest camp serà el identificatiu de l'usuari:

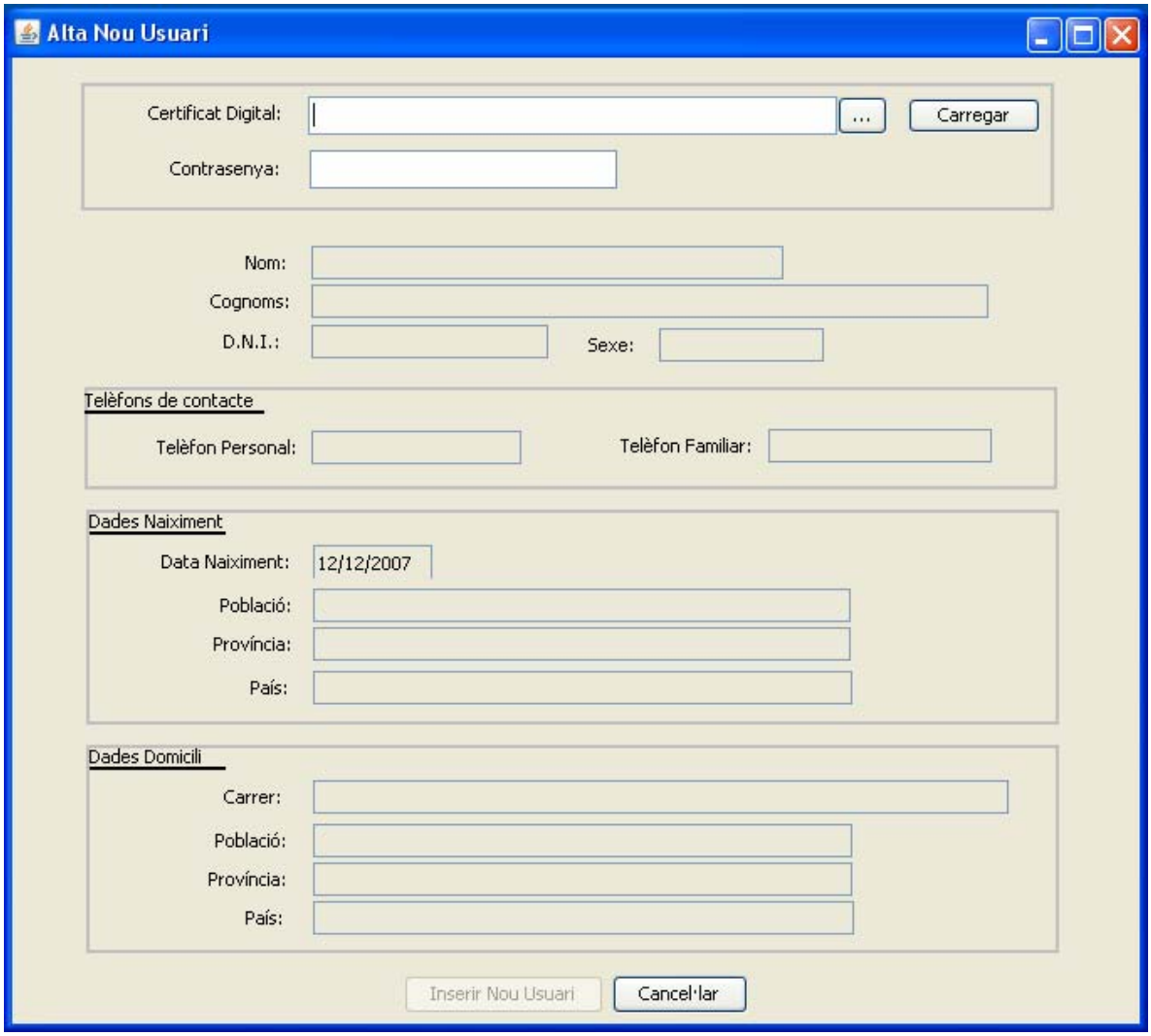


Figura 7.22 Formulari Alta Nou Usuari (1)

