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.

Estudiant: Pascual Sorando Pinilla.
Enginyeria en Informàtica

Consultor: Jordi Castellà Roca.

5 de Març de 2008

Resum.

En el PFC implementarem en Java una aplicació per gestionar historials mèdics de manera segura. Està desenvolupada seguint el model de capa de dades, capa de negoci i capa de presentació i el resultat són tres parts que anomenarem “part del gestor”, “part del pacient” i “part del metge”.
La capa de dades utilitza MySQL,
La columna vertebral de la aplicació és el paquet de classes criptogràfiques IAIK de la que es pot destacar X509Certificate, EncryptedData, SignedData i SignedAndEncryptedData.
La capa de presentació ha estat considerada unicamente en la seva faceta de entrada i sortida de dades i ha estat desenvolupada amb el paquet AWT.

Paraules clau.

Certificat.
Clau privada.
Clau pública.
Clau simétrica.
EnvelopedData.
EncryptedData.
SignedData.
Protecció de dades.
Dissociació de dades.

Nom de l’àrea de PFC.

PFC-Seguretat Informàtica.

Índex de continguts i índex de figures.

3.1.3. Gestió de la informació.
3.2. Protocols criptogràfics.

Capítol 4. XML.
4.1. Generalitats.
4.1.1. XML.
4.1.2. DTD.
4.1.3. DOM.
4.2. Estructura de dades de gestió d'historials mèdics.
4.2.1. DTD: objecteXML.dtd
4.2.2. XML: historial.xml, visita.xml

Capítol 5. RMI.
5.1. Concepte.
5.2. Generalitats.
5.3. Plantejament.

Capítol 6. Base de dades. (BD).
6.1. Generalitats de Bases de dades.
6.2. La estructura de les dades.
6.3. Taules – SQL.
6.4. Gestió d'informació.
6.5. SQL i Java.

Capítol 7. Vistes client.
7.1. Generalitats referents a les aplicacions gràfiques.
7.2. Pacient. Login
7.2.1. Consulta dades generals.
7.2.2. Consulta visita.
7.3. Metge. Login.
7.3.1. Consulta dades generals.
7.3.2. Historial, consulta i edició de dades mèdiques.
7.3.3. Visites.

Capítol 8. Vista gestor.
8.1. Login
8.2. Gestió de dades de Persona i Usuari.
8.3. Gestió de dades d'historial.
8.3.1. Dades generals.
8.3.2. Assignacions metge-pacient.

Capítol penúltim amb la valoració econòmica, si s'escau al tipus de projecte. (Sense desenvolupar).

Capítol 9 i últim amb les conclusions.

Glossari.

Bibliografia.

Annexos.
1. La llibreria IAIK.
2. El producte.
3. Els jocs de proves.

2. Índex de figures

(consultar el fitxer Indexdimatges.txt al directori bin/doc/imatges)

Capítol 1. Introducció.

En el PFC implementarem en Java un conjunt de protocols criptogràfics (esquema criptogràfic) bàsics per garantir els requisits de seguretat que cal assegurar en el cas de la gestió d'historials mèdics.

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

Les aplicacions informàtiques s'han d'ajustar al paràmetres de qualitat que exigeix la societat actual. En particular, es una obligació legal custodiar i protegir les dades de caràcter personal.
La Llei Orgànica de Protecció de Dades defineix les dades de caràcter personal com qualsevol informació relativa a persones físiques identificades o identificables.

Els principis de la protecció de dades són la qualitat i la adequació a les finalitats determinades i legítimes, el dret d'informació i el consentiment de l'afectat, la seguretat de les dades i el deure de secret professional.

Dades especialment protegides. Son dades que gaudeixen de la protecció Constitucional i que per ser tractades requereixen el consentiment exprés i per escrit de l'interessat. Amb tot, les dades necessàries per la prevenció i el tractament mèdics sí poden ser utilitzades, tot respectant el secret professional. També podran ser objecte de tractament les dades necessàries per salvaguardar l'interès vital de l'afectat.

L'interessat tindrà dret a sol·licitar i obtenir informació de les seves dades de caràcter personal sotmeses a tractament.

Procediment de dissociació de dades. És el tractament de dades personals de manera que la informació que s'obtingui no pugui ser associada amb una persona identificada o identificable.

Dades relatives a la salut. Les institucions i els centres sanitaris públics i privats i els professionals corresponents podran procedir al tractament de les dades de caràcter personal relatius a la salut de les persones

1.2. Objectius del PFC.

L'objectiu d'aquest PFC és 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.
Com a resultat del PFC s'obtindrà un sistema amb els components software següents:
Aplicatiu metge: (SistemaMetge)permet que un metge pugui consultar i afegir informació a l'historial d'un pacient de forma segura.
Aplicatiu pacient: (SistemaPacient) el pacient utilitza aquest aplicatiu per consultar les dades del seu historial.
Gestor central: (SistemaPacient) 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, assignar un pacient a un metge, donar accés a un metge a l'historial d'un pacient,etc.

1.3. Enfocament i mètode seguit.

1.3.1. Instal·lació de JAVA. Jdk-6u1-windows-i586-p.exe.
Definició de les considicions de seguretat: jce-policy-6.zip:

default_local.policy

// Country-specific policy file for countries with no limits on crypto strength.

```
grant {
   // There is no restriction to any algorithms.
   permission javax.crypto.CryptoAllPermission;
};
```

1.3.2. Instal·lació del Openssl : Win32OpenSSL-0_9_8e.exe. Openssl.cnf
1.3.3. Instal·lació de IAIK.: iaik-jec-full.jar (jdk1.6.0_01/jre/lib/ext/jdom.jar) i (jre1.0.6_01/lib/ext/jdom.jar)
1.3.4. Instal·lació de MySQL. (incluir com.mysql.jdbc.Driver en el path).

Instal·lació de la base de dades MySQL:

Fitxers font: mysql-4.1.12a-win32.zip i mysql-connector-java-3.1.8a.zip
Queda instal·lat MySQL Server 4.1

MySQL Server 4.1
Create a new free MySQL.com account
Skip Sign-Up
¿Config file?
¿service?¿dedicated port?¿set the password?¿root account?
Developer Machine
Multifunctional Database InnoDB
directory \MySQL\
FileSystem NTFS
(DSS)/OLAP
Enable TCP/IP Networking
Port Number 3306
Charset Latin 1
Service Name MySQL
Launch MySQL Server automatically
New root password: laucsaps
Enable root access from remote machine
configuration file my.ini

MySQL Administrator:

new connection
localhost
root
pswd: laucsaps

crear un nou usuari:

usuario
pswd: oirausu

Es necesari que l'usuari "root" pugui crear la Base de Dades "historial". (Amb MySQL Query Browser)

CREATE DATABASE IF NOT EXISTS historial

1.3.5. Instal·lació de org.jdom. jdom.jar

1.4. Planificació del projecte.

El projecte es desenvoluparà als mesos de febrer, març, abril, maig i juny de 2008.
Hi està previst fer les següents feines seguint un model de cicle de vida en espiral:

Instal·lar IAIK PKI	 openssl					setmana 1	PAC1
Implementació protocols	objectiu criptografia	disseny	Implementació	test	documentació	 setmana 2 setmana 5	PAC 2
Implementació documents	objectiu XML	disseny	Implementació	test	documentació	setmana 6 setmana 7	PAC 3
Comunicacions Servidor RMI Client base RMI	objectiu RMI	disseny	Implementació	test	documentació	 setmana 8 setmana 9	PAC 4
BD servidor Registre usuaris	objectiu model	disseny	Implementació	test	documentació	setmana 10 setmana 11	PAC 5
Vista client	objectiu aplicatiu client	disseny	Implementació	test	documentació	setmana 12 setmana 13	PAC 6
Vista gestor	objectiu aplicatiu gestor	disseny	Implementació	test	documentació	setmana 14	PAC 7
Documentacio	objectiu conclusions			test	documentació	Setmana 15 Setmana 16	PAC 8

Figura 1. Llista de feines.

Planificació del PFC

Setmana	dilluns	dimarts	dimecres	dijous	divendres	dissabte	diumenge	
1				28	29	1	2	febrer
2	3	4	5	6	7	8	9	març
3	10	11	3	4	5	6	7	
4	17	18	19	20	21	22	23	
5	24	25	26	27	28	29	30	
6	31	1	2	3	4	5	6	abril
7	7	8	9	10	11	12	13	
8	14	15	16	17	18	19	20	
9	21	22	23	24	25	26	27	
10	28	29	30	1	2	3	4	maig
11	5	6	7	8	9	10	11	
12	12	13	14	15	16	17	18	
13	19	20	21	22	23	24	25	
14	26	27	28	29	30	31	1	juny
15	2	3	4	5	6	7	8	
16	9	10	11					

Figura 2. Calendari.

1.5. Productes obtinguts (esmentar-los i explicar-los breument; els productes en si s'explicaran extensament als altres capítols de la memòria i/o seran altres productes lliurats junt amb la memòria).

1. Base de dades:
1.1. GestorBD.java: conté totes les funcionalitats relacionades amb gestió de bases de dades i llenguatge SQL.

2. Capa de negoci.
2.1. Generals.
2.1.1. SistemaGestor.java: conté el métode main. És el començament de la part del gestor. Fa la identificació criptogràfica del gestor.
2.1.2. Gestor.java: incloú totes les funcionalitats de la aplicació corresponents al gestor.
2.1.3. SistemaMetge.java: conté el métode main. És el començament de la part del metge. També fa autenticació.
2.1.4. Metge.java: incloú totes les funcionalitats de la aplicació corresponents al gestor. Inclou la comunicació remota (RMI) amb la classe gestor.
2.1.5. SistemaPacient.java: conté el métode main. És el començament de la part del Pacient. Fa autenticació d'usuaris.
2.1.6. Pacient.java: incloú les funcionalitats de la aplicació corresponents al pacient.
2.2. Criptografia.
2.2.1. CipherManager.java : permet treballar amb objectes iaik.pkcs.pkcs7.EnvelopedData.
2.2.2. SignerCipherManager.java: permet treballar amb objectes pkcs7_signedAndEnvelopedData
2.2.3. SignerManager.java : permet fer signatures de text en clar. (dades en clar no incloses).
2.2.4. P12.java: permet tenir accés a un fitxer pkcs#12.
2.2.5. CertificateFile.java : permet obtenir un X509Certificate a partir d'un fitxer .crt. Cal aportar el nom complet del fitxer. Per exemple: "./doc/pki/Gestor.crt"
2.2.6. CertificateValid.java: permet fer la validació de les propietats d'un X509Certificate.

3. Capa de presentació
3.1. AWTInici.java : és la finestra on es fa la autenticació del gestor, el metge i el pacient. Cal introduïr el nom del certificat corresponent. En el cas del Gestor cal introduïr "Gestor", per al metge: "Metge" i per al pacient "Pacient".
3.2. Pantalla.java : permet la sortida de missatges cap a l'usuari.
3.3. AWTGestor.java: es una finestra amb un menú amb opcions per fer altes, modificacions i consultes.
3.4. AWTMetge.java : és una finestra amb un menú.
3.5. AWTPacient.java : és una finestra amb un menú.

Documentació del producte.
Bin/doc/javadoc/: Documentació de les classes java fent us de javadoc per generar-la.

Presentació:
Fitxet psorando_presentacio.ppt (generada amb PowerPoint).

1.6. Breu descripció dels altres capítols de la memòria.

1. PKI
En aquest capítol es fa una presentació dels fonaments de la criptografia moderna basada en funcions matemàtiques. Se presenten les estructures de dades i la codificació DER que permeten la implementació de funcions criptogràfiques de encriptat, desencriptat, signatura i resum de dades, fent us de claus simétriques (SecretKey) i privades i públiques (PrivateKey i PublicKey)

2. Esquema criptogràfic.
És la part del treball on es consideren tots els problemes de la seguretat de les dades. ¿ón es produeixen?, ¿qui les pot veure? ¿com s'han de conservar? ¿cal signar-les?.
En general cal afegir funcions de confidencialitat, integritat i no repudi a totes les funcions que es puguin implementar. També cal complir el principi de la dissociació de dades.
Les decisions preses en aquest capitol han estat matizades per les imposicions del marc de treball que soposava la utilització de les classes del paquet IAIK.
De fet, adonar-me massa tard de les funcions per encriptar i desencriptar de les classes EnvelopedData, EncryptedData, SignedData i SignedAndEnvelopedData ha supossat no poder obtenir un millor resultat.

3. XML.
El llenguatge de marques aporta estructura i claredat de conceptes a qualsevol ambit de la vida cotidiana. En aquesta memoria també té el seu espai encara que, malauradament, no s'ha pogut aplicar a la pràctica per culpa d'un mal enfocament de les feines de programació relacionades amb RMI. (feia una mica de respecte com podia respondre RMI amb objectes complexes com son les estructures XML).

4. RMI.
RMI es un dels métodes que dispossa Java per implementar aplicacions distribuides. En el nostre cas les funcionalitats remotes s'han definit dintre de les clases principals "Gestor.java", "Metge.java" i "Pacient.java", encara que resta l'impressió que hi ha altres possibilitats mes apropiades.

5. Base de Dades.
Aquest capitol ha estat resolt amb MySQL.
Es fonamental advertir que per començar cal generar la Base de Dades "historial" des de fora de la aplicació i crear els usuaris i els password des de fora de la aplicació.
Totes les funcions relatives al nivel de dades i al SQL s'han implementat dintre de la classe GestorBD.java.
Comentar les complicacions que s'han trobat a l'hora d'utilitzar el tipus sql.BLOB, ampliats amb la incompatibilitat de Java i Latin-1. (Java utilitza Unicode i MySQL ha estat configurada en el meu cas per treballar amb Latin-1).

6. Capa de presentació.
De les classes generades per resolver la presentació només s'ha tingut en compte que calia generar estructures amb variables per informació d'entrada, variables per informació de sortida, objectes AWT de configuració de finestres i finalment variables auxiliares per control de flux.
Es a dir, qualsevol funcionalitat d'aquestes classes ha estat traslladada a les classes que feien la crida.

Capítol 2. PKI.

1. Infraestructura de clau pública.

1.1 Fonaments de criptografia.

La comunicació és pot representar com un flux de senyals codificades des d'un emissor fins a un receptor amb l'ajut d'un mitjà de comunicació que anomenem canal.

Aquest procés basic es complica a la pràctica i apareixen molts punts febles que son aprofitats per terceres persones, alienes a l'emissor i el receptor, per interferir, amb més o menys gravetat, la comunicació.

Els objectius d'una infraestructura criptogràfica són resoldre els problemes de la comunicació que poden afectar a la informació (interceptació de senyals, canvi de missatges en ruta, falsificació, destrucció d'informació, intrusions, etc.) i a les persones que intervenen en la comunicació (usurpació d'identitat, problema de l'home al mig, rebuig de missatges, acords).

Totes les tècniques orientades en aquesta direcció formen la criptografia, es a dir, la ciència que estudia els mètodes per protegir la informació.
Des del punt de vista de la seguretat s'ha de demanar que la informació sigui confidencial, autèntica i complerta i que tant l'emissor com el receptor es puguin identificar.

La criptografia moderna es basa en l'aplicació pràctica de conceptes matemàtics. S'utilitzen algorismes amb funcions fàcils de calcular, però amb una funció inversa que sigui pràcticament impossible de calcular.
Aquest algorisme es pot expressar com:

C = f(M);
M = f inversa(C);

on M és el text en clar i C és el text xifrat.

Normalment l'algorisme és una funció amb un paràmetre anomenat clau:

C = e(k,M);
M = d(x,C) = d(x, e(k,M));

on "e" és la funció de xifrat, "d" és la funció de desxifrat, k i x son la clau de les funcions.

Aquest sistema permet complir la suposició de Kerckhoff: tot el mecanisme de xifratge, excepte el valor de la clau secreta, és conegut pel criptoanalista enemic.

1.2. Clau compartida o simètrica.

Els sistemes criptogràfics de clau simètrica es caracteritzen perquè la clau de desxifratge "x" és idèntica a la clau de xifratge "k".

C = e(k,M);
M = d(k,C) = d(k, e(k,M));

La seguretat d'aquest sistema rau en mantenir en secret la clau k. Les implementacions d'aquest sistema són senzilles i ràpides.

DES (Data Encryption Standard) ha estat l'algorisme més estudiat i alhora més utilitzat. Actualment és vulnerable als atacs de "força bruta" i ha estat substituït per el Triple DES que utilitza claus de 112 bits o 168 bits.

En la xifra de bloc, si dos blocs de text en clar són iguals i es fa servir la mateixa clau, els blocs xifrats també seran iguals. Aquest problema ha estat corregit amb l'ús dels modes d'operació: ECB (Electronic Codebook), CBC (Cipher Block Chaining) i CFB (Cipher Feedback).

El punt feble de la clau compartida o simètrica és l'intercanvi de claus. El canal és, per definició, insegur i qualsevol podria escoltar la informació que s'hi transmet. La transformació del codi o

de l'estructura de les dades amb algorismes de clau compartida garanteix la confidencialitat de la comunicació, però no garanteix la integritat de les dades ni la identitat del receptor.

1.3. Clau pública.

El sistema de clau pública és la solució al problema del canal insegur.

Els algorismes de clau pública o asimètrica es basen en problemes matemàtics fàcils de plantejar però difícils de resoldre, i compleixen la suposició de Kerckhoffs.

Permeten xifrar amb la clau pública (del receptor, k publica) i desxifrar amb la clau privada (també del receptor, k privada).

C = e(k publica, M);
M = d(k privada, C);

d'aquesta manera es pot garantir el secret davant de tercers interposats a la comunicació.

Els algorismes que permeten xifrar amb la clau privada (de l'emissor) i desxifrar amb la clau pública (de l'emissor) permeten garantir la identitat de l'origen de les dades. Es pot representar per:

C = e(k privada, M);
M = d(k privada, C);

RSA és l'algoritme de clau pública més utilitzat. La clau pública està formada per un número n, calculat com a producte de dos factors primers molt grans (n = p*q), i un exponent e.
La clau privada és un altre exponent d calculat a partir de p, q, i e.

1.4. Funció Hash.

Les funcions "hash" ens permeten transformar un missatge de longitud arbitraria en una cadena de bits de longitud fixa.

H = h(M);

M és el missatge d'entrada i H és el "resum" hash.

Les funcions hash unidireccionals i resistents a col·lisions (hash segures) es poden utilitzar per donar un servei d'integritat del missatge.

L'algorisme SHA-1 dóna resums de 160 bits. Hi ha versions que generen resums de 256, 384 i 512 bits.

1.5. Infraestructura pública criptogràfica.

L'objectiu dels sistemes criptogràfics es arribar a complir el principi de l'anomenada xifra de Vernam: si "k" és una clau seqüencial totalment aleatòria que no es repeteix cíclicament, estem davant d'un exemple de xifratge incondicionalment segur.

La tècnica del sobre digital combina els avantatges de la clau pública i de la clau compartida. El missatge s'envia xifrat amb un criptosistema de clau compartida, sota una clau aleatòria, i tot seguit s'envia la clau aleatòria xifrada amb la clau pública del destinatari. En aquest cas, la clau compartida només s'utilitza una vegada.

La tècnica de la signatura digital permet garantir el reconeixement de l'emissor del missatge, es a dir, permet donar servei de "no repudi".

Amb les tècniques esmentades més amunt es pot donar solució als problemes de canal insegur que presenta la comunicació a nivell de xarxa. Però, encara queden els problemes de la comunicació a nivell d'actors i el problema de la intrusió als sistemes informàtics.

Fa falta algun sistema que ens permeti tenir la convicció que l'emissor és qui diu que és o que

el missatge ha estat rebut pel destinatari. Aquest problema de confiança ha estat resolt amb la infraestructura de clau pública (PKI).

La infraestructura de clau pública PKI, està pensada per sistemes insegurs que poden ser atacats per intrusos amb l'intenció de robar les claus privades dels usuaris, i la seua base es el compromís i la confiança mútua dels usuaris organitzats al voltant de les CA.

Les autoritats de certificació (CA) ens asseguren que coneixen el receptor i que les claus públiques pertanyen als seus suposats propietaris. Aquestes organitzacions signen digitalment els certificats de clau pública i en donen publicitat.

Les CA estan organitzades en jerarquies de certificació que formen cadenes de certificats (o de confiança) i que s'actualitzen amb les llistes de revocació de certificats (CRL).

Un certificat de clau pública es un document amb la identificació d'usuari, el valor de la clau pública de l'usuari i la signatura (amb la clau privada de la CA) de les dues parts anteriors. El format dels certificats està especificat a la definició del servei de directori X.500.

PKCS (Public Key Cryptography Standards) són un conjunt d'estàndards per la criptografia de clau pública, desenvolupats pels Laboratoris RSA conjuntament amb un consorci de grans empreses i centres d'investigació. Gràcies a la sintaxi especificada als PKCS podem intercanviar-nos dades en un format estàndard, aconseguint una gran interoperabilitat.

A continuació es descriuen els PKCS existents actualment:

1. PKCS #1 defineix els mecanismes per xifrar i signar dades emprant el criptosistema de clau pública RSA.
2. PKCS #3 defineix el protocol d'intercanvi de claus de Diffie-Hellman.
3. PKCS #5 descriu un mètode per xifrar un cadena de dades amb una clau privada obtinguda a partir d'una contrasenya.
4. PKCS #6 descriu algunes extensions per un certificat digital, però actualment no s'utilitza per afavorir l'ús de la versió 3 de l'estàndard X.509.
5. PKCS #7 defineix una sintaxi general pels missatges que contenen afegits criptogràfics com pot ser una signatura digital o dades xifrades. Trobem **pkcs7-data**, **pkcs7-digestedData**, **pkcs7-encryptedData**, **pkc7-envelopedData**, **pkcs7-signedData** i **pkcs7-signedAndEnvelopedData**.
6. PKCS #8 descriu un format per guardar la informació de la clau privada. Aquesta informació inclou la clau privada per un cert algorisme de criptografia asimètrica, i opcionalment un conjunt d'atributs.
7. PKCS #9 defineix alguns tipus d'atributs que s'empren en altres PKCS, com per exemple al PKCS#6, als missatges signats al PKCS#7, a la informació privada del PKCS#8, i a la petició de certificat del PKCS#10.
8. PKCS #10 defineix la sintaxi per la petició d'un certificat. L'estructura d'una petició de certificat és paral·lela a la del certificat que es vol obtenir; la diferència fonamental és que la petició de certificat és una estructura de dades auto-signada. D'aquesta manera l'Autoritat de Certificació pot provar que l'usuari/entitat que fa la petició està en possessió del parell de claus asimètriques.
9. PKCS #11 defineix una interfície de programació independent de la tecnologia, anomenada Cryptoki, per dispositius criptogràfics com una tarja intel·ligent i targes PCMCIA.
10. PKCS #12 especifica un format intercanviable per guardar o transportar la clau privada d'un usuari, certificats i diversa informació secreta, etc.
11. PKCS #13 intenta definir un mecanisme per xifrar i signar dades emprant Criptografia de Corbes El·líptiques
12. PKCS #14 actualment esta desenvolupant-se i cobreix la generació de nombres pseudoaletòris.
13. PKCS #15 és un complement al PKCS #11 aportant un estàndard pel format de les credencials criptogràfiques que es guarden en un dispositiu segur contra manipulacions.

Els PKCS compleixen la sintaxi ASN.1 (Abstract Syntax Notation One) de l'ISO i es codifiquen seguint les regles BER/DER de transferència de dades. Aquesta codificació permet garantir que les transformacions per codificació i descodificació de les dades són 1:1.

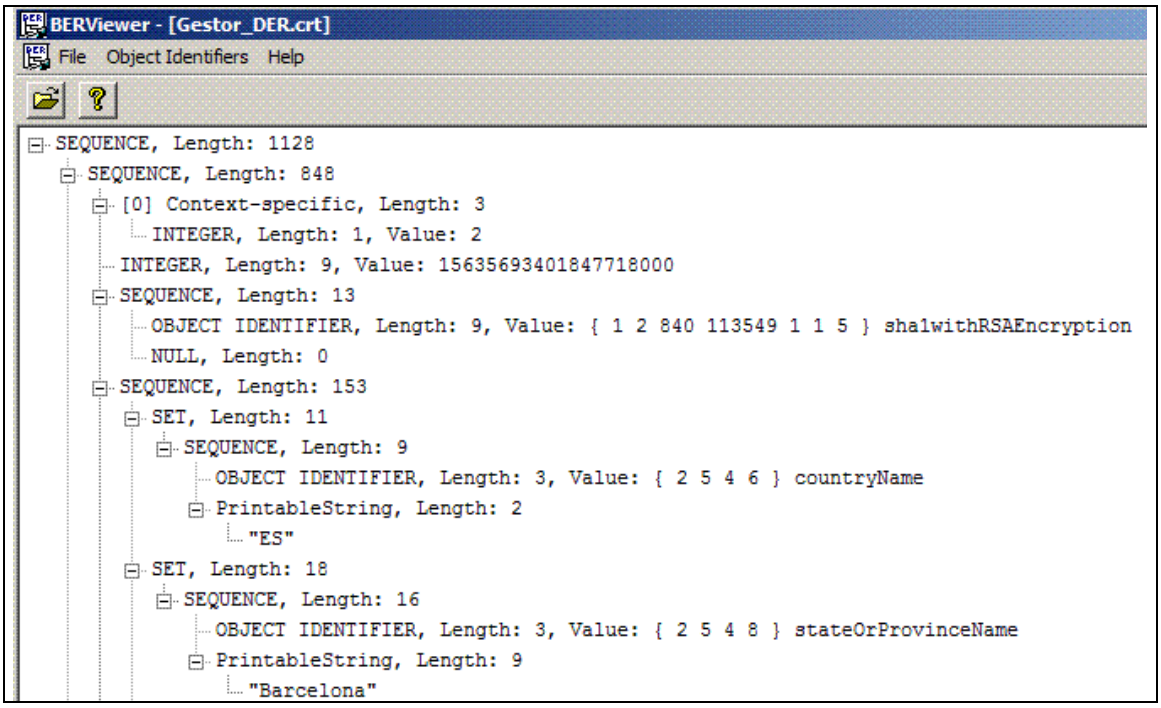


Figura 3. Estructura ASN.1

2.- PKI en Java (API).

Les classes de seguretat de java les podem trobar als packages:

- java.security
- java.security.acl
- java.security.cert
- java.security.interfaces
- java.security.spec
- javax.crypto
- javax.crypto.interfaces
- javax.crypto.spec

i en JCE.

Amb aquestes classes es poden fer les següents funcions:

- Crear un parell de claus
- Guardar la clau privada xifrada
- Crear una petició de certificat
- Crear un certificat arrel
- Crear un certificat d'usuari
- Crear un P12
- Signar dades.
- Encryptar i desencryptar dades.
- Resumir dades per garantir la integritat.

3. Els certificats. Openssl.

Per crear els certificats que es faran anar en la pràctica es pot recórrer a tres mètodes diferents.
El més senzill es utilitzar openssl per generar-los des de fora de la aplicació i instal·lar-los al directori corresponent, a on es puguin trobar sense cap problema.
La segona opció és utilitzar les classes Runtime, (<static> Runtime.getRuntime()), i Process.
La tercera opció es utilitzar la criptografia de iaik.

El procés per a generar tots els certificats necessaris per al projecte es descriu a continuació. Utilitzarem "uoc0506" com a contrasenya en tots els casos.

1.- Autoritat Certificadora:
1.1. Generar una parella de claus de 2048 bits: openssl genrsa – des3 –out CA.key 2048
1.2. Generar un Certificat autosignat: openssl req –new –sha1 –x509 –key CA.key –out CA.crt –days 360
1.3. Generar un PKCS#12

2.- Pacient:
2.1. Generar una parella de claus de 1024 bits: openssl genrsa -des3 -out Pacient.key 1024
2.2. Generar una petició de certificat PKCS#10: - openssl req -new -sha1 -key Pacient.key -out Pacient.csr -config openssl.cnf
2.3. Obtenir un Certificat signat per CA: openssl x509 -req -in Pacient.csr -days 180 -CA CA.crt -CAkey CA.key -CAcreateserial -extfile openssl.cnf –extensions usr_cert -out Pacient.crt
2.4. Generar un PKCS#12: openssl pkcs12 –in Pacient.crt –inkey Pacient.key –name Pacient –chain –CAfile CA.crt –export –out Pacient.p12

3.- Metge:
3.1. Generar una parella de claus de 1024 bits: openssl genrsa -des3 -out Metge.key 1024
3.2. Generar una petició de certificat PKCS#10: - openssl req -new -sha1 -key Metge.key -out Metge.csr -config openssl.cnf
3.3. Obtenir un Certificat signat per CA.: openssl x509 -req -in Metge.csr -days 180 -CA CA.crt -CAkey CA.key -CAcreateserial -extfile openssl.cnf –extensions usr_cert -out Metge.crt
3.4. Generar un PKCS#12: - openssl pkcs12 -in Metge.crt -inkey Metge.key –name Metge -chain -CAfile CA.crt -export -out Metge.p12

4.- Gestor:
4.1. Generar una parella de claus de 1024 bits: openssl genrsa -des3 -out Gestor.key 1024
4.2. Generar una petició de certificat PKCS#10: openssl req -new -sha1 -key Gestor.key -out Gestor.csr -config openssl.cnf
4.3. Obtenir un Certificat signat per CA.: openssl x509 -req -in Gestor.csr -days 180 -CA CA.crt -CAkey CA.key -CAcreateserial -extfile openssl.conf –extensions usr_cert -out Gestor.crt
4.4. Generar un PKCS#12: - openssl pkcs12 -in Gestor.crt -inkey Gestor.key –name Gestor -chain -CAfile CA.crt -export -out Gestor.p12

Tots aquest fitxers obtinguts, s'han situat al directori Linux : bin/doc/pki (.\bin\doc\pki per Windows). (en java cal escriure els caracters d'escapament del String: ".\\bin\\doc\\pki\\")

Capítol 3. Esquema criptogràfic.

3.1. Estudi de les necessitats del sistema.

3.1.1. Actors
En una primera aproximació al problema trobem un actor bàsic en totes les aplicacions al que anomenarem “usuari”. Aquest usuari el definim per contraposició a les persones que no son identificats com a usuaris del sistema. Hi ha diverses maneres de fer la identificació, però nosaltres implementarem la identificació amb “certificats”.

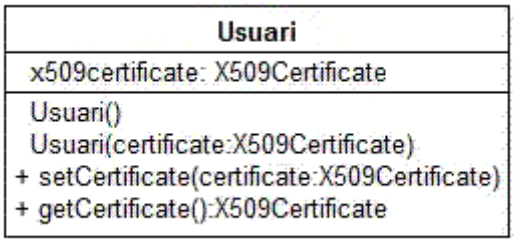


Figura 4. Esquema de la classe Usuari.

El sistema serà utilitzat bàsicament per personal sanitari dintre de les seves competències i també pels pacients per tal de consultar les seves pròpies dades.
La sintaxi dels certificats X509 permet fer classificacions de personal sanitari en funció de les seves funcions, però en el nostre estudi no desenvoluparem aquestes possibilitats. Les persones que no estiguin subjectes al secret professional no tindran accés. Farem un esquema senzill que pot ser ampliat més endavant.

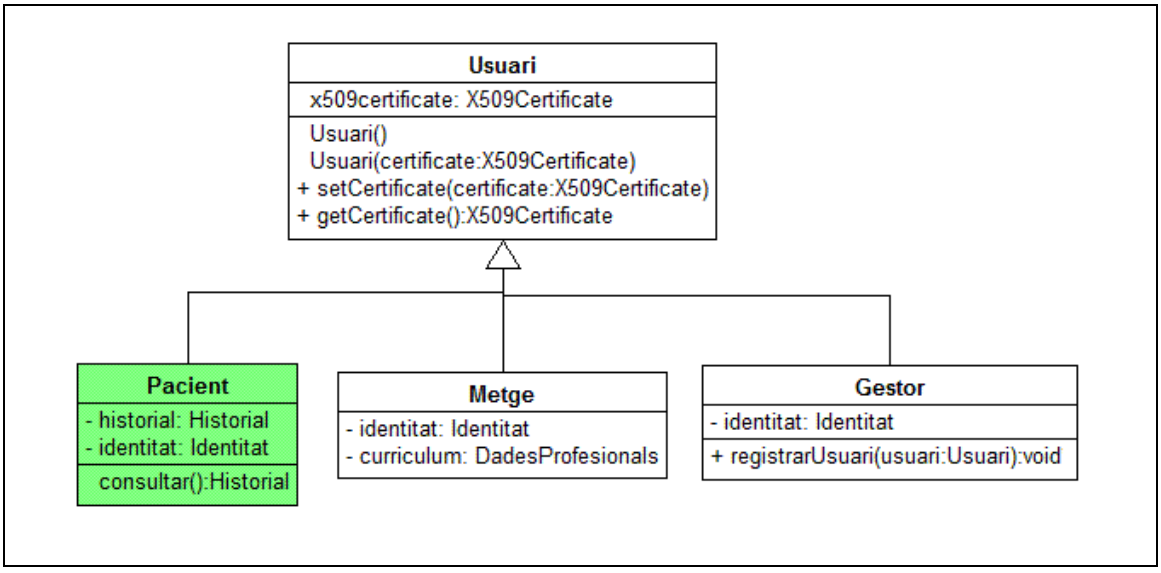


Figura 5. Herencia entre las classes d’usuari.

Utilitzarem el camp “dnQualifier” del certificat per guardar la informació “Id_usuari” que identificarà als usuaris.
Utilitzarem el camp “O U Name” del certificat per classificar als usuaris com pacients, metges o gestors (“Pacients”, “Metges”, “Administradors”);

3.1.2. Informació. Definicions.
L’historial mèdic d’un pacient pot contenir informació molt diversa, i alguna informació pot ser considerada més confidencial que una altra. Per exemple, el grup sanguini o les al·lèrgies del pacient haurien de poder ser consultades per qualsevol metge. Si el pacient ha tingut un

accident i cal fer-li una transfusió aquesta dada és vital. No obstant, si el pacient rep ajuda psicològica hauria de ser confidencial, i només ho hauria de poder consultar el seu metge.

Totes les dades relatives a la salut de les persones gaudeixen de protecció constitucional i es consideren dades especialment protegides. Requereixen el consentiment de l'interessat per al seu tractament, encara que les dades que resulten necessàries per a la prevenció o el tractament mèdics o la gestió de serveis sanitaris el tractament de dades està autoritzat en el cas que sigui fet per personal professional sanitari subject al secret professional. També poden ser objecte de tractament les dades necessàries per salvaguardar l'interés vital de l'afectat.

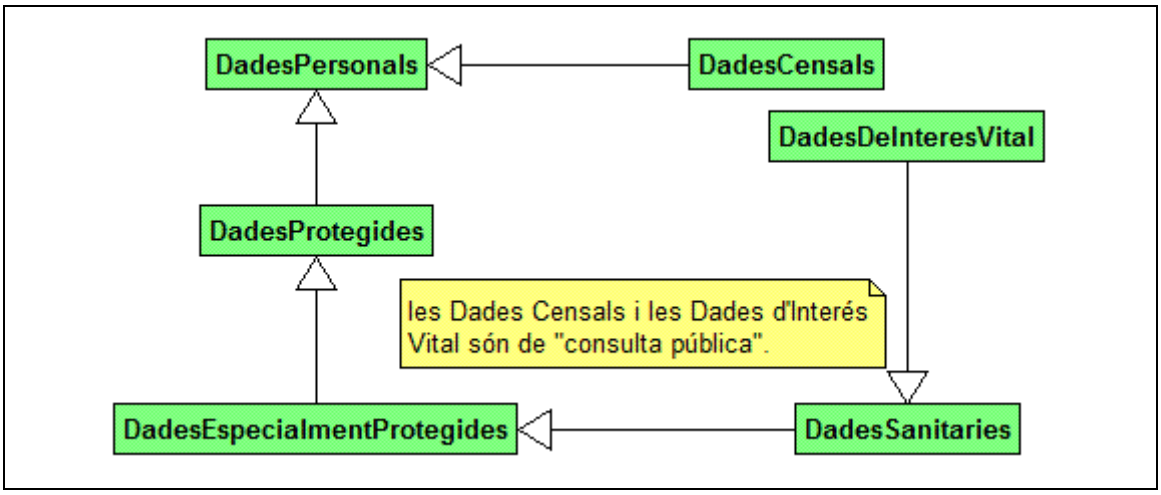


Figura 6. Classificació de les dades.

L'historial conté les dades generals del pacient. Són les dades censals o d'identitat i les dades de interès vital següents:

- Nom;
- Cognoms;
- Número de targeta sanitària;
- Número de DNI;
- Grup sanguini;
- Al·lèrgies;
- Certificat digital X509 v3.

Una visita consta de l'anamnesi, la diagnosi del metge i el tractament que ha de rebre el pacient.

3.1.3. Gestió de la informació.

L'objectiu del sistema es garantir les quatre propietats de seguretat següents:
Confidencialitat: s'ha de preservar la confidencialitat o privadesa de les dades de l'historial mèdic dels pacients. Amb pkcs7-envelopedData, pkcs7-encryptedData,
Autenticitat: la informació que es guarda en el sistema ha de disposar d'una prova de la seva autenticitat. Amb el sistema de claus asimetriques, PublicKey, PrivateKey.
Integritat: un cop la informació ha estat generada s'ha de garantir en tot moment la seva integritat. Amb pkcs7-digestedData.
No-repudi: si un usuari del sistema fa una certa acció més tard no pot negar que l'hagi realitzada. Amb pkcs7-signedData.
En tot cas, cal garantir que les dades no patiran alteració, pèrdua, tractament o accés no autoritzat. Per tal de garantir la privadesa de les persones es dissenyarà un procediment de dissociació de dades de manera que la informació que s'obtingui fora del procediment establert, no es pugui associar a cap persona identificada o identificable.

Les visites es guarden a la Base de Dades (BD) sense cap informació que les pugui vincular a un pacient. Si algú accedís a la BD no podria saber quina visita correspon amb un pacient en concret.

Els sistemes de protecció són els següents:
- Llista de metges protegida que conserva el Gestor. Es un EnvelopedData que forma part d'Historial.
- Llista de descriptors de visites per Historial, protegida.
- Descriptor de visita, protegit. Es un EnvelopedData.

A l'historial definit en el punt anterior l'afegirem la llista protegida de metges que tenen dret a consultar-lo i la llista protegida de les visites relacionades.

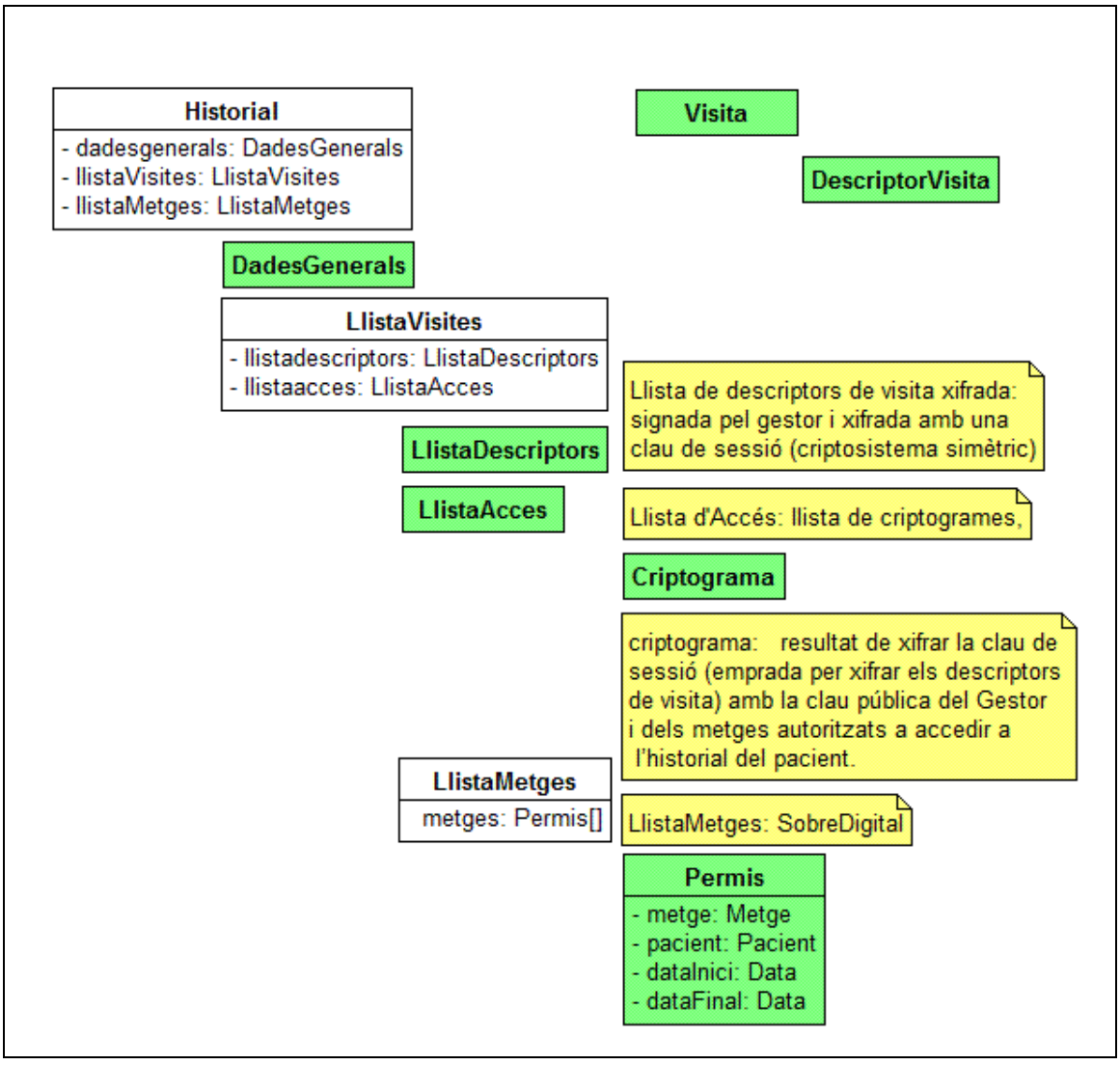


Figura 7. Estructura de l'historial.

La llista de visites es pot implementar amb un SignedAndEncryptedData, amb un X509Certificate[] _chain que contindrà els certificats de tots els usuaris autoritzats per fer la consulta. (en el nostre cas: el gestor, el pacient, i un o més metges autoritzats).

A la visita definida en el punt anterior l'hi afegirem un "descriptor" de visita i una signatura digital. El descriptor de visita conté la informació bàsica d'una visita, i està identificat de forma única per un identificador de visita.

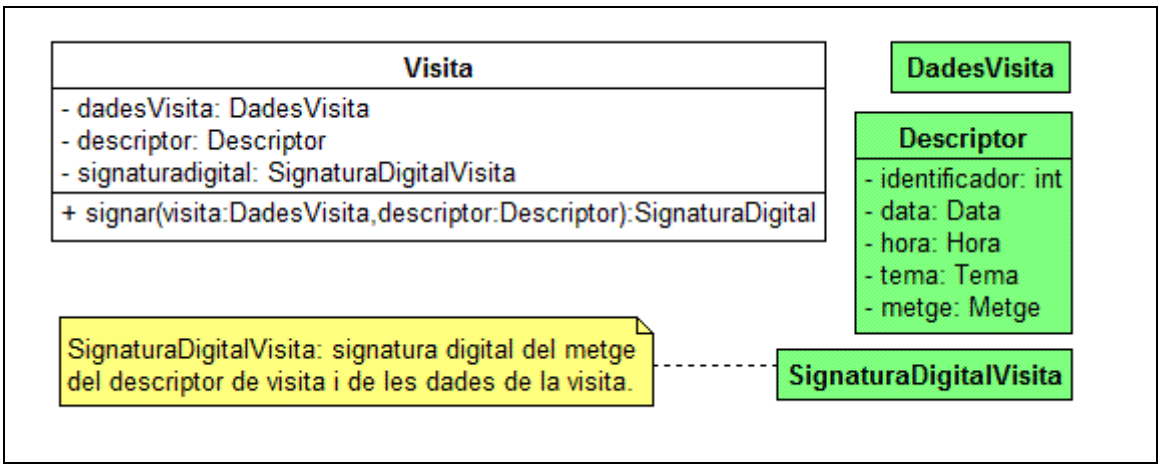


Figura 8. Dades d'una visita.

Els identificadors d'Historials i de les Visites estan definits de manera que a partir d'un Historial es pugui arribar a consultar les Visites, mentre que a partir de l'identificador d'una Visita sigui impossible arribar a obtenir l'Historial corresponent. D'aquesta manera podem garantir que existeix dissociació entre les dades mèdiques i les dades d'identificació del pacient i que es pot garantir la privacitat de la informació del sistema.

3.1.4. Flux d'informació

Les accions previstes amb la informació són: creació, encriptació, conservació en BD, desencriptació, consulta, destrucció / exportació. La modificació només es farà amb informació general.

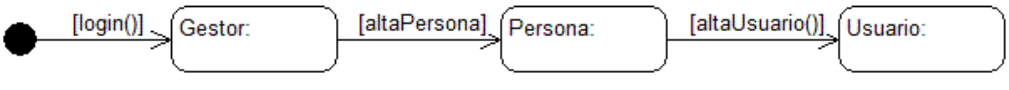


Figura 9. Diagrama d'estats (1),

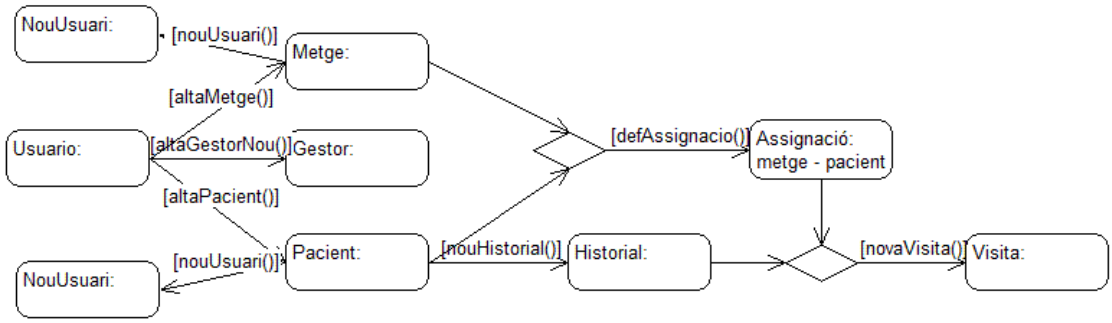


Figura 10. Diagrama d'estats (i 2).

3.1.4.1. Alta d'usuaris i assignació de metge.
El Gestor és l'encarregat de registrar els usuaris (metges i pacients) al sistema. Després de registrar un metge li afegirà una llista amb els pacients assignats. De la mateixa manera, després de registrar un pacient, li afegirà una llista amb els metges que tenen dret a afegir informació al seu historial.

