Projecte Fi de Carrera:
Esquema criptogràfic per la gestió d'expedients mèdics.

David Màlaga Mezquita
Enginyeria en Informàtica

Jordi Castellà Roca
Consultor

Data Lliurament
11 de Juny de 2008

Agraïments i dedicatòria.

Moltes gracies a totes les persones que m'han ajudat a arribar fins aquí, i que no enumeraré per què segurament em deixaria a algú i no em sembla just. A tots ells.

No obstant, vull agrair a en Jordi Castellà Roca la seva bona disposició a ajudar-nos sempre que ha calgut, i els seus consells sobre l'estructura i metodologia alhora d'afrontar el Projecte.

Una **menció especial** a l'Eva, per tot el que hem deixat de fer, i tots els llocs on no em anat durant aquest darrers anys mentre jo "jugava amb l'ordinador".

Avui en dia, d'entre les nombroses possibilitats que ens ofereixen les noves tecnologies, en podem destacar les que ens permeten accedir a la informació independentment del lloc on ens trobem i de l'hora que sigui.

Aquesta alta disponibilitat horària i geogràfica fa que els camps on aplicar les anomenades noves tecnologies siguin pràcticament il·limitats: educació, defensa, sanitat, transport, assegurances, banca...

En el cas concret d'aquest projecte ens centrem en la gestió d'expedients mèdics, concretament en oferir la possibilitat d'accedir a les dades mèdiques d'una persona de forma remota. Ara bé, donada la naturalesa de la informació que conté un expedient mèdic, és de vital importància que tant l'accés a les dades, com l'emmagatzemament d'aquestes, es porti a terme amb la major seguretat possible.

En la gestió d'expedients mèdics la informació és d'àmbit totalment privat, i només ha de ser accessible per a la pròpia persona interessada i pel seu metge. Així cal garantir, com a mínim, les següents propietats: confidencialitat de dades (ningú que no estigui autoritzat ha de poder veure les dades), autenticitat de les dades mèdiques (només els metges poden introduir informació als expedients), integritat dels expedients, totes les modificacions han de quedar registrades i amb l'autoria clarament identificada.

Complir amb les condicions esmentades dins d'un entorn informàtic no és una tasca fàcil, ja que la seguretat s'ha d'aplicar a diferents nivells, on destaquen el nivell de comunicacions o de xarxa, i el nivell d'aplicació. La seguretat a nivell de xarxa dependrà de les condicions d'accés a la xarxa de comunicacions, però per a garantir la seguretat a nivell de l'aplicació s'han dissenyat una sèrie de protocols criptogràfics, l'esquema criptogràfic, un per cada operació a realitzar. Tots aquest protocols estan basats en la criptografia de clau pública.

La combinació de les aplicacions desenvolupades en aquest projecte amb una tecnologia de connexió a Internet amb dispositius mòbils (ordinadors portàtils, PDAs...), podria servir per a poder disposar de tota la informació de l'expedient mèdic d'una persona en un cas en que calgui oferir servei a un pacient fora de les instal·lacions habituals (a casa del pacient, un accident de tràfic...), o augmentant la mobilitat del metge dins de l'entorn habitual de treball, no limitant

l’accés a la informació del pacient quan es disposa d’un ordinador i reduint l’ús del paper amb tots els avantatge que això comporta.

Índex

Índex imatges.

1 Introducció.

En aquest Projecte s'aborda el disseny i la implementació d'una petita aplicació, que permetrà emmagatzemar dades d'historials mèdics de forma segura, i proporcionar eines per accedir i modificar aquests historials d'una forma segura i remota.

Seguint l'arquitectura *client-servidor*, es dissenyaran dos components que permetran, respectivament, accedir a les dades (client) i gestionar els accessos i les operacions realitzades (servidor).

Tots els components es desenvoluparan amb tecnologia Java[20], el que en assegurarà la seva portabilitat i escalabilitat. S'utilitzarà criptografia de clau pública per a garantir la confidencialitat de les dades, i les comunicacions estaran basades en RMI[2].

1.1 Justificació del PFC i context de desenvolupament.

Cada cop més, es va imposant la tendència de disposar de la informació de forma mòbil, és a dir, disposar de la informació sense la necessitat d'un lloc fix d'accés i sense un horari establert.

La proliferació de les tecnologies de la informació, i aquí Internet hi té un paper determinant, ha incrementat la oferta d'aplicacions que proporcionen accés sense lligams a la informació. No obstant, no tots els camps s'han vist afavorits en aquest aspecte.

Una primera justificació per a la realització d'aquest Projecte sorgeix de la necessitat d'implantar un sistema que permeti l'emmagatzemament segur de les dades d'expedients mèdics i, alhora, permeti l'accés remot a aquestes.

Una segona justificació pel Projecte és la facilitat de la gestió i l'augment de la seguretat. Si els expedients mèdics estan emmagatzemats en un únic lloc i en format electrònic, és més senzill gestionar-lo, facilitant compartir la informació entre els diferents usuaris. Permet augmentar la seva seguretat (redundància, copies de seguretat...), a més dels avantatges respecte a disposar d'aquestes mateixes dades en paper (volum, vulnerabilitat a incendis, inundacions...).

Aquest projecte no hagués tingut sentit tant sols uns anys enrere, on la possibilitat de connexió remota a xarxes de comunicacions, internes d'una organització o globals com Internet, eren molt reduïdes: dispositius d'accés, punts de connexió, velocitats disponibles...

Actualment les xarxes de comunicació estan molt esteses, gairebé tothom disposa de connexió des del seu domicili, les connexions sense fils són cada cop més habituals i les velocitats d'accés disponibles són més que suficients per a desenvolupar aplicacions remotes.

Així, aplicacions com la que s'ha desenvolupat en aquest Projecte, han de facilitar la feina de molts professionals, aprofitant els avantatges que suposa l'accés remot a la informació.

1.2 Objectius del Projecte Final de Carrera.

L'objectiu d'aquest projecte és implementar una sèrie d'aplicacions que permetin a metges i pacients accedir a historials mèdics de forma remota i segura.

Per tant, es poden distingir dos objectius principals:

- Obtenir una aplicació segura. La seguretat de la gestió dels expedients mèdics haurà de garantir, com a mínim, les següents propietats:

    - **Confidencialitat**: les dades emmagatzemades han de ser secretes, només han de ser accessibles pel propi pacient i per l'equip de metges als que estigui assignat.

    - **Autenticació**: s'ha de poder demostrar qui ha incorporat dades a un expedient mèdic.

    - **Integritat**: no s'han de poder manipular les dades d'una visita mèdica un cop s'han incorporat a l'expedient d'un pacient. Tampoc s'han de poder eliminar fins que s'elimini l'expedient mèdic en la seva totalitat.

    - **No repudi**: un metge no ha de poder negar que ha incorporat dades a un expedient mèdic.

    - **Classificació de la informació**: no tota la informació tindrà les mateixes restriccions d'accés. Cal diferenciar els diferents nivells de privadesa.

- Accés remot. Proporcionar un accés segur a la informació, però limitar l'accés a determinats usuaris i des de determinats llocs, podria limitar l'ús de les aplicacions.

    - Utilització d'**XML**[1] com a llenguatge d'intercanvi d'informació.

    - Utilització d'**RMI**[2] com a protocol de comunicació.

Altres objectius a aconseguir, també importants encara que no siguin els principals, són:

- **Accessibilitat** i **facilitat d'ús**. Perquè una aplicació sigui útil, a més de fer les tasques per les que ha estat pensada, ha de ser senzilla d'utilitzar, amb una interfície clara i entenedora.

- **Traçabilitat**. S'ha de deixar constància de totes les operacions efectuades: què s'ha fet i qui ho ha fet. Com a complement de les propietats de seguretat esmentades, s'ha de proporcionar la opció de poder monitoritzar els accessos a les dades sense la necessitat d'haver de recórrer al magatzem de dades.

1.3 Enfocament i mètode seguit.

Un cop definits els objectius del Projecte, estem en disposició d'enumerar les diferents parts que formaran les diferents aplicacions:

- **Esquema criptogràfic**: és la *pedra angular* sobre la que es fonamenta la gestió segura dels expedients mèdics.

- **Base de dades**: tota la informació es desarà dins d'una base de dades.

- **XML**[1]: llenguatge de marques que ens ha de servir per a facilitar l'accés a les dades i la comunicació entre els diferents aplicatius.

- **RMI**[2]: serà el protocol de comunicacions utilitzat. La comunicació entre les diferents aplicacions, entre els metges i

pacients, que seran els clients, i el gestor de dades que serà el servidor, és una part essencial del Projecte.

- **Interfícies d'accés**: tots els usuaris del Projecte, metges i pacients, necessiten una interfície per a dur a terme les seves tasques. També es proporcionarà una interfície per al gestor, per a efectuar les tasques d'administració necessàries.

- **Auditoria**: totes les accions han de quedar registrades per a poder disposar d'una forma ràpida de conèixer l'activitat de l'aplicació.

El disseny i desenvolupament dels components esmentats s'ha fet pensant que qualsevol d'ells s'ha de poder integrar en altres entorns sense gaires dificultats, és a dir, s'han pensat com a unitats autònomes amb capacitat d'oferir els seus serveis a qui ho sol·liciti.

El mètode aplicat ha consistit en un desenvolupament per fases, de forma incremental. S'ha construït el Projecte de forma gradual, construint els mòduls un darrera l'altre, portant a terme proves unitàries per cadascun d'ells i proves d'integració amb la resta de mòduls.

Així, els passos efectuats han estat:

- Definició i implementació dels **protocols criptogràfics**.

- Disseny i implementació dels **documents XML**[1] d'intercanvi de dades, i dels components encarregats de emmagatzemar-los i recuperar-los.

- Dissenys i implementació dels **components de comunicació** RMI[2]: clients i servidors. Proves d'integració amb els documents XML[1]: proves d'enviament i recepció.

- Disseny de la **base de dades**. Integració amb els components encarregats d'emmagatzemar i recuperar els documents XML[1].

- Disseny i implementació de l'**aplicació client**. Integració de l'aplicació client amb els components de comunicacions.

- Disseny i implementació de l'**aplicació gestor**. Integració de l'aplicació gestor amb els components de comunicacions i de seguretat. Generació d'informació d'auditoria.

- Revisió i finalització de la **documentació** aportada.