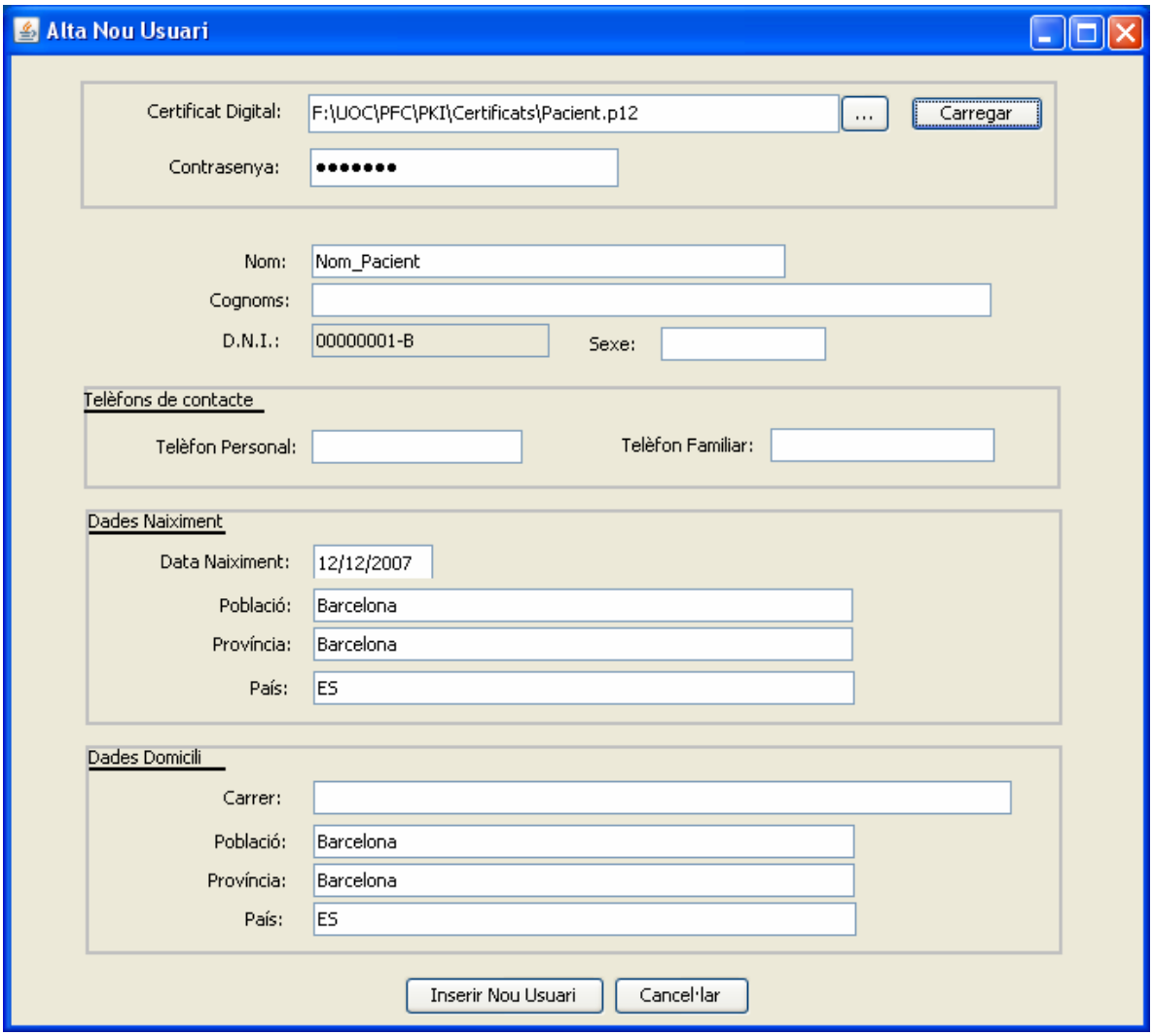


Figura 7.23 Formulari Alta Nou Usuari (2)

- **Baixa Usuaris**

La baixa dels usuaris, es una funcionalitat integrada dintre de la classe “Llistat d’usuaris”, on a partir de l’usuari seleccionat en la consulta dels usuaris, prement el boto “Baixa Usuari” i confirmat l’operació, s’elimina l’usuari de la base de dades.