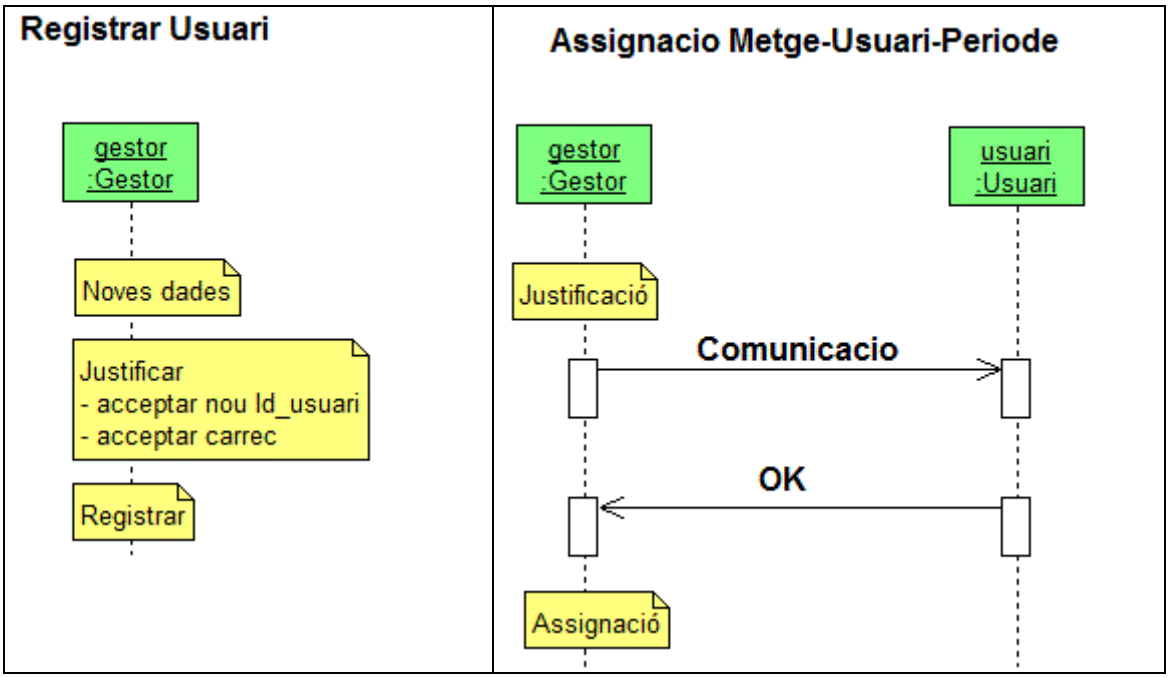


Figura 11. Registre i assignació d'usuaris.

El registre d'usuaris es pot descompondre en les següents subtasques:
- Recollida de dades d'identitat. Omplir el camp Id_usuari amb el valor del DNI o NSS.
- Definició del tipus d'usuari: Metge / Pacient. (Resultat: un Usuari sense certificat.)
- (Només Pacients) Crear Pacient. Introduir dades d'identitat, (El Sistema Pacient generarà un parell de claus i una petició de certificat) Crear Certificat amb el camp Org. Unit Name = "Pacients" , i camp dnQualifier = "Id_usuari". Crear l'historial.
- (Només Metges) Crear Metge. Recollir dades professionals: especialitat, núm. Col·legiat, (El SistemaMetge generarà un parell de claus i una petició de certificat) Crear Certificat amb el camp Org. Unit Name = "Metges". Afegir el nou metge a la Llista Xifrada<$_{Gestor}$> General De Metges.

El contingut del camp Id_usuari, (Id_pacient, Id_metge) és crítica. És la que es farà servir per a la identificació i classificació dels usuaris. Aquesta informació és redundant amb el contingut del Certificat de l'usuari.

L'assignació de metge a un pacient inclou les següents subtasques:
- actualitzar la Llista Xifrada<$_{Gestor}$> Metges que té un Pacient.
- Actualitzar la Llista Xifrada<$_{Metge}$> Pacients que té un Metge

(Aquesta funcionalitat queda amagada dins un SignedAndEncryptedData)

3.1.4.2. Afegir Visita
Els metges que figuren a la llista de metges d'un pacient són els encarregats de generar els documents que documenten les visites. Aquestes visites no es podran modificar posteriorment. Les correccions s'afegiran en visites posteriors. Per tant, es pot dir que la mecànica de l'historial del pacient és la producció d'un "diari".

La feina de Afegir Visita a l'Historial d'un pacient es pot descompondre en les següents sub-feines:

1.- El metge formarà el “Descriptor”.
2.- El metge afegirà les dades tècniques (anamnesi, diagnosi, tractament).
3.- El metge és l’encarregat de signar la visita.
4.- El gestor és l’encarregat de conservar-la i actualitzar l’Historial.

(Cal tenir present que, per tal de complir el principi de dissociació d’informació, no es pot arribar a identificar l’historial a partir del descriptor de la vista. Per tant, el gestor ha de identificar un historial abans que el metge demani la inclusió d’una nova visita).

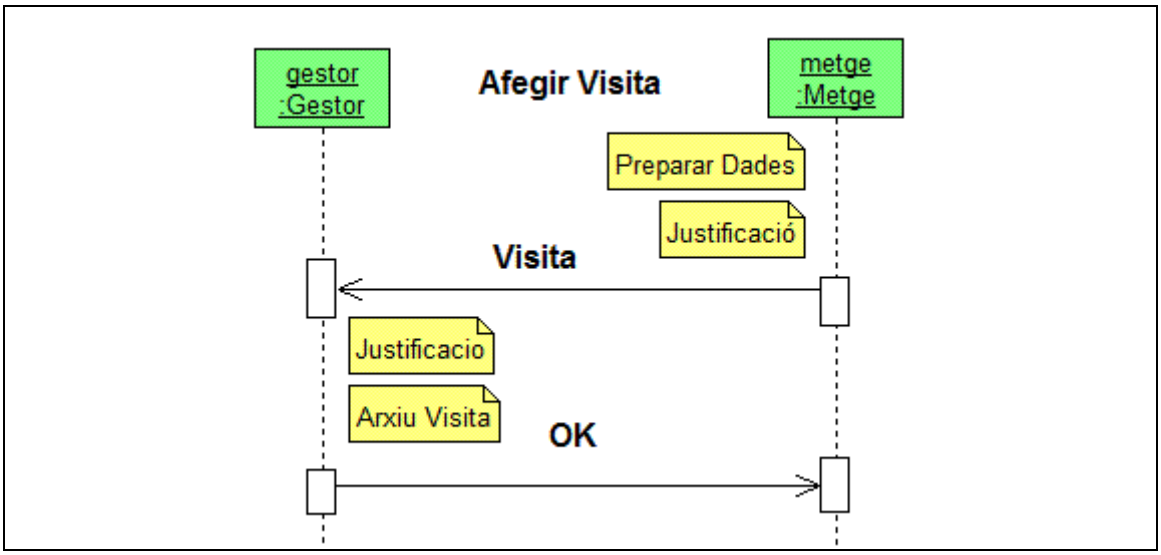


Figura 12. Afegir visita.

3.1.4.2. Consulta de dades.
La consulta de dades té tres modalitats segons qui fa la consulta. En primer lloc, els pacients tenen dret a consultar el contingut del seu propi historial. En segon lloc, tot hom amb qualificació sanitària (i que respecti el secret professional) pot consultar les dades anomenades “vitals”, (dades generals). Finalment, només els metges amb opció a crear “Visites” pot consultar el contingut de l’Historial del pacient.

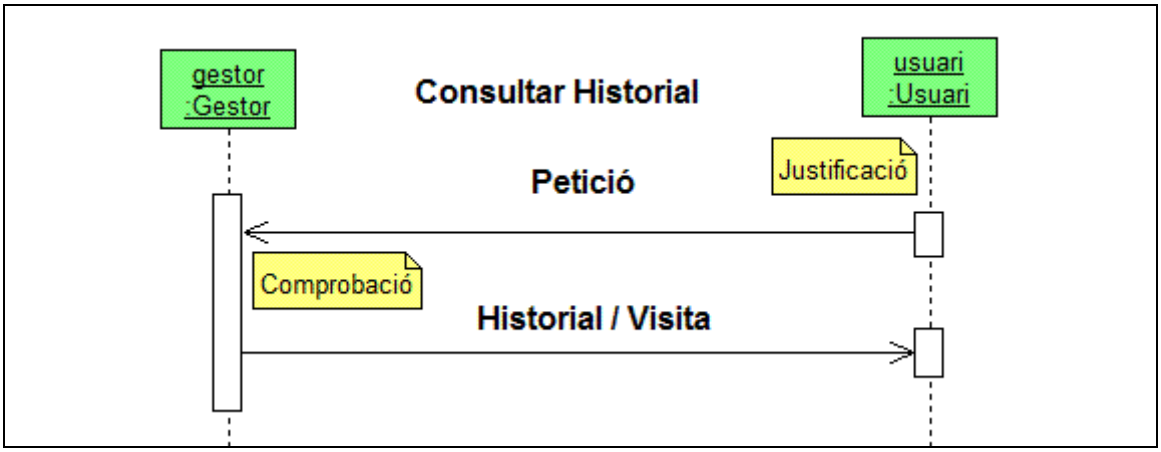


Figura 13. Consultar historial.

Les consultes es poden subdividir en les següents sub-tasques:

Primera part: consulta de l’historial.
1.- L’usuari s’ha de identificar davant del gestor (es pot fer amb la firma del seu Id_usuari)

2.- El gestor comprova la firma i contrasta Id_usuari amb la Llista Xifrada<> General d'Usuaris.
3.- L'usuari informa al Gestor del tipus de consulta que vol fer (dades generals / historial) i Id_pacient.
4.- El gestor comprova els permisos de l'usuari i decideix :

- Obrir un registre temporal de petició i enviar l'historial (El sistema genera un criptograma amb el Id_usuari_signat, Id_historial, Data, Tipus de consulta).
- rebutjar la petició.

5.- L'usuari rep l'historial i pot consultar les dades generals en clar.

Segona part: consulta d'una visita.
1.- El metge desxifra la Llista de descriptors continguda en l'historial i selecciona la visita que vol consultar. Envia al gestor una petició amb Id_metge_signat, tipus de consulta: Visita, i descriptor de visita (cal remarcar que no apareix la identitat del pacient en cap objecte de la petició).
2.- El gestor executa el procedure nº 5 de comprovacions. Desxifra i identifica Id_metge, i els seus permisos, comprova la identitat del descriptor (utilitza el Id_pacient que ha rebut en la primera part: consulta de l'historial) i decideix:

- Recuperar la visita identificada en Descriptor, xifrar-la amb P_{METGE} i enviar-la. (El sistema genera un criptograma amb Id_metge_signat, Id_descriptor, Dia:Hora).
- Rebutjar la petició.

Amb IAIK, aquesta funcionalitat es redueix a desencriptar un SignedAndEnvelopedData i obtenir la llista de visites que formen l'historial.

3.1.4.3. Modificació de dades.
En general no està prevista la modificació de dades. (Està prohibit). Només les dades generals es podran modificar. El Gestor és l'encarregat de fer les modificacions permeses.

3.1.4.4. Baixa d'usuaris.
Quan un usuari deixi de formar part del sistema, serà donat de baixa. Les seves dades seran esborrades quan sigui procedent, o transferides a una altra Base de Dades, tot respectant el principi de dissociació de dades, de manera que no es pugui relacionar la informació tecnico-sanitaria amb cap persona. El Gestor serà l'encarregat de portar a terme aquesta tasca.

3.1.5. Diagrama de classes.

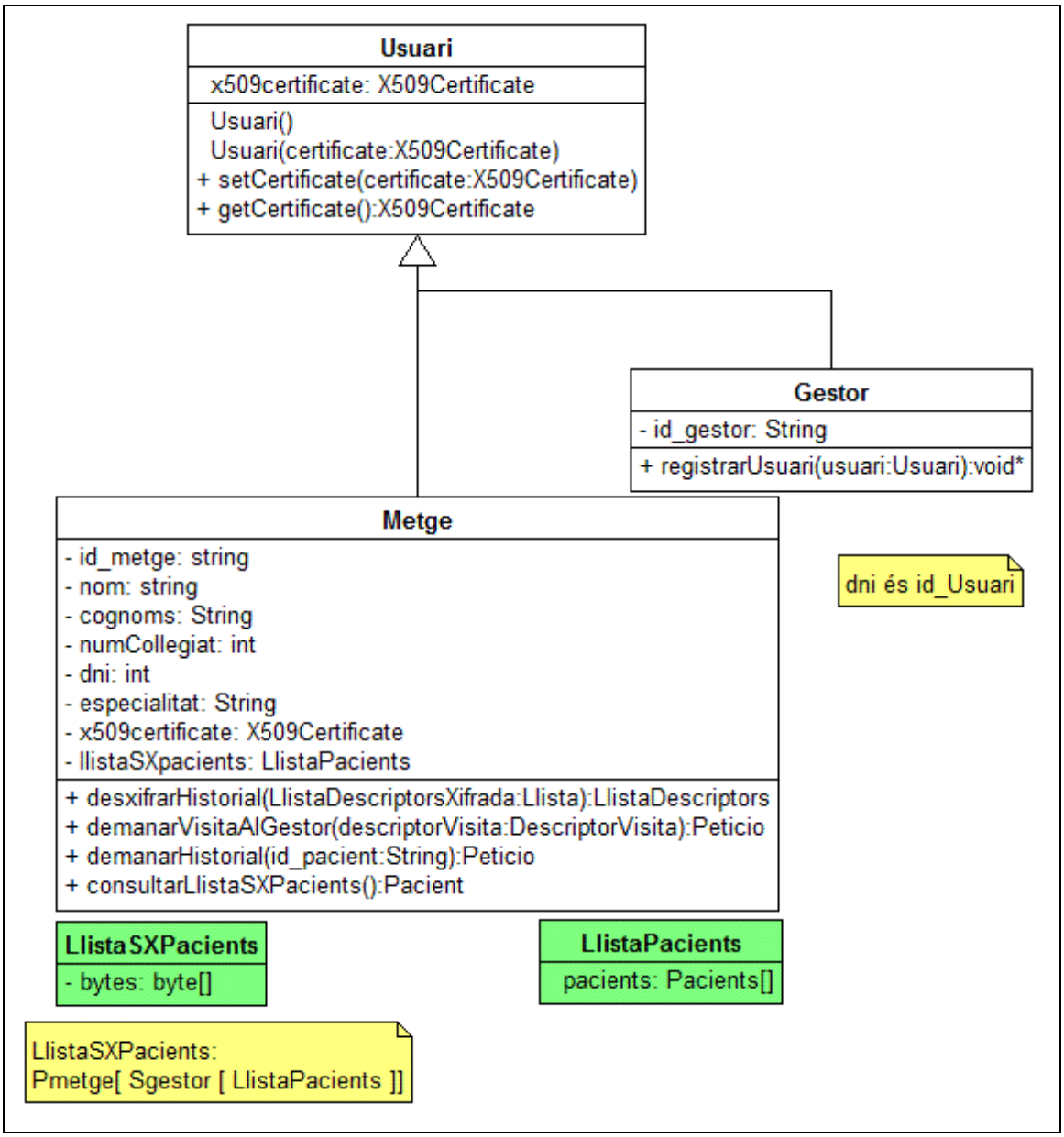


Figura 14. Diagrama de classes.

Historial:

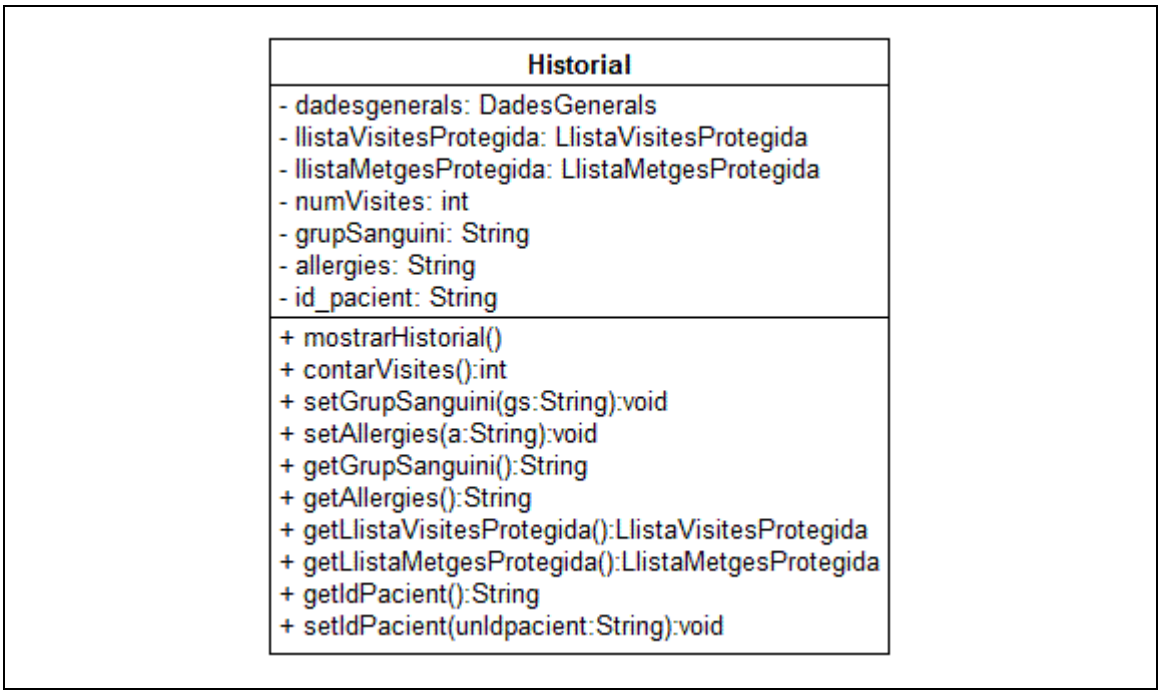


Figura 15. Clase Historial.

LlistaVisitesProtegida:

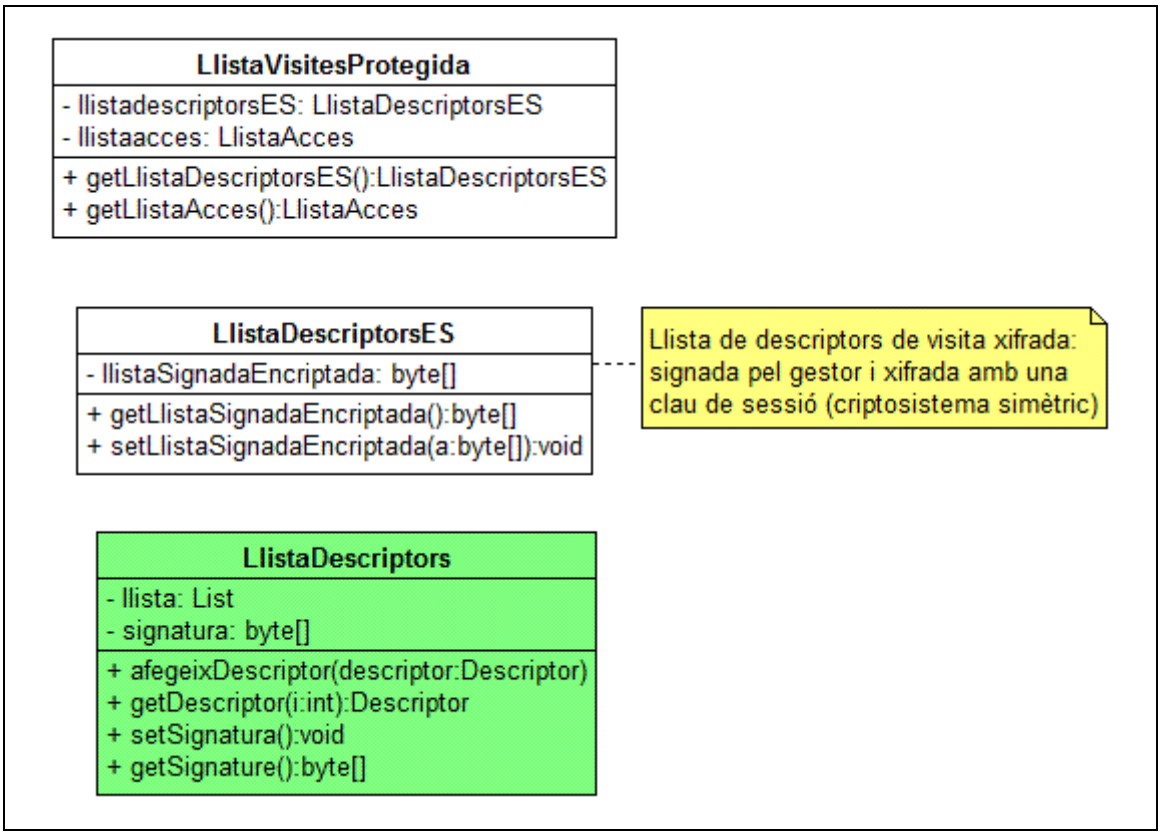


Figura 16. LlistaVisitesProtegida.

LlistaMetgesProtegida:

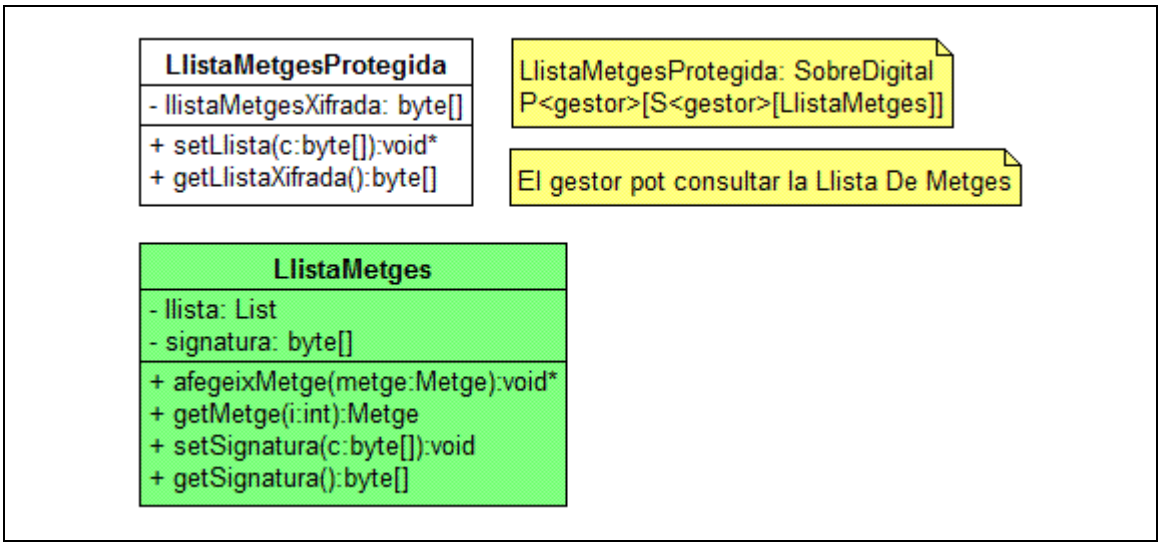


Figura 17. LlistaMetgesProtegida.

3.2. Protocols criptogràfics.

3.2.1. Notació. A la descripció dels protocols s'empra la notació següent:

NOTACIO	**CONCEPTE**
GESTOR	És el subjecte gestor del sistema.
USUARI	Pot ser el pacient interessat, o un metge (autoritzat o no).
METGE	Pot ser un metge autoritzat, o no.
PACIENT	Pacient a qui correspon l'historial.
K	Clau d'un criptosistema simètric
$E_K(M)$	Xifratge simètric d'un missatge M amb la clau K
$D_K(C)$	Desxifratge simètric del criptograma C amb la clau K
($P_{Entitat}$, $S_{Entitat}$)	Parella de claus asimètriques propietat d'Entitat on P correspon a la clau pública i S a la privada
$S_{Entitat}[M]$	Signatura digital del missatge M amb la clau privada S d'Entitat
$P_{Entitat}[M]$	Xifratge del missatge M amb la clau asimètrica pública $P_{Entitat}$ d'Entitat
H(M)	Sortida d'una funció resum criptogràfica del missatge M, aquestes funcions reben el nom de funcions hash.

$E_K(M)$ produirà un pkcs7-encryptedData, i **D_K(pkcs7-encryptedData)** produirà un pkcs7-data.
$E_K(M)$ i $P_{Entitat}[K]$ produirà un pkcs7-envelopedData.
$S_{Entitat}[M]$ produirà un pkcs7-signedData
H(M) produirà un pkcs7-digestedData.

3.2.2. PROTOCOL DE NEEDHAM-SCHROEDER

PROCEDURE 1
L'Usuari [Pacient / Metge] informa al Gestor que vol iniciar una acció tipificada.
Usuari genera un número aleatori: $\mathbf{N_i}$
Usuari recupera "**id_usuari**" que identifica l'Usuari.
Usuari vol enviar una "Petició Prèvia" composta per ($\mathbf{N_i}$, "IniciSessio", **id_usuari**) al Gestor. Si la petició prèvia fos interceptada es perdria informació "id_usuari" per tant estarà xifrada. Si volem que el Gestor pugui verificar la identitat de l'usuari, estarà signada.

$$\mathbf{P_{GESTOR}[S_{USUARI}[PeticióPrevia]]}$$

La aplicació Gestor permetrà:
a) desxifrar la petició prèvia signada i xifrada: $\mathbf{S_{GESTOR}[PeticioPreviaPS]}$
b) verificar la signatura de la petició prèvia signada, si fos el cas: P_{USUARI}[PeticioPreviaS]
c) obtenir $\mathbf{N_i}$ i **id_usuari** i obrir un registre temporal de sessions amb (N_i, id_usuari).

PROCEDURE 2
El Gestor informa l'Usuari que està preparat per a rebre la petició.
El Gestor genera un número aleatori: $\mathbf{N_g}$, que guardarà amb (N_i, id_usuari).
El Gestor prepara la "Resposta Prèvia" composta per ($\mathbf{N_i'}$, $\mathbf{N_g}$, **id_gestor**).
El Gestor xifrarà la resposta prèvia: P_{USUARI}[$\mathbf{N_i'}$, $\mathbf{N_g}$, **id_gestor**] i l'enviarà a l'usuari.

L'usuari desxifrarà la resposta prèvia: $S_{USUARI}[P_{USUARI}[\mathbf{N_i'}$, $\mathbf{N_g}$, **id_gestor**]] i recuperarà $\mathbf{N_i'}$, $\mathbf{N_g}$, **id_gestor**

PROCEDURE 3
El Gestor envia de manera segura l'historial corresponent a id_pacient, a l'usuari identificat per id_usuari, que pot ser un metge o el pacient a qui correspon l'historial.
El Gestor recupera l'historial corresponent a id_pacient : $\mathbf{Historial_{PACIENT}}$
El Gestor xifrarà l'historial amb la clau pública de l'usuari que ha fet la petició: $P_{USUARI}[\mathbf{Historial_{PACIENT}}]$ i el farà arribar a id_usuari que ha fet la petició.

PROCEDURE 4
1.- L'usuari desxifrarà l'historial que ha rebut i verificarà que és correcte: $S_{USUARI}[P_{USUARI}[\mathbf{Historial_{PACIENT}}]]$ (Pot consultar les dades generals en clar).
2.- L'historial conté la llista d'accés. Un dels criptogrames està xifrat amb la seva clau pública: $P_{USUARI}[\mathbf{K}]$ (es pot definir un conveni segons el qual el primer criptograma correspon al gestor, el segon al pacient, el tercer al metge, els següents a altres metges, o incloure informació del titular de P_{USUARI})
Farà, per tal de recuperar la clau simètrica que protegeix l'historial: $S_{USUARI}[P_{USUARI}[\mathbf{K}]]$: K
3.- Amb la clau simètrica **K** desxifrarà la llista de descriptors de visites xifrada: $\mathbf{D_K}$(LlistaDescriptorsK) : LlistaDescriptors, en clar. Ara ja pot seleccionar un descriptor de visita.
4.- Els descriptors contenen la signatura digital del Gestor. L'usuari la pot verificar: P_{GESTOR}[Signatura]

3.2.3. PROTOCOL DE CONSULTA DELS PACIENTS ASSIGNATS A UN METGE.

1. (PROCEDURE 1)
L'Usuari (Metge) informa al Gestor que vol iniciar una acció tipificada.
Usuari genera un número aleatori: $\mathbf{N_i}$
Usuari recupera "**id_usuari**" que identifica l'Usuari.
Usuari vol enviar una "Petició Prèvia" composta per ($\mathbf{N_i}$, "IniciSessio", **id_usuari**) al Gestor. Si la petició prèvia fos interceptada es perdria informació "id_usuari" per tant estarà xifrada. Si volem que el Gestor pugui verificar la identitat de l'usuari, estarà signada.
$\mathbf{P_{GESTOR}[S_{USUARI}[PeticióPrevia]]}$

El Gestor permetrà:
a) desxifrar la petició prèvia signada i xifrada: $\mathbf{S_{GESTOR}[PeticioPreviaPS]}$
b) verificar la signatura de la petició prèvia signada, si fos el cas: P_{USUARI}[PeticioPreviaS]
c) obtenir $\mathbf{N_i}$ i **id_usuari** i obrir un registre temporal de sessions amb (N_i, id_usuari).

2. (PROCEDURE 2)
El Gestor vol informar l'Usuari que està preparat per rebre la petició.
El Gestor genera un número aleatori: $\mathbf{N_g}$, que guardarà amb (N_i, id_usuari).
El Gestor prepara la "Resposta Prèvia" composta per ($\mathbf{N_i'}$, $\mathbf{N_g}$, **id_gestor**).
El Gestor xifrarà la resposta prèvia: P_{USUARI}[$\mathbf{N_i'}$, $\mathbf{N_g}$, **id_gestor**] i l'enviarà a l'usuari.

L'usuari desxifrarà la resposta prèvia: $S_{USUARI}[P_{USUARI}[\mathbf{N_i'}$, $\mathbf{N_g}$, **id_gestor**]] i recuperarà $\mathbf{N_i'}$, $\mathbf{N_g}$, **id_gestor**

3.- L'usuari metge autoritzat, realitza les operacions següents:
a1) comprovar que Gestor ha obert una sessió: $\mathbf{N_i' = N_i}$
a2) preparar la petició de la LlistaPacientsProtegida: Petició = [$\mathbf{N'_G}$, "Llista Pacients", **id_metge**]
Xifrarà la petició: P_{GESTOR}[$\mathbf{N'_G}$, "Llista Pacients", **id_metge**] i la enviarà al Gestor.
b) Si $\mathbf{N_i' \; != N_i}$ enviarà [$\mathbf{N'_G}$, "Error", **id_usuari**]

4.- El gestor realitza les operacions següents:
a) desxifrar $S_{GESTOR}[P_{GESTOR}[\mathbf{N'_G}$, "Llista Pacients", **id_metge**]] i obtenir : $\mathbf{N'_G}$, el tipus de petició "Llista Pacients", i l'identificador del metge que fa la petició **id_metge.**
El gestor comprova que id_metge forma part de la LlistaGeneralDeMetges.
Recuperar LlistaPacientsProtegida corresponent a id_metge. (Aquesta llista havia estat generada amb : $P_{METGE}[S_{GESTOR}$[LlistaPacients]]. Es a dir, estava signada pel gestor y encriptada amb la clau pública del metge).
El gestor envia la LlistaPacientsProtegida : $P_{METGE}[S_{GESTOR}$[LlistaPacients]].

El gestor anul·la la sessió.

5.- El metge autoritzat realitza les operacions següents:
a) desxifrar la llista Pacients Protegida: $S_{METGE}[P_{METGE}[S_{GESTOR}$[LlistaPacients]]] i obté S_{GESTOR}[LlistaPacients]
b) comprovar la signatura del gestor i obtenir la Llista Pacients en clar: $P_{GESTOR}[S_{GESTOR}$[LlistaPacients]]

A partir de la Llista Pacients en clar, pot trobar l'id_pacient que vol estudiar.

3.2.4. PROTOCOL DE CONSULTA DE DADES GENERALS D'UN PACIENT.

1. (PROCEDURE 1)
L'Usuari [Pacient / Metge] informa al Gestor que vol iniciar una acció tipificada.
Usuari genera un número aleatori: $\mathbf{N_i}$
Usuari recupera "**id_usuari**" que identifica l'Usuari.
Usuari vol enviar una "Petició Prèvia" composta per ($\mathbf{N_i}$, "IniciSessio", **id_usuari**) al Gestor. Si la petició prèvia fos interceptada es perdria informació "id_usuari" per tant estarà xifrada. Si volem que el Gestor pugui verificar la identitat de l'usuari, estarà signada.

$$\mathbf{P_{GESTOR}[S_{USUARI}[PeticióPrevia]]}$$

El Gestor permetrà:
a) desxifrar la petició prèvia signada i xifrada: $\mathbf{S_{GESTOR}[PeticioPreviaPS]}$
b) verificar la signatura de la petició prèvia signada, si fos el cas: P_{USUARI}[PeticioPreviaS]
c) obtenir $\mathbf{N_i}$ i **id_usuari** i obrir un registre temporal de sessions amb (N_i, id_usuari).

2. (PROCEDURE 2)
El Gestor vol informar la aplicació Usuari que està preparada per rebre la petició.
El Gestor genera un número aleatori: $\mathbf{N_g}$, que guardarà amb (N_i, id_usuari).
El Gestor prepara la "Resposta Prèvia" composta per ($\mathbf{N_i'}$, $\mathbf{N_g}$, **id_gestor**).
El Gestor xifrarà la resposta prèvia: P_{USUARI}[$\mathbf{N_i'}$, $\mathbf{N_g}$, **id_gestor**] i l'enviarà a l'usuari.

L'usuari desxifrarà la resposta prèvia: $S_{USUARI}[P_{USUARI}[\mathbf{N_i'}$, $\mathbf{N_g}$, **id_gestor**]] i recuperarà $\mathbf{N_i'}$, $\mathbf{N_g}$, **id_gestor**

3.- L'usuari realitza les operacions següents:
a1) comprovar que Gestor ha obert una sessió: $\mathbf{N_i' = N_i}$
a2) preparar la petició de l'historial corresponent a id_pacient. (id_pacient ha estat consultat prèviament). Petició = [$\mathbf{N'_G}$, "Consulta Dades Generals", **id_pacient**]
Xifrarà la petició: P_{GESTOR}[$\mathbf{N'_G}$, "Consulta Dades Generals", **id_pacient**] i la enviarà al Gestor.
("Consulta Dades Generals" vol dir que l'usuari "id_usuari" vol consultar les dades generals del pacient identificat amb id_pacient. Les dades generals són de consulta pública, per tant qualsevol metge reconegut pel sistema pot demanar-les).
b) Si $\mathbf{N_i' != N_i}$ enviarà [$\mathbf{N'_G}$, "Error", **id_usuari**]

4.- Gestor realitza les operacions següents:
a) Desxifrar la petició: $S_{GESTOR}[P_{GESTOR}[\mathbf{N'_G}$, "Consulta Dades Generals", **id_pacient**]] i obtenir la referència de la sessió $\mathbf{N'_G}$, el tipus de petició "Consulta Dades Generals", i l'identificador del pacient: **id_pacient**
b) Comprovarà que existeix una sessió: $\mathbf{N_G' = N_G}$
Verificarà que id_pacient correspon a un pacient enregistrat en el sistema.
Comprovarà si l'usuari demana les seves pròpies dades: id_usuari = id_pacient
O que id_usuari és un metge enregistrat en el sistema.
I si el metge id_metge està assignat al pacient id_pacient.

(PROCEDURE 3)
El Gestor vol enviar de manera segura l'historial corresponent a id_pacient, a l'usuari identificat per id_usuari, que pot ser un metge o el pacient a qui correspon l'historial.
El Gestor recupera l'historial corresponent a id_pacient : $\mathbf{Historial_{PACIENT}}$
El Gestor xifrarà l'historial amb la clau pública de l'usuari que ha fet la petició:
$P_{USUARI}[\mathbf{Historial_{PACIENT}}]$ i el farà arribar a id_usuari que ha fet la petició.

Si el Gestor vol recordar qui ha demanat què i quan necessitarà guardar informació de la consulta: (id_usuari, id_pacient, tipus de consulta, data:hora)
Si vol garantir que no hi haurà repudi pot guardar: (S_{USUARI}[id_usuari], id_pacient, tipus de consulta, data:hora). (Ver procedure 1)

Si vol donar fe que la consulta ha existit farà: $S_{GESTOR}[(S_{USUARI}[id\_usuari], id\_pacient, tipus\ de\ consulta, data:hora)]$ i qualsevol estarà en condicions de llegir el registre.
Però si només vol garantir la privacitat farà: $P_{GESTOR}[(S_{USUARI}[id\_usuari], id\_pacient, tipus\ de\ consulta, data:hora)]$. (Només qui tingui S_{GESTOR} podrà recuperar aquesta informació).

Si el metge id_metge no està assignat al pacient id_pacient, el sistema determina que la sessió ha terminat.
Si l'usuari és el propi pacient, o un metge assignat al pacient, (i la assignació està vigent), el sistema no pot determinar que la sessió ha acabat, perquè l'usuari pot demanar una "visita" a continuació.

5.- L'usuari realitzarà les operacions següents:
(PROCEDURE 4)
1.- L'usuari desxifrarà l'historial que ha rebut i verificarà que és correcte: $S_{USUARI}[P_{USUARI}[\mathbf{Historial_{PACIENT}}]]$ (Pot consultar les dades generals en clar).
2.- Per als usuaris autoritzats, l'historial conté la llista d'accés. Un dels criptogrames està xifrat amb la seva clau pública: $P_{USUARI}[\mathbf{K}]$ (es pot definir un conveni segons el qual el primer criptograma correspon al gestor, el segon al pacient, el tercer al metge, els següents a altres metges amb assignació, o es pot definir una estructura amb informació del titular de P).
Farà, per tal de recuperar la clau simètrica que protegeix l'historial: $S_{USUARI}[P_{USUARI}[\mathbf{K}]] : K$
3.- Amb la clau simètrica **K** desxifrarà la llista de descriptors de visites xifrada: $\mathbf{D_K}$(LlistaDescriptorsK) : LlistaDescriptors, en clar. Ara ja pot seleccionar un descriptor de visita.
4.- Els descriptors contenen la signatura digital del Gestor. L'usuari la pot verificar: P_{GESTOR}[Signatura]

3.2.5. PROTOCOL DE CONSULTA DE VISITES D'UN PACIENT.

L'usuari coneix el Descriptor de la visita que vol consultar.

1. (PROCEDURE 1)
L'Usuari [Pacient / Metge] informa al Gestor que vol iniciar una acció tipificada.
Usuari genera un número aleatori: $\mathbf{N_i}$
Usuari recupera "**id_usuari**" que identifica l'Usuari.
Usuari vol enviar una "Petició Prèvia" composta per ($\mathbf{N_i}$, "IniciSessio", **id_usuari**) al Gestor. Si la petició prèvia fos interceptada es perdria informació "id_usuari" per tant estarà xifrada. Si volem que el Gestor pugui verificar la identitat de l'usuari, estarà signada.

$$\mathbf{P_{GESTOR}[S_{USUARI}[PeticióPrevia]]}$$

El Gestor permetrà:
a) desxifrar la petició prèvia signada i xifrada: $\mathbf{S_{GESTOR}[PeticioPreviaPS]}$
b) verificar la signatura de la petició prèvia signada, si fos el cas: P_{USUARI}[PeticioPreviaS]
c) obtenir $\mathbf{N_i}$ i **id_usuari** i obrir un registre temporal de sessions amb (N_i, id_usuari).

2. (PROCEDURE 2)
El Gestor vol informar l' Usuari que està preparat per rebre la petició.
El Gestor genera un número aleatori: $\mathbf{N_g}$, que guardarà amb (N_i, id_usuari).
El Gestor prepara la "Resposta Prèvia" composta per ($\mathbf{N_i'}$, $\mathbf{N_g}$, **id_gestor**).
El Gestor xifrarà la resposta prèvia: $P_{USUARI}[\mathbf{N_i'}$, $\mathbf{N_g}$, **id_gestor**] i l'enviarà a l'usuari.

L'usuari desxifrarà la resposta prèvia: $S_{USUARI}[P_{USUARI}[\mathbf{N_i'}$, $\mathbf{N_g}$, **id_gestor**]] i recuperarà $\mathbf{N_i'}$, $\mathbf{N_g}$, **id_gestor**

3.- L'usuari realitza les operacions següents:
a1) comprovar que Gestor ha obert una sessió: $\mathbf{N_i'} = \mathbf{N_i}$
a2) preparar la petició de la visita corresponent al Descriptor seleccionat prèviament.
Petició = [$\mathbf{N'_G}$, "Consulta Visita", **id_usuari, Descriptor**]
Xifrarà la petició: P_{GESTOR}[$\mathbf{N'_G}$, "Consulta Visita", **id_usuari, Descriptor**] i la enviarà al Gestor.
b) Si $\mathbf{N_i'} \mathbf{!= N_i}$ enviarà [$\mathbf{N'_G}$, "Error", **id_usuari**]

4.- Gestor realitza les operacions següents:
a) Desxifrar la petició: $S_{GESTOR}[P_{GESTOR}[\mathbf{N'_G}$, "Consulta Visita", **id_usuari, Descriptor]**] i obtenir la referència de la sessió $\mathbf{N'_G}$, el tipus de petició "Consulta Visita", l'identificador de l'usuari: **id_usuari** i el Descriptor de la visita.
b) Comprovarà que existeix una sessió: $\mathbf{N_G'} = \mathbf{N_G}$
Comprovarà si l'usuari demana les seves pròpies dades: Descriptor forma part de LlistaVisitesProtegides de l'Historial de id_usuari.
O que id_usuari és un metge enregistrat en el sistema.
I si el metge id_metge està assignat al pacient id_pacient.
c) Obtenir la Visita identificada per Descriptor i calcular P_{USUARI}[Visita].
d) Enviar P_{USUARI}[Visita] a l'usuari.
El SistemaGestor registrarà la petició de visita. (id_usuari_signat, descriptor, data).

En el cas que alguna comprovació fos errònia enviarà: enviarà [$\mathbf{N'_G}$, "Error", **id_usuari**]
El gestor traurà la sessió $\mathbf{N'_G}$ de la memòria.

5. L'usuari desxifrarà la visita: $S_{USUARI}[P_{USUARI}[Visita]]$.

3.2.6. PROTOCOL PER AFEGIR UNA VISITA A L'HISTORIAL D'UN PACIENT.

En aquest protocol suposem que prèviament a la inserció de les dades, el metge ha consultat l'historial del pacient, i per tant, coneix id_pacient. (També suposem que Gestor sap quin és id_pacient).

1. (PROCEDURE 1)
L'Usuari (Metge autoritzat) informa al Gestor que vol iniciar una acció tipificada.
Usuari genera un número aleatori: $\mathbf{N_i}$
Usuari recupera "**id_usuari**" que identifica l'Usuari.
Usuari vol enviar una "Petició Prèvia" composta per ($\mathbf{N_i}$, "IniciSessio", **id_usuari**) al Gestor. Si la petició prèvia fos interceptada es perdria informació "id_usuari" per tant estarà xifrada. Si volem que el Gestor pugui verificar la identitat de l'usuari, estarà signada.

$$\mathbf{P_{GESTOR}[S_{USUARI}[PeticióPrevia]]}$$

La aplicació Gestor permetrà:
a) desxifrar la petició prèvia signada i xifrada: $\mathbf{S_{GESTOR}[PeticioPreviaPS]}$
b) verificar la signatura de la petició prèvia signada, si fos el cas: P_{USUARI}[PeticioPreviaS]
c) obtenir $\mathbf{N_i}$ i **id_usuari** i obrir un registre temporal de sessions amb (N_i, id_usuari).

2. (PROCEDURE 2)
El Gestor vol informar l'Usuari que està preparat per rebre la petició.
El Gestor genera un número aleatori: $\mathbf{N_g}$, que guardarà amb (N_i, id_usuari).
El Gestor prepara la "Resposta Prèvia" composta per ($\mathbf{N_i'}$, $\mathbf{N_g}$, **id_gestor**).
El Gestor xifrarà la resposta prèvia: $P_{USUARI}[\mathbf{N_i'}$, $\mathbf{N_g}$, **id_gestor**] i l'enviarà a l'usuari.

L'usuari desxifrarà la resposta prèvia: $S_{USUARI}[P_{USUARI}[\mathbf{N_i'}, \mathbf{N_g}, \mathbf{id\_gestor}]]$ i recuperarà $\mathbf{N_i'}$, $\mathbf{N_g}$, **id_gestor**

3.- El metge realitza les operacions següents:
a1) comprovar que Gestor ha obert una sessió: $\mathbf{N_i' = N_i}$
a2) preparar les Dades de la nova visita: Nova visita = [DadesVisita,]
a3) preparar el Descriptor de visita: id_visita, data:hora, tema, metge
a4) signar Dades + Descriptor: S_{METGE}[Dades+Descriptor] i afegir-les a la visita.
a5) preparar la petició de nova visita.

Petició = [$\mathbf{N'_G}$, "Afegir Visita", **id_metge, Visita**]
Xifrarà la petició: $P_{GESTOR}[\mathbf{N'_G}$, "Afegir Visita", **id_metge, Visita**] i la enviarà al Gestor.
b) Si $\mathbf{N_i' \; != N_i}$ enviarà [$\mathbf{N'_G}$, "Error", **id_usuari**]
(Al començament del protocol hem apuntat que Gestor coneix id_pacient, de protocols anteriors).

4.- El Gestor realitza les operacions següents:
a) desxifrar la petició: $S_{GESTOR}[P_{GESTOR}[\mathbf{N'_G}$, "Afegir Visita", **id_metge, Visita**]] i obtenir $\mathbf{N'_G}$, el tipus de petició "Afegir Visita", l'identificador del metge **id_metge,** i la nova **Visita**.
b) verificar que id_metge correspon a un metge del sistema.
Verificar que id_pacient és un pacient assignat a id_metge.
Verificar la signatura digital incorporada en visita: $P_{METGE}[S_{METGE}$[Dades+Descriptor]]
c) Afegir el descriptor a la LlistaDescriptors:
Obtenir el Descriptor de Visita.
Obtenir l'Historial corresponent a id_pacient.
Recuperar el criptograma P_{GESTOR}[K] de la LlistaAcces.
Obtenir K de sessió: $S_{GESTOR}[P_{GESTOR}$[K]]
Obtenir la LlistaVisitesProtegida de l'historial. Desencriptar $E_K[S_{GESTOR}$[LlistaDescriptors]] : $D_K[E_K[S_{GESTOR}$[LlistaDescriptors]]] i obtenir S_{GESTOR}[LlistaDescriptors].
Obtenir LlistaDescriptors en clar amb $P_{GESTOR}[S_{GESTOR}$[LlistaDescriptors]].
I finalment afegir Descriptor a la Llista de descriptors.
Signar la nova Llista de descriptors: S_{GESTOR}[LlistaDescriptors]
Encriptar amb una nova K : $E_K[S_{GESTOR}$[LlistaDescriptors]]
Generar els nous criptogrames de la LlistaAcces: P_{GESTOR}[K], $P_{PACIENT}$[K], P_{METGE}[K], ...
d) Afegir Visita a la base de dades.

En cas que les verificacions siguin incorrectes cal retornar: [$\mathbf{N'_G}$, "Error", **id_metge**]

Capítol 4. XML.

1. Generalitats. XML i DTD.

1.1. XML

XML és un estàndard de W3C (World Wide Web Consortium) integrat plenament en el món Internet. Es susceptible de utilització per distints tipus de programes en distints tipus de sistemes operatius.