Cal fer notar, que s'ha prioritzat l'acompliment dels objectius principals del Projecte, seguretat i accés remot.

1.4 Planificació del projecte.

La planificació del projecte mostra la temporalització que s'ha seguit per a l'execució de les fases mencionades en el punt anterior.

Cada fase implica aconseguir els objectius necessaris per a poder iniciar la següent, ja que la integració entre els diferents components així o requereix.

Les principals dates, fites, de la planificació del projecte són les següents:

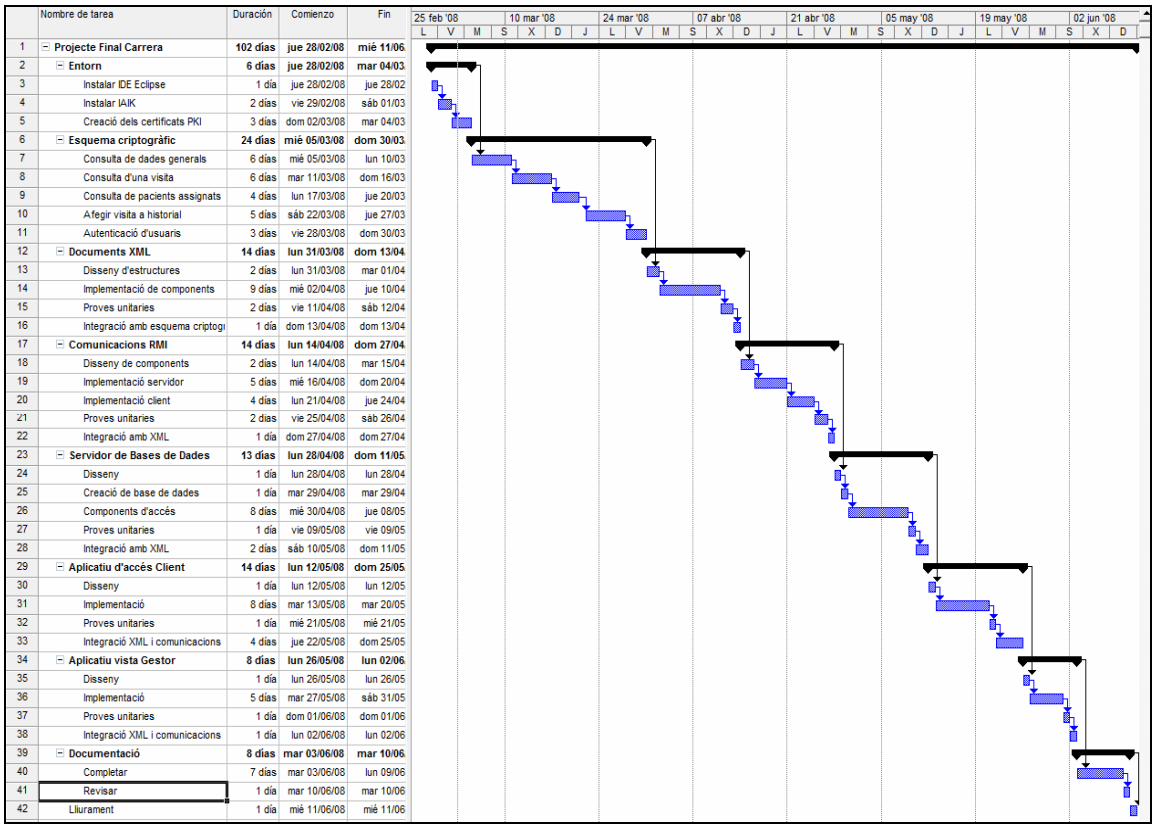
1. **Inici del projecte**.
2. **4 de Març 2008**: Finalització de la instal·lació de l'entorn de treball: IDE Eclipse[3], IAIK[4] i creació de certificats.
3. **30 de Març 2008**: Finalització de l'esquema criptogràfic dels diferents protocols.
4. **13 d'Abril 2008**: Finalització de la definició i implementació dels documents XML[1] d'intercanvi de dades.
5. **27 d'Abril 2008**: Finalització de la definició i implementació dels components RMI[2] de comunicació.
6. **11 de Maig 2008**: Finalització de la instal·lació de la base de dades i del disseny i implementació dels components d'accés.
7. **25 de Maig 2008**: Finalització de l'aplicació d'accés dels clients.
8. **03 de Juny 2008**: Finalització de l'aplicació *vista* del gestor.
9. **10 de Juny 2008**: Finalització de la documentació.
10. **11 de Juny 2008**: Lliurament del projecte.

La planificació ha sofert petites variacions respecte a la proposada inicialment.

La següent figura mostra de forma gràfica la planificació del Projecte realitzada amb MSProject. En aquesta figura es pot veure com la fase que ha consumit més temps ha estat la definició de l'esquema criptogràfic, ja que és la base del desenvolupament.

Cal remarcar que, encara que aparegui al final de la planificació com a una fase independent, la documentació s'ha anat generant durant tot el projecte, deixant per al final la revisió definitiva.

El document MSProject que mostra la imatge es proporciona juntament amb la memòria.



	Nombre de tarea	Duración	Comienzo	Fin
1	**Projecte Final Carrera**	**102 días**	**jue 28/02/08**	**mié 11/06**
2	**Entorn**	**6 días**	**jue 28/02/08**	**mar 04/03**
3	Instalar IDE Eclipse	1 día	jue 28/02/08	jue 28/02
4	Instalar IAIK	2 días	vie 29/02/08	sáb 01/03
5	Creació dels certificats PKI	3 días	dom 02/03/08	mar 04/03
6	**Esquema criptogràfic**	**24 días**	**mié 05/03/08**	**dom 30/03**
7	Consulta de dades generals	6 días	mié 05/03/08	lun 10/03
8	Consulta d'una visita	6 días	mar 11/03/08	dom 16/03
9	Consulta de pacients assignats	4 días	lun 17/03/08	jue 20/03
10	Afegir visita a historial	5 días	sáb 22/03/08	jue 27/03
11	Autenticació d'usuaris	3 días	vie 28/03/08	dom 30/03
12	**Documents XML**	**14 días**	**lun 31/03/08**	**dom 13/04**
13	Disseny d'estructures	2 días	lun 31/03/08	mar 01/04
14	Implementació de components	9 días	mié 02/04/08	jue 10/04
15	Proves unitaries	2 días	vie 11/04/08	sáb 12/04
16	Integració amb esquema criptogr	1 día	dom 13/04/08	dom 13/04
17	**Comunicacions RMI**	**14 días**	**lun 14/04/08**	**dom 27/04**
18	Disseny de components	2 días	lun 14/04/08	mar 15/04
19	Implementació servidor	5 días	mié 16/04/08	dom 20/04
20	Implementació client	4 días	lun 21/04/08	jue 24/04
21	Proves unitaries	2 días	vie 25/04/08	sáb 26/04
22	Integració amb XML	1 día	dom 27/04/08	dom 27/04
23	**Servidor de Bases de Dades**	**13 días**	**lun 28/04/08**	**dom 11/05**
24	Disseny	1 día	lun 28/04/08	lun 28/04
25	Creació de base de dades	1 día	mar 29/04/08	mar 29/04
26	Components d'accés	8 días	mié 30/04/08	jue 08/05
27	Proves unitaries	1 día	vie 09/05/08	vie 09/05
28	Integració amb XML	2 días	sáb 10/05/08	dom 11/05
29	**Aplicatiu d'accés Client**	**14 días**	**lun 12/05/08**	**dom 25/05**
30	Disseny	1 día	lun 12/05/08	lun 12/05
31	Implementació	8 días	mar 13/05/08	mar 20/05
32	Proves unitaries	1 día	mié 21/05/08	mié 21/05
33	Integració XML i comunicacions	4 días	jue 22/05/08	dom 25/05
34	**Aplicatiu vista Gestor**	**8 días**	**lun 26/05/08**	**lun 02/06**
35	Disseny	1 día	lun 26/05/08	lun 26/05
36	Implementació	5 días	mar 27/05/08	sáb 31/05
37	Proves unitaries	1 día	dom 01/06/08	dom 01/06
38	Integració XML i comunicacions	1 día	lun 02/06/08	lun 02/06
39	**Documentació**	**8 días**	**mar 03/06/08**	**mar 10/06**
40	Completar	7 días	mar 03/06/08	lun 09/06
41	Revisar	1 día	mar 10/06/08	mar 10/06
42	Lliurament	1 día	mié 11/06/08	mié 11/06

Imatge 1 - Planificació del projecte.

1.5 Productes Obtinguts.

Un cop implementat el Projecte, s'ha obtingut un sistema que està format pels següents components:

- **Base de dades**: Inclou el servidor de base de dades que ens servirà per emmagatzemar les dades dels expedients.

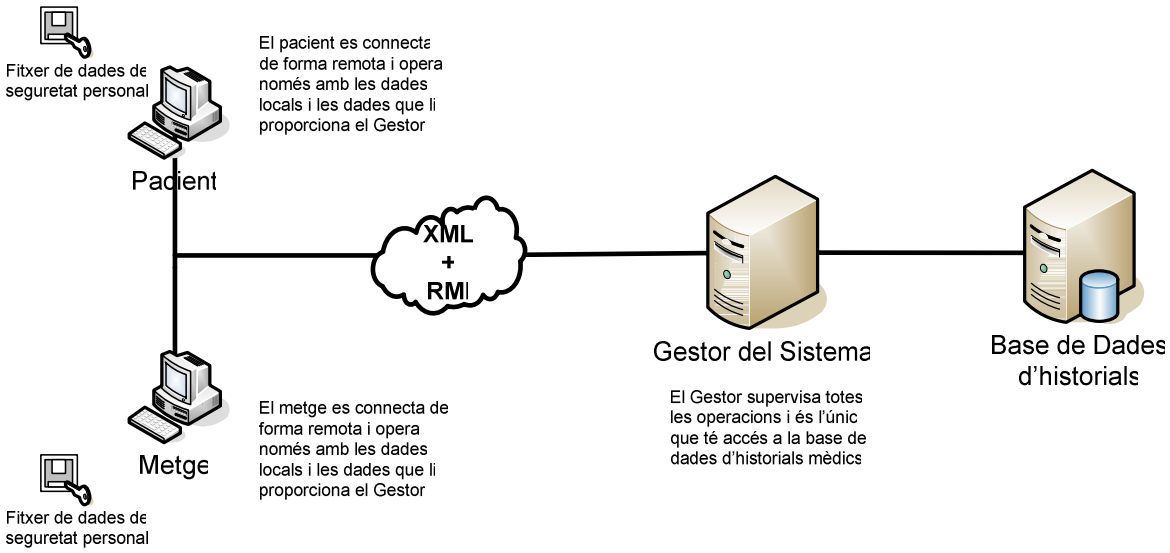
- Aplicació **gestor**: Inclou el servidor RMI[2], que proporciona la capacitat als clients d'executar codi de forma remota per tal de recuperar de forma segura les dades dels expedients, també inclourà el generador d'informació d'auditoria.