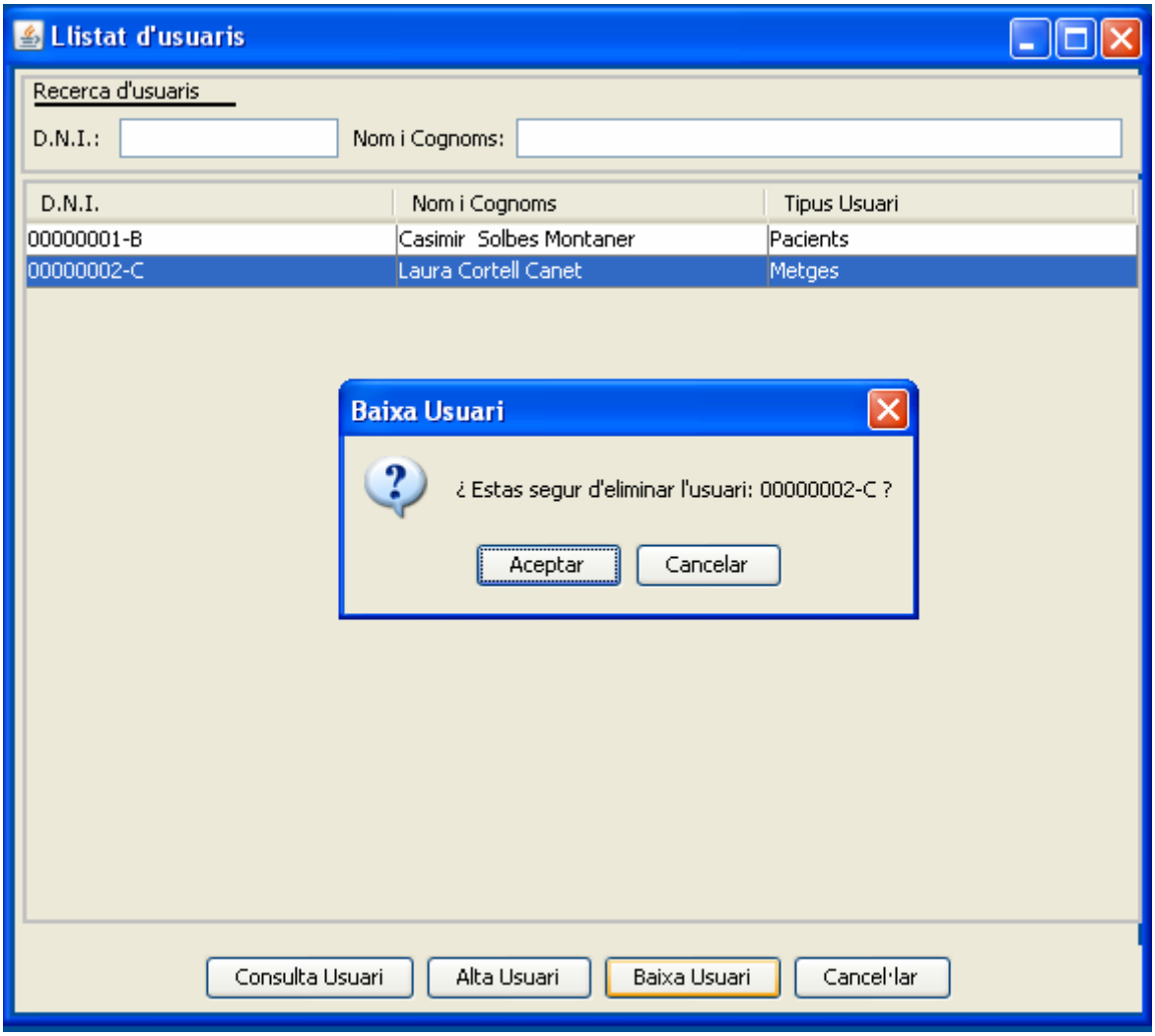


Figura 7.24 Formulari Llistat Usuaris – Baixa Usuari

- **Configuració Gestor**

En el menú “Editar” disposem de l’opció “Configuració” per a gestionar el fitxer de configuració del gestor, anomenat “configuracioGestor.xml” i que cada instal·lació de l’aplicació gestor ha de tindre. Mostrem l’aspecte del formulari comentat:

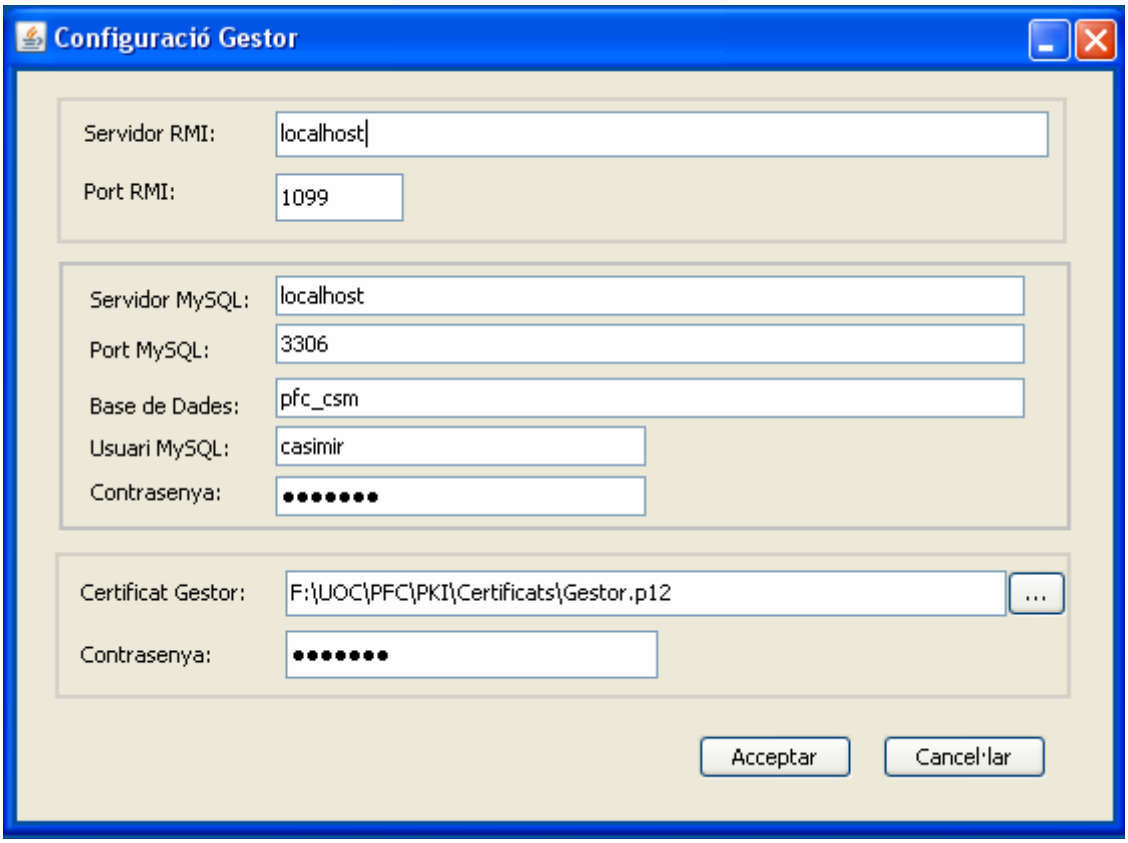


Figura 7.25 Formulari Configuració Gestor

7.5 CAPTURA DELS ERRORS

En la interfície de l'aplicació s'ha implementat la captura dels errors que es pugen generar durant l'execució de l'aplicació, així s'ha personalitzat el missatge amb el text "No s'ha pogut realitzar la petició Sol·licitada:" i a afegint el missatge de l'excepció capturada:

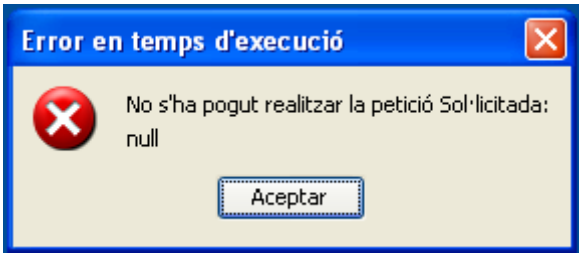


Figura 7.26 Captura errors

CONCLUSIONS

Una vegada desenvolupat ja tot el PFC, s'ha complit l'objectiu de la implementació d'un esquema criptogràfic per a garantir les propietats de seguretat (confidencialitat, autenticitat, integritat, no-repudi) d'un historial mèdic gestionat mitjançant una xarxa de comunicacions.

Així, podem concloure que en la realització del PFC s'han satisfet els propòsits de la present assignatura, ja que s'han posat en pràctica diversos coneixements adquirits i consolidats al llarg de l'estudi de diverses assignatures cursades en l'enginyeria informàtica, com:

- Criptografia: tant en conceptes com en la definició, implementació de l'esquema criptogràfic, i en la utilització de la llibreria criptogràfica Apache XML-Security 1.2.1.
- Seguretat en Xarxes de Computadors: creació de la infraestructura de clau pública PKI, utilitzant l'eina OpenSLL
- Comerç electrònic i Enginyeria del programari de components i sistemes distribuïts: en la utilització del sistema de base de dades MySQL.
- Arquitectura de sistemes distribuïts: la comunicació entre les aplicacions amb la tecnologia Remote Invocation RMI de Java.