Les sigles signifiquen Extensible Mark-up Language, derivat del SGML, que s'utilitza per realitzar documents estàndard de intercanvi o emmagatzemament de dades. XML defineix unes regles que garanteixen l'estructura de la informació. XML s'assembla a HTML per ser un llenguatge de marques (tags) basat en documents de text plans, i es distingeix, entre d'altres coses, per renunciar a definir la interface d'usuari. Tots els documents XML indiquen de manera explícita la codificació utilitzada (<?xml version="1.0" encoding="UTF-8" ?>

SGML (Standard Generalizad Mark-up Language) fou creat per IBM el 1986 per a la creació i definició de documents, amb uns documents de validació anomenats DTD (Document Type Definition). XML hereta els plantejaments de SGML però té menys dificultat.

XML exigeix el compliment estricte de unes regles de validació y ofereix a canvi facilitat de procés.
Primera regla: un document XML conté u o més elements o etiquetes. Les etiquetes son marques que s'obren i es tanquen en el mateix ordre de la seva creació, de manera anidada. (S'obté una estructura a manera d'arbre amb un element root).
Segona regla: totes les etiquetes consten d'una etiqueta d'obertura <nom>, i una de tancament, amb idèntic nom que la d'obertura, precedit per "/" </nom>. XML també accepta etiquetes buides que indiquen la obertura i el tancament simultani <nom/>.
Tercera regla: XML és case-sensitive.
Les etiquetes en XML estan comportes per els signes de obertura (<) i tancament (>) , pel nom de la etiqueta i per una sèrie de atributs que aporten significat a la etiqueta.
Quarta regla: el valor dels atributs es presenta sempre entre comilles doble(") o simples(').
És recomanable que els atributs siguin valor atòmics, sense estructura. En cas contrari cal recorre a definir noves etiquetes. En altres paraules, XML accepta la substitució de elements per atributs.
Cinquena regla: els caràcters especials es representaran en base 10 (&#DDD) o en base 16 (&#xDDD) segons la codificació definida en la declaració.

1.2. DTD.

DTD són les sigles de Document Type Definition i el seu propòsit és definir la legalitat dels blocs d'un document XML. Bàsicament defineix l'estructura d'un document amb una llista dels elements legals.

DTD permet definir una sintaxi de regulació de la formació de les etiquetes XML. Es el conjunt de definicions que descriu una estructura de document, declarant i definit tots els tipus d'elements del document, l'ordre de cada tipus d'element i qualsevol atribut, entitat, anotació, etc. Els documents DTD serveixen per validar els documents xml. La validació es optativa, però garanteix l'absència d'errors.

DTD pot aparèixer dins dels documents XML o fora (DTD externa). Tots els documents XML porten la següent línea al capdavant: <!DOCTYPE document SYSTEM "document.dtd">
Les DTD externes presenten alguns avantatges davant DTD interna que les fan aconsellables: flexibilitzen el treball.

A continuació es presenta un quadre amb els trets bàsics de la sintaxi DTD:

ELEMENT:
Es correspon amb les etiquetes xml. La sintaxi és la següent: <!ELEMENT Name Type>
Type pot ser : EMPTY 1 | ANY | altres ELEMENT | Mixt amb elements i text.
Tindrà el següent aspecte:
"<!ELEMENT" nom de l'element "(" contingut ")" signe quantificador ">"
El contingut pot ser "#PCDATA" o "ELEMENT [ELEMENT | ELEMENT]
El contingut pot expressar ORDRE amb comes. Pot expressar alternatives amb "|".
Els signes quantificadors són +, *, ? . + vol dir una o varies vegades, * cero o varies vegades, ? una o cap vegada.
Sempre cal definir un ELEMENT arrel, amb el mateix nom del fitxer dtd.

Atributs ATTLIST:
"<!ATTLIST" nomElement nomAtribut Type Default ">".
NomElement és un ELEMENT definit.
NomAtribut és el nom de l'atribut que es defineix.
Type pot ser: CDATA, ENUMERATION, NOTATION, ENTITY, ENTITIES, ID, IDREF, IDREFS, NMTOKEN o NMTOKENS.
Default pot prendre els següents valors: #REQUIRED, #IMPLIED, #FIXED, default. Fa referència a la obligatorietat d'omplir l'atribut o no.

1.3. DOM.

DOM (Document Object Model) s'utilitza per fer la validació de documents XML. És una metodologia de treball amb documents XML. Els elements es transformen en NODES d'un arbre. Permet moure'ns pels nodes fent recorreguts en funció de la relació pare / fill / germà. Les tasques comuns que podem trobar són carrega, localització i creació. La càrrega permet fer la conversió del document XML en una jerarquia d'objectes per tal que sigui més senzill manipular el contingut. La creació permet fer modificacions del contingut del document XML.

2. Estructura de dades de gestió d'historials mèdics.

2.1. DTD.

En una primera aproximació al problema es pot dir que ens farà falta crear <ObjecteXML> que contindrà <Persona> (una o més) o bé <Document> (també u o més d'un).
<Persona> contindrà informació de <Autorititat Certificadora> o <Usuari> (només un).
<Usuari> contindrà informació de <Gestor>, <Metge> o <Pacient> (només un).
<Document> contindrà informació de <Historial>, <Visita> o <Missatge> (només un).

Es pot argumentar que <Visita> anirà dintre de <Historial>, perquè un Historial està format per moltes Visites, i que <Historial> anirà dintre de <Pacient> perquè no té sentit parlar de cap Historial que no pertany a un Pacient determinat. Aquesta estructura lògica cal descartar-la per tal de respectar el principi de dissociació de dades.
Per tant, farem un esquema en que <Historial> i <Visita> seran elements autònoms.

Els <ObjecteXML> tindrà metadades dintre del tag per tal de determinar la naturalesa de les dades:

<ObjecteXML data=DATE> la data ens permetrà sincronitzar els sistemes i conèixer el moment exacte de generació.

<Usuari> contindrà metadades de identificació física i electrònica. Una altra opció es incloure un tag <Identitat><Fisica/><Electronica><Certificat/></Electronica></Identitat>.

<Document> ha d'incloure suficients metadades per tal de poder ser processat.
<Document><Metadades></Metadades><Contingut></Contingut></Document>
Les metadades de <Document> aportaran informació de l'autor (identitat física, dades per identificar el certificat quan el document estigui encriptat..., comentaris d'altres persones i metadades referents al propi document: tipus, data, paraules claus, longitud, signatura, resum...

ObjecteXML.dtd

```
<!ELEMENT objecteXML (usuari | document)?>
<!ATTLIST objecteXML
        idObjecteXML CDATA #IMPLIED
        sincronitzat CDATA #IMPLIED>
<!ELEMENT adreca (#PCDATA)?>
<!ELEMENT allergia (#PCDATA)>
<!ELEMENT anamnesi (#PCDATA)>
<!ELEMENT certificat ()>
<!ELEMENT cognoms (#PCDATA)>
<!ELEMENT criptograma ()>
<!ELEMENT curriculum numCollegiat, especialitat*>
<!ELEMENT dadesAdministratives numTarjetaSanitaria>
<!ELEMENT dadesGenerals grupSanguini?, allergia*>
<!ELEMENT dadesSanitaries grupSanguini?, allergia*>
<!ELEMENT dadesVisita anamnesi, diagnosi, tractament>
<!ELEMENT dia (#PCDATA)?>
<!ELEMENT descriptor idVisita, dia, hora?, tema, idMetge+>
<!ELEMENT diagnosi (#PCDATA)>
<!ELEMENT dni (#PCDATA)>
<!ELEMENT document ( historial | visita | missatge)? >
<!ELEMENT especialitat (#PCDATA)?>
<!ELEMENT gestor (#PCDATA)? >
<!ELEMENT grupSanguini (#PCDATA)?>
<!ELEMENT historial idPacient, dadesGenerals, llistaVisites, llistaMetges, seguretat>
<!ELEMENT hora (#PCDATA)?>
<!ELEMENT identitatElectronica (certificat)*>
<!ELEMENT identitatFisica nom, cognoms, dni>
<!ELEMENT idMetge (#PCDATA)>
<!ELEMENT idPacient (#PCDATA)>
<!ELEMENT idVisita (#PCDATA)>
<!ELEMENT llistaAcces criptograma* >
<!ELEMENT llistaDescriptors descriptor* >
<!ELEMENT llistaMetges idMetge* >
<!ELEMENT llistaPacients idPacient* >
<!ELEMENT llistaVisites llistaDescriptors, llistaAcces >
<!ELEMENT metge curriculum, llistaPacients >
<!ELEMENT missatge (#PCDATA)?>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT numCollegiat (#PCDATA)>
<!ELEMENT pacient dadesAdministratives, dadesSanitaries >
<!ELEMENT resum (#PCDATA)?>
<!ELEMENT seguretat signatura?, resum? >
<!ELEMENT signatura ()>
<!ELEMENT telefon (#PCDATA)?>
<!ELEMENT tema (#PCDATA)>
```

<!ELEMENT tractament (#PCDATA)>
<!ELEMENT usuari identitatFisica, identitatElectronica, adreca*, telefon*, (gestor | metge | pacient)?, seguretat >
<!ELEMENT visita descriptor, dadesVisita, signatura >

2.2. XML.

Definició d'Historial amb XML.

Historial.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE objecteXML SYSTEM "objecteXML.dtd">
<objecteXML idObjecteXML =”1” sincronitzat=”08/04/08:16:03:00”><document>
<historial>
<idPacient>00000000-A</idPacient>
<dadesGenerals>
        <grupSanguini>A</grupSanguini>

</dadesGenerals>
<llistaVisites>
        <llistaDescriptors></llistaDescriptors>
        <llistaAcces></llistaAcces>
</llistaVisites>
<llistaMetges></llistaMetges>
<seguretat></seguretat>
</historial>
</document></objecteXML>
```

Definició de Visita amb XML.

Visita.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE objecteXML SYSTEM "objecteXML.dtd">
<objecteXML><document>
<visita>
        <descriptor>
                <idVisita>1-1</idVisita>
                <dia>08/04/08</dia>
                <hora>25:03:00</hora>
                <tema>alta pacient</tema>
                <idMetge>00000000-A</idMetge>
        </descriptor>
        <dadesVisita>
                <anamnesi></anamnesi>
                <diagnosi></diagnosi>
```

```
                <tractament></tractament>
        </dadesVisita>
        <signatura>
                “.....”
        </signatura>
</visita>
</document></objecteXML>
```

Capítol 5. RMI.

1. RMI. Concepte.

Quan utilitzem sockets, ens hem de preocupar de cóm es transmeten fisicament les dades entreels extrems d'una conexió (a nivel de bytes). RMI permet oblidar-nos dels detalls de la transmisió de dades i centrar-nos al disseny de la lògica de la nostra aplicació, perque ens permet accedir a un objecte remot com si fos d'un objecte local. RMI utilitza serialització per fer la transmissió de dades a través de la xarxa.

Està escrit integrament en Java.

2. Generalitats.

Les aplicacions RMI comprenen dos programes separats: un servidor i un client. Una aplicació servidor típica crea un objecte romot, fa accesible una referència a l'objecte remot, i espera que els clients cridin aquest mètode remot.
Una aplicació client típica obté una referència remota d'un o més objectes remots en el servidor i crida als seus métodes.

Es pot explicar el funcionament amb un exemple. Suposem que volem sincronitzar dos ordinadors. Ens farà falta coneixer la Data i l'Hora dels dos Sistemes Operatius. Definirem una interficie amb un métode consultaFecha()

```
public String consultaFecha();
```

I un Objecte Remot amb el métode:

```
public String consultaFecha(){
        java.util.Date d = new java.util.Date();
        return d.toString();
}
```

El client cridaría el métode remot:

```
try{
        Interfaz hm = (Interfaz)Naming.lookup(direccion + "ObjetoRemoto");
        String msgRebut = new String(hm.consultaFecha());
}
```

i obtindria la informació de java.util.Date corresponent al Servidor.

Depenent de la definició del métode remot podem aconseguir que una informació generada en el Servidor aparegui en el Client o a la inversa, que una informació introduida pel Client aparegui en el Servidor.

2.1. Estructura de les aplicacions RMI.
A la figura seguent es presenten els components de les aplicacions RMI.

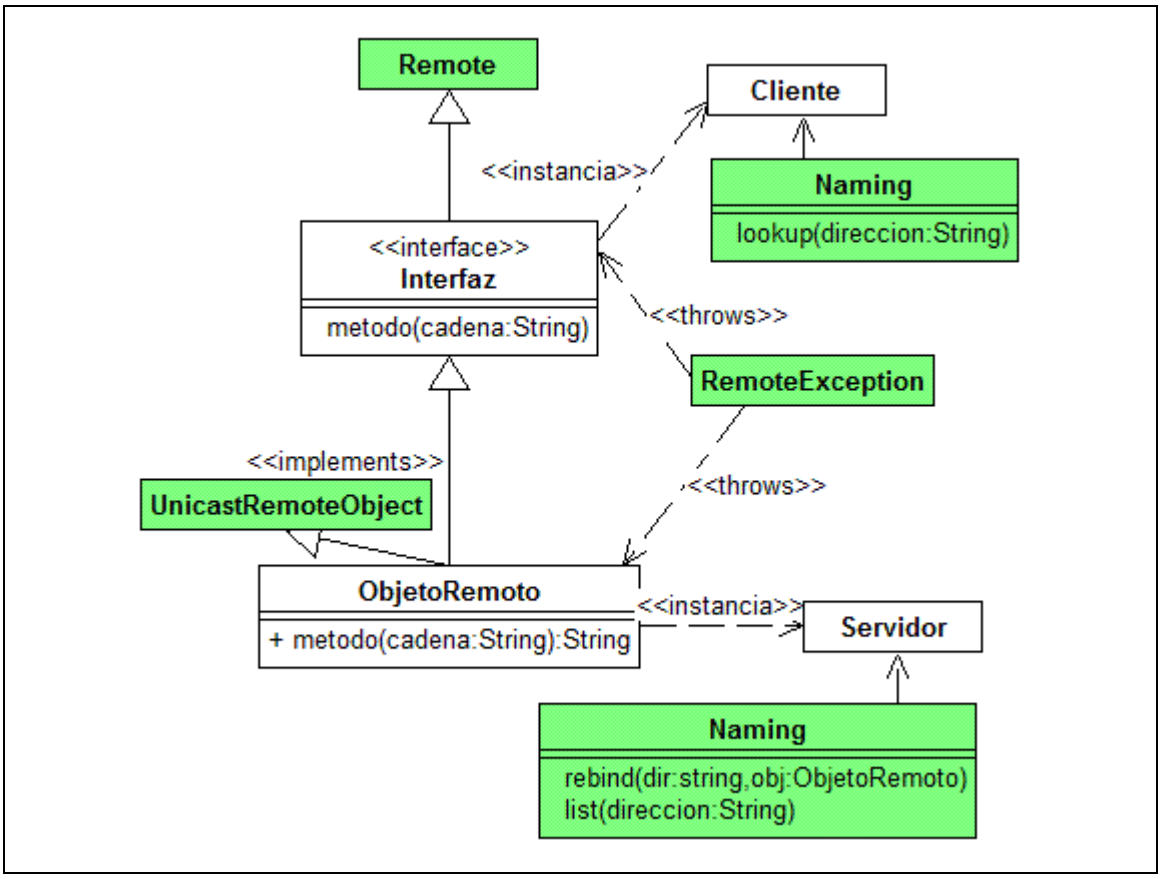


Figura 19. RMI

2.1.1. Interfaz
Conté la definició dels métodes remots que podrà utilitzar el client. Només els métodes publicats es podran cridar.

Interfaz.java

```
import java.rmi.*;
public interface Interfaz extends Remote{
        String metodeRemot(String cadena) throws RemoteException;
}
```

"Interfaz extends java.io.Serializable" permet crear objectes Serializable que es poden enviar per la xarxa.

2.1.2. ObjecteRemot

ObjetoRemoto.java

```
import java.rmi.*;
import java.rmi.server.*;
public class ObjetoRemoto extends UnicastRemoteObject implements Interfaz{
```

```
        public ObjetoRemoto() throws RemoteException{
                //super();
                }

        public String metodo(String dadesDelClient) throws RemoteException{
                return ("Dades per al Servidor: " +  dadesDelClient);
        }
}
```

2.1.3. Servidor

El servidor ha de registrar els objectes remots. Cadascun amb un nom. El SecurityManager s'encarrega de controlar les accions dels objectes descarregats per la xarxa. El controlador de seguretat determina si el còdi descarregat té accés al sistema de fitxers locals o si pot realitzar qualsevol altra operació privilegiada.

Servidor.java

```
import java.io.*;
import java.rmi.*;
import java.rmi.registry.*;
import java.rmi.server.*;
import java.net.*;

public class Servidor{
        public static void main(String args[]){
                try{
                        if(System.getSecurityManager() == null){
                                System.setSecurityManager(new RMISecurityManager());
                        }

                        //Es crea una instància de l'objecte remot
                        ObjetoRemoto obj = new ObjetoRemoto();
                        //Es defineix la URL
                        //rmi://host:port/remoteobjetname
                        //host i port son optatius. (El port per defecte es 1099)
                        //Aquestes mateixes dades serán utilitzades pel Client.
                        Naming.rebind("rmi://localhost:2002/ObjetoRemoto", obj);
                        System.out.println("Servidor esperando llamadas ");


                }catch(Exception e){
                        System.out.println("Error en main: " + e.toString());
                }
        }
}
```

2.1.4. Client

Només els mètodes publicats en la Interfaz es poden cridar des del client.
La referència a l'objecte remot es realitza amb el mètode "lookup" de la classe "Naming", i es passa com argument la direcció o nom de l'equip on es troba l'objecte remot, i el nom de l'objecte remot.

Cliente.java

```
import java.rmi.*;
public class Cliente{
```

```
	public static void main(String[] args){
		String direccion = "rmi://localhost:2002/";
		try{
			//Es crea una instància de la interfaz.
			Interfaz hm = (Interfaz)Naming.lookup(direccion + "ObjetoRemoto");
			//el nom pot ha de ser el utilitzat al Servidor(p.ej. “ObjetoRemoto”).
			//Utilitzem el métode definit a Interfaz
			System.out.println(hm.metodo("Dades generades al Client"));
			} catch(Exception e){
				e.printStackTrace();
			}
			System.exit(0);
	}
}
```

2.2. Compilació i utilització.

A continuació s’inclouen els quatre fitxers que fan falta per tal de compilar i utilitzar els métodes remots.

Per compilar l’aplicació utilitzarem les eines de Java: rmic.exe i rmiregistry.exe. Cal respectar l’ordre següent:

aplicacio1.bat

```
rem Compilar:
%JAVA%\javac Interfaz.java
%JAVA%\javac ObjetoRemoto.java
%JAVA%\javac Servidor.java
%JAVA%\javac Cliente.java
rem crear los fichero Stub y Skeleton: Stub se sitúa en la JVM cliente y Skel se sitúa en JVM rem Servidor
%JAVA%\rmic -vcompat -verbose -d %classpath% ObjetoRemoto
```

aplicacio2.bat

```
rem ejecutar el registro de objetos remotos
rem %JAVA%\start rmiregistry.exe port
rem para terminar pulsar [Control+C]
rem unset CLASSPATH
%JAVA%\rmiregistry 2002
```

aplicacio3.bat

```
rem ejecutar el servidor en una consola diferente a rmiregistry
rem %JAVA%\java -Djava.security.policy=java.policy Servidor
rem El Servidor funciona mentre hi hagi algun objecte registrat
rem para terminar pulsar [Control+C]
%JAVA%\java -Djava.security.policy=java.policy Servidor
```

aplicacio4.bat

```
rem ejecutar el cliente en una consola diferente a servidor
%JAVA%\java -Djava.security.policy=java.policy Cliente
```

3. Plantejament.

Per la nostra aplicació necesitem un métode que permeti al Client transferir al Servidor les dades (byte[]) següents:
byte[] peticioPrevia;

El client cridarà un metode remot que accepti “peticioPrevia” com a paràmetre, i que actualitzi una variable de la classe ObjecteRemot. Aleshores el Servidor accedirà a aquesta variable i fará la lectura de les dades enviades pel Client.

(Aquesta part es correspon amb el PROCEDURE 1 de Needham-Schroeder)

També necessitem un métodо remot que permeti al Servidor retornar la resposta al Client:
El servidor introduirà els byte[] de resposta dins la variable “byte[] resposta” de la classe ObjecteRemot, i el Client utilitzarà un métode getResposta() per tal de recuperar la informació.

(Aquesta part es correspon amb el PROCEDURE 2 de Needham-Schroeder)

Farà falta implementar una Interfaz que faci extensió de Serializable per tal de poder transferir objectes Historial.
El Servidor RMI actualitzará una variable de la classe ObjecteRemot, “Historial *un*historial”, i el client utilitzarà un métode getHistorial() per tal de recuperar l’historial.

(Part corresponent al PROCEDURE 3).

Capítol 6. Base de dades. (BD).

1. Generalitats de Bases de dades.

Un sistema de gestió de bases de dades relacional (SGBD) dona suport a la definició de dades mitjançant l'estructura de de dades del model relacional, la manipulació de les dades amb les operacions del model i, a més, assegura que se satisfan les regles d'integritat que el model estableix.

En general, tots els SGBD compleixen les regles següents:

- unicitat de la clau primària: no pot tenir valors repetits.
- Entitat de la clau primària: els atributs de la clau primària no poden tenir valors nuls.
- Integritat referencial o de clau forana: tots els valors que pren una clau forana han de ser valors existents a la clau primaria que referencia, o valors nulls.

1.1. La sintaxi de MySQL.

1.1.1. Definicio de Taules.

Per tal de no desesperar amb la utilització d'una base de dades és molt conveniet dominar la seva sintaxi, per aquest motiu s'inclou el següent cuadre:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    [(create_definition,...)]
    [table_options] [select_statement]



CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    [() LIKE old_tbl_name ()];

create_definition:
    column_definition
  | [CONSTRAINT [symbol]] PRIMARY KEY [index_type] (index_col_name,...)
  | KEY [index_name] [index_type] (index_col_name,...)
  | INDEX [index_name] [index_type] (index_col_name,...)
  | [CONSTRAINT [symbol]] UNIQUE [INDEX]
        [index_name] [index_type] (index_col_name,...)
  | [FULLTEXT|SPATIAL] [INDEX] [index_name] (index_col_name,...)
  | [CONSTRAINT [symbol]] FOREIGN KEY
        [index_name] (index_col_name,...) [reference_definition]
  | CHECK (expr)

column_definition:
    col_name type [NOT NULL | NULL] [DEFAULT default_value]
        [AUTO_INCREMENT] [[PRIMARY] KEY] [COMMENT 'string']
        [reference_definition]

type:
    TINYINT[(length)] [UNSIGNED] [ZEROFILL]
  | SMALLINT[(length)] [UNSIGNED] [ZEROFILL]
  | MEDIUMINT[(length)] [UNSIGNED] [ZEROFILL]
  | INT[(length)] [UNSIGNED] [ZEROFILL]
  | INTEGER[(length)] [UNSIGNED] [ZEROFILL]
  | BIGINT[(length)] [UNSIGNED] [ZEROFILL]
  | REAL[(length,decimals)] [UNSIGNED] [ZEROFILL]
  | DOUBLE[(length,decimals)] [UNSIGNED] [ZEROFILL]
  | FLOAT[(length,decimals)] [UNSIGNED] [ZEROFILL]
  | DECIMAL(length,decimals) [UNSIGNED] [ZEROFILL]
  | NUMERIC(length,decimals) [UNSIGNED] [ZEROFILL]
  | DATE
  | TIME
  | TIMESTAMP
  | DATETIME
  | CHAR(length) [BINARY | ASCII | UNICODE]
  | VARCHAR(length) [BINARY]
  | TINYBLOB
  | BLOB
  | MEDIUMBLOB
```

```
  | LONGBLOB
  | TINYTEXT
  | TEXT
  | MEDIUMTEXT
  | LONGTEXT
  | ENUM(value1,value2,value3,...)
  | SET(value1,value2,value3,...)
  | spatial_type

index_col_name:
    col_name [(length)] [ASC | DESC]

reference_definition:
    REFERENCES tbl_name [(index_col_name,...)]
               [MATCH FULL | MATCH PARTIAL]
               [ON DELETE reference_option]
               [ON UPDATE reference_option]

reference_option:
    RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT

table_options: table_option [table_option] ...

table_option:
    {ENGINE|TYPE} = {BDB|HEAP|ISAM|InnoDB|MERGE|MRG_MYISAM|MYISAM}
  | AUTO_INCREMENT = value
  | AVG_ROW_LENGTH = value
  | CHECKSUM = {0 | 1}
  | COMMENT = 'string'
  | MAX_ROWS = value
  | MIN_ROWS = value
  | PACK_KEYS = {0 | 1 | DEFAULT}
  | PASSWORD = 'string'
  | DELAY_KEY_WRITE = {0 | 1}
  | ROW_FORMAT = { DEFAULT | DYNAMIC | FIXED | COMPRESSED }
  | RAID_TYPE = { 1 | STRIPED | RAID0 }
        RAID_CHUNKS = value
        RAID_CHUNKSIZE = value
  | UNION = (tbl_name[,tbl_name]...)
  | INSERT_METHOD = { NO | FIRST | LAST }
  | DATA DIRECTORY = 'absolute path to directory'
  | INDEX DIRECTORY = 'absolute path to directory'
  | [DEFAULT] CHARACTER SET charset_name [COLLATE collation_name]

select_statement:
    [IGNORE | REPLACE] [AS] SELECT ...   (Some legal select statement)
```

1.1.2. Insertar dades.

```
INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
    [INTO] tbl_name [(col_name,...)]
    VALUES ({expr | DEFAULT},...),(...),...
    [ ON DUPLICATE KEY UPDATE col_name=expr, ... ]
```

```
INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
    [INTO] tbl_name
    SET col_name={expr | DEFAULT}, ...
    [ ON DUPLICATE KEY UPDATE col_name=expr, ... ]
```

```
INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
    [INTO] tbl_name [(col_name,...)]
    SELECT ...
```

`INSERT` inserts new rows into an existing table. The `INSERT ... VALUES` and `INSERT ... SET` forms of the statement insert rows based on explicitly specified values. The `INSERT ... SELECT` form inserts rows selected from another table or tables.

1.1.3. Consultar dades.

```
SELECT
    [ALL | DISTINCT | DISTINCTROW ]
      [HIGH_PRIORITY]
      [STRAIGHT_JOIN]
      [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
      [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr, ...
    [INTO OUTFILE 'file_name' export_options
      | INTO DUMPFILE 'file_name']
    [FROM table_references
      [WHERE where_definition]
      [GROUP BY {col_name | expr | position}
        [ASC | DESC], ... [WITH ROLLUP]]
      [HAVING where_definition]
      [ORDER BY {col_name | expr | position}
        [ASC | DESC] , ...]
      [LIMIT {[offset,] row_count | row_count OFFSET offset}]
      [PROCEDURE procedure_name(argument_list)]
      [FOR UPDATE | LOCK IN SHARE MODE]]
```

2. La estructura de les dades.

2.1. Persona.

El model relacional intenta imitar al màxim la informació que ofereix el món real. En el nostre cas podem parlar de “Persones”, “Documents”, “Historials” i “Visites” que són termes comuns en l’ambit del domini de l’ aplicació.

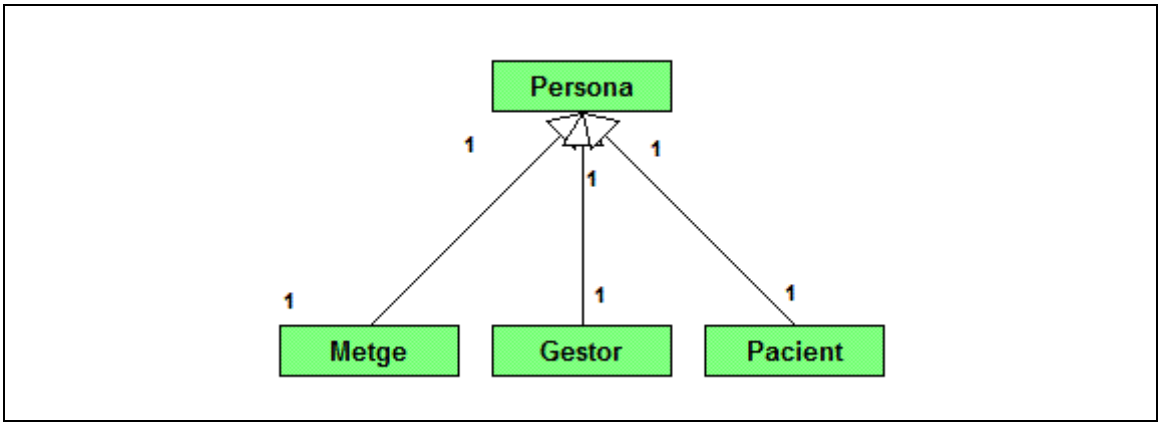


Figura 20. Identitat física.

Les persones del món real s’identifiquen, entre altres formes per les seves “Dades censals”, normalment amb el DNI, que es un número amb una serie de convencionalismes i d’assignació aleatoria.
Les “Persones” poden apareixer sota la forma de “Metge”, “Pacient” o “Gestor”, amb una relació 1:1, de manera que darrera un “Metge” només pot haver una “Persona” i viceversa. Però mentre que “Persona” és un concepte general, “Metge” és un concepte particular del domini “Món de la Medecina”.

Quan trobem una relació 1:1 surt la pregunta ¿Qué és millor, una taula o dues?
La presentació en dues taules dona més flexibilitat al model i permet afegir nous conceptes relacionats més amb la “Persona” en general o amb la activitat laboral. També dona facilitat a la compresió de la aplicació.

El Metge, que com a persona té una identitat (el DNI), en l'ambit de la seva activitat professional té una altra identitat (el número de col·legiat). En l'ambit de la aplicació electrónica també té una identitat específica (un certificat electrònic). En el cas del DNI hi ha la confiança en que aquest número és únic, però en el cas del certificat electrònic és general la situació de tenir-ne més d'un. (Vigents i caducats).

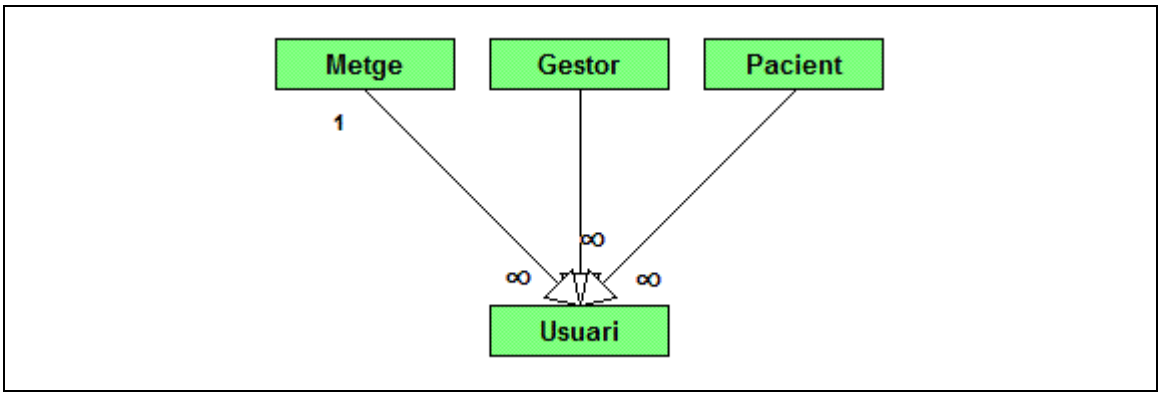


Figura 21. identitiat electrónica.

Per tant, a una identitat de "Metge" poden correspondre moltes identitats d'usuari electrónic, representat per un "certificat electrònic".

2.2. Historial.

La relació entre Pacient, Metge i Historial també pot representar-se de diferents maneres.

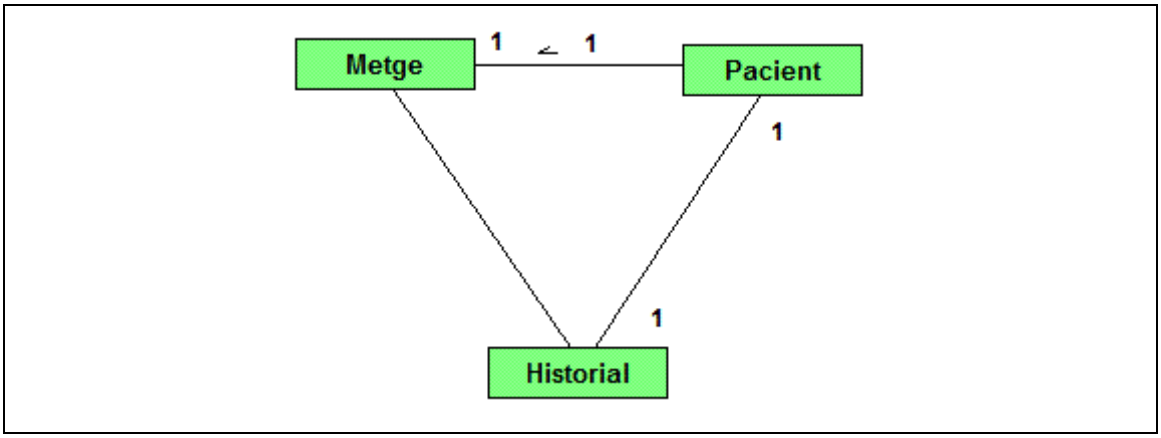


Figura. 22. Relacions

Des de el punt de vista del pacient, es pot considerar que la relació més importan és la que existeix entre Pacient i Historial. I que aquesta relació és 1:1, (" a un pacient ha de correspondre un historial, i només un "). També es pot supossar que la relació entre Pacient i Metge és una relació "rígida" de manera que "a un pacient sempre le correspondrà el mateix metge".

Des de el punt de vista del Metge, l'historial és "propietat" del metge que l'ha generat. I la relació amb el pacient es 1:molts.

A nosaltres ens interessa un model que sigui prou flexible per poder introduir modificacions davant de diferents plantejaments. Es propossa el següent model:

- la relació entre Pacient i Historial és 1: molts. Es a dir, un pacient pot tenir més d'un Historial (encara que la situació més correcta és que hi hagi només un historial, en casos especials es pot acceptar dos historials).
- La relació entre Pacient i Metge està controlada pel Gestor. Pren la forma de Assignació Metge_Pacient, amb data de caducitat. (Quan la relació entre Metge i Pacient caduca, el Gestor ha de crear una nova relació per a un nou periode. Per tant la tupla <metge, pacient> no pot ser clau primaria, perque poden apareixer repeticions).
- Finalment, la relació entre metge i historial serà indirecta, (primer cal consultar la identitat del pacient, abans de fer la consulta de l'historial).

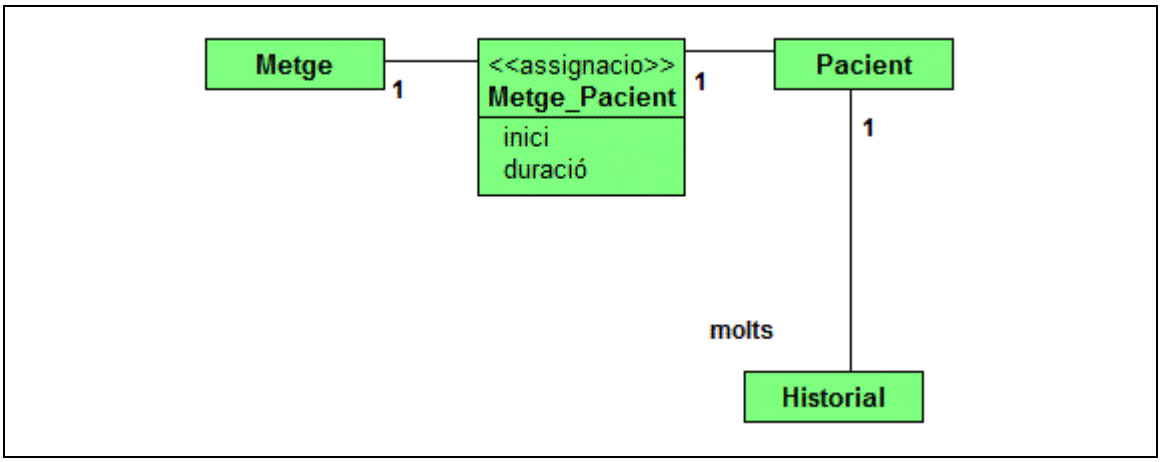


Figura 23. Relació metge-historial.

2.3. Visita.
En la figura seguent es mostra la relació entre l'historial i les visites médiques. Cal remarcar que l'objectiu es conservar secreta la identitat del pacient.

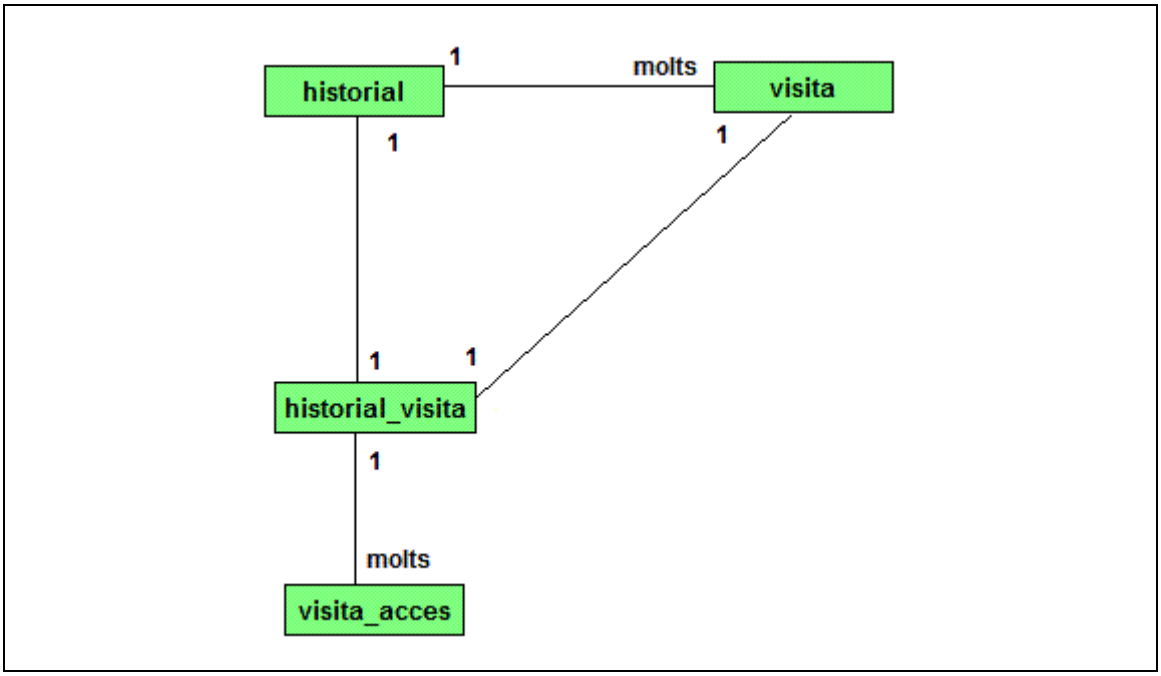


Figura 24. Relació historial – visita.

La taula historial_visita conté les tuples <historial, visitaEncriptada>. Però, el valor de "visita" es troba encriptat amb una clau simètrica. Es a dir, no es ben be una relació directa amb la taula visita. L'encriptat ho fa el Gestor, i res no l'impedeix de fer el xifratge diverses vegades. Per tant

podem tenir <historial, xifratge1>, <historial, xifratge2>... encara que, segons mostra la figura, l'ideal seria una relació 1:1, o sigui, només un xifratge.

La relació entre historial_visita i visita_acces conté un registre per cada usuari autoritzat a desxifrar la clau simètrica, <visita xifrada, usuari, clau xifrada>. El Gestor decideix qui pot desencriptar l'identificador de visita xifrada.

3. Taules - SQL

3.1. Taula persona.
Aquesta taula conté la informació de les "Dades censals" corresponents a una persona física.

CREATE TABLE IF NOT EXISTS persona(dni CHAR(20), nom CHARACTER VARYING (100), signedData BLOB, PRIMARY KEY(dni))"

3.2. Taula pacient.
Aquesta taula conté la informació de caracter sanitari corresponent a una persona física.

"CREATE TABLE IF NOT EXISTS pacient(dni CHAR(20), nom CHARACTER VARYING (100), signedData BLOB, PRIMARY KEY(dni))"

3.3. Taula metge.
Aquesta taula conté la informació de caracter profesional d'un metge.

"CREATE TABLE IF NOT EXISTS metge(dni CHAR(20), nom CHARACTER VARYING (100), signedData BLOB, PRIMARY KEY(dni))"

3.4. Taula gestor.
Aquesta taula conté la informació relativa al gestor.

"CREATE TABLE IF NOT EXISTS gestor (dni CHAR(20), nom CHARACTER VARYING(100), signedData BLOB, PRIMARY KEY(dni))"

3.5. Taula usuari.
Aquesta taula recollirà tota la informació referent als certificats de pacient, metges i gestor, es a dir, a la identitat electrònica dels usuaris.

"CREATE TABLE IF NOT EXISTS usuari (idUsuari INTEGER AUTO_INCREMENT, dni CHAR(20), numSerie CHAR(50) , tipus CHAR(20), signedData BLOB, PRIMARY KEY(idUsuari))"

Taula certificat.
Aquesta taula recollirà la informació dels certificats digitals que s'hagin utilitzat en qualsevol moment en el procès de signatura o xifrat de dades guardades a la base de dades. Els certificats poden caducar pel pas del temps i altres motius, i per tant cal identificar i conservar la informació dels certificats.

(Els certificats es guarden al directori: doc/pki)

La informació de la relació entre el metge i el pacient es guardarà dins de Historial, de manera secreta. Es un EnvelopedData amb la informació del metge i la data de caducitat.

3.6. Taula historial.
Aquesta taula conté la informació dels historials. La informació es guardarà de manera distribuida en tres taules:

"CREATE TABLE IF NOT EXISTS historialP(idHistorial INTEGER AUTO_INCREMENT, dni CHAR(20), signedData BLOB, PRIMARY KEY(idHistorial))"

"CREATE TABLE IF NOT EXISTS dadesGenerals (idHistorial INTEGER, dni CHAR(20), grupSanguini CHAR(10), allegies BLOB, signedData BLOB, PRIMARY KEY (idHistorial))"

“CREATE TABLE IF NOT EXISTS llistaVisites(idHistorial INTEGER, SignedAndEncryptedData BLOB , PRIMARY KEY(idHistorial))”

3.7. Taula visita.
Aquesta taula conté la informació relativa a una visita médica.

CREATE TABLE visita (idVisita CHAR (20), data DATE, hora TIME, tema CHAR VARYING (200), idMetge CHAR(20), anamnesi CHAR (200), diagnosi CHAR(200), tractament CHAR(200), signaturaMetge BLOB, numCertificat CHAR(20), PRIMARY KEY(idVisita))

La identificació d’un X509Certificate a partir del seu numSerie pot ser insegur. És més bona política utilitzar el FingerPrint per tal de fer una identificació segura d’un certificat.

4. Gestió d’informació.

4.1. Afegir informació.

INSERT INTO persona VALUES(‘0000000a’, ‘nom’, ‘010011...00..1.001’)

Persona		
NumDNI	nom	signedData
‘00000000a’	‘nom’	‘010011....00..1.001’

4.2. Consultar dades.

SELECT * FROM persona WHERE numDNI = ‘00000000a’

4.3. Modificar dades generals.

UPDATE persona SET carrer = ‘nou carrer’ WHERE numDNI = ‘00000000a’

Dades General				
Persona.dni	Persona.nom	Usuari.numSerie	Historial.allergies	Historial.grupsanguini
‘00000000a’	‘nom’	integer	‘llista d’allergies’	0

6.5. SQL i Java.

Les aplicacions Java utilitzen les clases Connection, DriverManager, ResultSet, Statement i SQLException del paquet java.sql.* per accedir a bases de dades en general.
Per tal que això sigui possible, es necesari disposar del connector java (mysql-connector-java-3.1.8a) instal·lat en el directori Java i haver preparat l’esquema, o base de dades que farem servir. Normalment això es fa amb l’usuari “root”, una sola vegada.

A continuació es presenta un ejemple de acces a una base de dades MySQL amb Java.

JDBCTest.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.sql.SQLException;

public class JDBCTest {

	public static void main(String args[]) {
		try {
			// Carregar driver de MySQL
			Class.forName("com.mysql.jdbc.Driver");
			System.out.println("Driver carregat");

			//Connectar-se a la base de dades "jdbc:mysql://host:port/database"

//Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/test","usuario","oirausu");
//la BD es diu “test” en el port 3306, la BD reconeix l’usuari “usuari” amb el password “oirausu”

		Connection conn = DriverManager.getConnection("jdbc:mysql://localhost/test","usuario","oirausu");

			// Aconseguir un "statement" de la connexio per poder realitzar operacions
			Statement stmt = conn.createStatement();
			// Crear una tabla
			try{
			stmt.execute("CREATE TABLE IF NOT EXISTS usuaris (nom CHAR(50), dni CHAR(9),
PRIMARY KEY (dni))");
			}catch(SQLException e){
				System.out.println("SQLException: CREATE: " + e);
			}

		// Insertar datos:
		try{
			stmt.execute("INSERT INTO usuaris VALUES (\"pascual\",\"xxxxxxxyK\")");
		}catch(SQLException e){
			System.out.println("SQLException: INSERT: " + e);
		}
		// Executar consulta
		try{
			ResultSet rs = stmt.executeQuery("SELECT * FROM usuaris");
			// Mostrar resultats
			while (rs.next()) {
			System.out.print("Nom: ");
			System.out.print(rs.getString("nom"));
			System.out.print(" - DNI: ");
			System.out.println(rs.getString("dni"));
		}
		}catch(SQLException sqle){
			System.out.println("SQLException: " + sqle);
		  //SQLException: com.mysql.jdbc.exceptions.MySQLSyntaxErrorException: Table 'test.usuaris'
doesn't exist
		  }
	} catch (SQLException e) {
		while (e != null) {
		System.out.println( "Estat   : " + e.getSQLState());
		System.out.println( "Missatge: " + e.getMessage());
		System.out.println( "Error   : " + e.getErrorCode());
		e = e.getNextException();
		}
	} catch (Exception e) {
		System.out.println(e);
	}
  }
}
```

Capítol 7. Vistes client.

AWT i API Swing de components gràfics

1. Generalitats referents a les aplicacions gràfiques.

Les interrupcions hardware.
La feina del microprocessador es redueix a llegir instruccions, decodificar-les i executar-les continuament. Però, de vegades aquest procès de treball es ve interromput per un succés extern. En el nostre cas, la majoria de les interrupcions provindran del mouse.

1.1. La programació d'events hardware en Java.
Java dispossa de dos paquets bàsics per a la programació gràfica d'interficies d'usuari aprofitant el mecanisme de les interrupcions hardware.

Java.awt.*;
Java.awt.event.*;

1.2.- Proposta per integrar la programació d'interrupcions hardware dins una aplicació clàsica.