- Aplicació **client**: Els clients, en aquest cas els metges i els pacients, utilitzen aquest aplicació per a consultar i modificar les dades dels expedients mèdics.

Els components esmentats s'expliquen més detalladament als capítols corresponents.

1.6 Esquema del projecte.

La següent imatge mostra l'esquema general del Projecte i les funcions que desenvolupen cadascú.



Imatge 2 - Esquema dels components del Projecte.

El clients de l'aplicació, el pacient i els metges, es connecten des dels seus llocs de treball amb el Gestor del sistema. Qualsevol informació que es vulgui obtenir o modificar haurà de passar pel Gestor, on es verificarà que la operació que es vol realitzar sigui vàlida.

L'aplicació s'executa en un entorn *client-servidor*, on la base de dades només és accessible pel Gestor.

Les operacions de xifrat i desxifrat es realitzen en local, y les dades obtingudes s'envien en estructures XML a través de RMI[2].

1.7 Descripció dels següents capítols.

En els següents capítols de la memòria es comenten amb profunditat els conceptes que engloba el Projecte, les decisions de disseny i els criteris i patrons que s'han seguit a cada fase.

L'estructura dels capítols segueix el descrit en la planificació del projecte, seguint la metodologia del desenvolupament incremental del Projecte:

- **PKI**: Com s'ha esmentat anteriorment en aquest mateix document, l'esquema criptogràfic es basa en la criptografia de clau pública. Això vol dir que cada usuari del Projecte haurà de disposar d'un parell de claus.

  En el capítol dedicat s'expliquen els conceptes generals de criptografia de clau pública i els elements necessaris per aplicar una infraestructura de PKI.

- **Esquema Criptogràfic**: En aquest capítol s'expliquen les diferents operacions que es poden portar a terme amb l'aplicació obtinguda i el protocol criptogràfic dissenyat per cadascuna d'elles.

- **Representació de les dades**: En l'aplicació obtinguda, les dades s'intercanvien en format XML[1], ja que proporciona un mecanisme bo per a la seva gestió interna i emmagatzemament. En el capítol corresponent es comentarà amb detall les estructures utilitzades i la llibreria Java[20] utilitzada, JDOM[11].

- **RMI**[2]: L'aplicació gestora, encarregada d'accedir a les dades i controlar els accessos, i l'aplicació d'accés client, s'executaran en diferents màquines, per tant, es necessita un mecanisme encarregat de gestionar les comunicacions. En aquest capítol s'ha realitzat la comunicació entre les dues parts.

- **Servidor Base de Dades**: La base de dades és un punt vital de l'aplicació, tant important com l'esquema criptogràfic. Serveix per a emmagatzemar els expedients mèdics i com a plataforma bàsica per a la integritat de les dades. El SGBD utilitzat és MySQL[5], s'ha escollit perquè és de lliure distribució i proporciona totes les funcionalitats necessàries.

- **Vista Client**: Es proporciona una aplicació que ha de servir per a que els clients, metges i pacients, puguin accedir a la informació emmagatzemada dins la base de dades i es puguin comunicar de manera senzilla i entenedora amb l' aplicació gestor. En aquest capítol es comenta la tecnologia escollida i com s'integra amb el sistema RMI[2].

- **Vista Gestor**: Es proporciona una aplicació que ha de servir per accedir de forma fàcil i entenedora a les dades de l'explicació des del punt de vista d'un administrador. En aquest capítol es comenta la tecnologia escollida i com s'integra amb el sistema RMI[2].

- **Auditor**: Com a totes les aplicacions on la informació emmagatzemada és de caràcter sensible, és necessari disposar d'un mecanisme que permeti fer un seguiment de l'activitat suportada per l'aplicació. En aquest capítol es comenta la tecnologia escollida, Log4J, i la seva configuració. el servidor de base de dades que emmagatzemarà les dades dels expedients

2 PKI.

2.1 Introducció.

L'esquema criptogràfic presentat en aquest projecte es basa en la criptografia de clau pública, això vol dir que totes les parts implicades (metges, pacients i gestor d'expedients mèdics) hauran de disposar d'una parella de claus, clau pública per xifrar i clau privada per desxifrar, amb el seu certificat corresponent.

Cal fer notar que bona part de la seguretat d'aquest tipus de sistema criptogràfic, rau en el fet en que és relativament fàcil obtenir la clau pública a partir de la clau privada, però que obtenir la clau privada a partir de la clau pública és pràcticament impossible (computacionalment és molt difícil). Així, la clau pública d'un usuari serà accessible, però la clau privada es mantindrà *en secret*.

Per tal de garantir que la utilització de la criptografia asimètrica sigui fiable, s'ha de garantir que una parella de claus són realment de qui diuen ser. És aquí on entra el concepte d'infraestructura de clau pública, PKI (Public Key Infraestructure).

Una PKI està formada per:

- Una autoritat de certificació, CA (Certification Authority): entitat de confiança, responsable d'emetre i revocar certificats digitals.

- Una autoritat de registre, RA (Registry Authority): entitat que canalitza les peticions dels usuaris a les CA.

- Subscriptors: són els usuaris finals, qui han sol·licitat els certificats.

- Repositoris: magatzems d'informació relacionada amb la infraestructura PKI (llista de certificats, llistes de revocació...)

Un cop definits els actors principals d'una PKI, es comenta el procés a seguir per obtenir un certificat:

1. Crear una parella de claus (l'usuari sol·licitant o un intermediari).

2. Realitzar una petició de certificat a través d'una RA.

3. La RA valida la identitat de l'usuari sol·licitant.

4. RA envia la petició a la CA.

5. CA emet els certificats.

De tot el comentat anteriorment, es pot deduir que la peça clau de tota la infraestructura és la CA, més concretament la clau privada de la CA, aquesta clau és la peça inicial de la cadena de certificació, i és per això que acostuma a estar desada en un entorn d'alta seguretat.

En cas de que un usuari sospiti que la seguretat de la seva clau privada ha estat compromesa, aquest haurà de comunicar-ho a la CA. En rebre una comunicació d'aquest tipus, la CA revocarà el certificat i l'inclourà dins una llista de certificats revocats, CRL (*Certificate Revocation List*).

Un cop es disposa dels certificats, aquests s'han de verificar, no es pot acceptar mai un certificat com a vàlid si no es realitzen els següents passos:

1. Saber quina CA ha emès el certificat, la CA ha de ser de confiança.

2. Verificar que el certificat no està revocat. Per fer això, es pot *preguntar* a la CA utilitzant un protocol dissenyat especialment per aquesta consulta, OCSP[6], o es pot accedir a la CRL de la CA i cercar el certificat dins d'aquesta llista.

Durant la realització d'aquest Projecte, s'ha utilitzat la llibreria OpenSSL[18] per a obtenir una PKI que s'ha utilitzat per al desenvolupament i les proves. Concretament, per a la generació de la PKI, s'ha utilitzat la versió per a Windows 0.9.8g d'Octubre de 2007.

Els certificats emesos en aquest Projecte segueixen l'estàndard X.509[8], la clau privada i el certificat corresponent s'han emmagatzemat seguint l'estàndard PKCS#12[7] (fitxers amb extensió .P12)

Per aprofundir més en el tema de la criptografia de clau pública, es poden consultar diferents apunts de la UOC sobre comerç electrònic[9] i criptografia[10].

2.2 Passos per a generar els arxius.

Com s'ha comentat, l'esquema proposat necessita que cadascun dels usuaris del Projecte disposi d'una parella de claus en format PKCS#12[7].

Aquest arxiu contindrà:

- Parella de claus pública i privada de l'usuari.
- Certificat de l'usuari emès per la CA.
- Certificat de la CA.

El primer que cal fer és obtenir el certificat de la CA. Els passos a seguir per a crear el certificat de la CA són els següents:

- Generar la parella de claus de la CA, aquestes claus tindran una longitud de 2048 bits. Per generar aquesta parella de claus s'utilitzarà el fitxer *'generarClaus'* que conté la seqüència de comandes necessària. El fitxer resultant amb la parella de claus s'anomena *CA.key*.
- Generar un certificat autosignat amb la parella de claus de la CA, aquest serà el certificat la CA. S'utilitzarà el fitxer *'generaCertificatAutosignat'* que conté la seqüència de comandes necessària. El fitxer resultat amb el certificat de la CA s'anomena *CA.crt*.

Un cop es disposa d'aquests dos fitxers, ja es podran crear els arxius per a la resta d'usuaris del sistema: metge, pacient i gestor.

Els passos a seguits han estat els següents (per als tres tipus d'usuaris):

- Generar una parella de claus. S'utilitza el fitxer *'generarClaus'*, amb una longitud de les claus de 1024 bits.
- Emetre una petició de certificat a la CA. S'ha utilitzat el fitxer *'generaPeticioCertificat'*, que conté la seqüència de comandes necessària per a tal fi.
- Emetre el certificat per part de la CA. Per a fer això s'ha utilitzat el fitxer *'generaCertificat'*, que conté la seqüència de comandes

necessària per a tal fi. En aquest cas, també s'ha utilitzat el fitxer de configuració *'openssl.cnf'* inclòs en els annexos del projecte.

- Generar el fitxer en format PKCS#12[7] que servirà com a contenidor de la parella de claus, el certificat i el certificat de la CA. Per això s'ha utilitzat el fitxer *'generaPKCS#12'*, que conté la seqüència de comandes necessària per a tal fi. Aquest tipus de fitxer està protegit per una paraula clau.

Al final del procés s'hauria d'obtenir un fitxer del tipus .P12 amb les claus i els certificats de l'usuari (metge.p12, pacient.p12 i gestor.p12), que són els que s'utilitzaran durant l'execució de l'aplicació.