I a més també s'ha requerit de la recerca de coneixements més específics que no s'han impartit en cap assignatura de la carrera, com ha segut amb la implementació de les interfícies gràfiques en el llenguatge Java, mitjançant la utilització de l'eina Visual Editor de l'Eclipse, ja que ha segut l'àrea de Java que menys coneixia i on la font principal d'informació que he utilitzat per a dur a terme aquesta recerca de informació, ha estat la web, cercant en els fòrums, articles i tutorials.

Respecte a la realització del projecte, cal dir que per a la planificació inicial de la implementació de cadascuna de les diferents fases, s'han complert perfectament els terminis de temps establerts. Així com els jocs de proves realitzats en cadascuna de les fases han segut de gran utilitat, pel fet que en les següents fases et podies despreocupar dels resultats de les fases anterior i centrar-te en la estaves realitzant.

Finalment, la realització del PFC ha resultat una experiència totalment satisfactòria pel fet de concentrar diversos coneixement en un únic treball, i especialment per considerar que les aplicacions que funcionen sobre xarxes de comunicacions requeriran cada vegada més, el compliment de les propietats de seguretat (confidencialitat, autenticitat, integritat, no-repudi) que s'han implementat mitjançant l'esquema criptogràfic definit en aquest projecte, i que de ben segur aquesta experiència em serà professionalment de gran utilitat.

GLOSSARI

API: (*Application Programming Interface*) Interfície de programació d'aplicacions. Programari que proporciona un conjunt de mètodes per a ser utilitzat com una capa d'abstracció.

CA: Autoritat de certificació de confiança, responsable d'emetre i revocar els certificats digitals.

DES: Data Encryption Standard, és un criptosistema de xifratge de bloc que xifra blocs de dades de 64 bits de llargada per mitjà d'una clan de 56 bits.

DTD: Document Type Definition. Definició de tipus del document (DTD) és una descripció de l'estructura i sintaxis d'un document XML.

IETF: (***I****nternet* ***E****ngineering* ***T****ask* ***F****orce*), Grup de Treball en Enginyeria de Internet. És una organització internacional considerada com una autoritat, que té com a objectius contribuir a la enginyeria de Internet.

JAVA: Llenguatge de programació orientat a objectes desenvolupat per Sun Microsystem.

JDBC: (*Java Database Connectivity*) Interfície que permet connectar i executar operacions sobre base de dades, amb el llenguatge Java.

OpenSSL: paquet d'eines d'administració i funcions criptogràfiques per gestionar certificats criptogràfics.

PKCS: Public-Key Cryptography Standards. Conjunt d'estàndards de criptografia de clau pública.

PKI: Infraestructura de clau pública, es el conjunt de maquinari i programari, politiques i procediments de seguretat que permeten l'execució amb garantía d'operacions criptogràfiques.

Plugin: (plug-in) És una aplicació informàtica que interactuar amb una altra aplicació per a apartar-li una funcionalitat o utilitat especifica.

RA: Autoritat de Registre. Entitat que per delegació d'una Autoritat de Certificació (CA), s'encarrega de donar fe de la identitat dels sol·licitants de certificats digitals, gestionar les seus sol·licituds i remetre-les a la CA per a que siguen expedits els certificats digitals.

RMI: (*Java Remote Method Invocation*) es un mecanisme ofert en Java per a invocar un mètode remotament.

RSA: Rivest, Shamir, Adelman. Algorisme d'encriptació de clau pública de clau pública desenvolupat per les ter persones mencionades, que utilitza un algorisme asimètric amb xifrat de blocs, on la clan pública se distribueix de forma autentificada, i la clau privada es guardada en secret pel seu propietari.

SGBD: Sistema de gestió de base de dades**,** és un programari específic que proporciona una interfície entre la base de dades, l'usuari i les aplicacions que l'utilitzen. Disposa d'un llenguatge de definició, manipulació i consulta de les dades.

SSL: Secure Sockets Layer. Protocol criptogràfic que proporciona comunicacions segures en Internet.

TLS: Transport Layer Security. Protocol criptogràfic que proporciona comunicacions segures en Internet.

Triple DES: Protocol de xifratge triple que utilitza el criptosistema DES com a base per a obtenir un xifrador amb un espai de claus superior.

UML: *Unified Modeling Language*. Llenguatge de modelació de sistemes de programari.

W3C: World Wide Web Consortium, abreviat W3C, és un consorci internacional que produeix estàndards per a Internet.

XML: *e**X**tensible **M**arkup **L**anguage.* Llenguatge de marques extensible que permet crear llenguatges d'etiquetes i que també s'utilitza per a l'intercanvi de informació estructurada entre diferents plataformes.