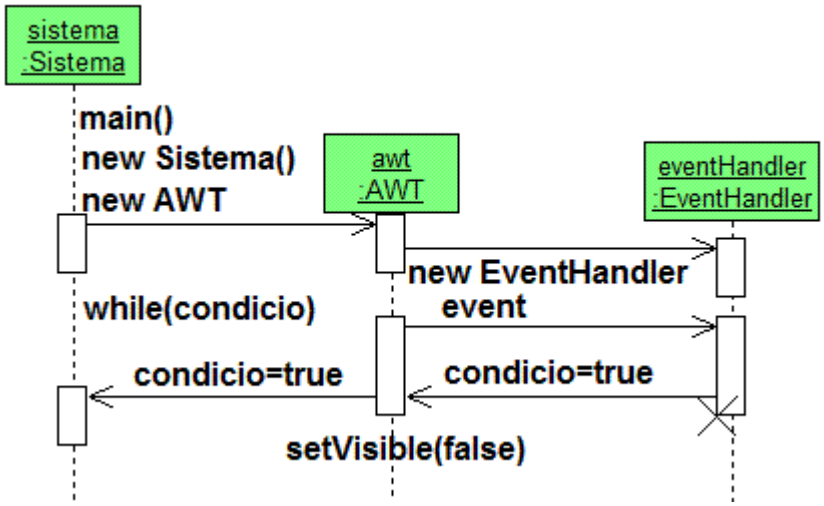


Figura 25 . Esquema general de control de fluxe amb captura de Events.

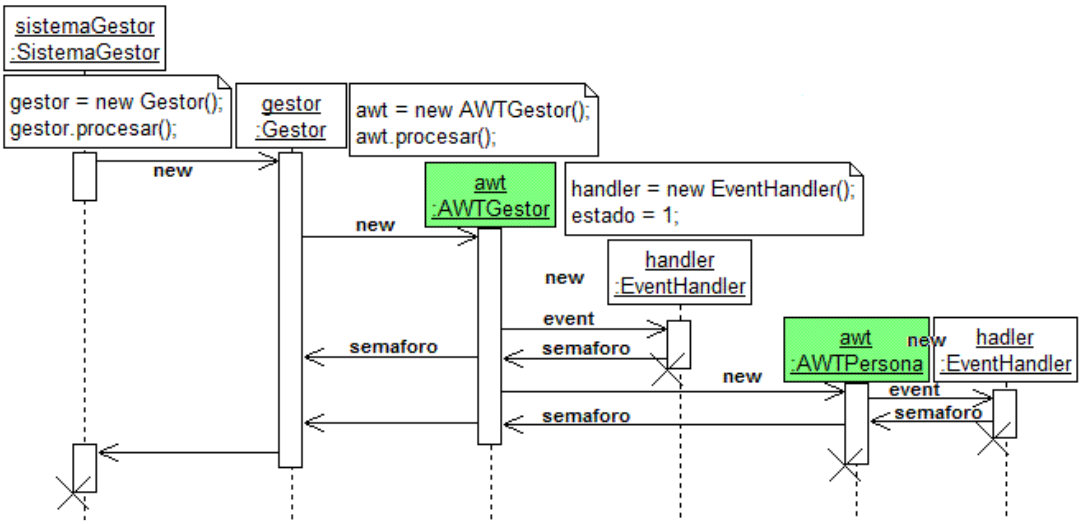


Figura 26. Esquema per al cas de la entrada de dades personals.

En la figura anterior es pot veure la situació que es planteja quan tenim un menú inicial (finestra del gestor) que permet navegar cap a unes altres funcions (finestra de entrada de dades personals).
L'objectiu d'aquest esquema tan complicat es evitar fer crides a métodes des de dins d'un gestor d'events per tal de mantenir el flux d'instruccions sota control. La variable "estado" permet retornar a la classe gràfica AWTGestor i des d'aquesta obrir una altra finestra: AWTPersona.

1.2.1. Funció login.
L'objectiu d'aquesta funció és fer la identificació segura (autenticacio) de la persona que vol utilitzar la aplicació. La aplicació presenta una Finestra amb dos camps de text per tal que l'usuari pugui introduir el seu identificador i una paraula password. Amb aquesta informació la aplicació intentarà obrir el fitxer PKCS#12 corresponent.
Hem de suposar que la persona que coneix el nom i el password del fitxer .p12 es un usuari autoritzat a fer servir la aplicació.
La funció login es el primer pas en les tres interficies d'usuari que se presenten: la interficie del gestor, la del metge i la del pacient. (ver SistemaGestor.java, SistemaMetge.java i SistemaPacient.java).

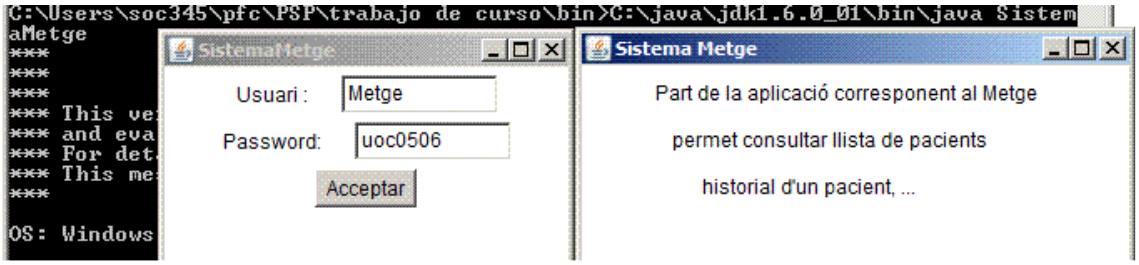


Figura 27. Funció login per Metge.p12

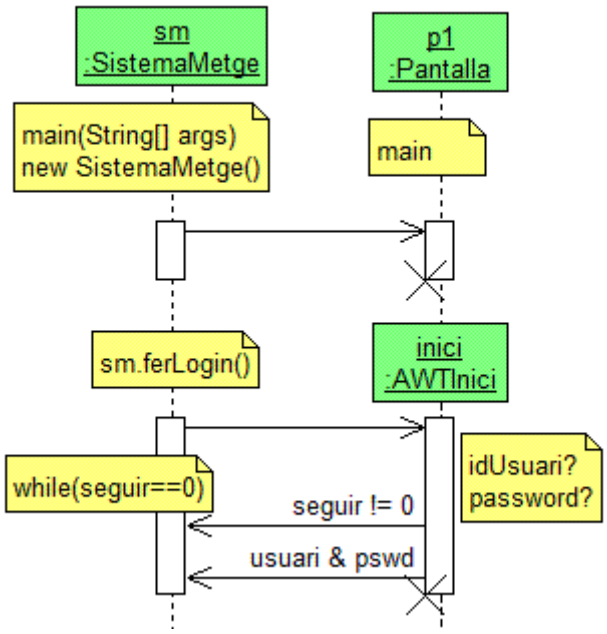


Figura 28. Esquema de programació d'interrupcions hardware.

En la figura anterior podem observar una classe SistemaMetge amb un método main(String[] args) (la programació clàssica) que genera una instancia de la classe AWTInici que utilitza java.awt.*; i java.awt.event.*;
AWTInici inici = new AWTInici();

La class AWTInici conté les variables usuari i password accessibles desde SistemaMetge que obtenen el seu valor a partir dels TextField equivalents id i pswd, a partir de la gestio d'un Event mouse generat sobre l'objecte Button.

També hi ha la variable semaforo que permet la coordinació entre el fluxe clàssic de la classe SistemaMetge i el fluxe controlat per interrupcions hardware de la classe AWTInici.
Mentre la persona usuaria de la aplicació es pren el seu temps per decidir quines dades introduirà en els camps TextField, la aplicació SistemaMetge roman bloquejada consultant el valor de la variable semaforo.
Quan l'usuari prem el Button Acceptar es genera una interrupció hardware que actualitza les dades de las variables usuari, password i semaforo, i SistemaMetge s'activa i recull la informació continguda en AWTInici.
(Sense aquest artifici la aplicació SistemaMetge generaria una instància de AWTInici i se'n oblidaria, continuant el seu fluxe fins al final.)

2. La interficie Pacient.
2.1. Consulta dades generals.
2.2. Consulta visita.
3. Interficie Metge.
3.1. Consulta dades generals.
Menú amb les funcions generals:

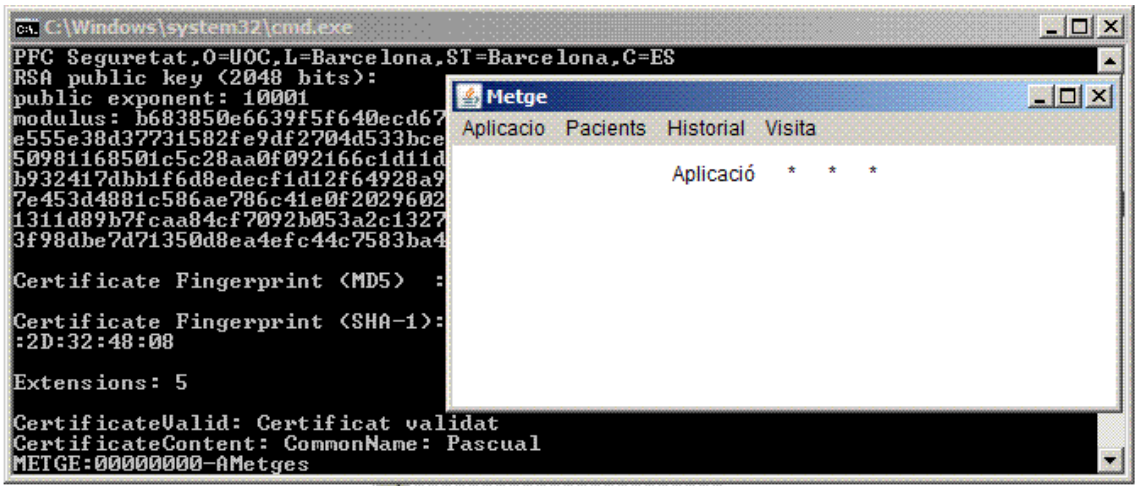


Figura 29. Pantalla bàsica de gestió de dades per al metge.

Funció selecció d'un pacient:

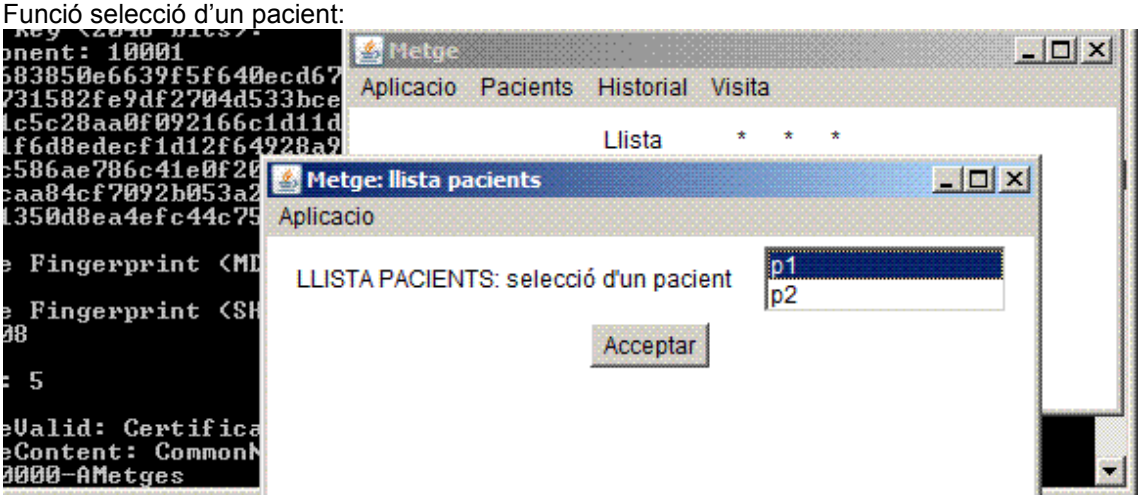


Figura 30. Pantalla de selecció d'un pacient.

3.2. Historial, consulta i edició de dades médiques.
3.3. Visites.

Capítol 8. Vista gestor.

1. Funció login: autenticació.

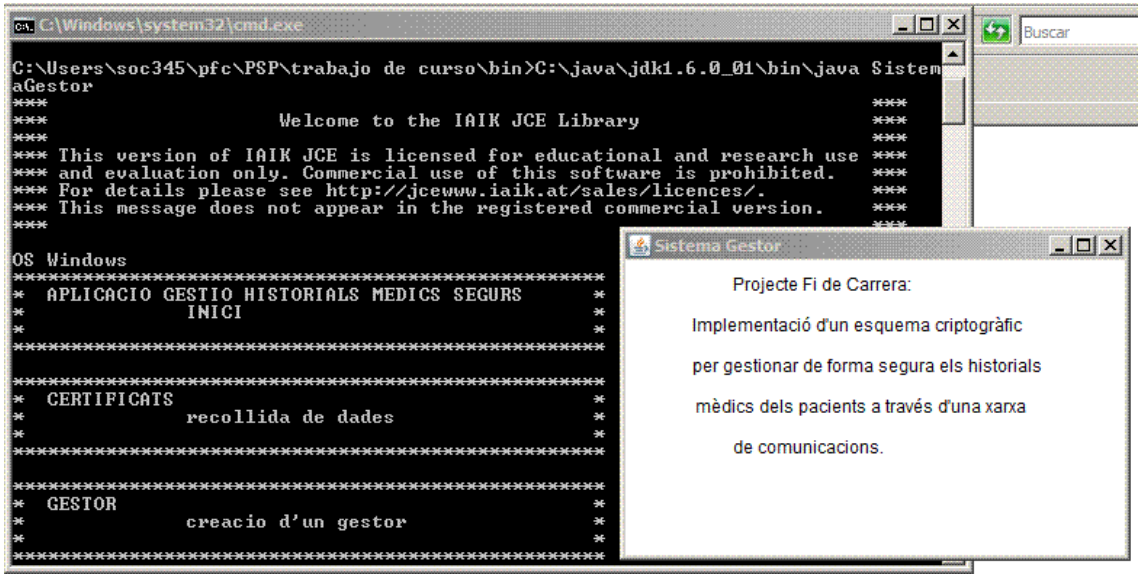


Figura 31. Pantalla de benvinguda amb informació general.

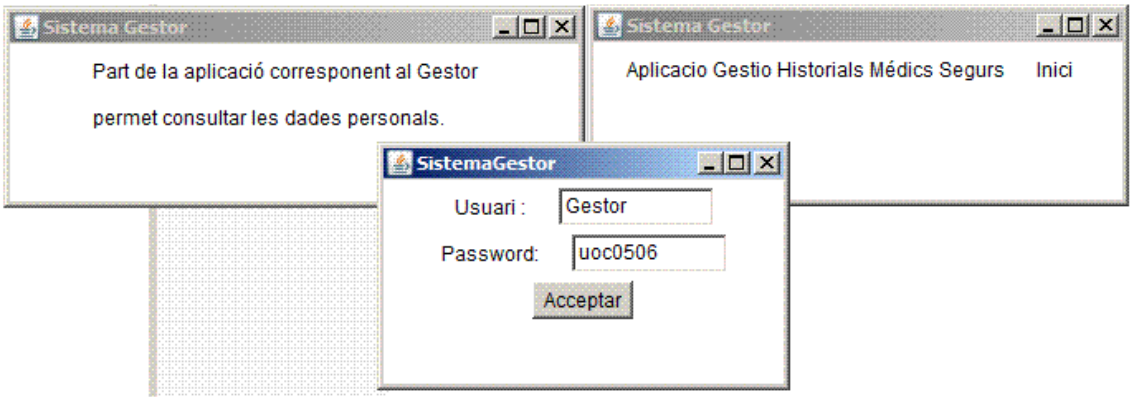


Figura 32. Funció login per al gestor: finestra d'entrada de dades.

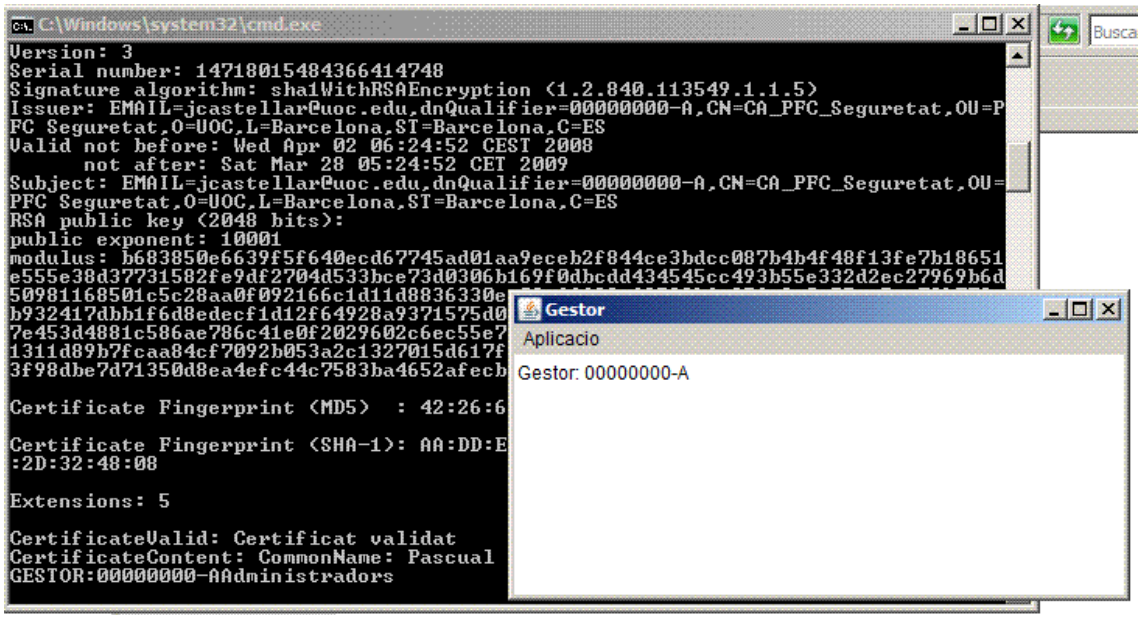


Figura 33. Funció login per al gestor: comprobació de identitat electrónica.

2. Gestió de dades de Persona i Usuari.
Funció Alta de persones autoritzades (identitat física).
La finestra de entrada de dades personals permet al gestor de registrar la identitat física d'una persona. La identificació física, a diferencia de la identificació electronica, només es pot fer de manera presencial, per la qual cosa, mai delegarem aquesta funcionalitat al propi pacient.
Per contra, el pacient i el metge, sí poden crear un PKCS#10 per fer una identificació electrónica remota.

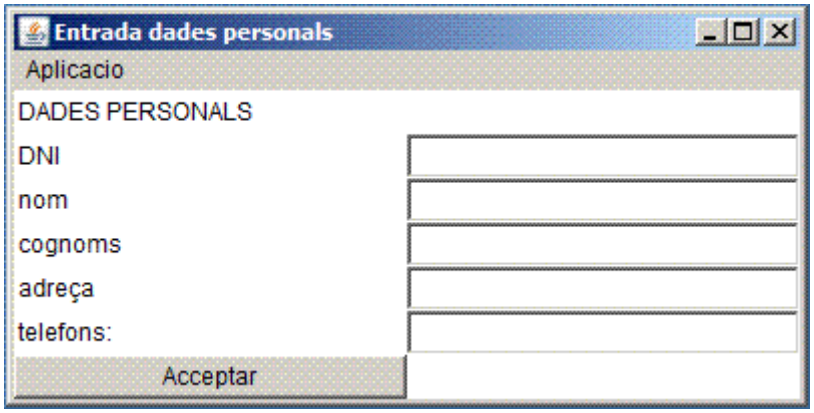


Figura 34. Finestra de entrada de dades censals.

Funcio Alta d'usuaris (identitat electrónica).
Quan el sistema té identificada una persona física, es pot procedir a incorporar les dades d'un certificat d'usuari. Aquest certificat és la identitat electrónica, i el sistema accepta una persona amb més d'un certificat.

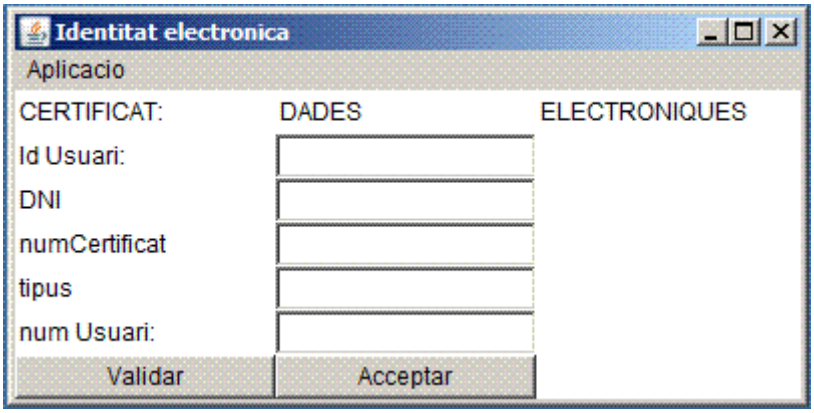


Figura 35. Finestra de entrada de identitats electróniques.

3. Gestió de dades d'Historial.
3.1. Dades generals.
3.2. Assignació metge-pacient.

Capítol penúltim amb la valoració econòmica, si s'escau al tipus de projecte. (Sense desenvolupar).

Capítol 9 i últim amb les conclusions.

Encara que el tema del projecte està molt lligat a la aplicació pràctica de conceptes criptogràfics, la seva implementació ha necessitat coneixements de moltes arees estudiades al llarg de la carrera. Han estat els problemes relacionats amb bases de dades i comunicació remota els que han requerit més esforç per trobar possibles solucions i els que han retardat més la consecució d'objectius. En particular cal mencionar la complicació presentada per a solucionar la utilització del tipus BLOB en MySQL, necessari per guardar informació de matrius de byte. També la manca de familiaritat amb RMI s'ha fet patent.

Per altra banda, cal mencionar que l'estudi del paquet IAIK ha estat molt engrescador, encara que només al final del temps del projecte he pogut arribar a tenir una idea clara de les possibilitat que ofereix.

El projecte ha estat una tasca agradable, des de el començament al març quan varem rebre l'enunciat, fins al dia d'avui, (11 de juny) que enllesteixo la memòria, i voldria tenir una mica més de temps per tal de poder arrodonir el codi que presento. En particular, vull mencionar que ha quedat al tinter la pràctica amb objectes serialitzables i rmi.

Agraeixo a l'area de PFC-Seguretat Informàtica com ha estat plantejada aquesta asignatura i a Jordi Castellà Roca la seva dedicació i suport.

Glossari.

ASN.1 Abstract Syntax Notation One, Notació de sintaxi abstracta normalitzada per ISO i ITU-T

atac estratègia o mètode que té per objectiu descobrir la clau de xifratge o bé el text en clar.

atac a sistemes informàtics explotació de les vulnerabilitats dels sistemes

atac criptoanalític explotació de les febleses dels algorismes de xifra

autenticació identificació fiable d'un emissor.

autenticitat propietat de la informació que no ha estat modificada de manera fraudulenta

autentificació comprovació de la autenticitat d'una informació. També és una forma de identificació d'usuaris amb criptografia.

autoritat de certificació tercera part fiable, que acredita l'autenticitat dels usuaris

CA Ver autoritat certificadora.

canal insegur és el canal de comunicació que no pot garantir la privadessa de la comunicació. (ni la integritat)

CBC modo de treball de DES (cipher bloc chaining)

certificat document electrònic amb la clau pública d'un usuari, amb la garantía d'una autoritat de certificació

certificat autosignat

certificat autosignat. Crear openssl req -new -sha1 -x509 -key CA.key -out CA.crt -days 360

certificat de CA és el certificat arrel d'una cadena de confiança.

CFB modo de treball de DES (cipher feedback)

clau paràmetre que controla els processos de xifratge i/o desxifratge

clau asimétrica ver clau pública

clau pública sistema criptografic que aprofita les propietats matemàtiques dels nombres primers.

clau secreta clau de xifratge/desxifratge protegida de consultes no autoritzades

clau simétrica la mateixa clau permet fer funcions de encriptat i desencriptat.

confidencialitat és la propietat de la comunicació que permet garantir només hi ha dos actors: l'emissor i el receptor.

criptoanàlisi ciència i estudi de l'escriptura secreta. s'ocupa de trencar xifres, descobrir la clau o el text en clar. De fet es la ciència que estudia la protecció de la informació

criptograma text xifrat

criptosistema xifra

criptosistema de clau compartida l'emissor i el receptor comparteixen una mateixa clau per a xifrar i desxifrar missatges

criptosistema de clau pública cada usuari té una clau pública i una de privada

criptosistema de xifratge de bloc és un tipus de criptosistema de clau compartida

criptosistema DES és un criptosistema de xifratge de bloc que xifra blocs de dades de 64 bits de llargada

criptosistema RSA (ver RSA)

dades especialment protegides les que gaudeixen de protecció constitucional.

DER Distinguished Encoding Rules, Regles de codificació distintives

DES algorisme de criptografia simetrica . Data Encryption Standard

desxifratge procés de transformació del text xifrat en text en clar

disociació de dades tractament de dades personals de manera que la informació que s'obtingui no pugui ser associada a una persona indentificable.

disociació de dades les dades personals (la identitat del pacient) es conserven a part de les dades de salut.

encapsular procediment de protecció de dades. S'aplica a certificats, claus, parelles de claus...

encapsular(parella de claus, certificat, certificat de CA) : PKCS#12

esquema criptografic conjunt d'algurismes que permeten garantir la seguretat de les dades

falsificació atac criptogràfic

gestor.parella de claus openssl genrsa -des3 -out Gestor.key 1024

gestor.petició de certificat openssl req -new -sha1 -key Gestor.key -out Gestor.csr -config openssl.conf

hash función que dóna com a sortida un resum, de longitud fixa, d'un missatge d'entrada de longitud variable.
hash unidireccional funció hash segura
IAIK paquet Java amb utilitats criptogràfiques.
integritat propietat de la informació que no ha sofert modificacions o supressions parcials no autoritzades
intercanvi de claus és el procés en que la clau pasa de l'emissor al receptor. És el punt feble de la comunicació.
llargada de claués un paràmetre que determina la resitencia de la clau als atacs de força bruta
OFB modo de treball de DES
parella de claus
parella de claus RSA
parella de claus. Cifra(triple des)
parella de claus. Crear = openssl genrsa -des3 -out CA.key 2048
parella de claus. Longitud
parella de claus. Password uoc0506
petició de certificat
pkcs#12 és un objecte criptogràfic protegit amb password per conservar claus PrivateKey i PublicKey
pkcs7-digestedData dades resumides amb una funció hash
pkcs7-encryptedData dades encriptades amb una clau simetrica
pkcs7-envelopedData dades (normalment una clau) encriptades amb una clau PublicKey
pkcs7-signedAndEnvelopedData dades signades i encriptades
pkcs7-signedData dades signades amb una PrivateKey
PKI
protocol criptogràfic
protocol criptogràfic
requisits de seguretat
RSA criptosistema de clau pública, basat en el problema de la factorització.
servei de no repudi l'emissor no pot negar l'autoria d'un text.
Signatura digital procediment per "signar" documents en format electrònic fent us de criptosistemes de clau pública
test en clar test que no ha estat xifrat
test xifrat test modificat d'acord a un algoritme, per tal que no pugui ser llegit per persones no autoritzades
Triple DES protocol de xifratge triple que utilitza el DES (criptografia simétrica)
VER Basic Encoding Rules, Regles bàsiques de codificació
xifra algoritme de transformació de textos en clar en textos xifrats
xifra de substitució xifra basada a modificar els elements d'un text, d'un en un. Cada element es "tradueix" sempre en un mateix element xifrat
xifra de transposició xifra basada a modificar l'ubicació que ocupen dos elements dintre d'un text.
xifratge procés de transformació d'un text en clar en un text xifrat.

Bibliografia.

Aleix Dorca Josa: Esquema criptogràfic per exàmens electrònics segurs. PFC. UOC 2005
Josep Lluis Ribas Gracia: Sistema de joc electrònic remot de BlackJack amb seguretat similar a la dels casinos tradicionals PFC. UOC - 2006
Pep Bisquert Illa: Esquema criptogràfic Joc electrònic remot segur POQUER. PFC. UOC 2005
Josep Domingo Ferrer (coordinador): Criptografia. UOC 1999
Jordi Herrera Joancomartí (coordinador): Seguretat en Xarxes de Computadors. UOC 2004
Jordi Castellà Roca: apunts de criptografia, Any 2002 (sense editar).
Xavier Perramon Tornil (coordinador): Sistemes de comunicacions. UOC 2000
Cristobal Romero Morales: Curso Básico de Java 2. Universidad de Córdoba y Obra Social y cultural de CajaSur 2002
Oscar Gonzalez: XML. Ediciones Anaya Multimedia. 2005.
Juan Diego Gutierrez Gallardo: XML. Ediciones Anaya Multimedia. 2005
Ian f. Darwin: Curso de Java. Ediciones Anaya Multimedia. Madrid 2005
Jaume Sistac Planas: Bases de dades I. UOC 1999
Jon Beltrán de Heredia: Lenguaje Ensamblador de los 80x86. Anaya Multimedia. Madrid 1995

Documents electrònics:
http://www.programacion.com/java/tutorial/rmi/2/
http://www.programacion.com/java/tutorial/rmi/4/
MySQL: MySQL Query Browser: The Information Browser: SQL Statement Syntax. (mysqlqb_statements.html). 2005

Annexos.

1. La llibreria IAIK

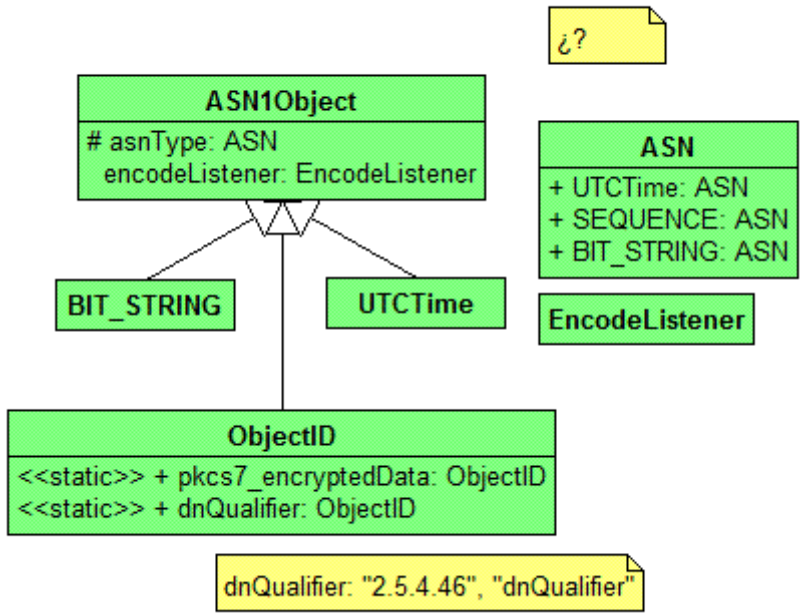
1.1 La llibreria IAIK està formada pels següents directoris:
asn1,
asn1/structures
jce
jce/com
jce/com/fourthpass
pkcs
pkcs/pkcs1
pkcs/pkcs10
pkcs/pkcs12
pkcs/pkcs5
pkcs/pkcs7
pkcs/pkcs8
pkcs/pkcs9
security
security/cipher
security/dh
security/dsa
security/keystore
security/mac
security/md
security/pbe
security/provider
security/random
security/rsa
security/spec
utils
x509
x509/attr
x509/extensions
x509/ocsp
x509/ocsp/extensions
x509/ocsp/net/application
x509/ocsp/utils
x509/qualified.

1.2. Les estructures de dades ASN1 i la codificació DER.
iaik.asn1.ASN: conté els tipus de dades primitius de la sintaxi ASN.
iaik.asn1.ASN1: permet treballar amb ASN1Object, amb InputStream i amb byte[]
iaik.asn1.ASN1Object: (no té constructors públics).
Iaik.asn1.ConstructedType extends ASN1Object
Iaik.asn1.SEQUENCE extends ConstructedType
Iaik.asn1.ObjectID: aquesta classe contè les definicions de totes les estructures que podem necessitar.
Iaik.asn1.DerCoder: permet treballar amb InputStream i amb OutputStream.
Iaik.asn1.structures.Name: (implements ASN1Type, Principal).

- java.lang.Object
  - iaik.asn1.**ASN** (implements java.lang.Cloneable)
  - iaik.asn1.**ASN1**
  - iaik.asn1.**ASN1Object**
    - iaik.asn1.ASN1String
      - iaik.asn1.**BMPString**
      - iaik.asn1.**GeneralString**
      - iaik.asn1.**IA5String**

        - iaik.asn1.**T61String**
        - iaik.asn1.**UNIString**
        - iaik.asn1.**UTF8String**
        - iaik.asn1.**VisibleString**
    - iaik.asn1.**BIT_STRING**
    - iaik.asn1.**BOOLEAN**
    - iaik.asn1.**ConstructedType**
        - iaik.asn1.**CON_SPEC**
        - iaik.asn1.**SEQUENCE**
        - iaik.asn1.**SET**
        - iaik.asn1.**UNKNOWN**
    - iaik.asn1.**ENUMERATED**
    - iaik.asn1.**GeneralizedTime**
    - iaik.asn1.**INTEGER**
    - iaik.asn1.**NULL**
    - iaik.asn1.**ObjectID**
    - iaik.asn1.**UTCTime**
- iaik.asn1.**DerCoder**
- java.io.InputStream (implements java.io.Closeable)
    - iaik.asn1.**DerInputStream**
- java.lang.Throwable (implements java.io.Serializable)
    - java.lang.Exception
        - iaik.asn1.**CodingException**

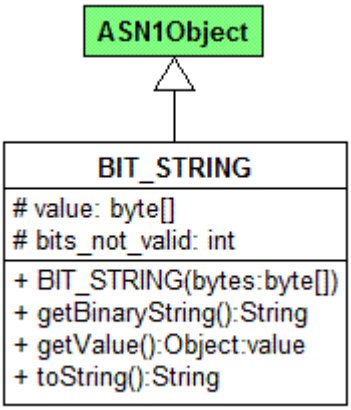
ASN1Object



Key

```
SubjectPublicKeyInfo ::= SEQUENCE {
  algorithm AlgorithmIdentifier,
  subjectPublicKey BIT STRING }

 AlgorithmIdentifier ::= SEQUENCE {
  algorithm OBJECT IDENTIFIER,
  parameters ANY DEFINED BY algorithm OPTIONAL }
```

BIT_STRING

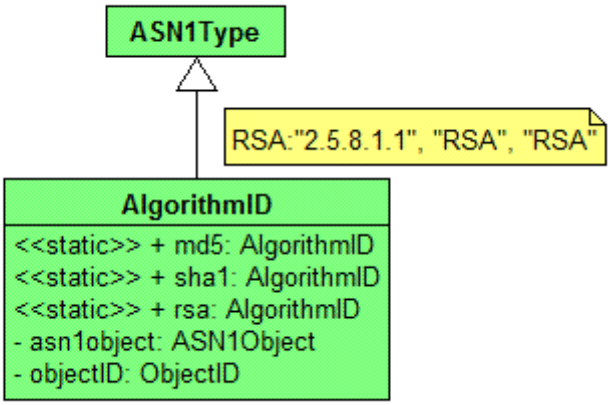


BIT_STRINGTest.java

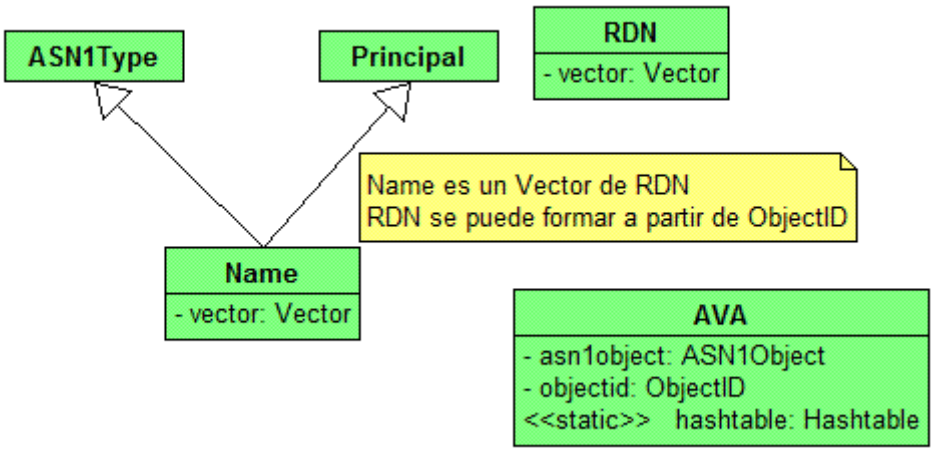
```
import iaik.asn1.BIT_STRING;

public class BIT_STRINGTest{
	public static void main(String[] args){
		BIT_STRING bs = new BIT_STRING("BIT_STRING".getBytes());
		System.out.println(bs.getBinaryString()); //secuencia de 0 y 1
		System.out.println(bs.toString());//salida formateada: "BIT STRING = "..."byte(s);"..."bit(s) not valid"
		byte[] b = (byte[]) bs.getValue(); //byte[]   ES EL VALOR ORIGINAL
		//comprobació:
		for(int i=0; i<b.length; i++){
			int c = b[i];
			System.out.println((char) c);
		}
		//Transformació a Códi DER
		byte[] derb = DerCoder.encode(bs);
		//objectes DER: tipo , longitud, contenido
		for(int i=0; i<derb.length; i++){
			int c = derb[i];
			System.out.print(c);
		}
		//Transformació a codi ASN
		try{
			ASN1Object asn = DerCoder.decode(derb);
			BIT_STRING bs2 = (BIT_STRING) DerCoder.decode(derb);
			//Comprobacio:
```

```
                    System.out.println(bs2.toString());
                }catch(CodingException ce){System.out.println(ce);}
            }
}
```

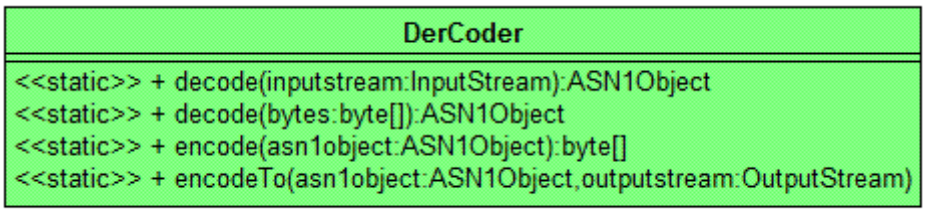
AlgorithmID



Name



DerCoder



1.3. Els pkcs i els certificats.

PKCS8
EncryptedPrivateKeyInfo

PrivateKeyInfo
Variables de clase: AlgorithmID, ASN1, int

PKCS12

PKCS12

Class PKCS12
java.lang.Object
└ **iaik.pkcs.pkcs12.PKCS12**

Class AuthenticatedSafe
java.lang.Object
└ **iaik.pkcs.pkcs12.AuthenticatedSafe**
All Implemented Interfaces:
iaik.asn1.ASN1Type

Class KeyBag
java.lang.Object
└ iaik.pkcs.pkcs12.Attributes
└ iaik.pkcs.pkcs12.SafeBag
└ **iaik.pkcs.pkcs12.KeyBag**
All Implemented Interfaces:
iaik.asn1.ASN1Type
Direct Known Subclasses:
PKCS8ShroudedKeyBag

Un PKCS12 es pot obtenir a partir d'un fitxer "pkcs12.p12"

```
String fileName = "pkcs12.p12.crt";
FileInputStream fis = new FileInputStream(fileName);
PKCS12 pkcs12 = new PKCS12(fis);
```

PKCS12 ofereix els següents mètodes:

P12.verify(psw.toCharArray()); // per verificar la integritat

P12.decrypt(psw.toCharArray()); // PKCS12 utilitza els mètodes de la variable AuthenticatedSafe per tal de arribar a tenir un SafeBag utilitzable.

P12.getKeyBag();

KeyBag ofereix els següents mètodes:

PrivateKey Susuari = kbag.getPrivateKey();
Byte[] localKeyId = Kbag.getLocalKeyID();
String slocalKeyId = new String(localKeyId);

//Public Key
//El locakKeyID ens vincula la clau privada amb la publica. Obtenim el localKeyID de la clau privada.
//A continuacio cerquem a la llista de certificats el certificat que te el mateix localKeyID.

Certificat

Class X509Certificate
java.lang.Object
└ java.security.cert.Certificate
└ java.security.cert.X509Certificate
└ **iaik.x509.X509Certificate**
All Implemented Interfaces:
iaik.asn1.ASN1Type, java.io.Serializable, java.security.cert.X509Extension
Direct Known Subclasses:
QualifiedCertificate

Un Certificat es pot obtenir amb:

String fileName = “Certificat.crt”;
FileInputStream fis = new FileInputStream(fileName);
X509Certificate x509 = new X509Certificate(fis);

Ofereix els següents mètodes:

c.getEncoded(): byte[]
c.getFingerPrint(): byte[]
c.getIssuerDN(): java.security.Principal (ver Name)
c.getSubjectDN(): java.security.Principal
c.getNotAfter(): java.util.Date
c.getPublicKey(): java.security.PublicKey
c.getSignature(): byte[]

Name

Class Name
java.lang.Object
└ **iaik.asn1.structures.Name**
All Implemented Interfaces:
iaik.asn1.ASN1Type, java.security.Principal

Un Name es pot obtenir a partir d’un X509Certificate:

Name issuer = (Name) _x509.getIssuerDN();
Name subject = (Name)_x509.getSubjectDN();

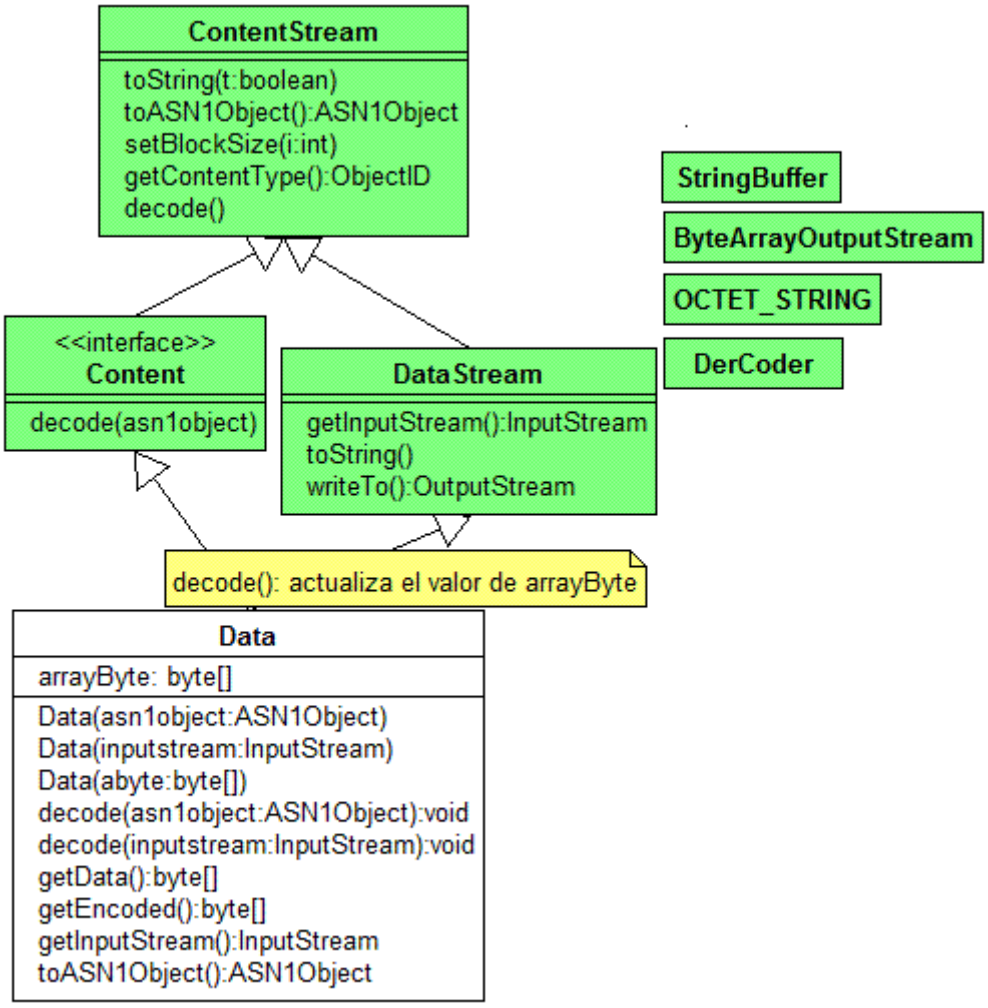
Ofereix els següents mètodes:

```
String id_usuari = n.getRDN(ObjectID.dnQualifier);
String tipus_usuari = n.getRDN(ObjectID.organizationalUnit);
```

PKCS7

Pkcs7-Data
Conté dades sense cap criptografia aplicada.

Data

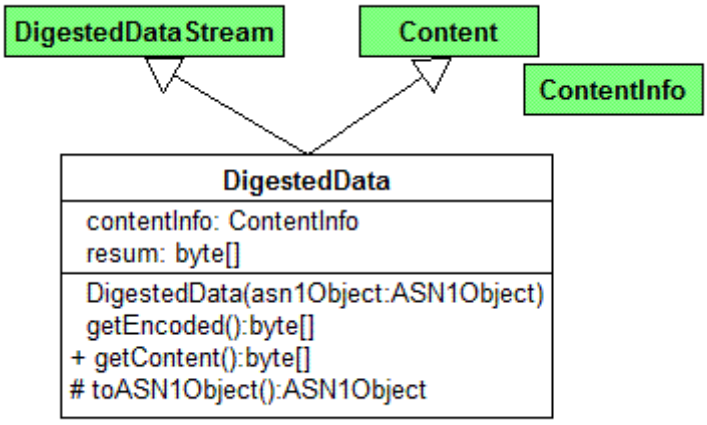


DataTest.java

```
import iaik.pkcs.pkcs7.Data;
import iaik.asn1.IA5String;
import iaik.asn1.DerCoder;


public class DataTest{
	public static void main(String[] args){
		IA5String ia5 = new IA5String("sequencia ASCII");

		try{
			byte[] b = DerCoder.encode(ia5);
			Data data = new Data(b);
			byte[] b2 = data.getData();
			for(int i=0; i<b2.length; i++){
				int c = b2[i];
				System.out.println("-"+i+" "+c);
			}
			byte[] b3 = data.getEncoded();
			for(int i=0; i<b3.length; i++){
				int c = b3[i];
				System.out.println("-"+i+" "+c);
				if(c == 4) System.out.println("OCTET STRING");
				if(c == 22) System.out.println("IA5String");
			}
		}
		catch(Exception e){System.out.println(e);}
	}
}
```

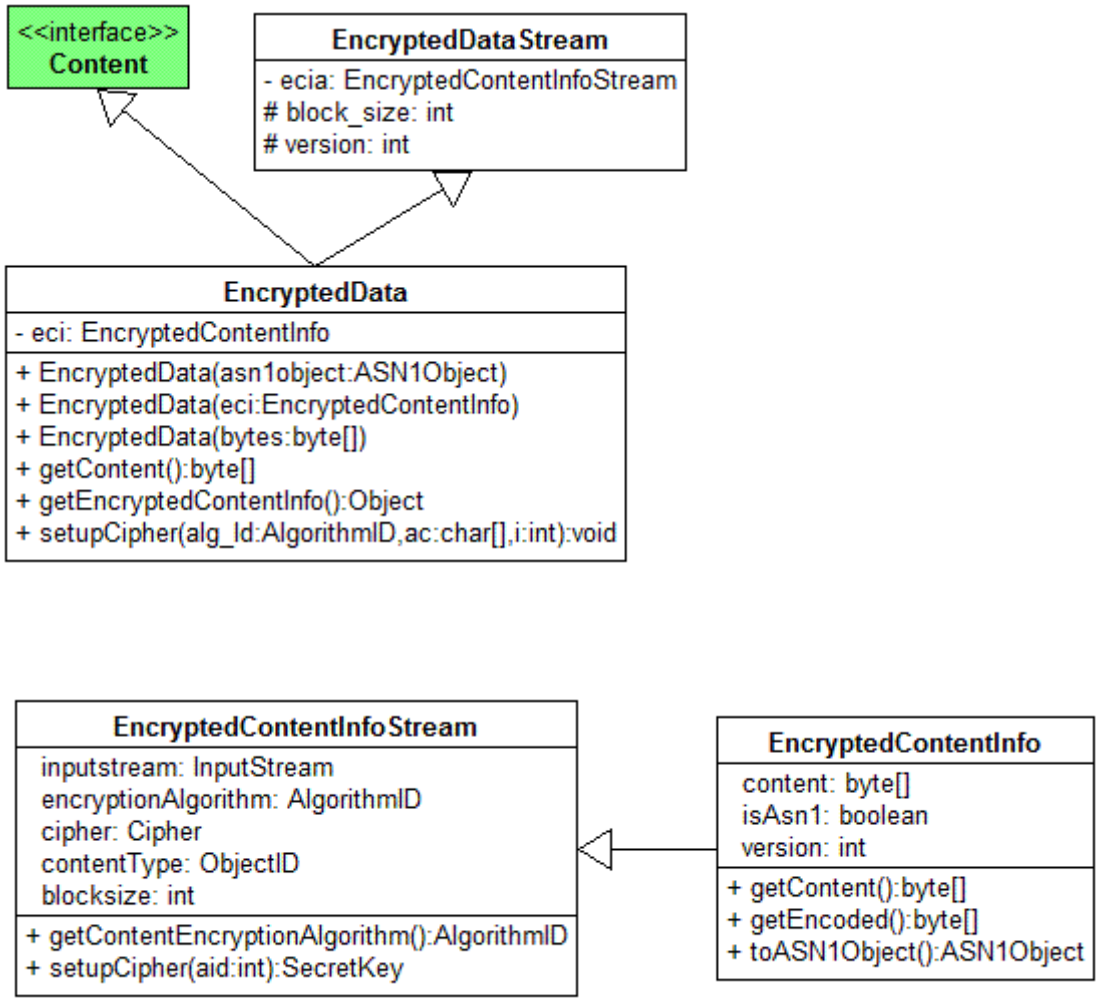
Pkcs7-digestedData
Conté dades amb un un resum.