Per cada nou usuari del sistema, s'hauran de repetir els passos descrits anteriorment, així, seria interessant disposar d'aquest procediment de forma automàtica, o bé en un mateix fitxer de seqüència de comandes o bé com a una nova funcionalitat del sistema.

2.2.1 Fitxers de generació.

Els fitxers proporcionats per a la generació de la PKI estaven pensats per a ser utilitzats en un sistema operatiu Unix/Linux, i es s'han adaptat per a la plataforma Windows. Els fitxers resultants són els següents:

generarClaus.bat

```
@echo off

if "%1" == "" goto :noparms
if "%2" == "" goto :noparms
if NOT "%3" == "" goto :random

echo creating key pair
openssl genrsa -des3  -out %1 %2
goto :fin

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

:noparms
      echo Usage: [key_file] [key_length]
      echo or
      echo Usage: [key_file] [key_length] [random_file_length]

:fin
```

generaCertificatAutosignat.bat

```
@echo off

if "%1" == "" goto :usage
if "%2" == "" goto :usage
if "%3" == "" goto :usage

openssl req -new -sha1 -x509 -key %1 -out %2 -days %3
goto :fin

:usage
      echo Usage:  [key_file] [file.crt] [dies]

:fin
```

generaPeticioCertificat.bat

```
@echo off

if "%1" == "" goto :usage
if "%2" == "" goto :usage
if "%3" == "" goto :usage

openssl req -new -sha1 -config %3 -key %1 -out %2
goto :fin

:usage
      echo Usage: [key_file] [file.csr] [config_file]

:fin
```

generaCertificat.bat

```
@echo off

if "%1" == "" goto :usage
if "%2" == "" goto :usage
if "%3" == "" goto :usage
if "%4" == "" goto :usage
if "%5" == "" goto :usage

openssl x509 -req -in %1 -days 180 -CA %4 -CAkey %5 -CAcreateserial -extfile
%3 -extensions usr_cert -out %2

goto :fin

:usage
echo   Usage: [file.csr] [file.crt] [config_file] [CA_certificate] [CA_key]

:fin
```

generaPKCS#12.bat

```
@echo off

if "%1" == "" goto :usage
if "%2" == "" goto :usage
if "%3" == "" goto :usage
if "%4" == "" goto :usage

openssl pkcs12 -in %2 -inkey %1 -chain -CAfile %3 -export -out %4
goto :fin

:usage
  echo Usage: [key_file] [file.crt] [CA_certificate] [file.p12]

:fin
```

3 Esquema criptogràfic.

3.1 Introducció.

En aquest capítol es descriuen els diferents protocols que s'han dissenyat per a proporcionar la gestió segura dels expedients mèdics.

Aquests protocols venen definits pel cicle de vida d'un expedient mèdic, les operacions que es poden portar a terme sobre aquest (creació, consulta, modificació...), i qui les pot portar a terme (pacient, metge o gestor del sistema)

El conjunt d'aquests protocols rep el nom d'esquema criptogràfic

3.2 Descripció d'un historial mèdic.

En aquest cas concret ens centrarem en un historial mèdic en format electrònic.

Un historial mèdic, a més de les dades identificatives de la persona, contindrà informació molt diversa: trets físics, al·lèrgies, el seu estat de salut actual i passat, tractaments seguits, resultats de proves mèdiques, historial de visites...

És evident que alguna de la informació esmentada pot ser considerada més confidencial que una altra, però el que queda clar és que l'historial mèdic d'un pacient conté informació de gran valor, i és per això que s'ha de protegir la seva integritat, autenticitat i confidencialitat.

Les dades mèdiques d'una persona són confidencials i només han de ser accessibles per la pròpia persona i pels seus metges.

3.2.1 Estructura del historial mèdic electrònic.

A continuació es presenta un resum de l'estructura de l'historial mèdic que s'utilitzarà en aquest Projecte.

La descripció completa i detallada d'aquesta estructura es farà en el capítol dedicat a l'estructura de dades, capítol 4, i al capítol dedicat a la base de dades, capítol 6 d'aquest mateix document, però es considera

interessant conèixer l'organització general de les dades de l'historial per entendre els protocols proposats.

L'estructura bàsica de l'historial contindrà: les dades generals del pacient, una llista de visites i una llista de metges.

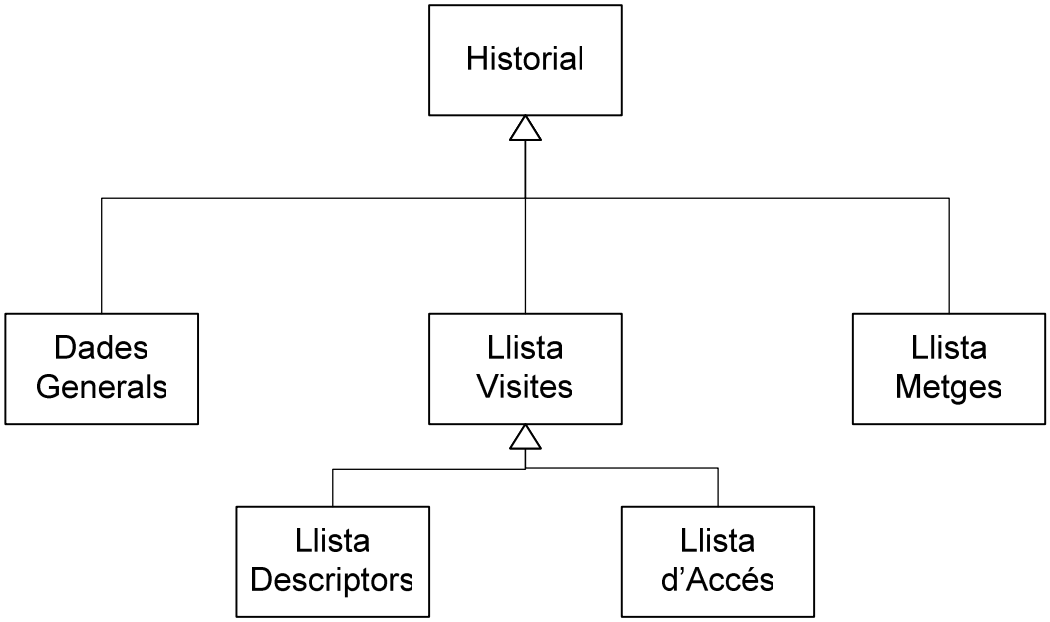
- La llista de visites estarà formada per:

    - Una llista de descriptors de visita xifrada i signada pel gestor del sistema, per tal d'assegurar la seva autenticitat, accessible només pel pacient i els metges assignats a aquest.

    - Una llista d'accés on els usuaris autoritzats, pacient i metges assignats, podran recuperar la clau de xifrat de la llista de descriptors de visita.

- La llista de metges de l'historial contindrà el conjunt de metges que poden accedir a l'historial i fins quan hi poden accedir, aquesta llista està xifrada amb a la clau pública del Gestor, només ha de ser accessible per l'ús intern del gestor.

La següent figura mostra un esquema de l'estructura de l'historial:



Imatge 3 - Estructura de l'historial mèdic.

3.3 Cicle de vida.

Com ja s'ha dit en la introducció d'aquest capítol, el cicle de vida d'un historial marcarà el disseny de l'esquema criptogràfic dissenyat, més concretament, serà el cicle de vida d'un historial electrònic.

Les operacions del cicle de vida que s'han tingut en compte per al disseny de l'esquema criptogràfic són les següents:

- **Registre d'usuaris**: es proporciona la possibilitat de donar d'alta a pacients i metges dins del sistema, aquesta operació és correspon amb la creació d'un historial.

- **Consulta de dades generals**: es permet que qualsevol metge accedeixi a les dades generals d'un pacient. El pacient també podrà accedir a les seves dades generals.

- **Consulta d'una visita**: es permet que els metges assignats a un pacient puguin accedir a les dades de les visites de l'historial, i que un pacient pugui accedir a les seves visites.

- **Consulta de pacients assignats**: un metge podrà accedir a la llista de pacients que té assignats.

- **Afegir visita**: un metge podrà afegir visites a l'historial d'un pacient.

I per poder fer totes les operacions descrites caldrà accedir al sistema i identificar-se de forma correcta, per tant, s'han d'afegir dues noves operacions:

- **Identificació dels usuaris**: es proporciona un sistema per tal de validar la identitat dels usuaris.

- **Autenticació dels usuaris**: s'ha de proporcionar un sistema d'autenticació segura, per saber qui és l'usuari i de quin tipus d'usuari es tracta, durant les operacions.

3.4 Actors del sistema.

De tot el comentat fins aquest moment es poden distingir tres actors del sistema:

- ***Pacient***: actor passiu del sistema, encara que la informació de la base de dades fa referència a aquest actor, aquest no la pot modificar, només consultar.

- ***Metge***: encarregat d'introduir i mantenir la informació dels historials mèdics.

- ***Gestor***: encarregat de gestionar tots els processos del sistema, de verificar que totes les operacions es porten a terme amb les garanties necessàries. És l'únic actor del sistema que té accés a la base de dades.

3.5 Notació utilitzada.

Durant la definició dels protocols s'utilitzarà la següent notació:

- K: clau d'un criptosistema simètric.

- $E_K(M)$: xifratge simètric d'un missatge M amb 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 del funcions *hash.*

3.6 Protocols proposats.

A continuació es detallen els diferents protocols que es proposen per a la gestió segura dels historials mèdics.

Abans, cal fer notar les decisions de disseny que s'han pres per a la implementació dels protocols:

- Tant en el servidor com en el client, totes les dades que es xifren són de tipus text.
- Tots els missatges que es passen entre el client i el servidor són en format text.
- Les dades que es xifren en un mateix missatge es concatenen, per executar només una operació de xifrat.

Els identificadors de les entitats del sistema es tractaran amb més detall dins del capítol 6, dedicat a l'estructura de dades, però per deixar-ho clar de cara a las definició de protocols:

- **IdUsuari**: Com a identificador de l'usuari s'utilitzarà el **número de Seguretat Social** d'aquest.
- **IdGestor**: Com a identificador del gestor s'utilitzarà el *hash* del certificat del Gestor.
- **IdVisita**: Com identificador de visita s'utilitzarà un número seqüencial, global a totes les visites.
- **DescVisita**: Com a descriptor de la visita s'utilitzarà el següent conjunt de dades: identificador de la visita, data i hora de la visita, identificador del metge, tema.