BIBLIOGRAFIA

- Apunts de Criptografia UOC, edició febrer 2006.
  Document annex XMLDsig i XMLEnc.

- Apunts de Seguretat en Xarxes de Computadors UOC, edició febre 2006.
  Document: L'eina OpenSSL febre 2005

- Apunts de Arquitectura de Sistemes Distribuïts UOC, edició febrer 2004.
  Document: Interacció entre Aplicacions: objectes distribuïts i invocació remota

- Tutorial DTD www.w3schools.com/dtd.

- Enginyeria del software Disseny I. Edicions UPC, 1999.

- Enginyeria del software Disseny II. Edicions UPC, 1999.

- Manual de Referència MySQL 5.0

- Wikipedia http://www.wikipedia.org

- Tutorial de swing: http://java.sun.com/docs/books/tutorial/uiswing/

ANNEXOS

ANNEX A. FITXER DE CONFIGURACIÓ OPENSLL

```
[ req ]
default_bits = 1024
default_keyfile = privkey.pem
distinguished_name = req_distinguished_name
attributes = req_attributes
x509_extensions = usr_cert # The extentions to add
# Passwords for private keys if not present they will be prompted for
# input_password = secret
# output_password = secret
# This sets a mask for permitted string types. There are several options.
# default: PrintableString, T61String, BMPString.
# pkix : PrintableString, BMPString.
# utf8only: only UTF8Strings.
# nombstr : PrintableString, T61String (no BMPStrings or UTF8Strings).
# MASK:XXXX a literal mask value.
# WARNING: current versions of Netscape crash on BMPStrings or UTF8Strings
# so use this option with caution!
string_mask = nombstr
# req_extensions = v3_req # The extensions to add to a certificate request
[ req_distinguished_name ]
countryName = Country Name (2 letter code)
countryName_default = ES
countryName_min = 2
countryName_max = 2
stateOrProvinceName = State or Province Name (full name)
stateOrProvinceName_default = Barcelona
localityName = Locality Name (eg, city)
localityName_default = Barcelona
0.organizationName = Organization Name (eg, company)
0.organizationName_default = UOC
organizationalUnitName = Organizational Unit Name (eg, section)
organizationalUnitName_default = PFC Seguretat
commonName = Common Name (eg, YOUR name)
commonName_max = 64
dnQualifier = D.N.I or N.S.S.
dnQualifier_default = 00000000-A
emailAddress = Email Address
emailAddress_max = 40
[ req_attributes ]
challengePassword = A challenge password
```

```
challengePassword_min = 4
challengePassword_max = 20
unstructuredName = An optional company name
[ usr_cert ]
# These extensions are added when ’ca’ signs a request.
# This goes against PKIX guidelines but some CAs do it and some software
# requires this to avoid interpreting an end user certificate as a CA.
basicConstraints=CA:FALSE
# Here are some examples of the usage of nsCertType. If it is omitted
# the certificate can be used for anything *except* object signing.
# This is OK for an SSL server.
# nsCertType = server
# For an object signing certificate this would be used.
# nsCertType = objsign
# For normal client use this is typical
nsCertType = client, email
# Copy extensions
# and for everything including object signing:
# nsCertType = client, email, objsign
# This is typical in keyUsage for a client certificate.
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
# This will be displayed in Netscape’s comment listbox.
nsComment = "PFC Seguretat"
# PKIX recommendations harmless if included in all certificates.
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer:always
# This stuff is for subjectAltName and issuerAltname.
# Import the email address.
subjectAltName=email:copy
# Copy subject details
issuerAltName=issuer:copy
#nsCaRevocationUrl = http://www.domain.dom/ca-crl.pem
#nsBaseUrl
#nsRevocationUrl
#nsRenewalUrl
```