DigestedData



Pkcs7-encryptedData
Conté dades xifrades simètricament.
Totes les funcions de encryptat de dades es fan de manera transparent per al programador, de manera que no cal ni generar, ni preocupar-se de guardar la clau secreta. (Resta amagada dintre de l'objecte ASN1 EncryptedData).

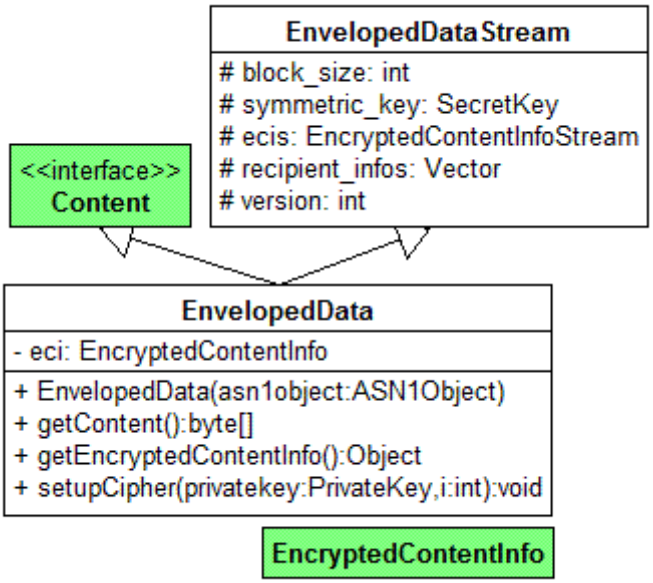
EncryptedData



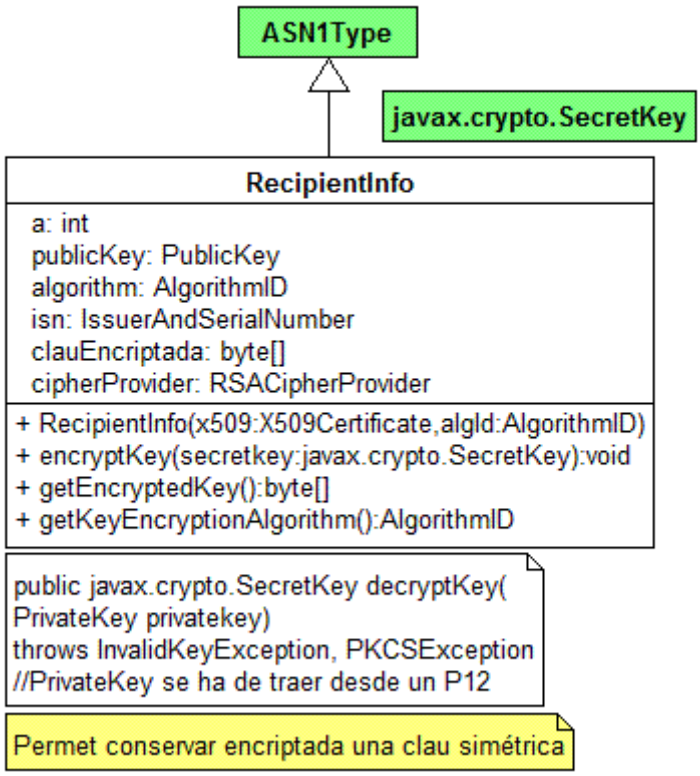
La utilització d'un objecte EncryptedData ens permet oblidar-nos de la clau secreta. Aquesta clau es genera internament en la clase i no es pot reutilitzar. En el cas que volguesim encryptar un mateix missatge i que es pogues obrir per més d'una persona, ens faria falta incluir dintre de X509Certificate[] _chain els certificats de totes les persones que poden llegir el contingut.

Pkcs7-envelopedData
Conté dades signades i xifrades. Té com a contingut un altre PKCS#7 encapsulat del tipus pkcs7-signedData.
A EnvelopedData se li poden afegir els RecipientInfo amb la clau pública dels usuaris que podran desxifrar la clau simetrica, (podran desxifrar el contingut).

EnvelopedData.



RecipientInfo



EnvelopedDataTest.java

```
import iaik.security.provider.IAIK;
import javax.crypto.SecretKey;
import java.security.*;

import iaik.pkcs.pkcs7.EnvelopedData;
import iaik.pkcs.pkcs7.EnvelopedDataStream;
import iaik.asn1.structures.AlgorithmID;
import java.security.NoSuchAlgorithmException;
import iaik.pkcs.pkcs7.EncryptedContentInfoStream;
import iaik.pkcs.pkcs7.EncryptedContentInfo;
import iaik.pkcs.PKCSException;
import java.io.IOException;
import java.io.ByteArrayInputStream;

import iaik.pkcs.pkcs7.RecipientInfo;
import iaik.pkcs.pkcs7.RSACipherProvider;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import iaik.x509.X509Certificate;
import java.security.cert.CertificateException;
import iaik.asn1.ASN1Object;
import iaik.asn1.CodingException;
import iaik.asn1.DerCoder;
import iaik.asn1.BIT_STRING;

/**
        En origen:
        fan falta les dades en clar i una clau simetrica
        (la clau simetrica es pot obtenir a partir de AlgorithmID)
        ->encriptem les dades en clar amb la clau simetrica

        fa falta la clau pública (el x509) del destinatari de les dades encriptades
        ->encriptem la clau simetrica amb la clau publica del destinatari
        guardem les dades encriptades (no pas les en clar)
        guardem la clau simetrica encriptada (però no la que es en clar)
        En destinació:
        ----------------------------: fa falta la clau privada del destinatari per desencryptar la clau simétrica
        ----------------------------: amb la clau simetrica desencripten les dades.
        SEQUENCE{
                contentType: pkcs-envelopedData
                content: SEQUENCE{
                        version: v1
                        recipientInfos: SEQUENCE OF{
                                recipientInfo: SEQUENCE{
                                        version: v1
                                        issuerAndSerialNumber: SEQUENCE{
                                                issuer: CN = root, O = UOC
                                                serialNumber: 5
                                                }
                                        keyEncryptionAlgorithm: pkcs1-rsaEncryption
                                        encryptedKey: "xxxxx clau simetrica xifrada amb publickey xx"
                                        }
                                }
                        encryptedContentInfo: SEQUENCE{
                                contentType: pkcs7-data  [!pkcs7-signedData]
                                contentEncryptionAlgorithm: SEQUENCE{
                                        algorithm: secsig-algorithm-des-cbc
                                        parameters: "xxxxxxxxxxxxxxxx"
                                        }
                                }
                        encryptedContent: "xxxxxx  dades encryptades   xxxx"
                        }
                }
```

```
                                                        EnvelopedData 432 bytes total
48 130 1 172                                            SEQUENCE(426)
        2 1 0                                                   INTEGER(3) version
49 130 1 88                                                     SET OF(423) recipientinfos[]
48 130 1 84                                                             SEQUENCE(419)
recipientInfo
```

```
        2 1 0                                                                   INTEGER(3)
version
48 129 188                                                                      SEQUENCE(413)
isuerAndSerialNumber
48 129 174
        SEQUENCE(410) issuer
49 11
        SET OF(11)
48 9
        SEQUENCE(9)
        6 3 85 4 6
                OBJECT ID: country 2.5.4.6
        19 2 69 83
        PRINTABLE STRING
49 18
        SET OF(18)
48 16
        SEQUENCE(16)
        6 3 85 4 8
                OBJECT ID stateOrprovince
        19 9 66 97 114 99 101 108 111 110 97                                    printable string
49 18
        SET OF(18)
48 16
        SEQUENCE(16)
        6 3 85 4 7
                OBJECT ID locality
        19 9 66 97 114 99 101 108 111 110 97                            printable string
49 12
        SET OF(12)
48 10
        SEQUENCE(10)
        6 3 85 4 10
                OBJECT ID organization
        19 3 85 79 67
                printable string
49 22
        SET OF(22)
48 20
        SEQUENCE(20)
        6 3 85 4 11
                OBJECT ID
        19 13 80 70 67 32 83 101 103 117 114 101 116 97 116             printable string
49 25
        SET OF(25)
48 23
        SEQUENCE(23)
        6 3 85 4 3
                OBJECT ID
        20 16 67 65 95 80 70 67 95 83 101 103 117 114 101 116 97 116 T61STRING
49 19
        SET OF(19)
48 17
        SEQUENCE(17)
        6 3 85 4 46
                OBJECT ID
        19 10 48 48 48 48 48 48 48 48 45 65                                     printable
string
49 33
        SET OF(33)
48 31
        SEQUENCE(31)
        6 9 42 134 72 134 247 13 1 9 1
                OBJECT ID "1.2.840.113549.1.9.1", "emailAddress"
        22 18 106 99 97 115 116 101 108 108 97 114 64 117 111 99 46 101 100 117 IA5STRING
        2 9 0 226 83 10 186 28 193 107 180                              INTEGER(11) serialNumber
48 13
        SEQUENCE(13)
        6 9 42 134 72 134 247 13 1 1 1
        OBJECT ID KeyEncryptionAlgorith: rsa "1.2.840.113549.1.1.1"
        5 0
                        NULL ¿?
        4 129 128 41 127 98 23 16 50 227 115 114 159 3 50 214 51 116 82 146 43 120 1 109
122 211 213 161 123 36 236 9 213 36 170 131 237 141 112 219 187 12 32 223 54 36 38 101
173 254 56 195 80 237 152 91 173 75 30 122 76 94 240 203 27 123 97 156 26 99 93 87 154
36 210 67 135 183 0 227 221 174 245 26 18 113 85 108 230 133 41 15 45 147 223 168 224
```

```
21 151 23 19 85 64 58 46 220 60 11 3 78 205 90 203 237 82 179 237 156 169 207 152 140
30 33 118 165 219 162 16 248 154                                    OCTET STRING(129) EncryptedKey:
clave simetrica encryptada
48 75
        SEQUENCE(75) EncryptedContentInfo
        6 9 42 134 72 134 247 13 1 7 1
        OBJECT ID     ContentType: PKCS#7_data
48 20
                        SEQUENCE(20)  ContentEncryptionAlgorithm
        6 8 42 134 72 134 247 13 3 7
                OBJECT ID AlgorithmID: des_EDE3_CBC "1.2.840.113549.3.7"
        4 8 243 126 175 176 50 160 116 175
        OCTET STRING parameters: ¿8 bytes?¿hash?
        128 40 39 218 41 74 193 243 175 98 123 4 247 225 89 138 98 238 221 92 224 182 38
173 154 62 132 93 133 36 120 133 154 130 163 102 212 104 26 96 255 40

                                CONTEXT_EXPECIFIC(40): EncryptedContent: texto encryptado



*/
public class EnvelopedDataTest{

        public static void main(String[] args){
                Security.insertProviderAt(new IAIK(), 2);
                //java.security.NoSuchAlgorithmException: 3DES KeyGenerator not available

                EnvelopedData enveloped_data;
                try{
                        byte[] _clearText = "1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0".getBytes();
                        enveloped_data = new EnvelopedData(_clearText, AlgorithmID.des_EDE3_CBC);


                        //busquem la clau pública del destinatari.
                        FileInputStream fileInput;
                        String fileName = "..\\doc\\pki\\Pacient.crt";
                        X509Certificate x509;
                        RecipientInfo ri;
                        fileInput = new FileInputStream(fileName);
                        x509 = new X509Certificate(fileInput);


                        //ri= new RecipientInfo(x509, AlgorithmID.des_EDE3_CBC);
                        ri= new RecipientInfo(x509, AlgorithmID.rsaEncryption); //¿rsa?
                        //hem creat un RecipientInfo amb la clau publica del destinatari:
x509certificate.getPublicKey();
        //hem guardat les dades del destinatari i del certificat:  IssuerAndSerialNumber(x509certificate);
        //hem guardat informació de la ¿clau simetrica?
                        //però la clau simetrica encriptada encara no està definida
                                //comprobació:
                                //byte[] riek = ri.getEncryptedKey();
                                //if(riek != null) System.out.println("riek: "+ riek.length);
                                //else System.out.println("1.- Clau simetrica sense definir");
                        //AlgorithmID a = ri.getKeyEncryptionAlgorithm();
                        //Comprobació:
                        //System.out.println(a.toString());
                        //System.out.println("Issuer: "+ ri.getIssuerAndSerialNumber().toString());


                        RecipientInfo[] recipients = new RecipientInfo[1];
                    recipients[0] = ri;
                    //EnvelopedDataStream: public void setRecipientInfos(RecipientInfo arecipientinfo[])

                        enveloped_data.setRecipientInfos(recipients);
                        //enveloped_data.addRecipientInfo(ri);
                        //Comprobacio: recuperem el RecipientInfo i consultem la clau secreta.

                        //RecipientInfo nouRi = enveloped_data.getRecipientInfo(x509);
                        //System.out.println("version: nouRi: "+ nouRi.getVersion());

                        //byte[] nriek = nouRi.getEncryptedKey();
                        //if(nriek != null) System.out.println("nriek: "+ nriek.length);
                        //else System.out.println("2.- Clau simetrica sense definir...");


                        byte[] _cipherText = enveloped_data.getEncoded();
```

```
                                for (int i =0 ; i < _cipherText.length; i++){
                                        int c = _cipherText[i];
                                        if(c > -1) System.out.print((char)c);
                                        else System.out.print((char)127+129+c);
                                }
                                for (int i =0 ; i < _cipherText.length; i++){
                                        short c = _cipherText[i];
                                        if(c>-1) System.out.print(" "+ (c));
                                        else System.out.print(" "+ (127+129+c));
                                }

                                BIT_STRING bs = new BIT_STRING(_cipherText);
                                System.out.println(bs.getBinaryString());


                                //_cipherText es un Sobre Digital amb el text encriptat (dintre d'un EnvelopedData) i la
clau simètrica encriptada

                                //1.? Consultar el EncryptedContentInfo
                                EncryptedContentInfo eci = (EncryptedContentInfo)
enveloped_data.getEncryptedContentInfo();
                                AlgorithmID aid = eci.getContentEncryptionAlgorithm();
                                SecretKey sk = eci.setupCipher(aid);
                                byte[] bitesSK = sk.getEncoded();
                                BIT_STRING bsSk = new BIT_STRING(bitesSK);
                                //System.out.println("bsSk: ");
                                //System.out.println(bsSk.getBinaryString());
                                //System.out.print("\n");

                                for (int i =0 ; i < bitesSK.length; i++){
                                        int c = bitesSK[i];
                                        //if(c > -1) System.out.print((char)c);
                                        //else System.out.print((char)127+129+c);
                                }
                                for (int i =0 ; i < bitesSK.length; i++){
                                        short c = bitesSK[i];
                                        //if(c>-1) System.out.print(" "+ (c));
                                        //else System.out.print(" "+ (127+129+c));
                                }
                                //System.out.println("fin SK.\n");

                                // repetició de la jugada:
                                EncryptedContentInfo eci3 = (EncryptedContentInfo)
enveloped_data.getEncryptedContentInfo();
                                AlgorithmID aid2 = eci3.getContentEncryptionAlgorithm();
                                SecretKey sk2 = eci3.setupCipher(aid2);
                                byte[] bitesSK2 = sk2.getEncoded();
                                BIT_STRING bsSk2 = new BIT_STRING(bitesSK2);
                                //System.out.println("bsSk2: ");
                                //System.out.println(bsSk2.getBinaryString());
                                //System.out.print("\n");

                                for (int i =0 ; i < bitesSK2.length; i++){
                                        int c = bitesSK2[i];
                                        //if(c > -1) System.out.print((char)c);
                                        //else System.out.print((char)127+129+c);
                                }
                                for (int i =0 ; i < bitesSK2.length; i++){
                                        short c = bitesSK2[i];
                                        //if(c>-1) System.out.print(" "+ (c));
                                        //else System.out.print(" "+ (127+129+c));
                                }
                                //System.out.println("fin SK2.\n");

                                //CONCLUSION: se genera una clave nueva cada vez.


                                //Analizar enveloped_data
                                System.out.println("componentes evd: " +
(enveloped_data.toASN1Object()).countComponents());
                                for(int ii= 0; ii<3; ii++){
                                                ASN1Object ax =
(enveloped_data.toASN1Object()).getComponentAt(ii);
                                                System.out.println(ax.toString());
```

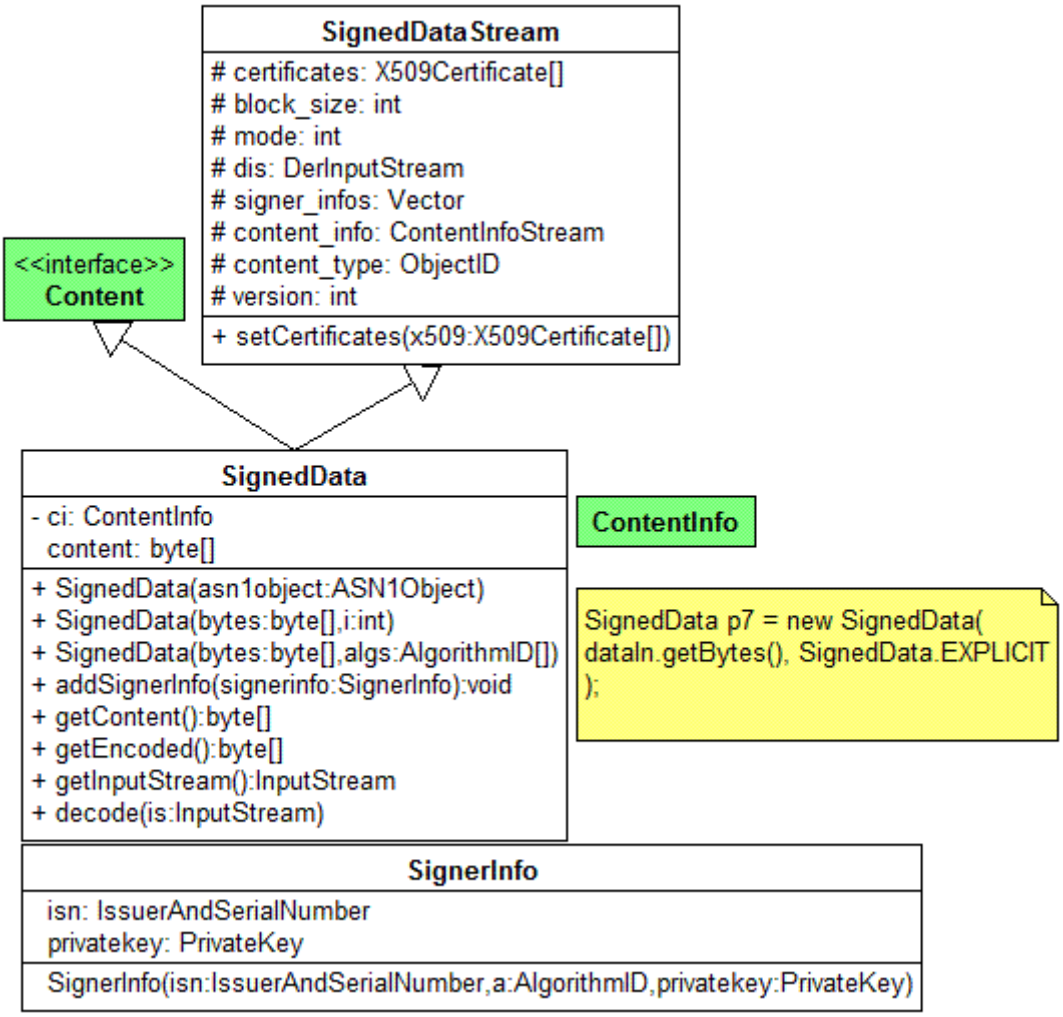
```
                                        //componentes evd: 3
                                        //INTEGER = 0
                                        //SET[C] = 1 elements
                                        //SEQUENCE[C] = 3 elements

                        }
                        //System.out.println("componentes eci: " + (eci3.toASN1Object()).countComponents());
                        for(int ii= 0; ii<3; ii++){
                                        ASN1Object ax = (eci3.toASN1Object()).getComponentAt(ii);
                                        //System.out.println(ax.toString());
                                        //componentes eci: 3
                                        //OBJECT ID = PKCS#7 data
                                        //SEQUENCE[C] = 2 elements
                                        //CONTEXTSPECIFIC = [0] IMPLICIT

                        }
                        ASN1Object ax = (eci3.toASN1Object()).getComponentAt(2);
                        //System.out.println("eci3(3) :  " + ax.countComponents());//eci3(3) :  1

                        //iaik.asn1.CON_SPEC contextspecific = (iaik.asn1.CON_SPEC) ax.getComponentAt(0);
                        iaik.asn1.OCTET_STRING contextspecific = (iaik.asn1.OCTET_STRING)
ax.getComponentAt(0);
                        Object obj2 = contextspecific.getValue();
                        //System.out.println(contextspecific.toString());//OCTET STRING = 56 bytes:
C5:B0:EF:85:56...
                        //System.out.println("eci: " + contextspecific.toString(true));//OCTET STRING = 56 bytes:
C5:B0:EF:85:56:1B:F8:39:9A:93:A4:E6:C0:DE:CD:1E:AF:49:B9:7D:4C:8B:CC:3F:70:F3:27:E9:FC:86:1C:41:44:B0:0B:0
B:A8:11:32:D8:9F:0A:54:BC:CC:C9:21:68:94:AC:88:61:67:9D:2F:19



                        iaik.asn1.SET set1 =
(iaik.asn1.SET)(enveloped_data.toASN1Object()).getComponentAt(1);
                        System.out.println(set1.toString());//SET[C] = 1 elements
                        iaik.asn1.SEQUENCE seq3 = (iaik.asn1.SEQUENCE) set1.getComponentAt(0);
                        //System.out.println("sequence 3:" + seq3.toString());//sequence 3:SEQUENCE[C] = 4
elements
                        System.out.println("componentes seq3");
                        for(int ii= 0; ii<4; ii++){
                                        ASN1Object ax1 = seq3.getComponentAt(ii);
                                        System.out.println(ax1.toString());
                                        //componentes seq3
                                        //INTEGER = 0
                                        //SEQUENCE[C] = 2 elements
                                        //SEQUENCE[C] = 2 elements
                                        //OCTET STRING = 128 bytes:   = clave simetrica encriptada para
Pacient.
                        }
                        iaik.asn1.SEQUENCE seq4 = (iaik.asn1.SEQUENCE) seq3.getComponentAt(2);
                        System.out.println("sequence 4:" + seq4.toString());//sequence 4:SEQUENCE[C] = 2
elements



                        iaik.asn1.OCTET_STRING clavesimetrica = (iaik.asn1.OCTET_STRING)
seq3.getComponentAt(3);
                        Object obj3 = clavesimetrica.getValue();
                        //System.out.println(contextspecific.toString());//
                        System.out.println(clavesimetrica.toString(true));//OCTET STRING = 128 bytes:
86:B6:29:DC:9E:E2:63:15:25:B5:BD:10:34:88:83:BE:10:14:22:9B:3C:2B:F2:94:B2:4B:E4:A2:B3:28:C8:0B:65:00:20:34:
BD:01:8E:E9:C5:05:1C:6B:8E:E6:E3:D5:8C:62:57:56:82:A5:28:22:53:28:07:66:6A:B9:7D:54:F7:1B:56:34:95:22:7F:24:7
9:62:90:2E:5C:9A:01:3B:FF:17:52:A2:72:CC:D2:37:16:49:BB:2F:CE:58:B4:1F:E7:3A:CA:26:1D:E8:DA:BC:20:63:44:5C:
26:C9:AA:9E:3C:94:D4:E6:F6:DF:84:6A:7E:E8:B0:A6:C2:FC:1E:65


                        iaik.asn1.SEQUENCE seq1 =
(iaik.asn1.SEQUENCE)(enveloped_data.toASN1Object()).getComponentAt(2);
                        //System.out.println(seq1.toString());//SEQUENCE[C] = 3 elements
                        System.out.println("componentes seq1");
                        for(int ii= 0; ii<3; ii++){
                                        ASN1Object ax1 = seq1.getComponentAt(ii);
                                        System.out.println(ax1.toString());
                                        //componentes seq1
                                        //OBJECT ID = PKCS#7 data
                                        //SEQUENCE[C] = 2 elements
```

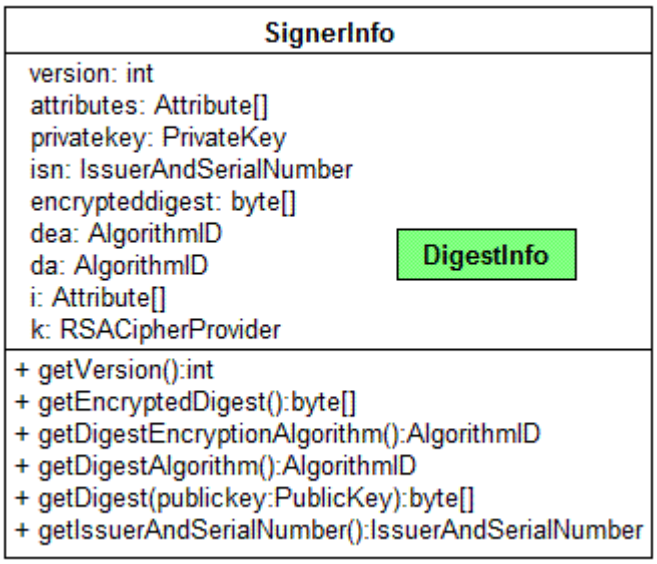
```
                                        //CONTEXTSPECIFIC = [0] IMPLICIT  // ¿es el texto cifrado?
                                }
                                iaik.asn1.SEQUENCE seq2 = (iaik.asn1.SEQUENCE) seq1.getComponentAt(1);
                                System.out.println("componentes seq2");
                                for(int ii= 0; ii<2; ii++){
                                        ASN1Object ax1 = seq2.getComponentAt(ii);
                                        System.out.println(ax1.toString());
                                        //componentes seq2
                                        //OBJECT ID = DES-EDE3-CBC
                                        //OCTET STRING = 8 bytes: D6:38:72:D2:A2...//parametros?
                                }


                                //2.- Desencriptar:

                                //encontrar la clave privada a partir de Pacient.p12

                                P12 p12 = null;
                                try{
                                //System.out.println(fileP12+" "+password);
                                p12 = new P12("..\\doc\\pki\\Pacient.p12","uoc0506");
                                }catch(Exception e){
                                e.printStackTrace();
                                System.exit(0);
                                }

                                // If the EnvelopedData is supplied as DER encoding, (byte[]) first decode it to an
ASN1Object:????
                                ASN1Object obj = DerCoder.decode(_cipherText);
                                System.out.println(obj.toString());
                                // Create an EnvelopedData structure from the supplied ASN.1 object, and parse the
internal structure:
                                EnvelopedData enveloped_data2 = new EnvelopedData(obj);
                                // Get information about the inherent EncryptedContentInfo:
                                EncryptedContentInfo eci2 =
(EncryptedContentInfo)enveloped_data2.getEncryptedContentInfo();

                                System.out.println("Content type: "+eci2.getContentType().getName());
                                System.out.println("Content encryption algorithm:
"+eci2.getContentEncryptionAlgorithm().getName());

                                // Get information about the included RecipientInfos:
                                RecipientInfo[] recipients2 = enveloped_data2.getRecipientInfos();

                                System.out.println("Included RecipientInfos:");

                                for (int i=0; i < recipients2.length; i++) {
                                   System.out.print("Recipient "+(i+1)+":");
                                   System.out.println(recipients2[i].getIssuerAndSerialNumber());
                                }


        // Use some recipient s specific private key to decrypt the inherent encrypted secret key
        // to be used for subsequently performing encrypted-content decryption:

        //setup cipher for recipient 1:
                                int recipientInfoIndex = 0;
                                enveloped_data2.setupCipher(p12.getPrivateKey(), recipientInfoIndex);

        // Unlike the stream supporting EnvelopedDataStream class where the setupCipher method
        // only initializes the cipher for decryption, whole the encrypted-content decryption
        // already is performed inside the setupCipher method of this class.

        // Get the recovered content:

                                byte[] _clearText2 = enveloped_data2.getContent();
                                for (int i =0 ; i < _clearText2.length; i++){
                                        int c = _clearText2[i];
                                        //System.out.print((char)c);
                                }



                        }catch(NoSuchAlgorithmException nsae){
                                System.out.println(nsae);
                        }
```

```
                catch(InvalidKeyException ike){System.out.print(ike);}
                catch(PKCSException pkcse){System.out.print(pkcse);}
                catch(IOException ioe){System.out.print(ioe);}
                //catch(FileNotFoundException fnfe){System.out.println(fnfe);}
                catch(CertificateException ce){System.out.println(ce);}
                catch(CodingException cde){};
                //RecipientInfo permet conservar una clau encriptada.


        }
}
```

Pkcs7-signedData. En el cas que el PKCS#7 sigui amb informació no inclosa, en lloc de contenir la informació, el camp “content.contentInfo.content” no està present.

SignedData





SignerManagerTest.java

```
import java.security.PrivateKey;
import java.security.PublicKey;
import iaik.x509.X509Certificate;
import iaik.pkcs.pkcs7.SignerInfo;
import iaik.pkcs.pkcs7.IssuerAndSerialNumber;
import iaik.asn1.DerCoder;
import iaik.asn1.structures.AlgorithmID;
import iaik.asn1.ASN1Object;
import iaik.security.provider.IAIK;
import java.security.*;
import iaik.pkcs.PKCSException;
import iaik.pkcs.pkcs7.SignedData;

/** Utilitza la classe P12 */
public class SignerManagerTest{
	public static void main(String[] args){
		Security.insertProviderAt(new IAIK(), 2);
		boolean verify = true;
		// per signar dades en fa falta la clau secreta. La classe P12 ens permet obtenir les claus i X509
		P12 p12 = null;
		String p12File = ".\\doc\\pki\\Pacient.p12";
		String password = "uoc0506";
		try{
		   p12 = new P12(p12File, password);
		} catch(Exception e){
		e.printStackTrace();
		System.exit(0);//si no tenim les claus no hi ha res a fer.
		}
		PrivateKey _privateKey = p12.getPrivateKey();
		PublicKey  _publicKey = p12.getPublicKey();
		X509Certificate[] _chain = p12.getCertificates();

		//Crearem un SignedData amb les dades que volem signar ( en format byte).
	// public SignedData(byte[] bytes, AlgorithmID[] algorithmid, int i) throws NoSuchAlgorithmException
		SignedData p7 = new SignedData("dataIn".getBytes(), SignedData.EXPLICIT);
		//SignedData.EXPLICIT = 2 = mode
			//comprobació de les dades introduides:
			byte[] sdb = p7.getContent();
			for(int i = 0; i<sdb.length; i++){
				int c = sdb[i];
```

```
                                        System.out.print((char)sdb[i]);
                                }//se obtenen els bytes introduits.
                        //Crearem un objecte SignerInfo per guardar les dades de l'usuari:
//public SignerInfo(IssuerAndSerialNumber issuerandserialnumber, AlgorithmID algorithmid, PrivateKey privatekey)
                SignerInfo signer = new SignerInfo(new IssuerAndSerialNumber(_chain[0]), AlgorithmID.sha1, _privateKey);
                        //comprobació :
                        System.out.println("signer isn: "+signer.getIssuerAndSerialNumber());
                        //Per fer la verificació fa falta introduir les dades dels certificats:
                        //public void setCertificates(X509Certificate ax509certificate[]) //en SignedDataStream
                        p7.setCertificates(_chain);
                        //public void addSignerInfo(SignerInfo signerinfo)throws NoSuchAlgorithmException
                        try{
                                p7.addSignerInfo(signer);
                        }catch(NoSuchAlgorithmException nsae){System.out.println(nsae);}
                        // Generem la signatura:
                        //    public byte[] getEncoded() throws PKCSException
                        try{
                                byte[] encoded = p7.getEncoded();
                                //comprobació:
                                for(int i=0; i<encoded.length;i++){
                                        int c = encoded[i];
                                        System.out.print((char)encoded[i]);
                                }
                                //Consulta de l'objecte SignerInfo actualitzat:
                                try{
                                        System.out.println("signer:getDigest: ");
                                        byte[] db = signer.getDigest(_publicKey);
                                        for(int i = 0; i<db.length; i++){
                                                int c = db[i];
                                                System.out.print((char)db[i]);
                                        }
                                }
                                catch(InvalidKeyException ike){}
                                catch(SignatureException se){}
                                byte[] bed = signer.getEncryptedDigest();
                                System.out.println("signer:getEncryptedDigest: ");
                                for(int i = 0; i<bed.length; i++){
                                        int c = bed[i];
                                        System.out.print((char)bed[i]);
                                }
                                //Comprobació de la signatura:
                                AlgorithmID[] algIDs = { AlgorithmID.sha1, AlgorithmID.md5 };
                           //public SignedData(byte abyte0[], AlgorithmID aalgorithmid[]) throws NoSuchAlgorithmException
                                SignedData signature = null;
                                try{
                                        byte[] data = "dataIn".getBytes();
                                        signature = new SignedData(data, algIDs);
                                }catch(NoSuchAlgorithmException nsae){nsae.printStackTrace();}
                                ASN1Object objExpP7 = null;
                                try{
                                   objExpP7 = DerCoder.decode(encoded);
                                   signature.decode(objExpP7);
                                   SignerInfo[] signerInfos = signature.getSignerInfos();
                                   System.out.println("\nsi: "+ signerInfos.length);
                                   for (int i=0; i < signerInfos.length; i++){
                                        //   public X509Certificate verify(int i) throws SignatureException
                                        try{
                                                //El métode verify(i) retorna un X509Certificate en el cas que la
                                                //verificació sigui correcta. Si la verificació es incorrecta es generen
                                                //diferents tipus d'excepcions.
                                                System.out.println("s.v:"+ (signature.verify(i)).toString());
                                        }catch(SignatureException se){
                                                System.out.println(se);
                                                //verify ha estat definida com 'true'. Si es genera la excepció: false
                                                verify = false;
                                        }
                                   }
                                }catch(Exception e){e.printStackTrace();}
                        }catch(PKCSException pkcse){System.out.println(pkcse);}
                        System.out.println("verify: "+verify);
                }
}
```

Pkcs7-signedAndEnvelopedData

1.4. La criptografia de clau simètrica. (està inclosa dins de les classes pkcs7)
TripleDES

- java.lang.Object
  - javax.crypto.CipherSpi
    - iaik.security.cipher.**DES**
    - iaik.security.cipher.**TripleDES**
  - iaik.security.cipher.**DESParameterSpec** (implements java.security.spec.AlgorithmParameterSpec)
  - javax.crypto.KeyGeneratorSpi
    - iaik.security.cipher.**DESKeyGenerator**
    - iaik.security.cipher.**TripleDESKeyGenerator**
  - javax.crypto.spec.SecretKeySpec (implements java.security.spec.KeySpec, javax.crypto.SecretKey)
    - iaik.security.cipher.**SecretKey** (implements java.io.Serializable)

1.5. Llista completa de classes de iaik.

...\iaik\iaik;4096
...\iaik\iaik\asn1;16384
...\iaik\iaik\asn1\ASN.class;7085
...\iaik\iaik\asn1\ASN1.class;4590
...\iaik\iaik\asn1\ASN1Object.class;2700
...\iaik\iaik\asn1\ASN1String.class;1311
...\iaik\iaik\asn1\ASN1Type.class;229
...\iaik\iaik\asn1\b.class;1027
...\iaik\iaik\asn1\BIT_STRING.class;1944
...\iaik\iaik\asn1\BMPString.class;1124
...\iaik\iaik\asn1\BOOLEAN.class;1401
...\iaik\iaik\asn1\c.class;635
...\iaik\iaik\asn1\CodingException.class;209
...\iaik\iaik\asn1\ConstructedType.class;3354
...\iaik\iaik\asn1\CON_SPEC.class;2347
...\iaik\iaik\asn1\d.class;569
...\iaik\iaik\asn1\DerCoder.class;3859
...\iaik\iaik\asn1\DerInputStream.class;7366
...\iaik\iaik\asn1\e.class;404
...\iaik\iaik\asn1\EncodeListener.class;213
...\iaik\iaik\asn1\ENUMERATED.class;1283
...\iaik\iaik\asn1\GeneralizedTime.class;1205
...\iaik\iaik\asn1\GeneralString.class;641
...\iaik\iaik\asn1\IA5String.class;633
...\iaik\iaik\asn1\INTEGER.class;1278
...\iaik\iaik\asn1\NULL.class;785
...\iaik\iaik\asn1\NumericString.class;641
...\iaik\iaik\asn1\ObjectID.class;10846
...\iaik\iaik\asn1\OCTET_STRING.class;5552
...\iaik\iaik\asn1\PrintableString.class;1972
...\iaik\iaik\asn1\SEQUENCE.class;765
...\iaik\iaik\asn1\SET.class;1739

...\iaik\iaik\asn1\structures;0
...\iaik\iaik\asn1\structures\a.class;778
...\iaik\iaik\asn1\structures\AccessDescription.class;2575
...\iaik\iaik\asn1\structures\AlgorithmID.class;15925
...\iaik\iaik\asn1\structures\Attribute.class;4470
...\iaik\iaik\asn1\structures\Attributes.class;4492
...\iaik\iaik\asn1\structures\AttributeValue.class;517
...\iaik\iaik\asn1\structures\AVA.class;5080
...\iaik\iaik\asn1\structures\ChoiceOfTime.class;4120
...\iaik\iaik\asn1\structures\DistributionPoint.class;4153
...\iaik\iaik\asn1\structures\GeneralName.class;5493
...\iaik\iaik\asn1\structures\GeneralNames.class;2218
...\iaik\iaik\asn1\structures\GeneralSubtree.class;2330
...\iaik\iaik\asn1\structures\Name.class;3629
...\iaik\iaik\asn1\structures\PolicyInformation.class;1843
...\iaik\iaik\asn1\structures\PolicyMapping.class;1376
...\iaik\iaik\asn1\structures\PolicyQualifierInfo.class;3613
...\iaik\iaik\asn1\structures\RDN.class;2611
...\iaik\iaik\asn1\structures\UnknownAttributeValue.class;878
...\iaik\iaik\asn1\T61String.class;633
...\iaik\iaik\asn1\UNIString.class;871
...\iaik\iaik\asn1\UNKNOWN.class;1463
...\iaik\iaik\asn1\UTCTime.class;1189
...\iaik\iaik\asn1\UTF8String.class;4277
...\iaik\iaik\asn1\VisibleString.class;641
...\iaik\iaik\Debug.class;210
...\iaik\iaik\iaik.xls;11776
...\iaik\iaik\jce;0
...\iaik\iaik\jce\com;0
...\iaik\iaik\jce\com\fourthpass;0
...\iaik\iaik\jce\com\fourthpass\a.class;5328
...\iaik\iaik\pkcs;4096
...\iaik\iaik\pkcs\NetscapeCertList.class;2853
...\iaik\iaik\pkcs\pkcs1;0
...\iaik\iaik\pkcs\pkcs1\RSACipher.class;9118
...\iaik\iaik\pkcs\pkcs10;0
...\iaik\iaik\pkcs\pkcs10\CertificateRequest.class;8836
...\iaik\iaik\pkcs\pkcs10\CertRequest.class;266
...\iaik\iaik\pkcs\pkcs12;0
...\iaik\iaik\pkcs\pkcs12\a.class;2946
...\iaik\iaik\pkcs\pkcs12\Attributes.class;1933
...\iaik\iaik\pkcs\pkcs12\AuthenticatedSafe.class;5628
...\iaik\iaik\pkcs\pkcs12\b.class;784
...\iaik\iaik\pkcs\pkcs12\CertificateBag.class;2067
...\iaik\iaik\pkcs\pkcs12\CRLBag.class;1865
...\iaik\iaik\pkcs\pkcs12\IvGenerator.class;234
...\iaik\iaik\pkcs\pkcs12\KeyBag.class;1495
...\iaik\iaik\pkcs\pkcs12\KeyFactory.class;1645
...\iaik\iaik\pkcs\pkcs12\MacKeyGenerator.class;238
...\iaik\iaik\pkcs\pkcs12\PKCS12.class;8330
...\iaik\iaik\pkcs\pkcs12\PKCS8ShroudedKeyBag.class;1811
...\iaik\iaik\pkcs\pkcs12\SafeBag.class;4544
...\iaik\iaik\pkcs\pkcs12\SafeContentsBag.class;1451
...\iaik\iaik\pkcs\pkcs12\SecretBag.class;1581
...\iaik\iaik\pkcs\pkcs12\SecretKeyGenerator.class;241
...\iaik\iaik\pkcs\pkcs5;0
...\iaik\iaik\pkcs\pkcs5\KeyFactory.class;1641
...\iaik\iaik\pkcs\pkcs5\PBKDF2.class;2809
...\iaik\iaik\pkcs\pkcs7;0
...\iaik\iaik\pkcs\pkcs7\a.class;2474
...\iaik\iaik\pkcs\pkcs7\b.class;1455
...\iaik\iaik\pkcs\pkcs7\c.class;773
...\iaik\iaik\pkcs\pkcs7\Content.class;216
...\iaik\iaik\pkcs\pkcs7\ContentInfo.class;5056
...\iaik\iaik\pkcs\pkcs7\ContentInfoStream.class;4765
...\iaik\iaik\pkcs\pkcs7\ContentStream.class;452
...\iaik\iaik\pkcs\pkcs7\d.class;204
...\iaik\iaik\pkcs\pkcs7\Data.class;2236
...\iaik\iaik\pkcs\pkcs7\DataStream.class;2065
...\iaik\iaik\pkcs\pkcs7\DigestedData.class;4651
...\iaik\iaik\pkcs\pkcs7\DigestedDataStream.class;6349
...\iaik\iaik\pkcs\pkcs7\DigestInfo.class;1861
...\iaik\iaik\pkcs\pkcs7\EncryptedContentInfo.class;4043
...\iaik\iaik\pkcs\pkcs7\EncryptedContentInfoStream.class;5554
...\iaik\iaik\pkcs\pkcs7\EncryptedData.class;4244
...\iaik\iaik\pkcs\pkcs7\EncryptedDataStream.class;5245

...\iaik\iaik\pkcs\pkcs7\EnvelopedData.class;5687
...\iaik\iaik\pkcs\pkcs7\EnvelopedDataStream.class;6141
...\iaik\iaik\pkcs\pkcs7\IssuerAndSerialNumber.class;1982
...\iaik\iaik\pkcs\pkcs7\RecipientInfo.class;4371
...\iaik\iaik\pkcs\pkcs7\RSACipherProvider.class;1483
...\iaik\iaik\pkcs\pkcs7\SignedAndEnvelopedData.class;8019
...\iaik\iaik\pkcs\pkcs7\SignedAndEnvelopedDataStream.class;7681
...\iaik\iaik\pkcs\pkcs7\SignedData.class;6849
...\iaik\iaik\pkcs\pkcs7\SignedDataStream.class;11415
...\iaik\iaik\pkcs\pkcs7\SignerInfo.class;8298
...\iaik\iaik\pkcs\PKCS7CertList.class;1985
...\iaik\iaik\pkcs\pkcs8;0
...\iaik\iaik\pkcs\pkcs8\EncryptedPrivateKeyInfo.class;5706
...\iaik\iaik\pkcs\pkcs8\PrivateKeyInfo.class;4078
...\iaik\iaik\pkcs\pkcs9;0
...\iaik\iaik\pkcs\pkcs9\ChallengePassword.class;1279
...\iaik\iaik\pkcs\pkcs9\ExtensionRequest.class;2453
...\iaik\iaik\pkcs\PKCSException.class;321
...\iaik\iaik\pkcs\PKCSParsingException.class;218
...\iaik\iaik\security;4096
...\iaik\iaik\security\cipher;0
...\iaik\iaik\security\cipher\a.class;1393
...\iaik\iaik\security\cipher\AESKeyGenerator.class;207
...\iaik\iaik\security\cipher\b.class;443
...\iaik\iaik\security\cipher\Blowfish.class;225
...\iaik\iaik\security\cipher\BlowfishKeyGenerator.class;213
...\iaik\iaik\security\cipher\c.class;2104
...\iaik\iaik\security\cipher\CAST128.class;224
...\iaik\iaik\security\cipher\CAST128KeyGenerator.class;211
...\iaik\iaik\security\cipher\CAST128KeyWrap.class;3529
...\iaik\iaik\security\cipher\CAST128Parameters.class;2890
...\iaik\iaik\security\cipher\CAST128ParameterSpec.class;658
...\iaik\iaik\security\cipher\CAST128WrapParameters.class;1960
...\iaik\iaik\security\cipher\CAST128WrapParameterSpec.class;299
...\iaik\iaik\security\cipher\d.class;2811
...\iaik\iaik\security\cipher\DES.class;220
...\iaik\iaik\security\cipher\DESKeyGenerator.class;1088
...\iaik\iaik\security\cipher\DESParameterSpec.class;667
...\iaik\iaik\security\cipher\e.class;1117
...\iaik\iaik\security\cipher\f.class;4177
...\iaik\iaik\security\cipher\g.class;1532
...\iaik\iaik\security\cipher\GeneralKeyFactory.class;2701
...\iaik\iaik\security\cipher\GOST.class;221
...\iaik\iaik\security\cipher\GOSTKeyGenerator.class;198
...\iaik\iaik\security\cipher\GOSTParameterSpec.class;590
...\iaik\iaik\security\cipher\h.class;1725
...\iaik\iaik\security\cipher\i.class;3142
...\iaik\iaik\security\cipher\IDEA.class;221
...\iaik\iaik\security\cipher\IDEAKeyGenerator.class;198
...\iaik\iaik\security\cipher\IDEAKeyWrap.class;1462
...\iaik\iaik\security\cipher\IvParameters.class;2289
...\iaik\iaik\security\cipher\j.class;3172
...\iaik\iaik\security\cipher\k.class;6176
...\iaik\iaik\security\cipher\l.class;3657
...\iaik\iaik\security\cipher\m.class;1721
...\iaik\iaik\security\cipher\MARS.class;227
...\iaik\iaik\security\cipher\MARSKeyGenerator.class;209
...\iaik\iaik\security\cipher\n.class;1512
...\iaik\iaik\security\cipher\o.class;16458
...\iaik\iaik\security\cipher\p.class;7500
...\iaik\iaik\security\cipher\PBEKey.class;1099
...\iaik\iaik\security\cipher\PBEKeyBMP.class;783
...\iaik\iaik\security\cipher\PbeWithMD5AndDES_CBC.class;3770
...\iaik\iaik\security\cipher\PbeWithSHAAnd3_KeyTripleDES_CBC.class;3959
...\iaik\iaik\security\cipher\PbeWithSHAAnd40BitRC2_CBC.class;3943
...\iaik\iaik\security\cipher\q.class;1965
...\iaik\iaik\security\cipher\r.class;430
...\iaik\iaik\security\cipher\RawBlockCipher128.class;3032
...\iaik\iaik\security\cipher\RawBlockCipher256.class;2777
...\iaik\iaik\security\cipher\RawMARS.class;9972
...\iaik\iaik\security\cipher\RawRC6.class;1854
...\iaik\iaik\security\cipher\RawRijndael.class;5094
...\iaik\iaik\security\cipher\RawRijndael256.class;5903
...\iaik\iaik\security\cipher\RawSerpent.class;12498
...\iaik\iaik\security\cipher\RawTwofish.class;5085
...\iaik\iaik\security\cipher\RC2.class;220