Com s'ha explicat en el punt 2, aquesta és una aplicació basada en criptografia asimètrica, però hi ha certa informació que estarà xifrada utilitzant criptografia simètrica. En aquest cas s'utilitzarà una clau de sessió, ***K***. Aquesta clau de sessió serà el *hash* en Base64[14][15] d'un número aleatori que es generarà cada cop que sigui necessari. L'algoritme utilitzat per la generació del *hash* serà l'MD5[19].

3.6.1 Identificació i autenticació.

Identificació.

Un punt essencial per a garantir la seguretat dels historials mèdics és la identificació i autenticació dels usuaris, és a dir, garantir la identitat de qui accedeix a la informació i quines operacions pot efectuar.

La identificació dels usuaris serà el primer pas que es farà en accedir a la aplicació, es portarà a terme només un cop.

La identificació del usuari es farà a partir de dos paràmetres:

- L'identificador del usuari dins l'aplicació, **IdUsuari**.

- La paraula clau per accedir al fitxer .P12. Cal recordar que aquests tipus de fitxers estan protegits per una paraula clau, tal com s'explica en l'apartat 2.2 d'aquest document.

Aquests dos paràmetres ens permetran accedir a les dades del fitxer .P12, i identificar a l'usuari.

Un cop s'identifiqui l'usuari, l'identificador de l'usuari l'utilitzarà el gestor per accedir al certificat d'aquest, aquest certificat ens permetrà identificar també a quin col·lectiu pertany l'usuari, a través del camp *Organizational Unit Name*.

Autenticació.

El protocol d'autenticació utilitzat en l'aplicació consisteix en una modificació del protocol Needham-Schroeder[12].

S'utilitzarà la següent notació:

- $\mathbf{N_i}$: número aleatori generat pel client.
- $\mathbf{N_g}$: número aleatori generat pel servidor.
- **IdClient**: Identificador del client que vol autenticar-se.
- **IdGestor**: Identificador del gestor del sistema.

A continuació es detallen els passos de la modificació del protocol, per conèixer la definició del protocol original visitar [13].

- Client:
  - Generació d'un número aleatori $\boldsymbol{N_i}$.

  - Xifrar amb la clau pública del gestor, $\boldsymbol{P_G}$, $\boldsymbol{N_i}$ juntament amb l'identificador del client a autenticar, obtenim $\boldsymbol{P_G[N_i, IdClient]}$.

  - $\boldsymbol{P_G[N_i, IdClient]}$ s'envia al gestor.

- Servidor:
  - Desxifrar el missatge rebut $P_G[Ni, IdClient]$ amb la clau privada del gestor S_g.
  - Generació d'un número aleatori N_g.
  - Desar a la base de dades N_i, N_g i *IdClient*.
  - Obtenir la clau pública del client, P_{client}.
  - Xifrar amb la clau pública del client N_i, N_g, i *IdGestor*, obtenim $P_{client}[N_i, N_g, IdGestor]$.
  - $P_{client}[N_i, N_g, IdGestor]$ s'envia al client.

La modificació del protocol estalvia una transferència de dades entre el client i el gestor en la inicialització del protocol, la que es portaria a terme per comprovar que el client ha rebut el número generat pel servidor.

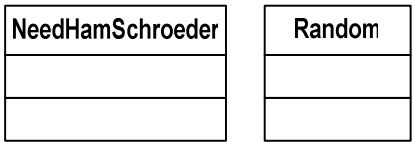
En aquesta variació, el servidor crea una entrada a la base de dades per marcar que el client ha iniciat un procés d'autenticació, després es podrà consultar aquesta entrada per a verificar l'autenticació.

Amb la resposta del servidor xifrada amb la clau pública del client, s'assegura que només el client coneix el número generat pel gestor, N_g, que s'enviarà en la petició del client per identificar-se durant l'operació.

Per evitar els atacs de rèplica, aquest procés d'autenticació es portarà a terme cada cop que el client realitzi una petició.

Per implementar el procés descrit en aquest apartat s'han implementat dues classes:

- **NeedHamSchroeder**: classe amb dos mètodes, un per cada pas dels descrits en el protocol d'autenticació.
- **Random**: classe per generar números aleatoris.



Imatge 4 - Classes NeedHamSchroeder i Random.

3.6.2 Consulta de dades generals.

La consulta de dades generals pot ser executada tant per un pacient com per un metge, per tant, s'haurà de verificar que qui sol·licita les dades generals és, o bé el mateix pacient, o bé un metge amb accés a les dades del pacient.

La notació que s'utilitzarà per descriure aquest protocol serà la següent:

- **IdUsuari**: Identificador del usuari del que es sol·liciten les dades.
- **IdUsuariU**: Identificador del usuari sol·licitant.
- **IdGestor**: Identificador del gestor.
- **La**: Llista d'accés a les visites.
- **Lv**: Llista de descriptors de visita.
- $\mathbf{P_{Gestor}}$: Clau pública del gestor.
- $\mathbf{P_{Client}}$: Clau pública del client.
- $\mathbf{S_{Gestor}}$: Clau privada del gestor.
- $\mathbf{S_{Client}}$: Clau privada del client.
- **K**: Clau de sessió.

Els passos a seguir són els següents:

- Com en totes les operacions, primer caldrà autenticar-se, utilitzant el protocol descrit en el punt 3.6.1.

- A partir de les dades rebudes del gestor en el segon pas del protocol d'autenticació, $\boldsymbol{P_{client}[N_i, N_g, IdGestor]}$, el client realitzarà les següent operacions:

  - Desxifrar les dades rebudes amb $\boldsymbol{S_U}$, i obtenir $\boldsymbol{N_i'}$, $\boldsymbol{N_g}$ i ***IdGestor***.

  - Comparar $\boldsymbol{N_i'}$ amb $\boldsymbol{N_i}$, si són iguals vol dir que el servidor està responent a la seva petició, ja que $\boldsymbol{N_i}$ l'ha generat el client. En cas contrari es genera un error.

  - Obtenir $\boldsymbol{P_{Gestor}}$.

  - Amb $\boldsymbol{P_{Gestor}}$ xifrar ***IdUsuari***, $\boldsymbol{N_g}$ (que ens servirà per identificar-nos amb el gestor), ***Consulta_Dades_Generals*** (identificador d'operació sol·licitada). Així s'obté: $\boldsymbol{P_{Gestor}[N_g,}$ ***Consulta_Dades_Generals, IdUsuari]***.

  - Enviar el resultat al gestor.

- Amb les dades rebudes del client, el gestor efectuarà les següents operacions:
    - Amb $\boldsymbol{S_g}$ desxifrar les dades rebudes del client i obtenir: $\boldsymbol{N_g}$, ***Consulta_Dades_Generals***, ***IdUsuari***.
    - Amb $\boldsymbol{N_g}$, accedir a la base de dades i obtenir ***IdUsuariU***.
    - Si no es recupera informació de la base de dades a partir de $\boldsymbol{N_g}$, es genera un error (procés d'autenticació incorrecte).
    - Comprovar que ***IdUsuariU*** sigui igual a ***IdUsuari***, o que ***IdUsuariU*** es tracti d'un metge. Si no és cap dels dos casos generar un error.
    - Obtenir l'historial, ***H***, de ***IdUsuariU***.
    - Obtenir la clau pública de ***IdUsuariU***, $\boldsymbol{P_{client}}$, i xifrar l'historial per obtenir $\boldsymbol{P_{client}[H]}$.
    - Eliminar $\boldsymbol{N_g}$, $\boldsymbol{N_i}$, ***IdUsuariU*** de la base de dades.
    - Enviar les dades obtingudes al client.
- Amb les dades rebudes del gestor, $\boldsymbol{P_{client}[H]}$, el client executa les següent operacions:
    - Obtenir $\boldsymbol{S_{client}}$.
    - Desxifrar les dades de l'historial. Desxifrar l'historial implica les següents accions:
        - Utilitzant $\boldsymbol{S_{client}}$, desxifrar l'historial.
        - Obtenir la clau de sessió, ***K***, a partir de l'entrada corresponent de la llista d'accés, ***La***.
        - Desxifrar la llista de descriptors de visita, ***Lv***, $\boldsymbol{D_K(Lv)}$
        - Verificar l'autenticitat de ***Lv***, verificant la signatura del Gestor.

Les classes utilitzades per a implementar les operacions descrites en aquest apartat són les següents:

- **Client**: aquesta classe conté tots els mètodes necessaris per executar les operacions descrites en aquest apartat des del punt de vista del client.

- **Gestor**: aquesta classe conté tots els mètodes necessaris per executar les operacions descrites en aquest apartat des del punt de vista del gestor del sistema.

- **Base64** [14][15]: una de les decisions de disseny és que totes les dades que es xifren i desxifren estan en format text, també ho estan les dades que s'intercanvien entre el client i el servidor. Aquesta classe serà l'encarregada de transformar totes els dades en format text Base64[14][15] per a poder complir les especificacions.

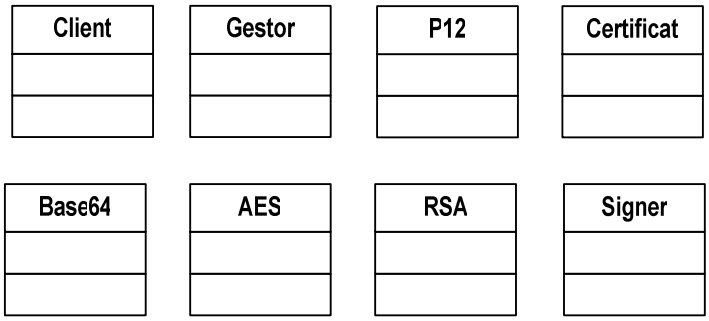
- **P12**: com s'ha comentat, les dades de la clau privada i del certificat de cada entitat, estaran emmagatzemades en local i en format PKCS#12[7]. Aquesta classe serà l'encarregada de accedir a aquest tipus de fitxers.

- **Certificat**: les claus públiques de cada client s'emmagatzemaran dins la base de dades dins un certificat, igual que la clau pública del gestor estarà disponible en un certificat en cada client. Aquesta classe serà l'encarregada d'accedir als fitxers dels certificats emmagatzemats en format X.509[8].