ANNEX B. SCRIPT DE CREACIÓ DE LA BASE DE DADES

```
create database pfc_csm;
use pfc_csm;
drop table if exists IdentificacioConnexio;
drop table if exists PacientsMetge;
drop table if exists HistorialPacient;
drop table if exists Usuaris;

create table Usuaris(
  idUsuari  varchar(12) not null,
  certificat LONGBLOB not null,
  dadesAdministratives BLOB not null,
  sequencia LONGBLOB,
  primary key (idUsuari)
 ) ENGINE=InnoDB;

create table IdentificacioConnexio(
  idUsuari  varchar(12) not null,
  Ni          INTEGER not null,
  Ng     INTEGER not null,
  primary key (Ng),
  foreign key (idUsuari) references Usuaris(IdUsuari) ON
DELETE RESTRICT ON UPDATE NO ACTION
 ) ENGINE=InnoDB;

create table PacientsMetge(
  idUsuariM  varchar(12) not null,
  idUsuariP  varchar(12) not null,
  primary key (idUsuariM, idUsuariP),
  foreign key (idUsuariM) references Usuaris(IdUsuari) ON
DELETE RESTRICT ON UPDATE NO ACTION,
  foreign key (idUsuariP) references Usuaris(IdUsuari) ON
DELETE RESTRICT ON UPDATE NO ACTION
 ) ENGINE=InnoDB;

create table HistorialPacient(
  idUsuariP  varchar(12) not null,
  X  INTEGER not null,
  T  DATETIME not null,
  VisitaXif LONGBLOB not null,
  VisitaSig LONGBLOB not null,
  primary key (idUsuariP, X),
  foreign key (idUsuariP) references Usuaris(IdUsuari) ON
DELETE RESTRICT  ON UPDATE NO ACTION
 ) ENGINE=InnoDB;
```

ANNEX C: FITXERS DE CONFIGURACIÓ DE L'APLICACIÓ

- **FITXER DE CONFIGURACIÓ DEL CLIENT**

```
<Configuracio>

<RMI_host>localhost</RMI_host>
<RMI_port>1099</RMI_port>

<Path_Cert_Gestor>F:\UOC\PFC\PKI\Certificats\Gestor.p12</Path_Cert_Gestor
>
<Pw_Cert_Gestor>uoc0506</Pw_Cert_Gestor>

<Path_Cert_Usuari>F:\UOC\PFC\PKI\Certificats\Metge.p12</Path_Cert_Usuari>

</Configuracio>
```

- **FITXER DE CONFIGURACIÓ DEL GESTOR**

```
<Configuracio>

<RMI_host>localhost</RMI_host>
<RMI_port>1099</RMI_port>

<MySQL_host>localhost</MySQL_host>
<MySQL_port>3306</MySQL_port>
<MySQL_database>pfc_csm</MySQL_database>
<MySQL_user>casimir</MySQL_user>
<MySQL_password>casimir</MySQL_password>

<Path_Cert_Gestor>F:\UOC\PFC\PKI\Certificats\Gestor.p12</Path_Cert_Gestor
>
<Pw_Cert_Gestor>uoc0506</Pw_Cert_Gestor>

</Configuracio>
```

ANNEX D: INICI DE L'EXECUCIÓ DE L'APLICACIÓ

1.- Programari extern que cal tenir instal·lat:

- Llibreria criptogràfica XML-SECURITY 1.2.1 amb els arxius commons-logging.jar, commons-logging-api.jar, xalan.jar, xercesImpl.jar, xml-apis.jar, xmlsec-1.2.1.jar, xmlsecSamples-1.2.1.jar i xmlsecTests-1.2.1.jar, copiats el directori arrel de java/jre/lib/ext i afegits a la variable d'entorn CLASSPATH

- Política de seguretat JCE, amb els arxius local_policy.jar, US_export_policy.jar copiats el directori arrel de java/jre/lib/security i afegits a la variable d'entorn CLASSPATH.

- Gestor de base de dades MySQL.

- La màquina virtual de Java de Sun.

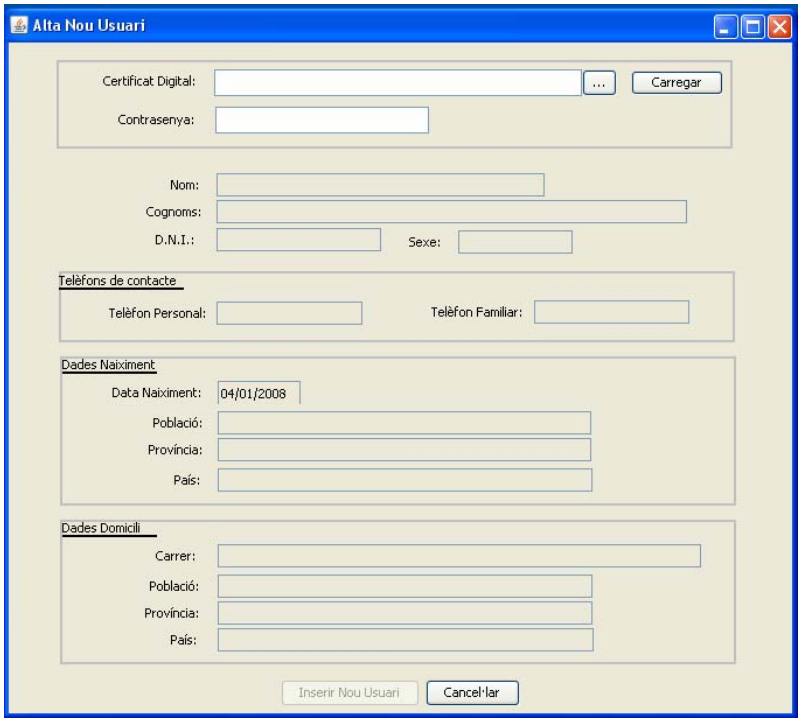
2.- Carregar la base de dades "ScripSQl_CreacioBBDD_i_Taules_PFC.sql" (AnnexB. Script de creació de la base de dades) en el sistema gestor de base de dades MySQL.