...\iaik\iaik\security\cipher\RC2KeyGenerator.class;203
...\iaik\iaik\security\cipher\RC2KeyWrap.class;3568
...\iaik\iaik\security\cipher\RC2Parameters.class;4892
...\iaik\iaik\security\cipher\RC2WrapParameters.class;1992
...\iaik\iaik\security\cipher\RC2WrapParameterSpec.class;302
...\iaik\iaik\security\cipher\RC4.class;220
...\iaik\iaik\security\cipher\RC4KeyGenerator.class;203
...\iaik\iaik\security\cipher\RC5.class;220
...\iaik\iaik\security\cipher\RC5KeyGenerator.class;203
...\iaik\iaik\security\cipher\RC5Parameters.class;3334
...\iaik\iaik\security\cipher\RC6.class;225
...\iaik\iaik\security\cipher\RC6KeyGenerator.class;203
...\iaik\iaik\security\cipher\Rijndael.class;235
...\iaik\iaik\security\cipher\Rijndael256.class;241
...\iaik\iaik\security\cipher\RijndaelKeyGenerator.class;217
...\iaik\iaik\security\cipher\s.class;661
...\iaik\iaik\security\cipher\SecretKey.class;1952
...\iaik\iaik\security\cipher\Serpent.class;233
...\iaik\iaik\security\cipher\SerpentKeyGenerator.class;215
...\iaik\iaik\security\cipher\t.class;6292
...\iaik\iaik\security\cipher\TripleDES.class;226
...\iaik\iaik\security\cipher\TripleDESKeyGenerator.class;923
...\iaik\iaik\security\cipher\TripleDESKeyWrap.class;2099
...\iaik\iaik\security\cipher\Twofish.class;233
...\iaik\iaik\security\cipher\TwofishKeyGenerator.class;215
...\iaik\iaik\security\cipher\u.class;5368
...\iaik\iaik\security\cipher\v.class;1676
...\iaik\iaik\security\cipher\w.class;1021
...\iaik\iaik\security\dh;0
...\iaik\iaik\security\dh\DHKeyAgreement.class;2895
...\iaik\iaik\security\dh\DHKeyFactory.class;3029
...\iaik\iaik\security\dh\DHKeyPairGenerator.class;5735
...\iaik\iaik\security\dh\DHParameterGenerator.class;3802
...\iaik\iaik\security\dh\DHParameters.class;2671
...\iaik\iaik\security\dh\DHPrivateKey.class;3664
...\iaik\iaik\security\dh\DHPublicKey.class;3724
...\iaik\iaik\security\dh\ESDHKEKParameters.class;3330
...\iaik\iaik\security\dh\ESDHKEKParameterSpec.class;2637
...\iaik\iaik\security\dh\ESDHKeyAgreement.class;4562
...\iaik\iaik\security\dh\ESDHKeyFactory.class;3043
...\iaik\iaik\security\dh\ESDHKeyPairGenerator.class;15084
...\iaik\iaik\security\dh\ESDHParameterGenerator.class;175
...\iaik\iaik\security\dh\ESDHParameters.class;3265
...\iaik\iaik\security\dh\ESDHParameterSpec.class;2147
...\iaik\iaik\security\dh\ESDHPrivateKey.class;4027
...\iaik\iaik\security\dh\ESDHPrivateKeySpec.class;1193
...\iaik\iaik\security\dh\ESDHPublicKey.class;4668
...\iaik\iaik\security\dh\ESDHPublicKeySpec.class;1191
...\iaik\iaik\security\dsa;0
...\iaik\iaik\security\dsa\DSA.class;1639
...\iaik\iaik\security\dsa\DSAKeyFactory.class;3121
...\iaik\iaik\security\dsa\DSAKeyPairGenerator.class;12170
...\iaik\iaik\security\dsa\DSAParameters.class;1995
...\iaik\iaik\security\dsa\DSAParams.class;2083
...\iaik\iaik\security\dsa\DSAPrivateKey.class;3272
...\iaik\iaik\security\dsa\DSAPublicKey.class;3442
...\iaik\iaik\security\dsa\RawDSA.class;5732
...\iaik\iaik\security\keystore;0
...\iaik\iaik\security\keystore\IAIKKeyStore.class;10744
...\iaik\iaik\security\mac;0
...\iaik\iaik\security\mac\HMac.class;1602
...\iaik\iaik\security\mac\HMacMd5.class;248
...\iaik\iaik\security\mac\HMacRipeMd128.class;260
...\iaik\iaik\security\mac\HMacRipeMd160.class;260
...\iaik\iaik\security\mac\HMacSha.class;248
...\iaik\iaik\security\mac\HMacSha256.class;254
...\iaik\iaik\security\mac\HMacSha384.class;258
...\iaik\iaik\security\mac\HMacSha512.class;258
...\iaik\iaik\security\md;0
...\iaik\iaik\security\md\b.class;966
...\iaik\iaik\security\md\Md2.class;2020
...\iaik\iaik\security\md\Md5.class;4117
...\iaik\iaik\security\md\Md5Old.class;3029
...\iaik\iaik\security\md\RipeMd128.class;3400
...\iaik\iaik\security\md\RipeMd160.class;4143
...\iaik\iaik\security\md\SHA.class;3768

...\iaik\iaik\security\md\SHA256.class;4626
...\iaik\iaik\security\md\SHA384.class;692
...\iaik\iaik\security\md\SHA512.class;692
...\iaik\iaik\security\md\SHA64bit.class;5315
...\iaik\iaik\security\pbe;0
...\iaik\iaik\security\pbe\PBEGenParameterSpec.class;371
...\iaik\iaik\security\pbe\PBEParameterGenerator.class;1695
...\iaik\iaik\security\pbe\PBEParameters.class;2571
...\iaik\iaik\security\provider;0
...\iaik\iaik\security\provider\IAIK.class;19475
...\iaik\iaik\security\random;0
...\iaik\iaik\security\random\a.class;1024
...\iaik\iaik\security\random\AnsiRandom.class;1515
...\iaik\iaik\security\random\AutoSeedGenerator.class;3523
...\iaik\iaik\security\random\AWT10SeedGenerator.class;953
...\iaik\iaik\security\random\AWT11SeedGenerator.class;2308
...\iaik\iaik\security\random\FIPS140Test.class;3813
...\iaik\iaik\security\random\HashObjectSeedGenerator.class;1219
...\iaik\iaik\security\random\JDKSeedGenerator.class;532
...\iaik\iaik\security\random\MD5Random.class;293
...\iaik\iaik\security\random\MessageDigestRandom.class;719
...\iaik\iaik\security\random\MetaSeedGenerator.class;891
...\iaik\iaik\security\random\RandomException.class;227
...\iaik\iaik\security\random\RandomInputStream.class;602
...\iaik\iaik\security\random\SecRandom.class;1909
...\iaik\iaik\security\random\SeedGenerator.class;2286
...\iaik\iaik\security\random\SeedGenListener.class;120
...\iaik\iaik\security\random\SHA1Random.class;248
...\iaik\iaik\security\rsa;0
...\iaik\iaik\security\rsa\a.class;783
...\iaik\iaik\security\rsa\Md2RSASignature.class;368
...\iaik\iaik\security\rsa\Md5RSASignature.class;414
...\iaik\iaik\security\rsa\RawRSASignature.class;2239
...\iaik\iaik\security\rsa\RipeMd128RSASignature.class;386
...\iaik\iaik\security\rsa\RipeMd160RSASignature.class;386
...\iaik\iaik\security\rsa\RSAKeyFactory.class;3546
...\iaik\iaik\security\rsa\RSAKeyPairGenerator.class;2435
...\iaik\iaik\security\rsa\RSAPrivateKey.class;5709
...\iaik\iaik\security\rsa\RSAPublicKey.class;3322
...\iaik\iaik\security\rsa\RSASignature.class;3176
...\iaik\iaik\security\rsa\Sha256RSASignature.class;377
...\iaik\iaik\security\rsa\Sha384RSASignature.class;377
...\iaik\iaik\security\rsa\Sha512RSASignature.class;377
...\iaik\iaik\security\rsa\ShaRSASignature.class;368
...\iaik\iaik\security\rsa\SSLRSASignature.class;2068
...\iaik\iaik\security\spec;0
...\iaik\iaik\security\spec\IaikPBEParameterSpec.class;1161
...\iaik\iaik\security\spec\PBEKeyAndParameterSpec.class;594
...\iaik\iaik\utils;16384
...\iaik\iaik\utils\a.class;343
...\iaik\iaik\utils\ArrayEnumeration.class;636
...\iaik\iaik\utils\ASN1InputStream.class;1513
...\iaik\iaik\utils\b.class;723
...\iaik\iaik\utils\Base64Exception.class;163
...\iaik\iaik\utils\Base64InputStream.class;1973
...\iaik\iaik\utils\Base64OutputStream.class;2318
...\iaik\iaik\utils\c.class;331
...\iaik\iaik\utils\CipherInputStream.class;1977
...\iaik\iaik\utils\ConcatEnumeration.class;841
...\iaik\iaik\utils\CriticalObject.class;5223
...\iaik\iaik\utils\CryptoUtils.class;3927
...\iaik\iaik\utils\d.class;331
...\iaik\iaik\utils\e.class;342
...\iaik\iaik\utils\EnhancedByteArrayOutputStream.class;479
...\iaik\iaik\utils\EOFListener.class;179
...\iaik\iaik\utils\ExtendedProperties.class;3468
...\iaik\iaik\utils\ExtendedProvider.class;885
...\iaik\iaik\utils\f.class;575
...\iaik\iaik\utils\Factory.class;1514
...\iaik\iaik\utils\IaikSecurity.class;2232
...\iaik\iaik\utils\InitBufferedInputStream.class;484
...\iaik\iaik\utils\InternalErrorException.class;556
...\iaik\iaik\utils\KeyAndCertificate.class;3424
...\iaik\iaik\utils\LineInputStream.class;1119
...\iaik\iaik\utils\LineOutputStream.class;824
...\iaik\iaik\utils\MacInputStream.class;1393

...\iaik\iaik\utils\MacOutputStream.class;1375
...\iaik\iaik\utils\NotifyEOFInputStream.class;1006
...\iaik\iaik\utils\NumberTheory.class;3143
...\iaik\iaik\utils\PemOutputStream.class;845
...\iaik\iaik\utils\ReplaceInputStream.class;2047
...\iaik\iaik\utils\RFC2253NameParser.class;7664
...\iaik\iaik\utils\RFC2253NameParserException.class;408
...\iaik\iaik\utils\SmtpException.class;208
...\iaik\iaik\utils\SmtpMailer.class;4663
...\iaik\iaik\utils\SSLeayPrivateKey.class;6586
...\iaik\iaik\utils\StreamCopier.class;1756
...\iaik\iaik\utils\TracedInputStream.class;854
...\iaik\iaik\utils\Util.class;14621
...\iaik\iaik\x509;4096
...\iaik\iaik\x509\a.class;767
...\iaik\iaik\x509\attr;0
...\iaik\iaik\x509\attr\AttCertIssuer.class;438
...\iaik\iaik\x509\attr\AttributeCertificate.class;13560
...\iaik\iaik\x509\attr\Holder.class;4448
...\iaik\iaik\x509\attr\IssuerSerial.class;3124
...\iaik\iaik\x509\attr\ObjectDigestInfo.class;4686
...\iaik\iaik\x509\attr\V1Form.class;1672
...\iaik\iaik\x509\attr\V2Form.class;3434
...\iaik\iaik\x509\CertificateFactory.class;2337
...\iaik\iaik\x509\ChainVerifier.class;4380
...\iaik\iaik\x509\extensions;12288
...\iaik\iaik\x509\extensions\AuthorityInfoAccess.class;673
...\iaik\iaik\x509\extensions\AuthorityKeyIdentifier.class;3515
...\iaik\iaik\x509\extensions\BasicConstraints.class;2305
...\iaik\iaik\x509\extensions\CertificateIssuer.class;1277
...\iaik\iaik\x509\extensions\CertificatePolicies.class;1860
...\iaik\iaik\x509\extensions\CRLDistPointsSyntax.class;1853
...\iaik\iaik\x509\extensions\CRLDistributionPoints.class;615
...\iaik\iaik\x509\extensions\CRLNumber.class;1326
...\iaik\iaik\x509\extensions\DeltaCRLIndicator.class;1350
...\iaik\iaik\x509\extensions\ErrorExtension.class;1055
...\iaik\iaik\x509\extensions\ExtendedKeyUsage.class;3599
...\iaik\iaik\x509\extensions\FreshestCRL.class;595
...\iaik\iaik\x509\extensions\HoldInstructionCode.class;1240
...\iaik\iaik\x509\extensions\InfoAccess.class;2360
...\iaik\iaik\x509\extensions\InhibitAnyPolicy.class;1338
...\iaik\iaik\x509\extensions\InvalidityDate.class;1585
...\iaik\iaik\x509\extensions\IssuerAltName.class;1281
...\iaik\iaik\x509\extensions\IssuingDistributionPoint.class;5104
...\iaik\iaik\x509\extensions\KeyUsage.class;2153
...\iaik\iaik\x509\extensions\NameConstraints.class;3284
...\iaik\iaik\x509\extensions\netscape;0
...\iaik\iaik\x509\extensions\netscape\NetscapeBaseUrl.class;1213
...\iaik\iaik\x509\extensions\netscape\NetscapeCaPolicyUrl.class;1233
...\iaik\iaik\x509\extensions\netscape\NetscapeCaRevocationUrl.class;1253
...\iaik\iaik\x509\extensions\netscape\NetscapeCertRenewalUrl.class;1248
...\iaik\iaik\x509\extensions\netscape\NetscapeCertType.class;1772
...\iaik\iaik\x509\extensions\netscape\NetscapeComment.class;1214
...\iaik\iaik\x509\extensions\netscape\NetscapeRevocationUrl.class;1243
...\iaik\iaik\x509\extensions\netscape\NetscapeSSLServerName.class;1244
...\iaik\iaik\x509\extensions\ocsp;0
...\iaik\iaik\x509\extensions\ocsp\NoCheck.class;728
...\iaik\iaik\x509\extensions\PolicyConstraints.class;2650
...\iaik\iaik\x509\extensions\PolicyMappings.class;2042
...\iaik\iaik\x509\extensions\PrivateKeyUsagePeriod.class;2260
...\iaik\iaik\x509\extensions\qualified;4096
...\iaik\iaik\x509\extensions\qualified\BiometricInfo.class;2063
...\iaik\iaik\x509\extensions\qualified\QCStatements.class;2489
...\iaik\iaik\x509\extensions\qualified\structures;4096
...\iaik\iaik\x509\extensions\qualified\structures\a.class;799
...\iaik\iaik\x509\extensions\qualified\structures\BiometricData.class;5285
...\iaik\iaik\x509\extensions\qualified\structures\etsi;0
...\iaik\iaik\x509\extensions\qualified\structures\etsi\QcEuCompliance.class;769
...\iaik\iaik\x509\extensions\qualified\structures\etsi\QcEuLimitValue.class;2594
...\iaik\iaik\x509\extensions\qualified\structures\etsi\QcEuRetentionPeriod.class;1304
...\iaik\iaik\x509\extensions\qualified\structures\QCStatement.class;3742
...\iaik\iaik\x509\extensions\qualified\structures\QCStatementInfo.class;511
...\iaik\iaik\x509\extensions\qualified\structures\SemanticsInformation.class;2620
...\iaik\iaik\x509\extensions\qualified\structures\UnknownQCStatementInfo.class;982
...\iaik\iaik\x509\extensions\ReasonCode.class;1801
...\iaik\iaik\x509\extensions\SubjectAltName.class;1283

...\iaik\iaik\x509\extensions\SubjectDirectoryAttributes.class;2405
...\iaik\iaik\x509\extensions\SubjectInfoAccess.class;670
...\iaik\iaik\x509\extensions\SubjectKeyIdentifier.class;1910
...\iaik\iaik\x509\NetscapeCertRequest.class;1805
...\iaik\iaik\x509\ocsp;8192
...\iaik\iaik\x509\ocsp\a.class;772
...\iaik\iaik\x509\ocsp\BasicOCSPResponse.class;11186
...\iaik\iaik\x509\ocsp\CertID.class;4742
...\iaik\iaik\x509\ocsp\CertificateResponse.class;417
...\iaik\iaik\x509\ocsp\CertStatus.class;2705
...\iaik\iaik\x509\ocsp\extensions;0
...\iaik\iaik\x509\ocsp\extensions\AcceptableResponses.class;1867
...\iaik\iaik\x509\ocsp\extensions\ArchiveCutoff.class;1414
...\iaik\iaik\x509\ocsp\extensions\CrlID.class;2699
...\iaik\iaik\x509\ocsp\extensions\Nonce.class;1032
...\iaik\iaik\x509\ocsp\extensions\ServiceLocator.class;2234
...\iaik\iaik\x509\ocsp\net;0
...\iaik\iaik\x509\ocsp\net\application;0
...\iaik\iaik\x509\ocsp\net\application\ocsp_response.class;1459
...\iaik\iaik\x509\ocsp\net\HttpOCSPRequest.class;2290
...\iaik\iaik\x509\ocsp\net\OCSPContentHandlerFactory.class;513
...\iaik\iaik\x509\ocsp\OCSPException.class;212
...\iaik\iaik\x509\ocsp\OCSPExtensions.class;3445
...\iaik\iaik\x509\ocsp\OCSPRequest.class;10112
...\iaik\iaik\x509\ocsp\OCSPResponse.class;3501
...\iaik\iaik\x509\ocsp\ReqCert.class;4862
...\iaik\iaik\x509\ocsp\Request.class;3055
...\iaik\iaik\x509\ocsp\ResponderID.class;3009
...\iaik\iaik\x509\ocsp\Response.class;715
...\iaik\iaik\x509\ocsp\ResponseBytes.class;2777
...\iaik\iaik\x509\ocsp\RevokedInfo.class;1933
...\iaik\iaik\x509\ocsp\SingleResponse.class;5116
...\iaik\iaik\x509\ocsp\UnknownInfo.class;1040
...\iaik\iaik\x509\ocsp\UnknownResponseException.class;593
...\iaik\iaik\x509\ocsp\utils;0
...\iaik\iaik\x509\ocsp\utils\ResponseGenerator.class;17459
...\iaik\iaik\x509\ocsp\utils\TrustedResponders.class;1563
...\iaik\iaik\x509\PublicKeyInfo.class;4318
...\iaik\iaik\x509\qualified;0
...\iaik\iaik\x509\qualified\QualifiedCertificate.class;5421
...\iaik\iaik\x509\qualified\QualifiedCertificateException.class;233
...\iaik\iaik\x509\qualified\QualifiedCertificateFactory.class;1623
...\iaik\iaik\x509\RevokedCertificate.class;3664
...\iaik\iaik\x509\SimpleChainVerifier.class;1181
...\iaik\iaik\x509\UnknownExtension.class;1044
...\iaik\iaik\x509\V3Extension.class;647
...\iaik\iaik\x509\X509Certificate.class;13414
...\iaik\iaik\x509\X509CRL.class;13200
...\iaik\iaik\x509\X509ExtensionException.class;235
...\iaik\iaik\x509\X509ExtensionInitException.class;444
...\iaik\iaik\x509\X509Extensions.class;11004
...\iaik\iaik_jce_full.jar;637378
...\iaik\META-INF;0
...\iaik\META-INF\JCESIGN.DSA;2124
...\iaik\META-INF\JCESIGN.SF;37777
...\iaik\META-INF\MANIFEST.MF;37724

3.- El producte.

En el PFC implementarem en Java un conjunt de protocols criptogràfics (esquema criptogràfic) bàsics per garantir els requisits de seguretat que cal assegurar en el cas de la gestió d'historials mèdics.

La implementació serà en Java principalment per dos motius: java és multiplataforma, i hi ha llibreries criptogràfiques de Java ben documentades.

---

class Certificat.java

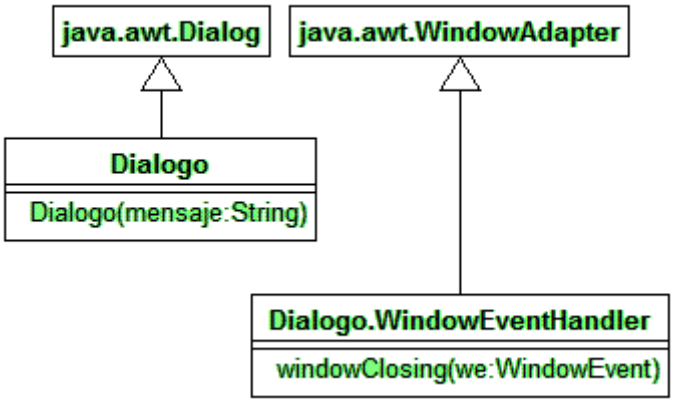
Permet treballar amb Certificats.crt

---

1. Capa de presentació:

Classe Dialogo

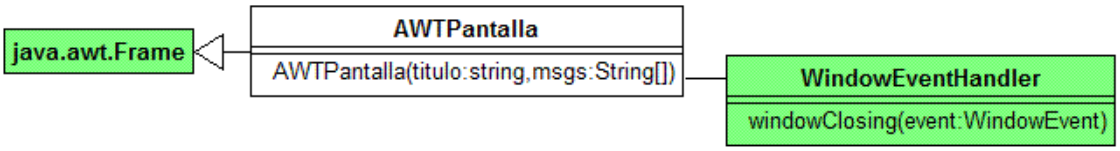
```
import java.awt.*;
import java.awt.event.*;

/**
Classe d'utilitat per enviar missatges gràfics a l'usuari.
*/
public class Dialogo extends Dialog{
	/**
	métode constructor Dialogo
	*@param mensaje: String amb la informacio que volem fer arribar a l'usuari
	*/
	public Dialogo(String mensaje){
		super(new Frame(),"Dialogo no modal", false);
		setSize(mensaje.length()*8 ,100);
		Label l = new Label(mensaje);
		add(l);
		addWindowListener(new Dialogo.WindowEventHandler());
		setVisible(true);
	}

	class WindowEventHandler extends WindowAdapter{
		public void windowClosing(WindowEvent we){
			dispose();
		}
	}
}
```



---

class Pantalla.java

```
import java.awt.*;
import java.awt.event.*;
/** permite presentar mensajes por pantalla */
public class Pantalla extends Frame{
	/** método constructor
	@param titulo : título de la pantalla
	@param msgs  mensaje presentado por pantalla
	*/
	public Pantalla(String titulo, String[] msgs){
		super(titulo);
		setLayout(new FlowLayout());
		addWindowListener(new Pantalla.WindowEventHandler());

		int maxlength= 25;
		for(int i =0; i< msgs.length; i++){
			String s = new String(msgs[i]);
			Label label = new Label(s);
			add(label);
			if(msgs[i].length()> maxlength)  maxlength = msgs[i].length();
		}
		//un digito equivale a 8 puntos horizontales,
		setSize(8 * maxlength,50 + (36 * msgs.length));
		setVisible(true);
	}
	/** classe incrustada que permite cerrar la ventana */
	class WindowEventHandler extends WindowAdapter{
		public void windowClosing(WindowEvent we){
			setVisible(false);
			//System.exit(0);
		}
	}
}
```

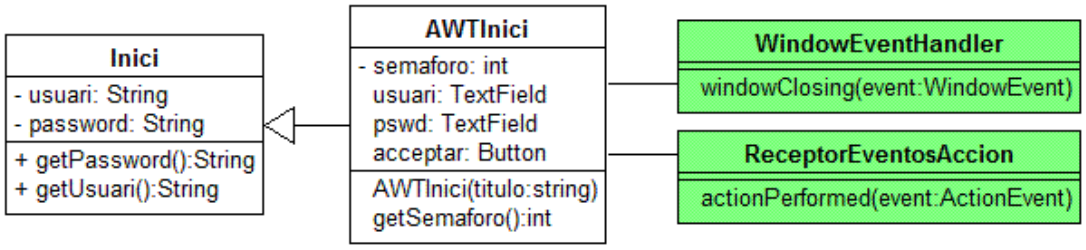


---

class AWTInici.java

```
import java.awt.*;
import java.awt.event.*;

/**
	Ventana para entrada gráfica de datos de login
	Sólo se da una opción para hacer login
*/

public class AWTInici{
	private String usuari;
	private String password;
	/** control de flujo desde la clase que hace la llamada */
	private int semaforo = 0;

	private TextField id = new TextField("Pacient ?", 10);
	private TextField pswd = new TextField("uoc0506", 10);
	private Button b = new Button("Acceptar");

	/**
	las 3 aplicaciones "Sistema..." necesitan el nombre de un fichero p12 para hacer identificación segura.
	los ficheros p12 están protegidos por contraseña: es el valor que se debe introducir en TextField pswd
	@param titulo
	*/
	public AWTInici(String titulo){

		Frame contenedor = new Frame(titulo);
		contenedor.setLayout(new FlowLayout());
		Label l1 = new Label("Usuari :  ");
		//l1.value = "usuari";
		contenedor.add(l1);
		contenedor.add(id);
		Label l2 = new Label("Password: ");
		//l2.setValue("uoc0506");
		contenedor.add(l2);
		contenedor.add(pswd);
		//Button b = new Button("Acceptar");
			//ReceptorEventosAction receptor_eventos_action = new
ReceptorEventosAction();
		b.addActionListener(new AWTInici.ReceptorEventosAction());
		b.setName("b1");
		contenedor.add(b);
		contenedor.setSize(250,150);
		contenedor.addWindowListener(new AWTInici.WindowEventHandler());
		contenedor.setVisible(true);
	}

	/**
	Las aplicaciones "Sistema..." quedan bloquedas mientras el usuario introduce datos en los TextField
	La pulsacion sobre el boton Acceptar genera un Evento que hace cambiar al semaforo.
	@return semaforo
	*/
	public int getSemaforo(){
		return semaforo;
	}

	/**
	La pulsacion sobre el boton Acceptar actualiza el valor de la variable usuari.
	este método es utilizado por las aplicaciones "Sistema..." para conocer el valor introducido por el usuario.
	@return usuari
	*/
	public String getUsuari(){
		return usuari;
	}

	/**
	@param s : idUsuari
	*/
	public void setUsuari(String s){
		usuari = new String(s);
	}

	/**
```

```
	La pulsación sobre el boton Acceptar actualiza el valor de la variable password (mediante un Evento).
	este metodo es utilizado por las aplicaciones "Sistema..." para conocer el valor introducido en el TextField
pswd
	@return password
	*/
	public String getPassword(){
		return password;
	}

	/**
	@param s : */
	public void setPassword(String s){
		password = new String(s);
	}

	/** método para pruebas */
	public static void main(String[] args){

		//se crea un proceso AWT:

		AWTInici inici = new AWTInici("Test");
		while(inici.getSemaforo() == 0);
		System.out.println("¿seguir? : " + inici.getSemaforo());
		while(inici.getSemaforo() == 0);

		System.out.println("Adios!");
		System.exit(0);
	}


	/**Gestor de Eventos de Botón Acceptar */
	class ReceptorEventosAction implements ActionListener{
		/** @param evento generado por el boton de acceptar */
		public void actionPerformed(ActionEvent evento){
			//System.out.println("event.getActionCommand() = " + evento.getActionCommand());
			if(evento.toString().indexOf("on b1") != -1){
				System.out.println("actionPerformed sobre el objeto Boton");
				//se actualiza la variable de instancia: usuari con el valor entrado en pantalla.
				usuari = new String(id.getText());
				//se actualiza password con el valor entrado en pantalla.
				password = new String(pswd.getText());
				//se oculta la ventana
				(b.getParent()).setVisible(false);
				//se libera el flujo en la clase Sistema---
				semaforo = 1;
			}
		}
	}

	/**Gestor de Eventos de Ventana*/
	class WindowEventHandler extends WindowAdapter{
		/** */
		public void windowClosing(WindowEvent we){
			System.out.println("id:  "+ id.getText());
			usuari = new String(id.getText());
			System.out.println("password: "+ pswd.getText());
			password = new String(pswd.getText());
			//se identifica el Frame original a partir del Evento para ocultar la ventana.
			System.out.println(we);
			(we.getWindow()).setVisible(false);
			//se libera el flujo en la clase Sistema---. Esta ventana no concluye la aplicación
			semaforo  = 1;
		}
	}
}
```



---

class AWTGestor.java

```
import java.awt.*;
import java.awt.event.*;

/** finestra principal de les funcions de Gestor */
public class AWTGestor extends Frame{
	/**control del flux del programa */
	private int semaforo = 0;
	/** selecció de la función a executar.
	Aquest artificie evita de cridar mètodes des de dins d'un EventHandler
	Esta información es recogida por la clase Gestor.
	estado == 0 : en espera.
	estado == 1 : alta de usuario
	*/
	private int estado = 0;

	private String taula; //persona
	private String funcio;//insert |
	private String numDNI;
	private String dades;

	/** etiqueta amb informació general */
	private Label label;

	/***/
	public AWTGestor(Gestor g){
		super("Gestor");
		setSize(400,200);
		addWindowListener(new AWTGestor.WindowEventHandler());
		MyMenuBar mmb = new MyMenuBar();
		setMenuBar(mmb);

		StringBuffer sb = new StringBuffer("Gestor: ");
		sb.append(g.getIDgestor());
		label = new Label(sb.toString());
		add(label, BorderLayout.NORTH);

		setLocation(10,10);
		setVisible(true);
		System.out.println("AWTGestor");
	}

	/**
	@return semaforo
	*/
	public int getSemaforo(){
		return semaforo;
	}

	/**
	@return estado
	*/
	public int getEstado(){
		return estado;
	}
	/** permite al Gestor modificat el estado de la ventana */
	public void setEstado(int st){
```

```
		estado = st;
}

/**@return taula*/
public String getTaula(){
		return taula;
}

/** @return funcio*/
public String getFuncio(){
		return funcio;
}

/**@return numDNI*/
public String getNumDNI(){
		return numDNI;
}
public void setNumDNI(){
		numDNI = null;
}
/**@return dades*/
public String getDades(){
		return dades;
}

/***/
public void setDades(){
		dades = null;
}



/**class que defineix el menu del Gestor.*/
class MyMenuBar extends MenuBar{
		public MyMenuBar(){
				super();
				//MenuBar mb = new MenuBar();
				Menu m = new Menu("Aplicacio");
				MenuItem mi = new MenuItem("Imprimir");
				//mi.setState(false);
				mi.setEnabled(false);
				mi.addActionListener(new AWTGestor.EventHandler());
				m.add(mi);
				m.addSeparator();
				MenuItem mi2 = new MenuItem("Exit");
				mi2.addActionListener(new AWTGestor.EventHandler());
				m.add(mi2);
				add(m);

				Menu m0 = new Menu("Usuari");
				MenuItem m0_1 = new MenuItem("Persona");
				m0_1.addActionListener(new AWTGestor.EventHandler());
				m0.add(m0_1);
				MenuItem m0_2 = new MenuItem("Certificat");
				m0_2.addActionListener(new AWTGestor.EventHandler());
				m0.add(m0_2);
				add(m0);
				Menu m1 = new Menu("Pacient");
				MenuItem m1_1 = new MenuItem("Alta-P");
				m1_1.addActionListener(new AWTGestor.EventHandler());
				m1.add(m1_1);
				MenuItem m1_2 = new MenuItem("Consulta-P");
				m1_2.addActionListener(new AWTGestor.EventHandler());
				m1.add(m1_2);
				MenuItem m1_3 = new MenuItem("Baixa-P");
				m1_3.addActionListener(new AWTGestor.EventHandler());
				m1.add(m1_3);
				add(m1);
				Menu m2 = new Menu("Gestor");
				MenuItem m2_1 = new MenuItem("Alta-G");
				m2_1.addActionListener(new AWTGestor.EventHandler());
				m2.add(m2_1);
				MenuItem m2_2 = new MenuItem("Consulta-G");
				m2_2.addActionListener(new AWTGestor.EventHandler());
				m2.add(m2_2);
				MenuItem m2_3 = new MenuItem("Baixa-G");
```

```
                        m2_3.addActionListener(new AWTGestor.EventHandler());
                        m2.add(m2_3);
                        add(m2);
                        Menu m3 = new Menu("Metge");
                        MenuItem m3_1 = new MenuItem("Alta-M");
                        m3_1.addActionListener(new AWTGestor.EventHandler());
                        m3.add(m3_1);
                        MenuItem m3_2 = new MenuItem("Consulta-M");
                        m3_2.addActionListener(new AWTGestor.EventHandler());
                        m3.add(m3_2);
                        MenuItem m3_3 = new MenuItem("Baixa-M");
                        m3_3.addActionListener(new AWTGestor.EventHandler());
                        m3.add(m3_3);
                        add(m3);
                        Menu m4 = new Menu("Historial");
                        MenuItem m4_1 = new MenuItem("Alta-H");
                        m4_1.addActionListener(new AWTGestor.EventHandler());
                        m4.add(m4_1);
                        MenuItem m4_2 = new MenuItem("Consulta-H");
                        m4_2.addActionListener(new AWTGestor.EventHandler());
                        m4.add(m4_2);
                        MenuItem m4_3 = new MenuItem("Baixa-M");
                        m4_3.addActionListener(new AWTGestor.EventHandler());
                        m4.add(m4_3);
                        add(m4);

                }
        }



        /**
        Permite extraer la clave simetrica encriptada de un EnvelopedData en unas condiciones muy determinadas.
        @param envelopedData
        requisitos:
        1.- el envelopedData ha sido creado para este Gestor.
        2.- el envelopedData ha sido creado por un método determinado
        */
//      public OCTET_STRING ClaveSecreta(EnvelopedData envelopedData){
                                                //1.- se espera que componentes envelopedData: 3
                                                //INTEGER = 0
                                                //SET[C] = 1 elements
                                                //SEQUENCE[C] = 3 elements

//                      iaik.asn1.SET set1 = (iaik.asn1.SET)(envelopedData.toASN1Object()).getComponentAt(1);
                        //System.out.println(set1.toString());//SET[C] = 1 elements ; se espera una sequence
//                      iaik.asn1.SEQUENCE seq3 = (iaik.asn1.SEQUENCE) set1.getComponentAt(0);
                                                //3.- se espera que componentes seq3:
                                                //INTEGER = 0
                                                //SEQUENCE[C] = 2 elements
                                                //SEQUENCE[C] = 2 elements
                                                //OCTET STRING = 128 bytes:   = clave simetrica encriptada
//                      iaik.asn1.SEQUENCE seq4 = (iaik.asn1.SEQUENCE) seq3.getComponentAt(2);
                        //System.out.println("sequence 4:" + seq4.toString());//sequence 4:SEQUENCE[C] = 2 elements
Información del algoritmo

//                      iaik.asn1.OCTET_STRING clavesimetrica = (iaik.asn1.OCTET_STRING) seq3.getComponentAt(3);
                                //Object obj3 = clavesimetrica.getValue();
                                //System.out.println(contextspecific.toString());//
                                //System.out.println(clavesimetrica.toString(true));//OCTET STRING = 128 bytes:
86:B6:29:DC:9E:E2:63:15:25:B5:BD:10:34:88:83:BE:10:14:22:9B:3C:2B:F2:94:B2:4B:E4:A2:B3:28:C8:0B:65:00:20:34:
BD:01:8E:E9:C5:05:1C:6B:8E:E6:E3:D5:8C:62:57:56:82:A5:28:22:53:28:07:66:6A:B9:7D:54:F7:1B:56:34:95:22:7F:24:7
9:62:90:2E:5C:9A:01:3B:FF:17:52:A2:72:CC:D2:37:16:49:BB:2F:CE:58:B4:1F:E7:3A:CA:26:1D:E8:DA:BC:20:63:44:5C:
26:C9:AA:9E:3C:94:D4:E6:F6:DF:84:6A:7E:E8:B0:A6:C2:FC:1E:65


//                      return clavesimetrica;
//      }


        /***/
        class EventHandler implements ActionListener, ItemListener{
                public void actionPerformed(ActionEvent event){
                        System.out.println("AWTGestor: pulsado un boton del menu");
                        String seleccion = event.getActionCommand();
```

```
//se cierra la aplicacion SistemaGestor
if("Exit".equals(seleccion)){
		//System.exit(0);
		(label.getParent()).setVisible(false);
		semaforo = 1;
//se oculta la ventana pero no cierra la aplicación SistemaGestor		System.exit(0);

}
//
if("Alta-P".equals(seleccion)){
		label.setText("Alta Pacient");
		estado = 1;
}

//
if("Consulta-P".equals(seleccion)){
		label.setText("Consulta Pacient");
}
//
if("Baixa-P".equals(seleccion)){
		label.setText("Baixa Pacient");
}
//
if("Alta-G".equals(seleccion)){
		label.setText("Alta Gestor");
		estado = 1;
}

//
if("Consulta-G".equals(seleccion)){
		label.setText("Consulta Gestor");
}
//
if("Baixa-G".equals(seleccion)){
		label.setText("Baixa Gestor");
}
//
if("Alta-M".equals(seleccion)){
		label.setText("Alta Metge");
		estado = 1;
}

//
if("Consulta-M".equals(seleccion)){
		label.setText("Consulta Metge");
}
//
if("Baixa-M".equals(seleccion)){
		label.setText("Baixa Metge");
}
//
if("Persona".equals(seleccion)){
		label.setText("Alta identitat física.");
		estado = 1;
}
//
if("Certificat".equals(seleccion)){
		label.setText("Alta de certificat.");
}
//
if("Alta-H".equals(seleccion)){
		label.setText("Alta Historial");
		estado = 1;
}

//
if("Consulta-H".equals(seleccion)){
		label.setText("Consulta Historial");
}
//
if("Baixa-H".equals(seleccion)){
		label.setText("Baixa Historial");
}

//
event = null;
```

```
		}

		public void itemStateChanged(ItemEvent e){
			//semaforo = 1;
		}
	}


	class WindowEventHandler extends WindowAdapter{
		public void windowClosing(WindowEvent e){
			(label.getParent()).setVisible(false);
			semaforo = 1;
			//se oculta la ventana pero no cierra la aplicación SistemaGestor	System.exit(0);
		}
	}
}
```

AWTGestorUsuari.java

```
import java.awt.*;
import java.awt.event.*;

/**
	Presentació gràfica de la aplicació: Part del Gestor:
	permet fer l'entrada de dades per l'alta dels Usuaris a partir d'un Certificat
*/
public class AWTGestorUsuari extends Frame{
	/**informació*/
	protected Label label = new Label("Alta d'un Usuari amb x509");
	/** informació d'entrada*/
	//cap
	/** informació de sortida: nom del fitxer ?.crt */
	protected String nombreCrt;
	/** variable de control*/
	protected int semaforo = 0;
	protected int estado = 0;
	/** AWT */
	protected TextField nombreFT = new TextField("nom de certificat", 20);
	protected Button b = new Button("Acceptar");


	/**
	Constructor
	*/
	public AWTGestorUsuari(){

		super("Gestor: Alta d'usuaris");
		setLayout(new FlowLayout());
		setSize(400,200);//X, Y
		addWindowListener(new AWTGestorUsuari.WindowEventHandler());
		MyMenuBar mmb = new MyMenuBar();
		setMenuBar(mmb);
		add(label);

		add(nombreFT);

		b.addActionListener(new AWTGestorUsuari.EventHandler());
		b.setName("b2");
		add(b);//Button

		setLocation(30,30);
		setVisible(true);
		System.out.println("AWTGestorUsuari");
	}

	/**
	Métode que permet relacionar la entrada de dades amb la aplicació
	*/
	public int getEstado(){
```

```
		return estado;
	}

	public int getSemaforo(){
		return semaforo;
	}

	/***/
	public void setEstado(int i){
		estado = i;
	}

	public void setSemaforo(int i){
		semaforo = 1;
	}
	/** métode de entrega d'informació a la aplicació */
	public String getNombreCrt(){
		return nombreCrt;
	}


	/** Clase incrustada per definir el menu de AWTMetge */
	class MyMenuBar extends MenuBar{
		public MyMenuBar(){
			super();
			//MenuBar mb = new MenuBar();
			Menu m = new Menu("Aplicacio");
			MenuItem mi = new MenuItem("Imprimir");
			//mi.setState(false);
			mi.setEnabled(false);
			mi.addActionListener(new AWTGestorUsuari.EventHandler());
			m.add(mi);
			m.addSeparator();
			MenuItem mi2 = new MenuItem("Exit");
			mi2.addActionListener(new AWTGestorUsuari.EventHandler());
			m.add(mi2);
			add(m);
		}
	}
	/** Classe incrustada per tal de gestionar els event de menú , boton */
	class EventHandler implements ActionListener, ItemListener{
		public void actionPerformed(ActionEvent evento){
			System.out.println("Event Handler: " + evento.getActionCommand());
			String seleccion = evento.getActionCommand();
			if("Exit".equals(seleccion)){
				Frame f0 = (Frame)label.getParent();
				estado = -1;
				f0.setVisible(false);
				//semaforo  = 1;//seguir
			}

			if(evento.toString().indexOf("on b2") != -1){
				System.out.println("actionPerformed sobre el objeto Boton");
				//se actualiza la variable de instancia: nombreCrt con el valor entrado en
pantalla.
				nombreCrt = nombreFT.getText();
				estado = 1;
				//se oculta la ventana
				(b.getParent()).setVisible(false);
				//se libera el flujo
				//semaforo = 1;
			}


		}
		/** captura de ItemEvent */
		public void itemStateChanged(ItemEvent evento){



			System.out.println("EventHandler:  "+  evento.paramString()+
evento.getItem().toString()+evento.getItemSelectable().toString());
			ItemSelectable lista2 = evento.getItemSelectable(); //e.getActionCommand();
			evento.paramString();
			evento.getItem().toString();
			evento.getStateChange();
```

```
                        //lista.addNotify();
                }
        }

        /**clase incrustada para cerra la ventana */
        class WindowEventHandler extends WindowAdapter{
                public void windowClosing(WindowEvent evento){
                        Frame f0 = (Frame)label.getParent();
                        estado = -1;
                        f0.setVisible(false);
                        //semaforo = 1; //seguir
                }
        }
}
```