- **RSA**[16]: aquesta serà la classe encarregada d'executar les operacions de xifrat i desxifrat asimètric.

- **AES**[17]: aquesta classe serà l'encarregada d'executar les operacions de xifrat i desxifrat simètric.

- **Signar**: aquesta classe serà l'encarregada d'executar les operacions de signatura i verificació de la signatura.



Imatge 5 - Classes involucrades en la consulta de dades generals.

3.6.3 Consulta de visita.

La consulta de dades d'una visita pot ser executada tant per un pacient com per un metge, per tant, s'haurà de verificar que qui sol·licita les dades de la visita és, o bé el mateix pacient, o bé un metge amb accés a les dades del pacient.

La notació que s'utilitzarà per descriure aquest protocol serà la següent:

- **IdUsuari**: Identificador del usuari del que es sol·liciten les dades.
- **IdUsuariU**: Identificador del usuari sol·licitant.
- **IdGestor**: Identificador del gestor.
- **DescVisita**: Descriptor de la visita que es sol·licita.
- **Lv**: Llista de descriptors de visites de IdUsuari.
- **La**: Llista d'accés a les visites de IdUsuari.
- **Lm**: Llista de metges de IdUsuariU.
- **Lp**: Llista de pacients protegida de IdUsuariU.
- $\mathbf{P_{Gestor}}$: Clau pública del gestor.
- $\mathbf{P_{Client}}$: Clau pública del client.
- $\mathbf{S_{Gestor}}$: Clau privada del gestor.
- $\mathbf{S_{Client}}$: Clau privada del client.

Els passos a seguir són els següents:

- Com en totes les operacions, primer caldrà autenticar-se, utilitzant el protocol descrit en el punt 3.6.1.

- A partir de les dades rebudes del gestor en el segon pas del protocol d'autenticació, $\boldsymbol{P_{client}[N_i, N_g, IdGestor]}$, el client realitzarà les següent operacions:

- Desxifrar les dades rebudes amb S_U, i obtenir N_i', N_g i ***IdGestor***.

- Comparar N_i' amb N_i, si són iguals vol dir que el servidor està responent a la seva petició, ja que N_i l'ha generat el client. En cas contrari es genera un error.

- Obtenir P_{Gestor}.

- Amb P_{Gestor} xifrar ***IdUsuari***, N_g (el número que ens servirà per identificar-nos amb el gestor), ***Consulta_Dades_Visita*** (identificador d'operació sol·licitada), ***IdVisista***. Així s'obté: $P_{Gestor}[N_g,$ ***Consulta_Dades_Generals, IdUsuari, IdVisita]***.

- Enviar el resultat al gestor.

- Amb les dades rebudes del client, el gestor efectuarà les següents operacions:

  - Amb S_g desxifrar les dades rebudes del client i obtenir: N_g, ***Consulta_Dades_Visita, IdUsuari, IdVisita***.

  - Amb N_g, accedir a la base de dades i obtenir ***IdUsuariU***.

  - Si no es recupera informació de la base de dades a partir de N_g, es genera un error (error d'autenticació).

  - Comprovar que ***IdUsuariU*** tingui accés a la visita sol·licitada. Es distingiran dos casos:

    - ***IdUsuariU = IdUsuari***.
      - Verificar que ***IdVisita*** està dins de ***Lv*** de ***IdUsuariU***. En cas contrari, retornar error.

    - ***IdUsuariU*** és metge.
      - Verificar que ***IdUsuari*** està dins ***Lp*** de ***IdUsuariU***. En cas contrari, retornar error.

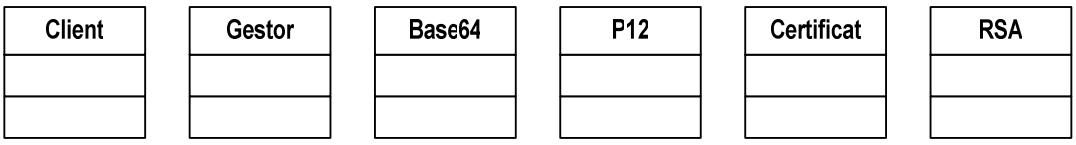
      - Verificar que ***IdUsuariU*** està dins de ***Lm*** de ***IdUsuari***. En cas contrari, retornar error.

      - Verificar que ***IdVisita*** està dins de ***Lv*** de ***IdUsuari***. En cas contrari, retornar error.

- Si es disposa d'accés a la visita:
  - Obtenir la visita, ***V***, identificada per ***IdVisita***.
  - Obtenir la clau pública de ***IdUsuariU***, P_{client}, i xifrar la visita per obtenir $P_{client}[V]$.
- Eliminar N_g, N_i, ***IdUsuariU*** de la base de dades.
- Enviar les dades obtingudes al client.

- Amb les dades rebudes del gestor, $P_{client}[V]$, el client executa les següent operacions:
  - Obtenir S_{client}.
  - Desxifrar les dades utilitzant S_{client}, i obtenir les dades de la visita.

Les classes utilitzades per a implementar les operacions descrites en aquest apartat són les mateixes que s'han utilitzat en l'apartat anterior, "*Consulta de dades generals*", excepte les classes de verificació de signatura i xifratge simètric.



Imatge 6 - Classes involucrades en la consulta de visita.

3.6.4 Consulta de pacients assignats.

La consulta de pacients assignats a un metge, només podrà ser executada per un metge, per tant, s'haurà de verificar que qui sol·licita la llista de pacients és un metge.

La notació que s'utilitzarà per descriure aquest protocol serà la següent:

- **IdUsuariU**: Identificador del usuari sol·licitant.
- **IdGestor**: Identificador del gestor.
- **Lp**: Llista de pacients protegida.

- $\mathbf{P_{Gestor}}$: Clau pública del gestor.
- $\mathbf{P_{Client}}$: Clau pública del client.
- $\mathbf{S_{Gestor}}$: Clau privada del gestor.
- $\mathbf{S_{Client}}$: Clau privada del client.

Els passos a seguir són els següents:

- Com en totes les operacions, primer caldrà autenticar-se, utilitzant el protocol descrit en el punt 3.6.1.

- A partir de les dades rebudes del gestor en el segon pas del protocol d'autenticació, $\boldsymbol{P_{client}[N_i, N_g, IdGestor]}$, el client realitzarà les següent operacions:

  - Desxifrar les dades rebudes amb $\boldsymbol{S_U}$, i obtenir $\boldsymbol{N_i'}$, $\boldsymbol{N_g}$ i ***IdGestor***.

  - Comparar $\boldsymbol{N_i'}$ amb $\boldsymbol{N_i}$, si són iguals vol dir que el servidor està responent a la seva petició, ja que $\boldsymbol{N_i}$ l'ha generat el mateix client. En cas contrari es genera un error.

  - Obtenir $\boldsymbol{P_{Gestor}}$.

  - Amb $\boldsymbol{P_{Gestor}}$ xifrar $\boldsymbol{N_g}$ (el número que ens servirà per identificar-nos al gestor), ***Llista_Pacients*** (identificador d'operació). Així s'obté: $\boldsymbol{P_{Gestor}[N_g, Llista\_Pacients]}$.

  - Enviar el resultat al gestor.

- Amb les dades rebudes del client, el gestor efectuarà les següents operacions:

  - Amb $\boldsymbol{S_g}$ desxifrar les dades rebudes del client i obtenir: $\boldsymbol{N_g}$, ***Llista_Pacients***.

  - Amb $\boldsymbol{N_g}$, accedir a la base de dades i obtenir ***IdUsuariU***.

  - Si no es recupera informació de la base de dades a partir de $\boldsymbol{N_g}$, es genera un error (error d'autenticació).

  - Comprovar que ***IdUsuariU*** sigui un metge, a partir del atribut *Organizational Unit Name* del certificat de l'usuari.

  - Obtenir ***Lp***.

- Obtenir la clau pública de ***IdUsuariU***, P_{client}, i xifrar la llista de pacients protegida, per obtenir $P_{client}[Lp]$.
- Eliminar N_g, N_i, ***IdUsuariU*** de la base de dades.
- Enviar les dades obtingudes al client.

- Amb les dades rebudes del gestor, $P_{client}[Lp]$, el client executa les següent operacions:
  - Obtenir S_{client}.
  - Desxifrar les dades utilitzant S_{client}, i obtenir la llista de pacients protegida, ***Lp***.
  - Tractar ***Lp*** per ser mostrada a l'usuari.

Les classes utilitzades per a implementar les operacions descrites en aquest apartat són les mateixes que s'han utilitzat en l'apartat "*Consulta de dades de visita*".

3.6.5 Afegir visita a l'historial.

Les operacions de manteniment d'historial mèdics només les pot executar un metge, així s'haurà de verificar que qui intenta afegir una visita a l'historial és un metge.

Aquesta operació té la particularitat de que prèviament s'ha d'haver accedit a les dades del historial del pacient, de totes formes, es torna a executar un procediment d'autenticació de l'usuari.

La notació que s'utilitzarà per descriure aquest protocol serà la següent:

- **IdUsuariM**: Identificador del metge que vol executar l'operació.
- **IdPacient**: Identificador del pacient a qui pertany l'historial.
- **DescVisita**: Descriptor de la visita.
- **IdGestor**: Identificador del gestor.
- **Lv**: Llista de visites protegida
- **H**: Historial.
- **V**: Visita a afegir.
- $\mathbf{P_{Gestor}}$: Clau pública del gestor.
- $\mathbf{P_{Client}}$: Clau pública del client.

- $\mathbf{S_{Gestor}}$: Clau privada del gestor.
- $\mathbf{S_{Client}}$: Clau privada del client.
- **K**: Clau de sessió.
- $\mathbf{N_k}$: Número aleatori que ens servirà per generar K.

Els passos a seguir són els següents:

- Com en totes les operacions, primer caldrà autenticar-se, utilitzant el protocol descrit en el punt 3.6.1.

- A partir de les dades rebudes del gestor en el segon pas del protocol d'autenticació, $\boldsymbol{P_{client}[N_i, N_g, IdGestor]}$, el client realitzarà les següent operacions:

  - Desxifrar les dades rebudes amb $\boldsymbol{S_U}$, i obtenir $\boldsymbol{N_i'}$, $\boldsymbol{N_g}$ i ***IdGestor***.