3.- Iniciar l'aplicació Gestor: "**java ve/Principal_Gestor**"

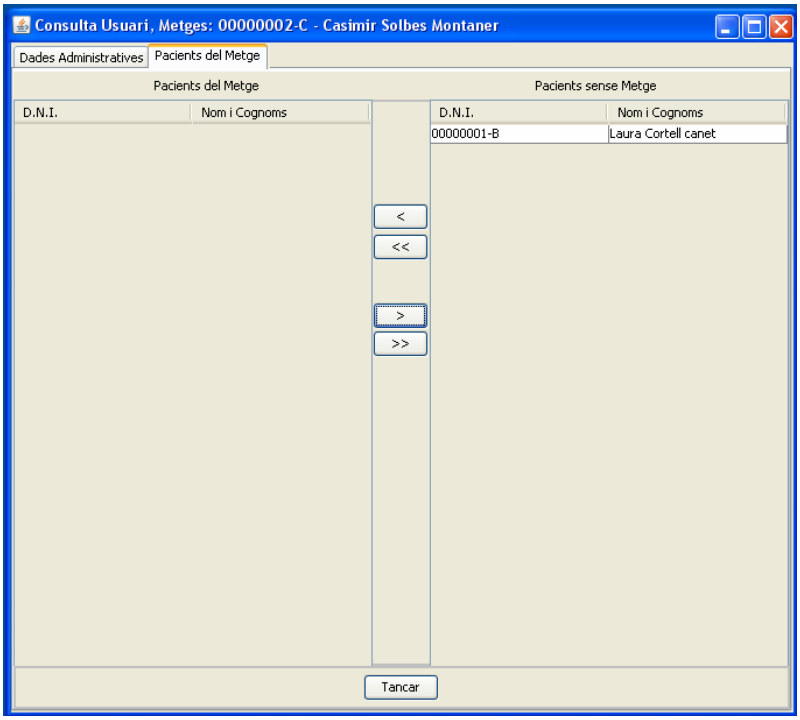
- Entrar en el Menú Editar/Configuració i configurar:

  - El servidor i el port RMI.
  - El servidor i el port de la base de dades MySQL, el nom de la base de dades "pfc_csm" i l'usuari i contrasenya de MySQL amb permisos per a la base de dades "pfc_csm".
  - El certificat del gestor amb la seva contrasenya.

- En el Menú Consulta/Llistat d'usuaris, podem donar d'alta els usuaris prement el botó "Alta Usuari". Introduir el certificat de l'usuari i la seva contrasenya i prémer el botó "Carregar" i acabar d'omplir les dades de l'usuari, finalment prem el botó "Inserir Nou Usuari":



Una vegada introduïts tots els usuaris, cal assignar als usuaris metges els seus pacients. Així en el llistat d'usuaris seleccionar l'usuari de tipus metge i prémer el botó "Consulta Usuari", el formulari de la consulta tindrà una segona pestanya "Pacients del Metge" allí caldrà seleccionar els pacients de la columna "Pacients sense Metge" i prémer el botó que indica que el pacient passa a la columna "Pacients del Metge":



4.- Engegar el registre RMI:

rmiregistry & (unix) start rmiregistry (windows)

5.- Instanciar i anunciar l'objecte Gestor:

Java rmi/GestorServer & (unix) start java rmi/GestorServer

6.- Iniciar l'aplicació Client: "**java ve/Principal**", bé amb el certificat d'un metge o d'un pacient.

- Entrar en el Menú Editar/Configuració i configurar:

  - El servidor i el port RMI.
  - El certificat del gestor amb la seva contrasenya.