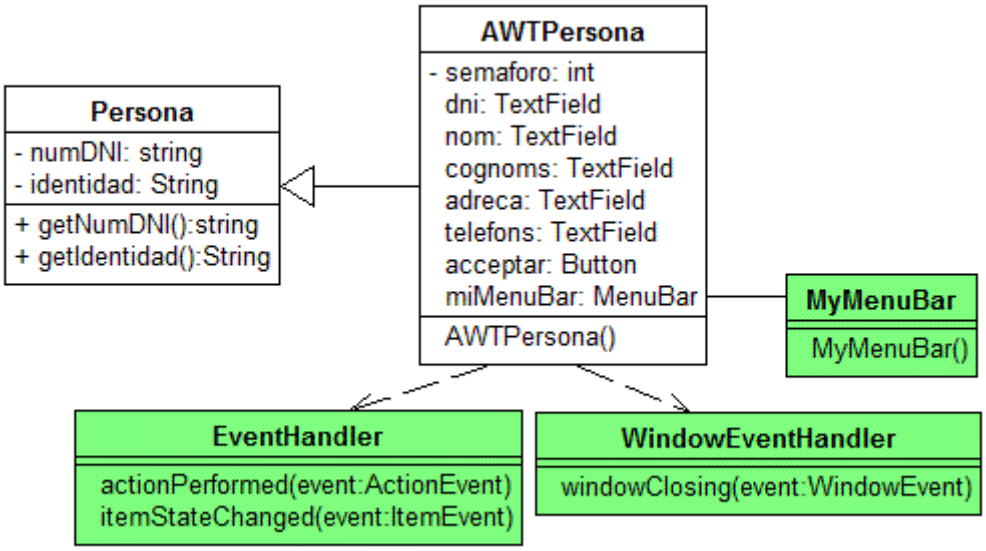
---

class AWTPersona.java

```
import java.awt.*;
import java.awt.event.*;
import java.lang.StringBuffer;

/** clase grafica que utilizará el Gestor para introducir la identidad fisica */
public class AWTPersona extends Frame{
        /**  control del flujo en la clase que hace la llamada. */
        private int semaforo  = 0;
        //private int estado = 0;
        /** Despues de pulsar el boton "Aceptar" contendra la identidad fisica de una persona */
        private String identidad;
        /** Aquest valor no pot ser null */
        private String numDNI;

        Button b ; //boton que permite acceptar y guardar los datos
        Label label = new Label("DADES PERSONALS"); //etiqueta de información
        Label l0 = new Label("DNI      ");
        TextField f0 = new TextField("",10);
        Label l1 = new Label("nom      ");
        TextField f1 = new TextField("",10);
        Label l2 = new Label("cognoms  ");
        TextField f2 = new TextField("",10);
        Label l3 = new Label("adreça   ");
        TextField f3 = new TextField("",10);
        Label l4 = new Label("telefons:");
        TextField f4 = new TextField("",10);// campos de entrada de datos

        Label x = new Label("     "); // artificio para dar formato a la pantalla.

        /** Constructor */
        public AWTPersona(){
                super("Entrada dades personals");
                setLayout(new GridLayout(7,2));
                setSize(400,200);
                addWindowListener(new AWTPersona.WindowEventHandler());
                MyMenuBar mmb = new MyMenuBar();
                setMenuBar(mmb);
                add(label);             add(x);
                add(l0);                add(f0);
                add(l1);                add(f1);
                add(l2);                add(f2);
                add(l3);                add(f3);
                add(l4);                add(f4);
                b = new Button("Acceptar");
                b.addActionListener(new AWTPersona.EventHandler());
                b.setName("b4");
                add(b);

                setLocation(15,15);
                setVisible(true);
        }
```

```
	/**
	control del fluxe en la classe que fa la crida
	@return semaforo
	*/
	public int getSemaforo(){
		return semaforo;
	}

	/**
	@return numDNI
	*/
	public String getNumDNI(){
		return numDNI;
	}

	/**
	@return identitat permet introduir en el sistema els valors dels TextField
	*/
	public String getIdentidad(){
		return identidad;
	}


	/** class incrustada per definir el menu */
	class MyMenuBar extends MenuBar{
		public MyMenuBar(){
			super();
			//MenuBar mb = new MenuBar();
			Menu m = new Menu("Aplicacio");
			MenuItem mi = new MenuItem("Imprimir");
			//mi.setState(false);
			mi.setEnabled(false);
			mi.addActionListener(new AWTPersona.EventHandler());
			m.add(mi);
			m.addSeparator();
			MenuItem mi2 = new MenuItem("Exit");
			mi2.addActionListener(new AWTPersona.EventHandler());
			m.add(mi2);
			add(m);
		}
	}
	/**clase que permite la gestion de eventos de menu y boton */
	class EventHandler implements ActionListener, ItemListener{
		public void actionPerformed(ActionEvent event){
			String seleccion = event.getActionCommand();

			if("Exit".equals(seleccion)){
				 //volem que la aplicació acabi en SistemaPacient, per tant no podem fer
System.exit(0);
				(b.getParent()).setVisible(false);
				//se libera el flujo en la clase que hizo la llamada
				semaforo = 1;
			}

			if(event.toString().indexOf("on b4") != -1){
				System.out.println("Action Performed sobre el objeto Boton");
				//Actualitza el valor de la variable numDNI
				numDNI = f0.getText();
				//Actualitza el valor de la variable identitat
				StringBuffer sb = new StringBuffer(f1.getText());
				sb.append(", ");
				sb.append(f2.getText());
				sb.append(", ");
				sb.append(f3.getText());
				sb.append(", ");
				sb.append(f4.getText());
				identidad = sb.toString();
				//oculta aquesta Finestra
				(b.getParent()).setVisible(false);
				//lliber el flux d'instruccions en la clase que ha fet la crida
				semaforo = 1;
			}
		}
		public void itemStateChanged(ItemEvent event){
		}
	}
```

```
	/** clase incrustada que permite la gestion de eventos. */
	class WindowEventHandler extends WindowAdapter{
		public void windowClosing(WindowEvent event){
			//seguir = 1;
			//Volem que la aplicació acabi en SistemaPacient, per tant no podem fer System.exit(0);
			(b.getParent()).setVisible(false);
			//se libera el flujo en la clase Sistema---
			semaforo = 1;
		}
	}


	/** método para pruebas */
	public static void main(String[] args){
		//se crea un proceso AWT:
		AWTPersona inici = new AWTPersona();
		while(inici.getSemaforo() == 0);
		System.out.println("Adios!");
		System.exit(0);
	}
}
```



AWTMetge.java

```
import java.awt.*;
import java.awt.event.*;

/** Presentació gràfica de la aplicació: Part del Metge */
public class AWTMetge extends Frame{
	/**control de fluxe */
	private int semaforo = 0;
	/**control de fluxe */
	private int estado = 0;
	/** informació per l'usuari : estat */
	Label label = new Label("Aplicació");

	private Pacient pacientSeleccionat;
	Label pLabel = new Label("*");
	private Historial historialSeleccionat;
	Label hLabel = new Label("*");
	private Visita visitaSeleccionada;
	Label vLabel = new Label("*");
```

```
/** Constructor */
public AWTMetge(){
	super("Metge");
	setLayout(new FlowLayout());
	setSize(400,200);//X, Y
	addWindowListener(new AWTMetge.WindowEventHandler());
	MyMenuBar mmb = new MyMenuBar();
	setMenuBar(mmb);
	//Label l = new Label("Aplicació");
	add(label);
	if(pacientSeleccionat != null) pLabel.setText(pacientSeleccionat.toString());
	if(historialSeleccionat != null) hLabel.setText(historialSeleccionat.toString());
	if(visitaSeleccionada != null) vLabel.setText(visitaSeleccionada.toString());
	add(pLabel);
	add(hLabel);
	add(vLabel);
	setLocation(10,10);
	setVisible(true);
}
/**Métode que permet sincronitzar la presentació gràfica amb la aplicació*/
public int getSemaforo(){
	return semaforo;
}

/**
Pot llançar Exception
*/
public Pacient getPacient(){
			return pacientSeleccionat;
}
/**
Pot llançar Exception
*/
public Historial getHistorial(){
			return historialSeleccionat;
}
/**
Pot llançar Exception
*/
public Visita getVisita(){
			return visitaSeleccionada;
}

/** métode redundant. Util per fer proves.*/
public void setPacient(String s){
	//pacientSeleccionat = s;
}


/** métode sense desenvolupar */
public void setVisita(){
	//visitaSeleccionada = doConsultVisita();
	//doConsultVisita();
}


/**
	métode que permet controlar l'execució del fluxe
*/
public void procesar(){
	if(estado == 1) {
			estado = 0;
			doLlista();
	}
}

/** permet seleccionar un pacient a partir de la consulta de la llista de pacients assignats al metge */
private void doLlista(){
	//obrir AWTMetgeLlista
	AWTMetgeLlista ml = new AWTMetgeLlista();
	//esperar
	while(ml.getSemaforo() == 0);

	//pacient = ml.getPacient();
```

```
                //ml.setVisible(false);
                pacientSeleccionat = ml.getPacientSeleccionat();
                System.out.println(pacientSeleccionat.getIDpacient());
        }
        /***/
        private void doConsultar(){
                //obrir AWTMetgeHistorial
                //AWTMetgeHistorial mh = new AWTMetgeHistorial(pacientSeleccionat);
                //PROCEDURE
        }

        /** */
        private void doAfegir(){
                //AWTMetgeVisitaAfegir mva = new AWTMetgeVisitaAfegir();

        }
        /***/
        private void doConsultVisita(){

        }
        /** Clase incrustada per definir el menu de AWTMetge */
        class MyMenuBar extends MenuBar{
                public MyMenuBar(){
                        super();
                        //MenuBar mb = new MenuBar();
                        Menu m = new Menu("Aplicacio");
                        MenuItem mi = new MenuItem("Imprimir");
                        //mi.setState(false);
                        mi.setEnabled(false);
                        mi.addActionListener(new AWTMetge.EventHandler());
                        m.add(mi);
                        m.addSeparator();
                        MenuItem mi2 = new MenuItem("Exit");
                        mi2.addActionListener(new AWTMetge.EventHandler());
                        m.add(mi2);
                        add(m);
                        Menu m2 = new Menu("Pacients");
                        MenuItem m2_1 = new MenuItem("Llista");
                        m2_1.addActionListener(new AWTMetge.EventHandler());
                        m2.add(m2_1);
                        add(m2);
                        Menu m3 = new Menu("Historial");
                        MenuItem m3_1 = new MenuItem("Consultar");
                        m3_1.addActionListener(new AWTMetge.EventHandler());
                        m3.add(m3_1);

                        MenuItem m3_2 = new MenuItem("Editar");
                        m3_2.addActionListener(new AWTMetge.EventHandler());
                        m3.add(m3_2);

                        add(m3);
                        Menu m4 = new Menu("Visita");
                        MenuItem m4_1 = new MenuItem("Afegir");
                        m4_1.addActionListener(new AWTMetge.EventHandler());
                        m4.add(m4_1);
                        MenuItem m4_2 = new MenuItem("Consult.Visita");
                        m4_2.addActionListener(new AWTMetge.EventHandler());
                        m4.add(m4_2);
                        add(m4);
                }
        }
        /** Classe incrustada per tal de gestionar els event de menú */
        class EventHandler implements ActionListener, ItemListener{
                public void actionPerformed(ActionEvent e){
                        String seleccion = e.getActionCommand();

                        if("Exit".equals(seleccion)){
                                //Volem que la aplicació es tanqui en SistemaMetge per tant no podem fer
System.exit(0);
                                Frame f0 = (Frame)label.getParent();
                                f0.setVisible(false);
                                semaforo = 1;
                        }
                        if("Llista".equals(seleccion)) {
                                label.setText("Llista");
                                //artifici per tal de evitar obrir una finestra des d'un Gestor d'Events.
```

```
                                        estado = 1; // doLlista();
                                }
                                if("Consultar".equals(seleccion)) label.setText("Consultar");
                                if("Afegir".equals(seleccion)) label.setText("Afegir");;
                                if("Consult.Visita".equals(seleccion)) label.setText("Consult.Visita");
                        }
                        public void itemStateChanged(ItemEvent e){

                        }

                }


                class WindowEventHandler extends WindowAdapter{
                        public void windowClosing(WindowEvent e){
                                //Volem que la aplicació es tanqui en SistemaMetge per tant no podem fer System.exit(0);
                                Frame f0 = (Frame)label.getParent();
                                f0.setVisible(false);

                                semaforo = 1;
                        }
                }
}
```

AWTPacient.java

```
import java.awt.*;
import java.awt.event.*;
import java.lang.StringBuffer;

public class AWTPacient extends Frame{
        private Historial historial;
        private int semaforo  = 0;
        private int estado = 0;

        Button b ;
        Label label;
        Label l1;
        Label l2;
        /** */
        public AWTPacient(Pacient pacient){
                super("Pacient");
                //setLayout(new BorderLayout());
                setSize(400,200);
                addWindowListener(new AWTPacient.WindowEventHandler());
                MyMenuBar mmb = new MyMenuBar();
                setMenuBar(mmb);

                StringBuffer bs = new StringBuffer(pacient.getIDpacient());
                bs.append(" ");
                bs.append(pacient.getCollectiu());
                label = new Label(bs.toString());
                add(label, BorderLayout.NORTH);
                l1 = new Label("");
                add(l1, BorderLayout.CENTER);
                l2 = new Label("H");
                add(l2);
                b = new Button("Consultar");
                b.addActionListener(new AWTPacient.EventHandler());
                b.setName("b3");
                add(b, BorderLayout.SOUTH);
                setLocation(10,10);
                setVisible(true);
        }

        /** control del fluxe en la classe que fa la crida */
        public int getSemaforo(){
                return semaforo;
        }

        /** permite atender al evento de boton */
        public int getEstado(){
```

```
		return estado;
	}

	/** definición de las acciones en funcion del estado */
	public void procesar(){
		if (estado == 0) ;
		if (estado == 1) doHistorial();
	}

	/**  métode que permet sol·licitar al Gestor l'historial */
	public void doHistorial(){
		System.out.println("Solicitando historial");
		//historial = unHistorial;
		l2.setText("Historial: ");
		estado = 0;
	}

	/**
	@return historial */
	public Historial getHistorial(){
		return historial;
	}

	/** class incrustada per definir el menu */
	class MyMenuBar extends MenuBar{
		public MyMenuBar(){
			super();
			//MenuBar mb = new MenuBar();
			Menu m = new Menu("Aplicacio");
			MenuItem mi = new MenuItem("Imprimir");
			//mi.setState(false);
			mi.setEnabled(false);
			mi.addActionListener(new AWTPacient.EventHandler());
			m.add(mi);
			m.addSeparator();
			MenuItem mi2 = new MenuItem("Exit");
			mi2.addActionListener(new AWTPacient.EventHandler());
			m.add(mi2);
			add(m);
		}
	}
	class EventHandler implements ActionListener, ItemListener{
		public void actionPerformed(ActionEvent event){
			String seleccion = event.getActionCommand();

			if("Exit".equals(seleccion)){
				 //volem que la aplicació acabi en SistemaPacient, per tant no podem fer
System.exit(0);
				(b.getParent()).setVisible(false);
				//se libera el flujo en la clase Sistema---
				semaforo = 1;
			}

			if(event.toString().indexOf("on b3") != -1){
				System.out.println("Action Performed sobre el objeto Boton");
				l1.setText("Consultando Historial");
				estado = 1;
			}
		}
		public void itemStateChanged(ItemEvent event){
			//
		}
	}

	class WindowEventHandler extends WindowAdapter{
		public void windowClosing(WindowEvent event){
			//seguir = 1;
			//Volem que la aplicació acabi en SistemaPacient, per tant no podem fer System.exit(0);
			(b.getParent()).setVisible(false);
			//se libera el flujo en la clase Sistema---
			semaforo = 1;
		}
	}
}
```

2. Capa de negoci.
2.1. Infraestructura criptologica.

---

class SignerManager.java

```
import java.security.PrivateKey;
import java.security.PublicKey;
import iaik.security.provider.IAIK;
import iaik.x509.X509Certificate;
import java.security.*;
import iaik.pkcs.pkcs7.*;
import iaik.asn1.DerCoder;
import iaik.asn1.structures.AlgorithmID;
import iaik.asn1.ASN1Object;
import iaik.utils.Util;

/**
 * Describe class <code>SignerManager</code> here.
 * La signatura d'un text requereix una clau privada.
 * produeix un objecte pkcs7-signedData
 * @author <a href="mailto:jcastellar@uoc.edu">Jordi Castella-Roca</a>
 * @version 1.0
 */
public class SignerManager{
    private PrivateKey _privateKey;
    private PublicKey _publicKey;
    private X509Certificate[] _chain;

    /**
     * Creates a new <code>SignerManager</code> instance.
     *
     * @param p12File a <code>String</code> value
     * @param password a <code>String</code> value
     */
    public SignerManager(String p12File, String password){

            P12 p12 = null;//P12 es una clase creada para simplificar un PKCS12

            try{
                p12 = new P12(p12File, password);
            } catch(Exception e){
                e.printStackTrace();
                System.exit(0);
            }

            _privateKey = p12.getPrivateKey();
            _publicKey = p12.getPublicKey();
            _chain = p12.getCertificates();

    }

    /**
     * Describe <code>signData</code> method here.
     * Transforma un text en clar en un array de byte pkcs7-signedData aprofitant la funcionalitat de la classe SignedData
     * @param dataIn a <code>String</code> value
     * @return a <code>byte[]</code> value
     */
    public byte[] signData(String dataIn){

            byte[] p7Enc = null;
            SignedData p7 = new SignedData(dataIn.getBytes(), SignedData.EXPLICIT);
            p7.setCertificates(_chain);
            SignerInfo signer = new SignerInfo(new IssuerAndSerialNumber(_chain[0]),
                                                    AlgorithmID.sha1,
                                                    _privateKey);
            try{
                p7.addSignerInfo(signer);
            }catch(Exception e){
                e.printStackTrace();
```

```
            System.exit(0);
        }

        try{
            p7Enc = p7.getEncoded();
        }catch(Exception e){
            e.printStackTrace();
            System.exit(0);
        }

        return p7Enc;
    }

    /**
     * Describe <code>verifySignature</code> method here.
     * L'objecte pkcs7-signedData ha estat realitzat amb la PrivateKey,
     * La verificació es fa amb la PublicKey del mateix usuari.
     * Es compara el valor data amb el producte de la desencriptació de p7.
     * @param p7 a <code>byte[]</code> value
     * @param data a <code>byte[]</code> value
     */
    public void verifySignature(byte[] p7, byte[] data){

        AlgorithmID[] algIDs = { AlgorithmID.sha1, AlgorithmID.md5 };

        SignedData signature = null;

        try{
            signature = new SignedData(data, algIDs);
        }catch(Exception e){
            e.printStackTrace();
            System.exit(0);
        }

        ASN1Object objExpP7 = null;

        try{
            objExpP7 = DerCoder.decode(p7);
        }catch(Exception e){
            e.printStackTrace();
            System.exit(0);
        }

        try{
            signature.decode(objExpP7);
        }catch(Exception e){
            e.printStackTrace();
            System.exit(0);
        }

        try{
            SignerInfo[] signerInfos = signature.getSignerInfos();
            for (int i=0; i < signerInfos.length; i++){
                    //System.out.println(signature.verify(i));
                    signature.verify(i);
                    //System.out.println("ERr: "+i);
            }
        }catch(Exception e){
            e.printStackTrace();
            System.exit(0);
        }
    }

}
```

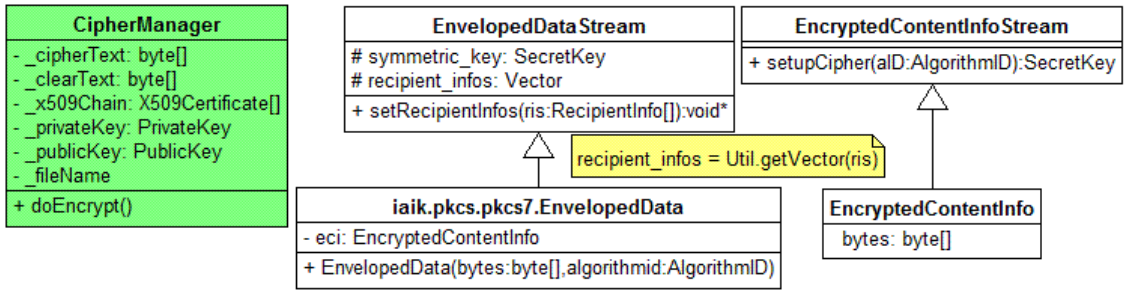
CipherManager.java

```
import java.security.cert.Certificate;
import iaik.x509.X509Certificate;
import java.security.PrivateKey;
import java.security.PublicKey;
import iaik.pkcs.pkcs7.EnvelopedData;
import iaik.asn1.structures.AlgorithmID;
import java.security.NoSuchAlgorithmException;
import iaik.pkcs.pkcs7.RecipientInfo;
import iaik.asn1.ASN1Object;
import iaik.asn1.DerCoder;
import iaik.pkcs.pkcs7.EncryptedContentInfo;
import iaik.pkcs.PKCSException;
import iaik.asn1.CodingException;
import iaik.pkcs.PKCSParsingException;
import java.security.InvalidKeyException;


//import javax.crypto.SecretKey;

/** Utilidad: generación de EnvelopedData */
public class CipherManager{
   private byte[] _cipherText;
   private byte[] _clearText;
   private X509Certificate[] _x509Chain;
   private PrivateKey _privateKey;
   private PublicKey _publicKey;
   private String _fileName;
   //private SecretKey secretKey;

   /***/
   public CipherManager(String fileName)
   {
                        _fileName = new String(fileName);
   }

   public CipherManager()
   {

   }

   public void setPrivateKey(PrivateKey privateKey)
   {
           _privateKey = privateKey;
   }

   //Optional!
   //public void setPublicKey(PublicKey publicKey)
   //{
   //      _publicKey = publicKey;
   //}

   public void setCertificateChain(X509Certificate[] x509Chain)
   {
           _x509Chain = x509Chain;
   }

           /**
           @param clearText  Es el texto en claro que queremos encryptar (con una clave simetrica)
           Se incluye  la clave simetrica encryptada con una PublicKey (la del destinatario)
           */
   public void encrypt(byte[] clearText)
   {
           copyIN(clearText);

           try{
              doEncrypt();
           }catch(Exception e){
              e.printStackTrace();
              System.exit(0);
           }
   }
```

```
        /**
        @param cipherText  es el texto encryptado con una clave simetrica
        */
public void decrypt(byte[] cipherText)
{


        copyOUT(cipherText);

        try{
           doDecrypt();
        }catch(Exception e){
           e.printStackTrace();
           System.exit(0);
        }
}
        /** método sin parametros (utiliza el contenido de las variables) que llama a doDecrypt
        */
public void decrypt()
{
        try{
           doDecrypt();
        }catch(Exception e){
           e.printStackTrace();
           System.exit(0);
        }
}
        /**
        @return _cipherText contiene el texto cifrado */
public byte[] getEncryptedData()
{
        return _cipherText;
}
        /**
        @return _clearText contiene el texto en claro */
public byte[] getDecryptedData()
{
        return _clearText;
}

        /** permite volcar el contenido de un array byte "in" en el array byte _clearText
        @param in */
private void copyIN(byte[] in)
{
        int i;

        _clearText = new byte[in.length];

        for(i=0;i<in.length;i++){
           _clearText[i]=in[i];
        }

}
        /**¡nos permite manipular el texto encryptado!
        @param in:
        */
private void copyOUT(byte[] in)
{
        int i;

        _cipherText = new byte[in.length];

        for(i=0;i<in.length;i++){
           _cipherText[i]=in[i];
        }

}
        /** Es el método que encierra el algoritmo de encriptación: TripleDES in CBC
        De hecho la encryptacion se realiza en la clase EnvelopedData
        EnvelopedData es una estructura ASN1Object (en formato DER) que puede ser "analizada" para
        extraer la clave simetrica al objeto de poder reutilizarla en otros cifrados.
        */
private void doEncrypt()throws NoSuchAlgorithmException,
                                        PKCSException
{
        int i;
```

```
        //use TripleDES in CBC mode for encrypting the content
        EnvelopedData enveloped_data = new EnvelopedData(_clearText, AlgorithmID.des_EDE3_CBC);

//analizar enveloped_data?
System.out.println(enveloped_data.getEncryptedContentInfo().toString());


        // For each intended recipient, create a RecipientInfo object, and add all RecipientInfos to
        // the EnvelopedData structure by calling the setRecipientInfos method, e.g. (assuming to add
        // two recipients with corresponding certificates cert1 and cert2; currently only the PKCS#1
        // rsaEncryption is supported as key- encryption algorithm):

        RecipientInfo[] recipients = new RecipientInfo[_x509Chain.length];

        for(i=0;i<_x509Chain.length;i++){
           recipients[i] = new RecipientInfo(_x509Chain[i], AlgorithmID.rsaEncryption);
        }

        enveloped_data.setRecipientInfos(recipients);

        // Prepare the EnvelopedData object for transmission by transforming it into an ASN1Object
        // or immediately encoding it. This step also will perform the encryption of the symmetric
        // content-encryption key for each participated recipient.
        // ASN1Object obj = enveloped_data.toASN1Object();
        // respectively

        _cipherText = enveloped_data.getEncoded();

}

/** Es el método básico de desencriptacion. (sobre un objeto ASN1Object)
a partir de la PrivateKey del Gestor = CipherManager se puede recuperar la clave simetrica y obtener el
texto en claro.
Este metodo es return = null; El resultado se guarda en la variable _clearText.
*/
private void doDecrypt()throws CodingException,
                                        PKCSParsingException,
                                        InvalidKeyException,
                                        NoSuchAlgorithmException,
                                        PKCSException
{


        // If the EnvelopedData is supplied as DER encoding, first decode it to an ASN1Object:
        ASN1Object obj = DerCoder.decode(_cipherText);

        System.out.println(obj.toString());
        // Create an EnvelopedData structure from the supplied ASN.1 object, and parse the internal structure:
        EnvelopedData enveloped_data = new EnvelopedData(obj);
        // Get information about the inherent EncryptedContentInfo:
        EncryptedContentInfo eci = (EncryptedContentInfo)enveloped_data.getEncryptedContentInfo();

        //System.out.println("Content type: "+eci.getContentType().getName());
        //System.out.println("Content encryption algorithm: "+eci.getContentEncryptionAlgorithm().getName());


        // Get information about the included RecipientInfos:

        RecipientInfo[] recipients = enveloped_data.getRecipientInfos();

        //System.out.pritnln("Included RecipientInfos:");
        //
        //for (int i=0; i < recipients.length; i++) {
        //    System.out.print("Recipient "+(i+1)+":");
        //    System.out.println(recipients[i].getIssuerAndSerialNumber());
        //}


        // Use some recipient s specific private key to decrypt the inherent encrypted secret key
        // to be used for subsequently performing encrypted-content decryption:

        //setup cipher for recipient 1:
        int recipientInfoIndex = 0;
        enveloped_data.setupCipher(_privateKey, recipientInfoIndex);
```

```
            // Unlike the stream supporting EnvelopedDataStream class where the setupCipher method
            // only initializes the cipher for decryption, whole the encrypted-content decryption
            // already is performed inside the setupCipher method of this class.

            // Get the recovered content:

            _clearText = enveloped_data.getContent();
    }

            /***/
            //public SecretKey getSecretKey(){}

}
```



CertificateFile.java

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import iaik.x509.X509Certificate;
//import java.security.PublicKey;
import java.io.IOException;
import java.io.FileNotFoundException;
//import iaik.security.provider.IAIK;
//import java.security.*;
//import iaik.asn1.structures.Name;
//import iaik.asn1.ObjectID;
import java.security.cert.CertificateException;

/**
Utilidad: recuperar un X509Certificate a partir de un fichero guardado en disco.
Utilizado en Gestor:
*/
public class CertificateFile{
   private X509Certificate x509;


            /**
            usa FileInputStream, InputStream,
            @param fileName : es el nombre completo incluida la dirección
            */
   public CertificateFile(String fileName ) throws FileNotFoundException,IOException, CertificateException
   {
                        //fileName hace referencia al certificado ubicado en bin\doc\pki
                        FileInputStream fileInput = new FileInputStream(fileName);
                        //System.out.println("Nombre OK\n");
                        x509 = new X509Certificate(fileInput); //X509Certificate(InputStream)

   }

            public X509Certificate getCertificate(){
                        return x509;
            }
}
```

CertificateValid.java

```
import iaik.asn1.CodingException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.SignatureException;
import java.security.cert.CertificateEncodingException;
import java.security.cert.CertificateException;
import java.security.cert.CertificateExpiredException;
import java.security.cert.CertificateNotYetValidException;
//import java.security.spec.AlgorithmParameterSpec;
import java.util.Date;


//import iaik.pkcs.pkcs12.PKCS12;
import java.io.FileInputStream;
//import java.security.cert.Certificate;
import iaik.x509.X509Certificate;
//import java.security.PrivateKey;
import java.security.PublicKey;
//import iaik.pkcs.pkcs12.KeyBag;
//import iaik.pkcs.pkcs12.CertificateBag;
//import iaik.pkcs.PKCSParsingException;
//import iaik.pkcs.PKCSException;
import java.io.IOException;
import java.io.FileNotFoundException;
//import iaik.security.provider.IAIK;
```

```
//import java.security.*;
//import iaik.asn1.structures.Name;
//import iaik.asn1.ObjectID;

/**
 * Describe class <code>CertificateValid</code> here.
 */
public class CertificateValid{
	/** Fa referencia a la clau pública de la CA */
	private PublicKey publicKey;
	private boolean caducat;

	/**
	*/
   public CertificateValid(X509Certificate x509) throws CertificateException
   {
	//Validació de date
		java.util.Date date = new java.util.Date();
		caducat = date.after(x509.getNotAfter());
		System.out.println("CertificateValid: caducat:" + caducat);
		if (caducat==true) System.exit(0);

		//Fa la mateixa funcio que Validació de date
		try{
			x509.checkValidity();
		}
		catch(CertificateNotYetValidException cnyve){}
		catch(CertificateExpiredException cee){}
		catch(Exception e){}

		//Validació de signatura:
		try{

			//¿El Certificat que volem validar ha estat signat per CA.? Sí
			//cal trobar la signatura pública de la Ca
			//cal obrir el certificat de la CA i trobar la clau pública de CA
			String direccioWindows = ".\\doc\\pki\\CA.crt";
			CertificateFile cfCA = new CertificateFile(direccioWindows);
			X509Certificate x509CA = cfCA.getCertificate();
			System.out.println(x509CA.toString());
			publicKey = x509CA.getPublicKey();

			x509.verify(publicKey);
			System.out.println("CertificateValid: Certificat validat");

		}
		catch(IOException ioe){System.out.println(ioe);}
		//catch(FileNotFoundException fnfe){System.out.println(fnfe);}
		catch(SignatureException se){System.out.println(se);}
		catch(NoSuchProviderException nspe){System.out.println(nspe);}
		catch(InvalidKeyException ike){System.out.println(ike);}
		catch(NoSuchAlgorithmException nsae){System.out.println(nsae);}
		//catch(CertificateException ce){System.out.println(ce);}


		//Validació d'integritat:
			//comprobació del método de resum

   }

   public void setPublicKey(PublicKey pk){
		publicKey = pk;
   }

   public PublicKey getPublicKey(){
		return publicKey;
   }
}
```

CertificateContent.java

```
import iaik.x509.X509Certificate;
import java.security.PublicKey;
import iaik.security.provider.IAIK;
import java.security.*;
import iaik.asn1.structures.Name;
import iaik.asn1.ObjectID;
import java.security.cert.CertificateException;
import java.math.BigInteger;

/**
 * Describe class <code>CertificateContent</code> here.
 * Utilizada en clase Gestor.
 * Dades del certificat: SerialNumber
 * Dades del Subject: organizationalUnit, dnQualifier, publicKey
 */
public class CertificateContent{
    //private X509Certificate certificate;
    //número de serie: BigInteger
    private BigInteger numSerie;
    /**permite comprobar la integridad del certificado */
    private byte[] fingerprintSHA;
    private PublicKey publicKey;
    private String collectiu;
    private String userID;
    private String nom;
    /***/
    private boolean integritat = false;

            /**
            usa FileInputStream, InputStream, Name,
            @param x509  permite extraer los datos de un X509Certificate
            */
    public CertificateContent(X509Certificate x509) throws CertificateException
    {

                        Name name = (Name)x509.getSubjectDN();
//x509.getSubjectDN(): java.security.Principal  //?? return d = Name
                        //System.out.println("Casting Name name OK\n");
                        //String country = name.getRDN(ObjectID.country);  //¿ObjectID.country?
                        //Está definido en ObjectID: public static ObjectID country = new ObjectID("2.5.4.6", "countryName",
"C");
                        //String stateprovince = name.getRDN(ObjectID.stateOrProvince);
                        //String locality = name.getRDN(ObjectID.locality);
                        //String organization = name.getRDN(ObjectID.organization);
                        //String organizationalUnit = name.getRDN(ObjectID.organizationalUnit);
                        collectiu = name.getRDN(ObjectID.organizationalUnit);
                        String commonName = name.getRDN(ObjectID.commonName);
                        nom = name.getRDN(ObjectID.commonName);
                        //String email = name.getRDN(ObjectID.emailAddress);
                        //String dni = name.getRDN(ObjectID.dnQualifier);
                        userID = name.getRDN(ObjectID.dnQualifier);

                        //System.out.println("Country: "+country);
                        //System.out.println("State or province: "+stateprovince);
                        //System.out.println("Locality: "+locality);
                        //System.out.println("Organization: "+organization);
                        //System.out.println("OrganizationalUnit: "+organizationalUnit);
                        //System.out.println("CertificateContent: CommonName: "+commonName);
                        //System.out.println("Email: "+email);
                        //System.out.println("DNI: "+dni);

                        //String sx509 = toString();
                        //System.out.println(sx509);

                        numSerie = x509.getSerialNumber();
                        publicKey = x509.getPublicKey();
                        //iaik.asn1.structures.AlgorithmID sa = x509.getSignatureAlgorithm();
                        //System.out.println("algorithmID: " + sa.toString());
                        fingerprintSHA = x509.getFingerprintSHA();
    }

    public PublicKey getPublicKey(){
```

```
		return publicKey;
	}

	public String getUserID(){
		return userID;
	}

	public String getCollectiu(){
		return collectiu;
	}

	public String getNom(){
		return nom;
	}

	public BigInteger getNumSerie(){
		return numSerie;
	}

	public byte[] getFingerPrint(){
		return fingerprintSHA;
	}

	private static boolean validar(X509Certificate x509){
			//obtener el fingerprint del certificado incognita.
			x509.getFingerprintSHA();
			//calcular el fingerprint del certificado incognita.
				//identificar el metodo de resum SHA1

			//comparar los dos valores del SHA

			return false;
	}
}
```

SistemaGestor.java

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

import java.io.File;
import java.io.FileNotFoundException;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;
import java.security.cert.CertificateException;
import iaik.x509.X509Certificate;

/**		Sistema Gestor  Inici de la part del Gestor.*/
public class SistemaGestor{
		/** sistema operatiu : fa falta per definir '/' o '\'*/
		private static String so;

   /** El gestor que treballa sobre el SistemaGestor */
		private Gestor gestor;

		/** certificat del gestor: fa falta per identificar el Gestor.
		El x509 es pot identificar pel figerprint o pel numero de serie */
		private X509Certificate x509g;

		/**
		Llista General de Metges del Sistema
		*/
		//private LlistaGeneralMetges lgm;
		/**
		Llista General de Pacients del Sistema.
		*/
		//private LlistaGeneralPacients lgp;

		//private Hastable LlistaGeneralUsuaris lgu; //Hastable: id_usuari -> certificate //Llista de certificats emitits.

		/**cadena de certificats:  P12 */

		/**
		@param unID  quan es coneix el gestor Es el nom del fitxers Gestor.crt i Gestor.p12
		@param pswd  es el password del fichero Gestor.p12*/
		public void setGestor(String unID, String pswd){
			X509Certificate x509g;
			gestor = new Gestor();
			try{
				//abrir el fichero Gestor.crt
				String nomFitxer;
				if(so.equals("Windows")){
					nomFitxer = new String(".\\doc\\pki\\Gestor.crt");
				}else {
					nomFitxer = new String("./doc/pki/Gestor.crt");
				}

				CertificateFile cf = new CertificateFile(nomFitxer);
				x509g = cf.getCertificate();

				//comprobar la validez del certificado
				CertificateValid cv = new CertificateValid(x509g);

				//asignar el certificat al gestor
				gestor.setCertificat(x509g);

				//leer la informacion del gestor
				CertificateContent cc = new CertificateContent(x509g);
				gestor.setPublicKey(cc.getPublicKey());
				//System.out.println("num serie: "+cc.getNumSerie());
				gestor.setNumCertificat(cc.getNumSerie());
				//System.out.println("usuari: "+cc.getUserID());
				gestor.setIDgestor(cc.getUserID());
				//System.out.println("collectiu: "+cc.getCollectiu());

				System.out.println("SG: setGestor:"+ gestor.getIDgestor() + gestor.getCollectiu());

				//abrir el fichero Gestor.p12  y extraer la clave privada del gestor.
```

```
                        String nomFitxer12;
                        if(so.equals("Windows")){
                                nomFitxer12 = new String(".\\doc\\pki\\Gestor.p12");
                        }else {
                                nomFitxer12 = new String("./doc/pki/Gestor.p12");
                        }

                        P12 p12 = null;//P12 es una clase creada para simplificar un PKCS12
                        try{
                        p12 = new P12(nomFitxer12, pswd);
                        } catch(Exception e){
                        e.printStackTrace();
                        System.exit(0);//problema fatal.
                        }

                        if(!(cc.getPublicKey()).equals(p12.getPublicKey())){
                                System.out.println("Error fatal");
                                System.exit(0);
                        }
                        gestor.setPrivateKey(p12.getPrivateKey());
                        //chain = p12.getCertificates();
                        gestor.setChain(p12.getCertificates());

                        System.out.println("SG: setGestor OK");
                        //Se pasa el control a la clase Gestor
                        //ejecuta las funcionalidades de gestor.
                        gestor.procesar();

                }catch(FileNotFoundException fnfe){System.out.println(fnfe);}
                catch(CertificateException ce){System.out.println(ce);}
                catch(IOException ioe){System.out.println(ioe);}
                catch(Exception e){System.out.println(e);}
        }


        /**
        Aplicacio Gestio Historials Medics Segurs.
        */
        public static void main(String[] args){


                //Informar a l'usuari de la funció
                String[] msg1= {"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.                          "};
                //Dialogo d1 = new Dialogo(msg1);
                Pantalla p1 = new Pantalla("Sistema Gestor", msg1);


                /** llanza Exception */
                Security.insertProviderAt(new IAIK(), 2);

                //Cal coneixer el sistema operatiu abans de llegir fitxers.
                sistemaOperatiu();


                //Generar un ambient no static: crear una instancia
                SistemaGestor sg = new SistemaGestor();

                String[] msg2 = {"Part de la aplicació corresponent al Gestor",
                                "permet consultar les dades personals.      "};
                Pantalla p2 = new Pantalla("Sistema Gestor", msg2);


                //filtro
                //if(benvinguda()){
                        System.out.println("*************************************************");
                        System.out.println("*"+"  APLICACIO GESTIO HISTORIALS MEDICS SEGURS     "+"*");
                        System.out.println("*"+"            INICI                             "+"*");
                        System.out.println("*"+"                                              "+"*");
                        System.out.println("*************************************************\n");

                        String[] msg3 = {"Aplicacio Gestio Historials Médics Segurs","Inici", "Cal introduir la
paraula: Gestor"};
```

```
                Pantalla p3 = new Pantalla("Sistema Gestor", msg3);

        //} else return;

        //Ara la aplicació inicia la recollida de dades:
                System.out.println("*****************************************************");
                System.out.println("*"+"  CERTIFICATS                                  "+"*");
                System.out.println("*"+"          recollida de dades                 "+"*");
                System.out.println("*"+"                                           "+"*");
                System.out.println("*****************************************************\n");

                //1.-
                //2.-
                //3.-

        //Ara la aplicació autentica l'Usuari: ¿es Gestor?
                System.out.println("*****************************************************");
                System.out.println("*"+"  GESTOR                                      "+"*");
                System.out.println("*"+"          creacio d'un gestor                "+"*");
                System.out.println("*"+"                                           "+"*");
                System.out.println("*****************************************************\n");
                //sg.autenticacio();


        //getPath();


        //abrir el servidor de RMI
        //RMIServidor rmiservidor = new RMIServidor();
        //RMIObjeto rmiobj = rmiservidor.Servidor();
        //System.out.println(rmiobj.getFromClient());//no esta sincronizado.

        //Comprobació de la base de dades.
        try{
                GestorBD gbd = new GestorBD();
                gbd.creaTables();
                String[] msg5 ={ "Test: Base de Datos OK"};
                Pantalla p5 = new Pantalla("Sistema Gestor", msg5);
        }catch(Exception e){
                String[] msg4 = {"ERROR: BD no accesible"};
                Pantalla p4 = new Pantalla("SistemaGestor : error",msg4);
        }

        //identificació previa
        sg.ferLogin();


        System.out.println("Adeu!");
}


/** usuari i contrasenya
a partir d'una entrada grafica obtenir usuari i password de p12
cal obrir el fitxer protegit Pacient.p12
*/
public void ferLogin(){

        AWTInici inici = new AWTInici("SistemaGestor");
        //dar tiempo a la entrada de datos en la ventana de login
        while(inici.getSemaforo() == 0){
        }

        System.out.println("login:usuari: " + inici.getUsuari());
        System.out.println("login:password: " + inici.getPassword());
        //formar el nombre de un fichero con el usuario entrado en AWT
        //se tiene en cuenta que Windows usa "\" y Linux "/"
        try{
                String p12File;
                String unFile;
                if (so.equals("Windows")){
                        StringBuffer sb = new StringBuffer(".\\doc\\pki\\");
                        unFile = (sb.append(inici.getUsuari())).toString();
                        sb.append(".p12");
                        p12File = new String(sb.toString());
                } else {
                        StringBuffer sb = new StringBuffer("./doc/pki/");
```

```
                                        unFile = (sb.append(inici.getUsuari())).toString();
                                        sb.append(".p12");
                                        p12File = new String(sb.toString());
                                }
                                //abrir el fichero "Gestor.p12"
                                P12 p12 = null;
                                String password = inici.getPassword();//"uoc0506";
                                try{
                                p12 = new P12(p12File, password);
                                        //PrivateKey _privateKey = p12.getPrivateKey();
                                        //PublicKey  _publicKey = p12.getPublicKey();
                                        //X509Certificate[] _chain = p12.getCertificates();

                                //cargar los datos en gestor:
                                System.out.println("login: OK");
                                setGestor(unFile, password);

                                } catch(Exception e){
                                e.printStackTrace();
                                System.exit(0);
                                }
                        }
                        //catch(FileNotFoundException fnfe){System.out.println(fnfe);}
                        //catch(CertificateException ce){System.out.println(ce);}
                        //catch(IOException ioe){System.out.println(ioe);}
                        catch(Exception e){System.out.println(e);}


        }



        /** Fa falta coneixer el Separador de directoris per tal de accedir als fitxers:*/
        private static void sistemaOperatiu(){
                //la clase File depende de la plataforma:
                //aprofitem això per identificar el SO
                File file = new File("file");
                if(file.separator.equals("\\")){
                        so = new String("Windows");
                        System.out.println("OS Windows");
                }
                else{
                        so = new String("Unix");
                        System.out.println("OS: No es Windows");
                }
        }

}
```