  - Comparar $\boldsymbol{N_i'}$ amb $\boldsymbol{N_i}$, si són iguals vol dir que el servidor està responent a la seva petició, ja que $\boldsymbol{N_i}$ l'ha generat el mateix client. En cas contrari es genera un error.

  - Obtenir les dades de la visita ***V***.

  - Obtenir $\boldsymbol{S_{Client}}$.

  - Signar la visita, ***V***, amb $\boldsymbol{S_{Client}}$. i obtenir $\boldsymbol{S_{Client}[V]}$.

  - Obtenir $\boldsymbol{P_{Gestor}}$.

  - Amb $\boldsymbol{P_{Gestor}}$ xifrar $\boldsymbol{N_g}$ (el número que ens servirà per identificar-nos al gestor), ***V***, $\boldsymbol{IdPacient, S_{Client}[V], Afegir\_Visita}$ (identificador d'operació). Així s'obté: $\boldsymbol{P_{Gestor}[N_g, Afegir\_Visita, V, IdPacient, S_{Client}[V]]}$.

  - Enviar el resultat al gestor.

- Amb les dades rebudes del client, el gestor efectuarà les següents operacions:

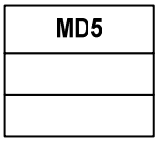
  - Amb $\boldsymbol{S_{Gestor}}$ desxifrar les dades rebudes del client i obtenir: $\boldsymbol{N_g, Afegir\_Visita, V, IdPacient, S_{Client}[V]}$.

  - Amb $\boldsymbol{N_g}$, accedir a la base de dades i obtenir ***IdUsuariM***.

- Si no es recupera informació de la base de dades a partir de N_g, es genera un error.
- Comprovar que ***IdUsuariM*** sigui un metge, a partir del atribut *Organizational Unit Name* del certificat de l'usuari, en cas contrari generar un error.
- Comporvar que ***IdPacient*** és un pacient assignat a ***IdUsuariM***, en cas contrari generar un error.
- Obtenir P_{client}.
- Amb P_{client} verificar la signatura $S_{Client}[V]$.
- Obtenir el descriptor de ***V***, ***DescVisita***.
- Afegir ***DescVisita*** a ***Lv***.
- Signar ***Lv*** amb S_{Gestor}.
- Crear una nova clau de sessió a partir d'un número aleatori, N_k, $H[N_k]$.
- Signar ***Lv*** amb la clau de sessió ***K***, $E_K[Lv]$
- Xifrar la clau de sessió ***K*** amb les claus públiques dels metges de la llista de metges de ***H***, obtenint una nova ***Lv***.
- Afegir ***V*** a la base de dades.
- Eliminar N_g, N_i, ***IdUsuariM*** de la base de dades.

Les classes utilitzades per a implementar les operacions descrites en aquest apartat són les mateixes que s'han utilitzat en l'apartat "*Consulta de visita*", excepte que en aquest cas s'afegeix una nova classe, que serà l'encarregada de:

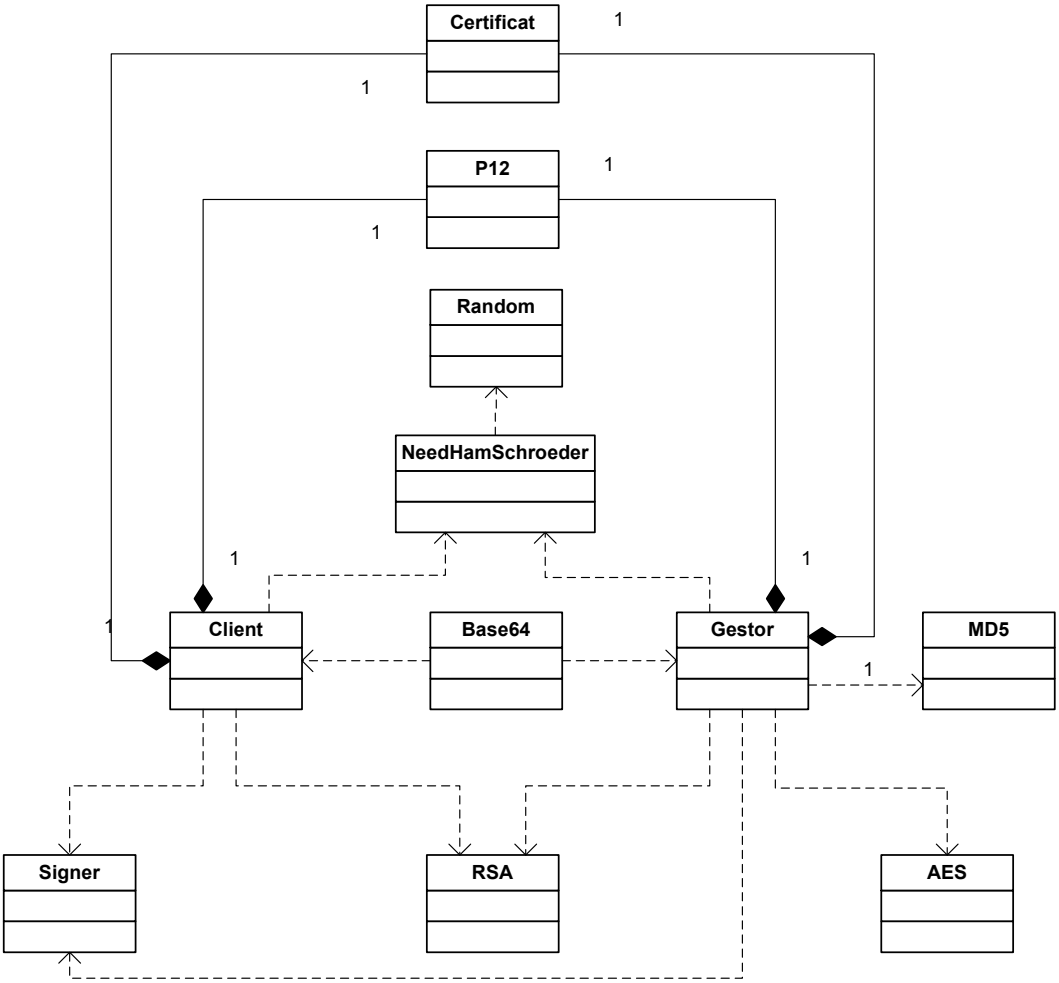
- **MD5**[19]: classe que calcularà el *hash* per a generar la clau de sessió.



Imatge 7 - Classe de funcions de *hash* MD5.

3.7 Diagrama de classes.

En aquest apartat es mostra un diagrama UML[21] amb les classes utilitzades per implementar l'esquema criptogràfic, i les seves relacions.



Imatge 8 - Relació entre les classes de l'esquema criptogràfic.

3.8 Nivell d'aplicació de la seguretat.

Com s'ha pogut veure durant aquest capítol, la seguretat sempre s'aplica a nivell d'aplicació. No només en la validació dels protocols de l'esquema criptogràfic, sinó també sobre les dades que es transmetran entre el gestor i el client.

Aquesta implementació independitza l'aplicació del mitja de comunicació utilitzat, i en facilita la seva portabilitat.

3.9 Proves.

Per a la realització de les proves d'aquest apartat s'ha generat una classe per cada protocol, aquestes classes simulen el diàleg entre el client, pacient o metge, i el gestor.

Un exemple de les classes de prova es pot trobar l'Apendix B d'aquest document.

4 Representació i gestió de les dades: XML

4.1 Introducció.

En poc temps, l'XML[1] ha esdevingut l'estàndard utilitzat per a l'intercanvi de dades entre aplicacions i la base per a una nova generació d'aplicacions i protocols. Això ha estat, principalment, per la seva capacitat d'extensió, la qual permet ampliar el conjunt de dades a representar molt fàcilment, i perquè a més de facilitar la gestió de les dades, els hi aporta estructura que en facilita la comprensió.

En aquest Projecte, s'ha utilitzat XML[1] per a l'emmagatzemament de les dades, com s'explicarà amb més detall en el capítol corresponent, i per a la transferència de dades entre els pacients i metges, i el gestor. Les raons principals per a utilitzar XML[1] han estat que permet representar les dades en format text, el que facilita la seva transferència, i que facilita la comunicació amb altres aplicacions.

Per a la gestió interna de les dades XML[1], s'ha utilitzat la llibreria JDOM[11].

4.2 Estructures de documents XML.

En aquest apartat es presentaran les estructures dels documents XML utilitzats per a la representació de les dades.

En aquest cas, l'aplicació generadora és la mateixa que la consumidora, això permet estalviar la definició dels DTD[1] dels documents, en cas de connexió amb altres aplicacions, caldria definir-los.

4.2.1 Autenticació.

Aquest document representa l'estructura de dades necessàries per al procés d'autenticació entre un metge/pacient i el gestor.

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

<autenticacio>
      <ni></ni>
      <ng></ng>
      <idUsuari></idUsuari>
</autenticacio>
```

El document conté els números aleatoris generats durant el procés d'autenticació, i l'identificador de l'usuari que efectua la comunicació.

4.2.2 Petició de servei.

Aquest document representa l'estructura de dades necessàries per a les peticions de servei d'un pacient/metge cap el gestor.

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

<peticio>
      <ng></ng>
      <operacio></operacio>
      <parametres>
            <parametre></parametre>
            <parametre></parametre>
            . . .
      </parametres>
</peticio>
```

Conté el número que identifica a l'usuari que genera la petició, l'identificador d'operació que sol·licita, i els paràmetres que necessita la operació sol·licitada, el nombre de paràmetres és variable en funció de l'operació, no s'identifiquen per nom, perquè l'aplicació ja sap l'ordre en que arribaran.

4.2.4 Visita i Descriptor de Visita.

Els documents comentats en aquest punt representen les estructures de dades necessàries per a representar les dades d'una visita.

En aquest cas es presenten dos documents, el corresponent a la visita, i el corresponent al descriptor de la visita. Això és així perquè els descriptors de visita es podran tractar de forma individual, i podran estar inclosos en altres documents, com ara l'historial d'un pacient.

Aquest document representa l'estructura de les dades d'una visita.