Gestor.java

```
import java.util.Vector;
import java.io.File;
import java.util.StringTokenizer;

import java.io.*;
import java.rmi.*;
import java.rmi.registry.*;
import java.rmi.server.*;
import java.net.*;

import java.math.BigInteger;
import iaik.x509.X509Certificate;
import java.security.PrivateKey;
import java.security.PublicKey;

//import iaik.security.provider.IAIK;
//import java.security.*;
import iaik.pkcs.pkcs7.*;
import iaik.pkcs.pkcs7.SignedData;
import iaik.asn1.DerCoder;
//import iaik.asn1.BIT_STRING;
import iaik.asn1.structures.AlgorithmID;
import iaik.asn1.ASN1Object;
import iaik.utils.Util;


/**	Gestor:
	Hace funciones de SignerManager;

*/
public class Gestor extends Usuari{

	/**id_gestor: */
	private String id_gestor;

	/** col·lectiu: */
	private String collectiu = "Administradors";

	/** X509Certificate*/
	private X509Certificate x509g;
	private BigInteger numCertificat;
	private PublicKey publickey;
   private X509Certificate[] _chain;

	/** PKCS#12*/
	private PrivateKey privatekey;

	/** RMI */
	//private String mensajeEntrante;
	Vector RMIClient;
	RMIObjeto rmiobj;



	/** Metode Constructor */
	public Gestor(){
		P12 p12 = null;
		String p12File = ".\\doc\\pki\\Gestor.p12";
		String password = "uoc0506";
		try{
		   p12 = new P12(p12File, password);
		} catch(Exception e){
		e.printStackTrace();
		System.exit(0);
		}
		privatekey = p12.getPrivateKey();
		publickey = p12.getPublicKey();
		_chain = p12.getCertificates();


	}
```

```
	/**
	@return id_gestor*/
	public String getIDgestor(){
		return id_gestor;
	}

	/**
	@return collectiu */
	public String getCollectiu(){
		return collectiu;
	}

	/** @param id  quan es coneix el gestor */
	public void setIDgestor(String id){
		id_gestor = id;
	}

	/** @param unCert*/
	public void setCertificat(X509Certificate unCert){
		x509g = unCert;
	}

	/**@param num : es el numero de serie del certificat asociado a este Gestor */
	public void setNumCertificat(BigInteger num){
		numCertificat = num;
	}

	/**@param pk  la PublicKey del gestor: */
	public void setPublicKey(PublicKey pk){
		publickey = pk;
	}

	/**@param sk  la PrivateKey del gestor: se utiliza para desencriptar Sobre digital y para firmar*/
	public void setPrivateKey(PrivateKey sk){
		privatekey = sk;
	}

	/**@param x509s se obtiene del certificado*/
	public void setChain(X509Certificate[] x509s){
		_chain = x509s;
	}



  /**
   * Describe <code>signData</code> method here.
   * El gestor pot signar les dades que vulgui conservar.
   * al definir SignedData.EXPLICIT el signedData no conte el text en clar
   * faria falta definir SignedData.IMPLICIT per tal que el text en clar fos incluit en el SignedData.
   * @param dataIn a <code>String</code> value
   * @return a <code>byte[]</code> value
   */
  public byte[] signData(String dataIn){

		byte[] p7Enc = null;
		SignedData p7 = new SignedData(dataIn.getBytes(), SignedData.EXPLICIT);
		p7.setCertificates(_chain);
		SignerInfo signer = new SignerInfo(new IssuerAndSerialNumber(_chain[0]), AlgorithmID.sha1,
privatekey);
		try{
		p7.addSignerInfo(signer);
		}catch(Exception e){
		e.printStackTrace();
		System.exit(0);//problema fatal.
		}

		try{
		p7Enc = p7.getEncoded();
		}catch(Exception e){
		e.printStackTrace();
		System.exit(0);
		}

		return p7Enc;
  }
```

```
/**
   * Describe <code>verifySignature</code> method here.
   * a partir de les dades originals, i de la signatura, es pot comprovar la integritat
   * @param p7 a <code>byte[]</code> value
   * @param data a <code>byte[]</code> value
   */
  public void verifySignature(byte[] p7, byte[] data){

                        AlgorithmID[] algIDs = { AlgorithmID.sha1, AlgorithmID.md5 };

                        SignedData signature = null;

                        try{
                        signature = new SignedData(data, algIDs);
                        }catch(Exception e){
                        e.printStackTrace();
                        System.exit(0);
                        }

                        ASN1Object objExpP7 = null;

                        try{
                        objExpP7 = DerCoder.decode(p7);
                        }catch(Exception e){
                        e.printStackTrace();
                        System.exit(0);
                        }

                        try{
                        signature.decode(objExpP7);
                        }catch(Exception e){
                        e.printStackTrace();
                        System.exit(0);
                        }

                        try{
                        SignerInfo[] signerInfos = signature.getSignerInfos();
                        for (int i=0; i < signerInfos.length; i++){
                                //System.out.println(signature.verify(i));
                                signature.verify(i);
                                //System.out.println("ERr: "+i);
              }
                        }catch(Exception e){
                        e.printStackTrace();
                        System.exit(0);
                        }
  }

        /** verifySignatura
        @param crtFile es el nombre completo ".\\doc\pki\\Pacient.crt" del certificado del firmante
        */
        public void verifySignature(byte[] signedDatabyte, String crtFile){
                X509Certificate x509 = null;
                try{
                        CertificateFile cf = new CertificateFile(crtFile);
                        x509 = cf.getCertificate();
                }catch(Exception e){
                        System.out.println("Gestor error "+ e);
                }
                ASN1Object objsd = null;
                try{
                        objsd = DerCoder.decode(signedDatabyte);
                        //System.out.println("objsd " + objsd.toString());
                }catch(iaik.asn1.CodingException e){
                        System.out.println("CodingException " + e);
                }
                SignedData sd = null;
                try{
                        sd = new SignedData(objsd);
                        //System.out.println("sd "+sd.toString(true));
                        //System.out.println("sd "+sd.toString(false));
                }catch(iaik.pkcs.PKCSParsingException e){
                        System.out.println("PKCSParsingException "+e);
                }
```

```
		//String sdobj = new String(sd.getContent());
		//System.out.println("sdobj " + sdobj);

		try{
			sd.verify(x509);
		}catch(Exception e){
			System.out.println("error en la verificacion");
		}

	}



	/** métode per controlar el flux del programa */
	public void procesar(){
		//Per comunicació remota:
		//Servidor de RMI?
		try{

			if(System.getSecurityManager() == null){
				System.setSecurityManager(new RMISecurityManager());
			}
			//java.policy
			//grant {
			//  permission java.net.SocketPermission "localhost:1024","listen"
		//	permission java.net.SocketPermission "*:1024-65535",		"connect,accept";
		//	permission java.net.SocketPermission "*:80", "connect";
			//};
			RMIClient = new Vector();

			rmiobj = new RMIObjeto();
			System.out.println("Servidor: se ha creado el objeto remoto");//1º
			//URL rmi://host:port/remoteobjetname , host y port son optativos.
			Naming.rebind("rmi://localhost:2002/objrmi", rmiobj);
			rmiobj.setSemaforo(0);
			System.out.println("Servidor esperando llamadas ");



			//Per el menu del Gestor:
			AWTGestor awtgestor = new AWTGestor(this);

			//fer temps per que el gestor entri dades en la finestra gràfica.
			while(awtgestor.getSemaforo() == 0){ //la finestra AWTGestor està oberta i el gestor pot
fer transaccions
				//se entra en un bucle que se cerrará cuando se cierre la ventana con el menu
Gestor.

				//Atendemos al objeto remoto
				if(rmiobj.getSemaforo()!=0){
					NeedhamShroeder();
				}

				//Atendemos al menu:
				if(awtgestor.getEstado() == 0); //no hacer nada
				//guardar los datos de un usuario
				if(awtgestor.getEstado() == 1){ //alta de usuario
					this.altaUsuari();
					//terminado
					awtgestor.setEstado(0);

				}
				//selecciona un pacient y da de alta el historial.
				if(awtgestor.getEstado() == 2){
					this.altaHistorial();
					//terminado
					awtgestor.setEstado(0);
				}

			}//fin while
		}//catch(java.rmi.RemoteException re){
			//System.out.println(re);
			//Informar a l'usuari del error
```

```
                        //String[] msg1= {"Error del Servidor RMI: conexion rechazada" };
                        //Dialogo d1 = new Dialogo(msg1);
                        //Pantalla p1 = new Pantalla("Gestor: Servidor RMI ", msg1);
                        //Error fatal.
                //}
                catch(Exception e){System.out.println(e);}
                System.out.println("Gestor.procesar() Fin");
}//fin metodo


/***/
public void atenderRemote(String s){

}


/**
@param id  correspon amb el nom d'un certificat */
public void altaGestor(String id){
                //Trobar el certificat del nou gestor
                        //format del nom del fitxer .crt
                //Validar el certificat
                //Guardar a la base de dades: volem que sigui el SistemaGestor qui ho faci.: gbd.afegirGestor()

                SistemaGestor sg = new SistemaGestor();
                //Generem una estructura de dades per al Sistema Gestor
                //dades
                //0 funció  "afegirGestor" |

                String[] dades = new String[1];
                dades[0] = "afegirGestor";
                //metadades:
                //0 - Quí : 0 idFisica , 2 IdElectronica = num certificat,  3 signatura
                //3 - A quí: id fisica
                //4 - Qué
                //5 - Quan
                //sg.guardar(dades, metadades);
}

/**
*while(awtgestor.getSemaforo() == 0);
*if(awtgestor.getEstado() == 1) //alta de usuario
*/
public void altaUsuari(){
                //Dades que es consultaran al sistema.
                //Entrada de dades per finestra: declarar la finestra de entrada de dades.
                AWTGestorUsuari awtgu = new AWTGestorUsuari();
                while(awtgu.getSemaforo() == 0){
                        //consultar el estado de AWTGestorUsuari
                                //si no se ha pulsado el boton Aceptar:
                                        //Se ha cerrado la ventana de forma irregular:
                        //if(awtgu.getEstado() == -1) awtgu = null;
                                        //Si Se está a la espera:
                        if(awtgu.getEstado() == 0); //no hacer nada = esperar
                                //se ha pulsado el boton Acceptar
                        if(awtgu.getEstado() == 1){
                                //identificar el sistema operativo:
                                String so;
                                String directoris;
                                        File file = new File("file");
                                        if(file.separator.equals("\\")){
                                                directoris = new String(".\\doc\\pki\\");
                                                so = "Windows";
                                                //System.out.println("OS Windows");
                                        }
                                        else{
                                                directoris = new String("./doc/pki/");
                                                so = "Linux";
                                                //System.out.println("OS: No es Windows");
                                        }
                                StringBuffer sb = new StringBuffer(directoris);
                                sb.append(awtgu.getNombreCrt());
                                sb.append(".crt");
                                String nomFitxer = sb.toString();
                                System.out.println("Gestor: alta: " + nomFitxer);
                                try{
```

```
//Analizar el certificado y extraer los datos.
CertificateFile cf = new CertificateFile(nomFitxer);
X509Certificate x509 = cf.getCertificate();
//leer la informacion del certificado
CertificateContent cc = new CertificateContent(x509);
//gestor.setPublicKey(cc.getPublicKey());

//System.out.println("num serie: "+ cc.getNumSerie());
java.math.BigInteger numSerie = cc.getNumSerie();
System.out.println("nunSerie "+ numSerie.toString());

//System.out.println("fp"+ cc.fingerPrint());
byte[] fingerPrint = x509.getFingerprint();

//System.out.println("usuari: "+ cc.getUserID());
String idUsuari = cc.getUserID();
System.out.println("dni " +idUsuari);

String nom = cc.getNom();
System.out.println("nom" + nom);


System.out.println("collectiu: "+ cc.getCollectiu());
String collectiu = cc.getCollectiu();

//es pot utilitzar fins:
java.util.Date fins = x509.getNotAfter();



//Guardar en la base de datos.
GestorBD gbd = new GestorBD();
//(persona(dni CHAR(20), nom CHARACTER VARYING (100),
signedData BLOB, PRIMARY KEY(dni))");
//(pacient(dni CHAR(20), nom CHARACTER VARYING (100),
signedData BLOB, PRIMARY KEY(dni))");
//(metge(dni CHAR(20), nom CHARACTER VARYING (100),
signedData BLOB, PRIMARY KEY(dni) )");
//(gestor (dni CHAR(20), nom CHARACTER VARYING(100),
signedData BLOB, PRIMARY KEY(dni))");
//(usuari (idUsuari INTEGER AUTO_INCREMENT, dni CHAR(20),
numSerie CHAR(50) , tipus CHAR(10), signedData BLOB, PRIMARY KEY(idUsuari) )");
try{//afegir persona
                                                        //Crear
una firma
                System.out.println("datosfirma:" + idUsuari + ","+
nom);
                String datosFirma = new String(idUsuari +","+
nom);
                byte[] signatura1 = signData(datosFirma);

        gbd.afegirPersona(idUsuari, nom, signatura1);
}catch(Exception e){
        System.out.println("Gestor: afegir persona - Error" + e);
}
try{//afegir Usuari
                                                        //Crear
una firma
                System.out.println("datosfirma:" + idUsuari + ","+
numSerie + "," + collectiu);
                String datosFirma = new String(idUsuari +","+
numSerie + "," +collectiu);
                byte[] signatura2 = signData(datosFirma);//firma
del gestor

        gbd.afegirUsuari(idUsuari, numSerie.toString(), collectiu,
signatura2);

}catch(Exception e){
                                                        //presentar una
ventana "Error"
        System.out.println("Gestor: afegirUsuari - Error" + e);
        Dialogo error = new Dialogo("Gestor: afegirUsuari - Error" +
e);
```

```
                                                //awtgu.setEstado(-1);//la ventana se debe cerrar de forma
inesperada.
                                                //awtgu.setSemaforo(1);
                                        }
                                        try{
                                                                                                //Crear
una firma
                                                System.out.println("datosfirma:" + idUsuari + ","+
nom);
                                                String datosFirma = new String(idUsuari +","+
nom);
                                                byte[] signatura3 = signData(datosFirma);//firma
del gestor

                                                if(collectiu.equals("Administradors")){//guardar en gestor
                                                        gbd.afegirGestor(idUsuari, nom, signatura3);
                                                }
                                                if(collectiu.equals("Pacients")){
                                                        gbd.afegirPacient(idUsuari, nom, signatura3);
                                                }
                                                if(collectiu.equals("Metges")){
                                                        gbd.afegirMetge(idUsuari, nom, signatura3);
                                                }

                                                //Presentar una ventana "Usuario guardado"
                                                Dialogo usuariOk = new Dialogo("Usuari guardat.");
                                        }catch(Exception e){

                                        }
                                        awtgu.setSemaforo(1);
                                }catch(Exception e){
                                        //presentar una ventana "Error"
                                        Dialogo error = new Dialogo("Error" + e);
                                        awtgu.setEstado(-1);//la ventana se debe cerrar de forma inesperada.
                                        awtgu.setSemaforo(1);
                                        //awtgu = null;
                                }
                        }//fin if estado == 1

                }//fin while
        }//fin altaUsuari

        /** método de prueba */
        public void altaHistorial(){
                //Seleccion de todos los pacients de la base de dades.
                String[] listaPacients = {"p1 ","p2 ","p3 "};
                //Pantalla para seleccionar un pacient.
                AWTGestorHistorial awtgh = new AWTGestorHistorial(listaPacients);
                while(awtgh.getSemaforo() == 0){
                        if(awtgh.getEstado() != 2);
                        if(awtgh.getEstado() == 2){
                                int indice = awtgh.getItemSeleccionado();
                                //Paciente seleccionado
                                //El indice de Historial es autoincrementado: guardar nuevo historial
                                awtgh.setEstado(0);
                                awtgh = null;
                        }
                }

        }
        //fin altaHistorial


        /**
        java utiliza byte con signo. (el octavo bit indica el singo). MySQL està preparado para interpretar Latin 1
        @param DATA  es una matriz de byte
        */
        public static String byteToString(byte[] DATA){

                        StringBuffer sb = new StringBuffer("0");
                        //este cero se añade porque ResultSet.next() da problemas en la primera iteració
                        for(int ii = 0; ii<DATA.length; ii++){
                                //System.out.print("* " + DATA[ii]);
                                String c32 = Integer.toBinaryString(DATA[ii]);
                                String c8 = null;
                                //System.out.print(c32 + "/" + c32.length());
```

```
                                        if(c32.length() == 1) {
                                                sb.append("0000000");
                                                sb.append(c32);
                                        }
                                        if(c32.length() == 2) {
                                                sb.append("000000");
                                                sb.append(c32);

                                        }
                                        if(c32.length() == 3) {
                                                sb.append("00000");
                                                sb.append(c32);
                                        }
                                        if(c32.length() == 4) {
                                                sb.append("0000");
                                                sb.append(c32);
                                        }
                                        if(c32.length() == 5) {
                                                sb.append("000");
                                                sb.append(c32);
                                        }
                                        if(c32.length() == 6) {
                                                sb.append("00");
                                                sb.append(c32);
                                        }
                                        if(c32.length() == 7) {
                                                sb.append("0");
                                                sb.append(c32);
                                        }
                                        if(c32.length() == 8) {
                                                sb.append(c32);
                                        }
                                        if(c32.length() >8) {
                                                c8 = c32.substring(c32.length()-8);
                                                //System.out.println(c8);
                                                sb.append(c8);
                                        }

                                        //System.out.print(":" + sb.toString());
                                        //sb.delete(0,c32.length());
                                        //sb.append(Integer.toHexString(DATA[ii]));
                                }

                                //System.out.println(sb.toString());
                                String binario = sb.toString();
                                //String hexadec = sb.toString();

        return binario;
}

/**
@param bits es una cadena de la forma "000010001110..."
@return cadena
*/
public String bitsToString(String bits){
        //desde cero hasta bits.length()
        //procesar de 8 en 8,
        String cadena = " ";
        System.out.println("Gestor: bitsToString: " + cadena);
        return cadena;
}


/**
@param bits es un String de la forma "110001011100..."
@return byte[]
*/
public byte[] bitsToByteArray(String bits){
        //¿Cuanto mide bits?
        int bits_size = bits.length();
        //¿Es bits_size multiplo de 8
        int resto = bits_size%8;
        if(resto != 0) System.out.println("Gestor: cadena erronea");
        //Declaramos el vector que recogera los "octetos"
        Vector octetos = new Vector();
        StringBuffer octeto = new StringBuffer();//java.lang.
```

```
		int j = 0; // indice para formar grupos de ocho.
		//Recorremos la cadena de "bits"
		for(int i=0; i<bits.length(); i++){
			//se forman grupos de ocho
			while(j<8){
				octeto.append(bits.charAt(i));
				j++;
			}
			if(j==8){
				octetos.addElement(octeto.toString());
				octeto=null;
				octeto = new StringBuffer();
				j=0;
			}
		}
		//Transformar los String de "octetos" en byte
		byte[] bytes = new byte[octetos.size()];
		for(int i =0; i<octetos.size(); i++){
			bytes[i] = Short.valueOf((String)octetos.elementAt(i),2).byteValue();
		}
		return bytes;
	}

	/** Needham Shroeder
	Este método solo se puede utilizar a partir de .procesar()
	*/
	public void NeedhamShroeder(){


		//while(rmiobj.getSemaforo() == 0);// el servidor espera mientras el cliente no ponga el semaforo a
!= 0
		if(this.rmiobj.getSemaforo() !=0 ){

			System.out.println("Servidor: " + rmiobj.getFromClient());//
			//extraer el contenido del mensaje.
			StringTokenizer st = new StringTokenizer(rmiobj.getFromClient(), "#");

			//para contener el idclient y el random:
			int[] idClient = new int[2];
			//el primer token es el identificador que asigna el servidor; o 11 por defecto.
			try{
				Integer itg = new Integer(st.nextToken());
				if(itg.intValue() == 11) {
					idClient[0] = RMIClient.size() + 1;
				}
				//else idClient[0] = itg.intValue();
			}catch(Exception e){}
			//el segundo token es el random (encryptado para el gestor)
			try{
				Integer itg = new Integer(st.nextToken());
				idClient[1] = itg.intValue();
			}catch(Exception e){}
			RMIClient.addElement(idClient);

			while(st.hasMoreTokens()) System.out.println(st.nextToken());

			//generar la respuesta con el valor del identificador
			String resposta = "#"+idClient[0]+ "#"+ idClient[1] ;//

			rmiobj.setFromServer(resposta); //(ORsfs)
			rmiobj.setSemaforo(0);//da el turno al client.

			String[] mensaje = {"Se ha conectado un cliente remoto "};
			Pantalla p = new Pantalla("Gestor RMI ", mensaje);
		}//fin if
	}
}
```

SistemaMetge.java

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

import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.security.cert.CertificateException;
import iaik.x509.X509Certificate;

/**	SistemaMetge:*/
public class SistemaMetge{
	/** sistema operatiu */
	private static String so;

	/** login: idusuari;
	idusuari per defecte: UOC
	*/
	private String idUsuari;
	/** login: password
	password per defecte: uoc0506
	*/
	private String passwd;


	/***/
	private Metge metge;

	/**certificat del gestor*/
	private X509Certificate x509g;

	/**	main:*/
	public static void main(String[] args){
		//Evita Exception
		Security.insertProviderAt(new IAIK(), 2);

		//determinació de la plataforma:
		sistemaOperatiu();

		//generem un entorn no static
		SistemaMetge sm = new SistemaMetge();
		String[] msg1 = {"Part de la aplicació corresponent al Metge",
				"permet consultar llista de pacients	",
				"historial d'un pacient, ...		"};
		Pantalla p1 = new Pantalla("Sistema Metge", msg1);
		//identificació previa
		sm.ferLogin();
		//comprovació del certificat del gestor: farà falta per enviar-le dades encryptades.
		sm.setGestor();
		//comprovació del certificat del metge i lectura de dades:
		sm.setMetge();

		//RMIClient rmiclient = new RMIClient();//en Metge.procesar()


		System.out.println("ADEU");
		System.exit(0);
	}


	/** usuari i contrasenya
	a partir d'una entrada grafica obtenir usuari i password de p12
	cal obrir el fitxer protegit Metge.p12
	*/
	public void ferLogin(){
		AWTInici inici = new AWTInici("SistemaMetge");
		//dar tiempo al usuario para introducir datos (se bloquea el flujo mientras AWTInici no cambie el
semaforo)
		while(inici.getSemaforo() == 0);
		//presentar los valores introducidos en los TextField
		System.out.println("usuari: " + inici.getUsuari());
		System.out.println("password: " + inici.getPassword());
		//formar el String con el nombre y el path del fichero Metge.p12
```

```
		try{
			String p12File;
			if (so.equals("Windows")){
				StringBuffer sb = new StringBuffer(".\\doc\\pki\\");
				sb.append(inici.getUsuari());
				sb.append(".p12");
				p12File = new String(sb.toString());
			} else {
				StringBuffer sb = new StringBuffer("./doc/pki/");
				sb.append(inici.getUsuari());
				sb.append(".p12");
				p12File = new String(sb.toString());
			}
			//hacer la identificación segura con el fichero Metge.p12 y su contraseña. (Metge.p12 está
protegido por contraseña)
			//abrir el fichero Metge.p12
			P12 p12 = null;
			String password = inici.getPassword();//"uoc0506";
			try{
			p12 = new P12(p12File, password);
				//PrivateKey _privateKey = p12.getPrivateKey();
				//PublicKey  _publicKey = p12.getPublicKey();
				//X509Certificate[] _chain = p12.getCertificates();

			//this.metge = new Metge();
			System.out.println("identificació correcta");
			} catch(Exception e){
			e.printStackTrace();
			System.exit(0);
			}

		}
		//catch(FileNotFoundException fnfe){System.out.println(fnfe);}
		//catch(CertificateException ce){System.out.println(ce);}
		//catch(IOException ioe){System.out.println(ioe);}
		catch(Exception e){System.out.println(e);}


	}



	/**	autenticacio del Gestor:
	accedir a disc dur, llegir fitxer Gestor.crt i recuperar les dades
	*/
	private void setGestor(){

		try{
			String nomFile;
			if (so.equals("Windows")){
				nomFile = new String(".\\doc\\pki\\Gestor.crt");
			} else {
				nomFile = new String("./doc/pki/Gestor.crt");
			}
			//abrir el fichero Gestor.crt
			CertificateFile cf = new CertificateFile(nomFile);
			x509g = cf.getCertificate();
			System.out.println("Gestor: "+ x509g.toString());

			//comprobar la validez del certificado: !esta firmado por CA¡
			CertificateValid cv = new CertificateValid(x509g);

		}catch(FileNotFoundException fnfe){System.out.println(fnfe);}
		catch(CertificateException ce){System.out.println(ce);}
		catch(IOException ioe){System.out.println(ioe);}
		catch(Exception e){System.out.println(e);}

	}


	/**	llegir la informació continguda en el certificat Metge.crt */
	public void setMetge(){
		X509Certificate x509m;
		metge = new Metge();
		try{
			//abrir el fichero metge.crt
```

```
                        String nomFitxer;
                        if(so.equals("Windows")){
                                nomFitxer = new String(".\\doc\\pki\\Metge.crt");
                        }else {
                                nomFitxer = new String("./doc/pki/Metge.crt");
                        }

                        CertificateFile cf = new CertificateFile(nomFitxer);
                        x509m = cf.getCertificate();
                        //System.out.println(x509m.toString());

                        //comprobar la validez del certificado
                        CertificateValid cv = new CertificateValid(x509m);

                        //asignar el certificat al pacient
                        metge.setCertificat(x509m);

                        //leer la informacion del metge
                        CertificateContent cc = new CertificateContent(x509m);
                        //System.out.println("num serie: "+cc.getNumSerie());
                        //System.out.println("usuari: "+cc.getUserID());
                        metge.setIDmetge(cc.getUserID());
                        //System.out.println("collectiu: "+cc.getCollectiu());

                        System.out.println("METGE:"+ metge.getIDmetge() + metge.getCollectiu());

                        metge.procesar();

                }catch(FileNotFoundException fnfe){System.out.println(fnfe);}
                catch(CertificateException ce){System.out.println(ce);}
                catch(IOException ioe){System.out.println(ioe);}
                catch(Exception e){System.out.println(e);}

        }


        /** Fa falta coneixer el Separador de directoris per tal de accedir als fitxers:*/
        private static void sistemaOperatiu(){
                //la clase File depende de la plataforma:
                //aprofitem això per identificar el SO
                File file = new File("file");
                if(file.separator.equals("\\")){
                        so = new String("Windows");
                        System.out.println("OS: Windows");
                }
                else{
                        so = new String("Unix");
                        System.out.println("OS: No es Windows");
                }
        }

}
```

Metge.java

```
import java.security.PrivateKey;
import java.security.PublicKey;
import iaik.security.provider.IAIK;
import java.security.*;
import iaik.pkcs.pkcs7.*;
import iaik.asn1.DerCoder;
import iaik.asn1.structures.AlgorithmID;
import iaik.asn1.ASN1Object;
import iaik.utils.Util;
import iaik.x509.X509Certificate;
import java.rmi.*;
import java.util.Random;



/**	Metge:*/
public class Metge extends Usuari{
	/**	id_metge*/
	private String id_metge;

	/**col·lectiu:*/
	private String collectiu = "Metges";

	/** certificat metge: ver Usuari */

	/**	parell de claus:*/
	private PrivateKey privatekey;
	private PublicKey publickey;
	private X509Certificate[] _chain;

	/** Llista de pacients:*/
	private LlistaPacientsProtegida llistaPacientsProtegida;


	/** RMI */
	RMIInterfaz hm;
	int identificador = 11;
	int unrandom;


	/***/
	public Metge(){
		P12 p12 = null;
		String p12File = ".\\doc\\pki\\Metge.p12";
		String password = "uoc0506";
		try{
		   p12 = new P12(p12File, password);
		} catch(Exception e){
		e.printStackTrace();
		System.exit(0);
		}
		privatekey = p12.getPrivateKey();
		publickey = p12.getPublicKey();
		_chain = p12.getCertificates();

	}

	/**
	@param s identitat del metge
	*/
	public void setIDmetge(String s){
		id_metge = new String(s);
	}

	/**	getIDmetge
	@return id_metge
	*/
	public String getIDmetge(){
		return id_metge;
	}

	/**
```

```
	@return collectiu
	*/
	public String getCollectiu(){
		return collectiu;
	}

	/**
	métode que permet rebre la derivacó del fluxe del metode main()

	*/
	public void procesar(){
		//part grafica
		AWTMetge awtmetge = new AWTMetge();

		//comunicació remota:
		String dirección = "rmi://localhost:2002/";
		//RMIInterfaz hm = null;

		try{
			//La referencia al objeto remoto se realiza con el método lookup de la clase Naming
			// al cual se le pasa como argumento la dirección o nombre del equipo donde se encuentra el objeto remoto,
			// así como el nombre de dicho objeto remoto.
			hm = (RMIInterfaz)Naming.lookup(dirección + "objrmi");

			NeedhamShroeder();
			while(awtmetge.getSemaforo() == 0)
			{
				//consultar el objeto remoto

				//consultar la ventana
				awtmetge.procesar();
			}
		} catch(Exception e){
			e.printStackTrace();
		}
	}



	/** NeedhamShroeder*/
	public void NeedhamShroeder(){
		unrandom= getRandom();
		System.out.println("Client envia: #"+ identificador+ "#"+ unrandom+ "# y signedData<>");

		//error se ha de transformar int en string
		//byte[] signedData0 = this.signData(unrandom);//los byte producidos pueden crear conflicto si Latin 1 > 127

		try{
			while(hm.getRsemaforo() != 0);// el cliente espera mientras el semaforo no este a 0
			String tipus = "Metge";
			hm.setFromClient("#"+ identificador+"#"+ unrandom +"#" + tipus + "#y signedData");
			hm.setRsemaforo(identificador);//libera al servidor (y el cliente se pone en espera)
			//hm.metodo("texto que se envia");
			while(hm.getRsemaforo() != 0 );// el cliente espera mientras el servidor no ponga el semaforo a 0
			System.out.println("Client: recibo: " + hm.getFromServer());//aparece por la pantalla del client.
		}catch(Exception e){
			System.out.println("Pacient: error"+ e);
		}
	}



  /**
   * Describe <code>signData</code> method here.
   * El gestor pot signar les dades que vulgui conservar.
   * al definir SignedData.EXPLICIT el signedData no conte el text en clar
   * faria falta definir SignedData.IMPLICIT per tal que el text en clar fos incluit en el SignedData.
   * @param dataIn a <code>String</code> value
   * @return a <code>byte[]</code> value
   */
  public byte[] signData(String dataIn){
```

```
                byte[] p7Enc = null;
                SignedData p7 = new SignedData(dataIn.getBytes(), SignedData.EXPLICIT);
                p7.setCertificates(_chain);
                SignerInfo signer = new SignerInfo(new IssuerAndSerialNumber(_chain[0]), AlgorithmID.sha1,
privatekey);
                try{
                p7.addSignerInfo(signer);
                }catch(Exception e){
                e.printStackTrace();
                System.exit(0);//problema fatal.
                }

                try{
                p7Enc = p7.getEncoded();
                }catch(Exception e){
                e.printStackTrace();
                System.exit(0);
                }

                return p7Enc;
  }



        /** verifySignatura
        @param signedDatabyte es un objeto SignedData en formato DER
        @param crtFile es el nombre completo ".\\doc\pki\\Pacient.crt" del certificado del firmante
        */
        public void verifySignature(byte[] signedDatabyte, String crtFile){
                X509Certificate x509 = null;
                try{
                        CertificateFile cf = new CertificateFile(crtFile);
                        x509 = cf.getCertificate();
                }catch(Exception e){
                        System.out.println("Metge "+ e);
                }
                ASN1Object objsd = null;
                try{
                        objsd = DerCoder.decode(signedDatabyte);
                        //System.out.println("objsd " + objsd.toString());
                }catch(iaik.asn1.CodingException e){
                        System.out.println("CodingException " + e);
                }
                SignedData sd = null;
                try{
                        sd = new SignedData(objsd);
                        //System.out.println("sd "+sd.toString(true));
                        //System.out.println("sd "+sd.toString(false));
                }catch(iaik.pkcs.PKCSParsingException e){
                        System.out.println("PKCSParsingException "+e);
                }


                //String sdobj = new String(sd.getContent());
                //System.out.println("sdobj " + sdobj);

                try{
                        sd.verify(x509);
                }catch(Exception e){
                        System.out.println("error en la verificacion");
                }

        }

        /** obtener un numero aleatorio
        @return i : un numero entero aleatorio.
        */
        public static int getRandom(){
                Random r = new Random();
                int i = r.nextInt();
                return      i;
        }
}
```

SistemaPacient.java

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

import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.security.cert.CertificateException;
import iaik.x509.X509Certificate;


/**	SistemaPacient:
utilitza CertificateValid; CertificateContent;
*/
public class SistemaPacient{
	/** sistema operatiu */
	private static String so;


	/** login: idusuari; per defecte: Pacient  */
	private String idUsuari;
	/** login: password
	password per defecte: uoc0506
	*/
	private String passwd;

	/**	Objecte Pacient */
	private Pacient pacient;


	/**certificat de CA */

	/**certificat del gestor: farà falta la PublicKey del gestor */
	private X509Certificate x509g;

	/**	main:*/
	public static void main(String[] args){
		//problema de SecurityException?
		Security.insertProviderAt(new IAIK(), 2);

		//java.util.Date date = new java.util.Date();
		//System.out.println(date.toString());
		//Fri Apr 25 20:57:58 CEST 2008

		//Cal coneixer el sistema operatiu abans de llegir fitxers. problema '\' '/'
		sistemaOperatiu();

		//generem un entorn no static
		SistemaPacient sp = new SistemaPacient();
		//informació per l'usuari
		String[] msg1 = {"Part de la aplicació corresponent al Pacient",
				"permet consultar les dades personals.	"};
		Pantalla p1 = new Pantalla("Sistema Pacient", msg1);

		//identificació previa amb criptografia.
		sp.ferLogin();
		//comprovació del certificat del Gestor : farà falta per enviar-le dades encryptades.
		sp.setGestor();
		//sp contiene una instancia de Pacient pacient = new Pacient(); amb les dades extretes de
Pacient.p12
		sp.setPacient();

		System.out.println("A D E U");
	}


	/** usuari i contrasenya
	a partir d'una entrada grafica obtenir usuari i password de p12
	cal obrir el fitxer protegit Pacient.p12
	*/
	public void ferLogin(){
		AWTInici inici = new AWTInici("SistemaPacient");
```

```
		//bloquea el flujo para dar tiempo al usuario a entrar datos en la ventana. AWTInici controla el semaforo.
		while(inici.getSemaforo() == 0);
		//informa de los datos introducidos en los TextField
		System.out.println("usuari: " + inici.getUsuari());
		System.out.println("password: " + inici.getPassword());
		//se da formato al nombre del fichero Pacient.p12
		try{
			String p12File;
			if (so.equals("Windows")){
				StringBuffer sb = new StringBuffer(".\\doc\\pki\\");
				sb.append(inici.getUsuari());
				sb.append(".p12");
				p12File = new String(sb.toString());
			} else {
				StringBuffer sb = new StringBuffer("./doc/pki/");
				sb.append(inici.getUsuari());
				sb.append(".p12");
				p12File = new String(sb.toString());
			}
			//abrir el fichero Pacient.p12 (tiene contraseña)
			P12 p12 = null;
			String password = inici.getPassword();//"uoc0506";
			try{
			p12 = new P12(p12File, password);
				//PrivateKey _privateKey = p12.getPrivateKey();
				//PublicKey  _publicKey = p12.getPublicKey();
				//X509Certificate[] _chain = p12.getCertificates();

			//this.pacient = new Pacient();
			System.out.println("identificació correcta");

			} catch(Exception e){
			e.printStackTrace();
			//Si la identificación es incorrecta se desconecta el programa.
			System.exit(0);
			}

		}
		//catch(FileNotFoundException fnfe){System.out.println(fnfe);}
		//catch(CertificateException ce){System.out.println(ce);}
		//catch(IOException ioe){System.out.println(ioe);}
		catch(Exception e){System.out.println(e);}


	}

	/**
	utilitza la classe CertificateFile per
	accedir a disc dur, llegir fitxer Gestor.crt i recuperar les dades
	*/
	public void setGestor(){

		try{
			String nomFile;
			if (so.equals("Windows")){
				nomFile = new String(".\\doc\\pki\\Gestor.crt");
			} else {
				nomFile = new String("./doc/pki/Gestor.crt");
			}
			//abrir el fichero Gestor.crt
			CertificateFile cf = new CertificateFile(nomFile);
			x509g = cf.getCertificate();
			System.out.println("Gestor: "+ x509g.toString());

			//comprobar la validez del certificado: !esta firmado por CA¡
			CertificateValid cv = new CertificateValid(x509g);

		}catch(FileNotFoundException fnfe){System.out.println(fnfe);}
		catch(CertificateException ce){System.out.println(ce);}
		catch(IOException ioe){System.out.println(ioe);}
		catch(Exception e){System.out.println(e);}

	}

	/**setPacient: permet llegir el fitxer Pacient.crt, validar i cargar el valor en pacient */
```

```
	public void setPacient(){
		X509Certificate x509p;
		pacient = new Pacient();
		try{
			String nomFile;
			if (so.equals("Windows")){
				nomFile = new String(".\\doc\\pki\\Pacient.crt");
			} else {
				nomFile = new String("./doc/pki/Pacient.crt");
			}
			//abrir el fichero pacient.crt
			CertificateFile cf = new CertificateFile(nomFile);
			x509p = cf.getCertificate();
			//System.out.println(x509p.toString());

			//comprobar la validez del certificado
			CertificateValid cv = new CertificateValid(x509p);

			//asignar el certificat al pacient
			pacient.setCertificat(x509p);

			//leer la informacion del pacient
			CertificateContent cc = new CertificateContent(x509p);
			//System.out.println("num serie: "+cc.getNumSerie());
			//System.out.println("usuari: "+cc.getUserID());
			pacient.setIDpacient(cc.getUserID());
			//System.out.println("collectiu: "+cc.getCollectiu());

			System.out.println("PACIENT:"+ pacient.getIDpacient() + pacient.getCollectiu());
			//per presentar una finestra AWTPacient:
			pacient.procesar();

		}catch(FileNotFoundException fnfe){System.out.println(fnfe);}
		catch(CertificateException ce){System.out.println(ce);}
		catch(IOException ioe){System.out.println(ioe);}
		catch(Exception e){System.out.println(e);}

	}


	/** Fa falta coneixer el Separador de directoris per tal de accedir als fitxers:*/
	private static void sistemaOperatiu(){
		//la clase File depende de la plataforma:
		//aprofitem això per identificar el SO
		File file = new File("file");
		if(file.separator.equals("\\")){
			so = new String("Windows");
			System.out.println("OS Windows");
		}
		else{
			so = new String("Unix");
			System.out.println("OS: No es Windows");
		}
	}
}
```

Pacient.java

```
import java.security.PrivateKey;
import java.security.PublicKey;
import iaik.security.provider.IAIK;
import java.security.*;
import iaik.pkcs.pkcs7.*;
import iaik.asn1.DerCoder;
import iaik.asn1.structures.AlgorithmID;
import iaik.asn1.ASN1Object;
import iaik.utils.Util;



import iaik.x509.X509Certificate;

import java.rmi.*;

import java.util.Random;



/**		Pacient: */
public class Pacient extends Usuari{
		/**		id_pacient*/
		private String id_pacient;

		/**		col·lectiu:*/
		private String collectiu = "Pacients";

		/**		certificat: */
		private X509Certificate x509;

		/**		parell de claus: */
		private PrivateKey privatekey;
		private PublicKey publickey;
		private X509Certificate[] _chain;

		/** Historial: */
		Historial historial;

		/** RMI */
		RMIInterfaz hm;
		int identificador = 11;
		int unrandom;


		/**
		@param s quan es coneix l'usuari, */
		public void setIDpacient(String s){
			id_pacient = new String(s);
		}

		/**
		@param c quan es coneix el certificat del pacient */
		public void setCertificat(X509Certificate c){
			x509 = c;
		}

		/***/
		public Pacient(){
			P12 p12 = null;
			String p12File = ".\\doc\\pki\\Pacient.p12";
			String password = "uoc0506";
			try{
			   p12 = new P12(p12File, password);
			} catch(Exception e){
			e.printStackTrace();
			System.exit(0);
			}
			privatekey = p12.getPrivateKey();
			publickey = p12.getPublicKey();
			_chain = p12.getCertificates();
```

```
	}

	/***/
	public Pacient(String s){
		//id_pacient = new String(s);
		id_pacient = s;
	}
	/**		getIDpacient
	@return id_pacient
	*/
	public String getIDpacient(){
		return id_pacient;
	}

	/**
	@return collectiu
	*/
	public String getCollectiu(){
		return collectiu;
	}

	/**
	@return x509
	*/
	public X509Certificate getCertificat(){
		return x509;
	}

	/**@param col quan es coneix el collectiu , per defecte es Pacient */
	public void setCollectiu(String col){
		if(! collectiu.equals(col)) System.out.println("ERROR");
		//collectiu = col;
	}

	/**
	@param h quan es coneix l'historial del pacient */
	public void setHistorial(Historial h){
		historial = h;
	}

	/**
	Demanar les dades de l'historial al gestor.
	*/
	public void doHistorial(Pacient p){
		//Procedure amb Gestor
		//historial = unHistorial;
	}

	/**metode per al SistemaPacient */
	public void procesar(){
		//part gràfica.
		AWTPacient awtpacient = new AWTPacient(this);
		//comunicació remota:
		String dirección = "rmi://localhost:2002/";
		//RMIInterfaz hm = null;

		try{
			//La referencia al objeto remoto se realiza con el método lookup de la clase Naming
			// al cual se le pasa como argumento la dirección o nombre del equipo donde se
encuentra el objeto remoto,
			// así como el nombre de dicho objeto remoto.
			hm = (RMIInterfaz)Naming.lookup(dirección + "objrmi");

			NeedhamShroeder();
			while(awtpacient.getSemaforo() == 0)
			{
				//consultar el objeto remoto

				//consultar la ventana
				awtpacient.procesar();
			}

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

```
	}


	/** obtener un numero aleatorio
	@return i : un numero entero aleatorio.
	*/
	public static int getRandom(){
		Random r = new Random();
		int i = r.nextInt();
		return	i;
	}


	/** NeedhamShroeder*/
	public void NeedhamShroeder(){
		unrandom= getRandom();
		System.out.println("Client envia: #"+ identificador+ "#"+ unrandom+ "# y signedData<>");

		//Error se ha de transformar int en String
		//byte[] signedData0 = this.signData(unrandom);//los byte producidos pueden crear conflicto si Latin
1 > 127

		try{
			while(hm.getRsemaforo() != 0);// el cliente espera mientras el semaforo no este a 0
			String tipus = "Pacient";
			hm.setFromClient("#"+ identificador+"#"+ unrandom +"#" + tipus + "#y signedData");
			hm.setRsemaforo(identificador);//libera al servidor (y el cliente se pone en espera)
			//hm.metodo("texto que se envia");
			while(hm.getRsemaforo() != 0 );// el cliente espera mientras el servidor no ponga el
semaforo a 0
			System.out.println("Client: recibo: " + hm.getFromServer());//aparece por la pantalla del
client.
		}catch(Exception e){
			System.out.println("Pacient: error"+ e);
		}
	}


   /**
    * Describe <code>signData</code> method here.
    * El gestor pot signar les dades que vulgui conservar.
    * al definir SignedData.EXPLICIT el signedData no conte el text en clar
    * faria falta definir SignedData.IMPLICIT per tal que el text en clar fos incluit en el SignedData.
    * @param dataIn a <code>String</code> value
    * @return a <code>byte[]</code> value
    */
   public byte[] signData(String dataIn){

		byte[] p7Enc = null;
		SignedData p7 = new SignedData(dataIn.getBytes(), SignedData.EXPLICIT);
		p7.setCertificates(_chain);
		SignerInfo signer = new SignerInfo(new IssuerAndSerialNumber(_chain[0]), AlgorithmID.sha1,
privatekey);
		try{
		p7.addSignerInfo(signer);
		}catch(Exception e){
		e.printStackTrace();
		System.exit(0);//problema fatal.
		}

		try{
		p7Enc = p7.getEncoded();
		}catch(Exception e){
		e.printStackTrace();
		System.exit(0);
		}

		return p7Enc;
   }



	/**
    * Describe <code>verifySignature</code> method here.
    * a partir de les dades originals, i de la signatura, es pot comprovar la integritat
    * @param p7 a <code>byte[]</code> value
```

```
 * @param data a <code>byte[]</code> value
 *@param nomFile es el nombre del X509Certificate correspondiente al firmante.
 */
public void verifySignature(byte[] p7, byte[] data, String nomFile){
        try{
                CertificateFile cf = new CertificateFile(nomFile);
                X509Certificate unx509 = cf.getCertificate();
                PublicKey pk = unx509.getPublicKey();
                X509Certificate[] chain = new X509Certificate[1];
                chain[0] = unx509;
                CipherManager cm = new CipherManager();
                cm.setCertificateChain(chain);
                cm.decrypt(p7);
                byte[] firmaEnClaro = cm.getDecryptedData();
                for(int i =0; i< data.length; i++){
                                System.out.println("Pacient"+ data[i] + "" + firmaEnClaro[i]);
                        }
           }catch(Exception e){
                        System.out.println(e);
           }

}

        /** verifySignatura
        @param crtFile es el nombre completo ".\\doc\pki\\Pacient.crt" del certificado del firmante
        */
        public void verifySignature(byte[] signedDatabyte, String crtFile){
                X509Certificate x509 = null;
                try{
                        CertificateFile cf = new CertificateFile(crtFile);
                        x509 = cf.getCertificate();
                }catch(Exception e){
                        System.out.println("Pacient "+ e);
                }
                ASN1Object objsd = null;
                try{
                        objsd = DerCoder.decode(signedDatabyte);
                        //System.out.println("objsd " + objsd.toString());
                }catch(iaik.asn1.CodingException e){
                        System.out.println("CodingException " + e);
                }
                SignedData sd = null;
                try{
                        sd = new SignedData(objsd);
                        //System.out.println("sd "+sd.toString(true));
                        //System.out.println("sd "+sd.toString(false));
                }catch(iaik.pkcs.PKCSParsingException e){
                        System.out.println("PKCSParsingException "+e);
                }


                //String sdobj = new String(sd.getContent());
                //System.out.println("sdobj " + sdobj);

                try{
                        sd.verify(x509);
                }catch(Exception e){
                        System.out.println("error en la verificacion");
                }

        }


/**comprobació*/

public static void main(String[] args){
                Security.insertProviderAt(new IAIK(), 2);



        Pacient pacient = new Pacient("01234567-q");
        int unint = getRandom();
        String s = "" + unint;
        System.out.println("pacient. main " + s);
                byte[] signedData = pacient.signData(s);
                for(int i = 0; i<10; i++){
```

```
                        System.out.print(signedData[i]);
                }

                pacient.verifySignature(signedData, s.getBytes(), ".\\doc\\pki\\Pacient.crt");

    }
}
```

---

4. Capa de dades.

GestorBD.java

4.1. Definició d'objectes:

AlgorithmSequence.java

```
import iaik.asn1.SEQUENCE;
import iaik.asn1.ASN;
import iaik.asn1.structures.AlgorithmID;
import iaik.asn1.BIT_STRING;
import javax.crypto.SecretKey;

import iaik.asn1.DerCoder;
import iaik.pkcs.pkcs7.EnvelopedData;
import iaik.pkcs.pkcs7.Data;
//import iaik.pkcs.PKCSParsingException;

/**
 EnvelopedData: ContentEncryptionAlgorithm

 AlgorithmID:
 public static AlgorithmID des_EDE3_CBC = new AlgorithmID("1.2.840.113549.3.7", "DES-EDE3-CBC",
"3DES/CBC/PKCS5Padding");
 public SecretKeyFactory getSecretKeyFactoryInstance() throws NoSuchAlgorithmException
 public ASN1Object getParameter()
 public ASN1Object toASN1Object()
 public ASN1Object toASN1Object(boolean flag)


 SecretKeyFactory:
 SecretKey generateSecret(KeySpec keySpec) Generates a SecretKey object from the provided key specification (key
material).

*/
public class AlgorithmSequence extends SEQUENCE{//DataSequence
        private AlgorithmID algorithmId = AlgorithmID.des_EDE3_CBC;
        private BIT_STRING parameters ;//OCTET STRING
        //private SecretKeyFactory;

        public AlgorithmSequence(){
                super.asnType = ASN.SEQUENCE;
                super.constructed = true;
                super.addComponent(algorithmId.toASN1Object(true));
                parameters = new BIT_STRING("");
                super.addComponent(parameters);
        }


        public AlgorithmID getAlgorithmId(){
                return algorithmId;
        }

        public BIT_STRING getParameters(){
```

```
		return parameters;
	}


	public static void main(String[] args){
		try{
			AlgorithmSequence seq = new AlgorithmSequence();
			System.out.println("seq.toString(): "+ seq.toString());
			System.out.println("seq.getAlgorithmId().toString(): " + seq.getAlgorithmId().toString());
			//
			System.out.println("seq.getParameters().toString(): " + seq.getParameters().toString());
			//

			Data data = new Data(DerCoder.encode(seq));

			byte[] b = data.getData();

			for(int i=0; i<b.length; i++){
				int c = b[i];
				System.out.print(" " + c);
			}
			System.out.println("");
			for(int i=0; i<b.length; i++){
				int c = b[i];
				System.out.print(" " + (char)c);
			}
		}
		//catch(PKCSParsingException pkpe){System.out.println(pkpe);}
		catch(Exception e){System.out.println(e);}
	}
}
```

4.2. Gestió de base de dades.

Classe Openssl

```
import java.io.*;
/** Clase que permite generar claves RSA utilizando Openssl */
public class Openssl{

	public void genrsa(String fileOut){
		System.out.println("En genrsa : ");
		Runtime runtime = Runtime.getRuntime();
		Process proceso = null;
		String[] comando = {"C:\\Openssl\\bin\\openssl.exe", "genrsa"};
		System.out.println(comando[0] + comando[1]);
		try{
			proceso = runtime.exec(comando);
			InputStream is = proceso.getInputStream();
			int tamano = 0;
			byte[] b = new byte[2048];//InputStream.read() retorna int obliga a hacer casting (byte)
			for(int i=0; i<2048; i++){
				b[2047] = (byte)is.read(); //is.read() retorna un entero.
				System.out.print(b[2047]);
				if(b[2047] == -1 ) i= 2048;
				else{
					b[i] = b[2047];
					tamano++;
				}
			}
			is.close();
			System.out.println("Mida: "+ tamano + " " + b);// Nos dice cual es la "dirección" de b.
			StringBuffer sb = new StringBuffer();
			for(int i = 0; i< tamano ; i++){
				System.out.println("byte: "+ b[i]+ " char: "+ (char)b[i]);
				sb.append((char)b[i]);
			}
			String s = sb.toString();
			System.out.println("" + tamano + " " + s);
```

```
                        System.out.println("fin genrsa");

                        byte[] b2 = new byte[tamano];
                        for(int i=0; i<tamano; i++){
                                b2[i] = b[i];
                        }
                        File f = new File(fileOut +".key");
                        FileOutputStream fos = new FileOutputStream(f);
                        fos.write(b2);
                        fos.close();


                }catch(Exception e){
                        System.out.println("error: "+ e);
                }
        }

public static void main(String[] args){
        Runtime runtime = Runtime.getRuntime();
        Process proceso = null;
        try{

                Openssl openssl = new Openssl();
                openssl.genrsa("sortida_2");

        }catch(Exception e){
                System.out.println("Error de ejecucion");
                System.out.println(e.toString());
        }
   }
}
```

---

xx.- El joc de proves.
La prova de la funcionalitat de cadascuna de les classes està incluida dins de la mateixa. En general s'ha optat pel mètode d'afegir un métode main(String[] args) dins de cada classe que es volia provar.

---

class Certificat.java

Permet treballar amb Certificats.crt

La prova de les funcionalitats RMI (?)

---

java.policy

```
// Standard extensions get all permissions by default

grant codeBase "file:${{java.ext.dirs}}/*" {
      permission java.security.AllPermission;
};

// default permissions granted to all domains

grant {
      // Allows any thread to stop itself using the
java.lang.Thread.stop()
```

```
      // method that takes no argument.
      // Note that this permission is granted by default only to
remain
      // backwards compatible.
      // It is strongly recommended that you either remove this
permission
      // from this policy file or further restrict it to code sources
      // that you specify, because Thread.stop() is potentially
unsafe.
      // See "http://java.sun.com/notes" for more information.
      permission java.lang.RuntimePermission "stopThread";

      // allows anyone to listen on un-privileged ports
      permission java.net.SocketPermission "localhost:1024-",
"listen";

      // "standard" properies that can be read by anyone

      permission java.util.PropertyPermission "java.version", "read";
      permission java.util.PropertyPermission "java.vendor", "read";
      permission java.util.PropertyPermission "java.vendor.url",
"read";
      permission java.util.PropertyPermission "java.class.version",
"read";
      permission java.util.PropertyPermission "os.name", "read";
      permission java.util.PropertyPermission "os.version", "read";
      permission java.util.PropertyPermission "os.arch", "read";
      permission java.util.PropertyPermission "file.separator",
"read";
      permission java.util.PropertyPermission "path.separator",
"read";
      permission java.util.PropertyPermission "line.separator",
"read";

      permission java.util.PropertyPermission
"java.specification.version", "read";
      permission java.util.PropertyPermission
"java.specification.vendor", "read";
      permission java.util.PropertyPermission
"java.specification.name", "read";

      permission java.util.PropertyPermission
"java.vm.specification.version", "read";
      permission java.util.PropertyPermission
"java.vm.specification.vendor", "read";
      permission java.util.PropertyPermission
"java.vm.specification.name", "read";
      permission java.util.PropertyPermission "java.vm.version",
"read";
      permission java.util.PropertyPermission "java.vm.vendor",
"read";
      permission java.util.PropertyPermission "java.vm.name", "read";

};
```