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

<visita>
      <descriptor>
            <identificador></identificador>
            <data></data>
            <hora></hora>
            <tema></tema>
            <idmetge></idmetge>
      </descriptor>
      <anamnesi></anamnesi>
      <diagnosi></diagnosi>
      <tractament></tractament>

      <signatura></signatura>
</visita>
```

Conté les dades que d'una visita: el descriptor de la visita que identifica la visita, l'anamnesi que conté la indagació portada a terme pel metge, la diagnosi i el tractament prescrit, i finalment una signatura de les dades del metge del descriptor.

Aquest document representa l'estructura d'un descriptor de visita.

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

<descriptor>
      <identificador></identificador>
      <data></data>
      <hora></hora>
      <tema></tema>
      <idmetge></idmetge>
</descriptor>
```

Conté les dades que identifiquen una visita: l'identificador únic de visita, la data i hora de la visita, el tema de la visita i l'identificador únic del metge que efectua la visita.

4.2.3 Metge.

Aquest document representa l'estructura de les dades d'un metge: dades de text en clar, i dades binàries representades en Base64[14][15], corresponents a camps xifrats i signatura.

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

<metge>
      <nom></nom>
      <cognoms></cognoms>
      <idColegiat></idColegiat>
      <dni></dni>
      <especialitat></especialitat>
      <certificat></certificat>

      <sobreDigital>
            <dades></dades>
            <clau></clau>
      </sobreDigital>

      <signatura></signatura>

      <llistaPacients>
            <pacient></pacient>
            <pacient></pacient>
            . . .
      </llistaPacients>

</metge>
```

Conté les dades generals del metge: nom, cognoms, identificador de col·legiat, DNI, especialitat i el certificat digital del metge.

A part dels camps generals, el document conté els següents camps:

- ***sobreDigital***: el qual conté un camp de dades amb la llista de pacients assignats al metge; i un camp amb la clau de xifrat del sobre, xifrada amb la clau privada del gestor.

- ***signatura***: signatura de la llista de pacients amb la clau privada del gestor

- ***llistaPacients***: pacients assignats al metge. Aquesta llista es carrega a partir del contingut del sobre digital.

4.2.5 Historial.

Aquest document representa l'estructura de les dades d'una visita: dades de text en clar, i dades binàries representades en Base64[14][15], corresponents a camps xifrats i claus.

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

<historial>
      <nom></nom>
      <cognoms></cognoms>
      <numTargetaSanitaria></numTargetaSanitaria>
      <dni></dni>
      <grupSanguini></grupSanguini>
      <certificate></certificate>

      <alergies>
            <alergia></alergia>
            <alergia></alergia>
            . . .
      </alergies>

      <llistaVisites>
            <descriptors></descriptors>
            <signatura></signatura>

            <llistaDescriptors></llistaDescriptors>

            <llistaAcces>
                  <clauAcces></clauAcces>
                  <clauAcces></clauAcces>
                  . . .
            </llistaAcces>
      </llistaVisites>

      <llistaMetges>
            <sobreDigital>
                  <dades></dades>
                  <clau></clau>
            </sobreDigital>
            <metge></metge>
            <metge></metge>
            . . .
      </llistaMetges>

</historial>
```

Conté les dades generals d'un pacient: nom, cognoms, número de targeta sanitària, DNI, grup sanguini, el seu certificat i una llista amb les al·lèrgies del pacient.

També conté una llista amb els descriptors de visites del pacient, aquesta llista està composta pels següents camps:

- ***descriptors***: aquest camp conté una llista dels descriptors de visites del pacient, signada pel gestor i xifrada amb una clau de sessió.

- ***signatura***: aquest camp conté la signatura de la llista de descriptors de visites del pacient.

- ***llistaDescriptors***: conté la llista de descriptors de visita del pacient desxifrada.

- ***llistaAcces***: conté la clau de sessió xifrada amb les claus del gestor, el pacient i els metges que tenen accés a la llista de descriptors de visita.

4.3 Utilització dels documents XML.

En aquest apartat s'explica com s'integren els documents XML[1] comentats en l'apartat anterior amb la resta de components del Projecte.

Les dades gestionades per l'aplicació s'emmagatzemaran en format XML[1], és a dir, a la base de dades de l'aplicació s'emmagatzemaran directament les estructures XML[1] amb les dades corresponents a pacients, metges i historials mèdics.

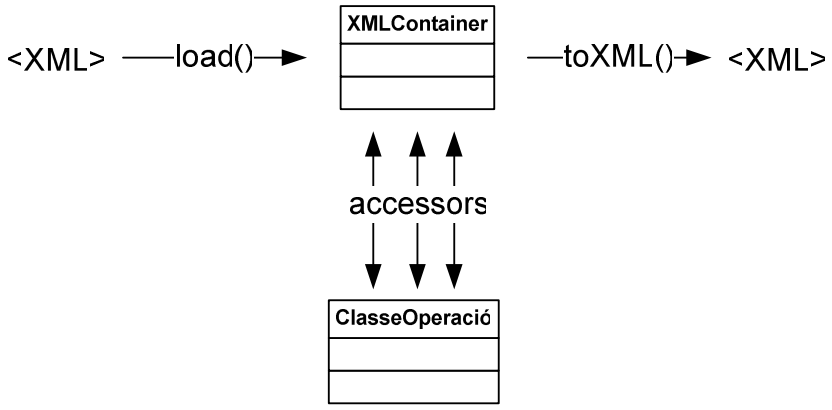
Les comunicacions entre els diferents actors també es portaran a terme mitjançant l'intercanvi de missatges en format XML[1].

Així com la representació XML[1] de les dades facilita la seva representació, emmagatzematge i utilització per a les comunicacions, aquesta representació no és gaire adequada per a la manipulació interna de dades.

És per això que s'han implementat una sèrie de classes contenidors, capaces de gestionar un document XML[1] concret, oferint mètodes per accedir als valors emmagatzemats en un document i, a partir d'aquests valors, crear un document XML[1] per ser emmagatzemat a la base de dades o utilitzat en una comunicació.

Les classes que contenen la lògica de l'aplicació, no coneixen el format dels documents XML[1], i en cap moment accedeixen a les dades en aquest format.

La següent figura mostra la relació entre les classes contenidors, el flux de dades, i les classes que contenen la lògica de l'aplicació.



Imatge 9 - Funcionament d'un contenidor XML.

Alhora de carregar les dades, aquestes classes porten a terme les següents validacions:

- Validació d'estructura dels documents: es valida si els documents estan *ben formats*, i que si contenen caràcters *extranys*.
- Validació d'estructura: es valida que contingui totes les dades necessàries per un tipus de document.

En cas de complir-se alguna de les validacions, es genera un error i no es permet continuar amb l'execució.

Les validacions de seguretat, és a dir, si alguna de les dades del document ha d'estar signada i/o xifrada, es porten a terme als components consumidors. Això és així per eliminar qualsevol lògica dels contenidors d' XML[1].

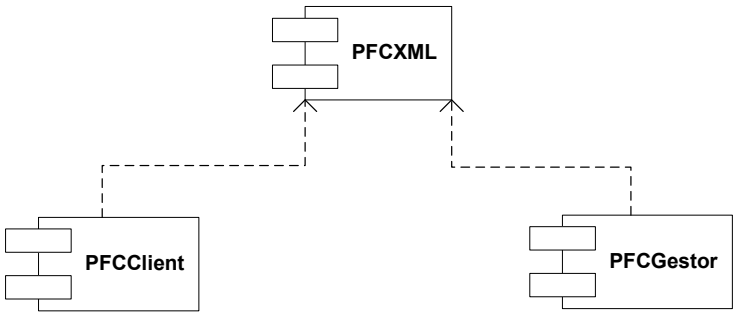
Així, quan es vol realitzar alguna operació amb dades XML[1], el que cal fer és crear una instancia del contenidor que calgui, carregar les dades dins aquest contenidor, aquest les validarà i ja estaran disponibles per a ser utilitzades con un objecte Java[20] *clàssic*.

Amb les dades del contenidor, es realitzaran les operacions que calgui i es podran *serialitzar* en format XML[1] per a tornar a ser emmagatzemades o enviades a un altre actor de l'aplicació.

4.4 Implementació.

S'ha desenvolupat un component específic, aïllat de la resta de components de l'aplicació, que conté les classes contenidor.

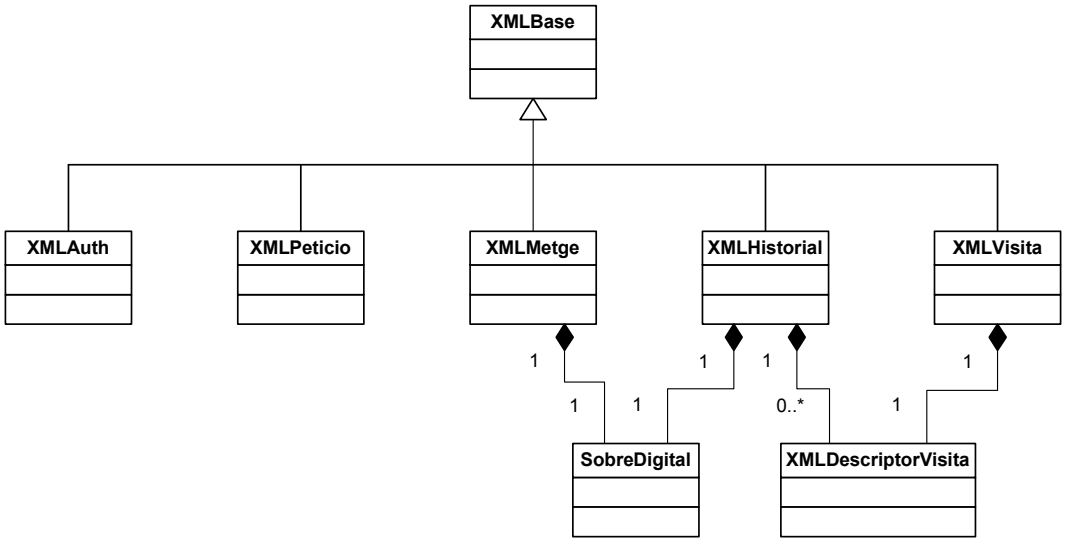
S'ha pres aquesta decisió per separar les funcionalitats dels diferents components de l'aplicació, així com hi ha dos components encarregats exclusivament de l'esquema criptogràfic, un pel client i un pel gestor, també hi ha un component encarregat exclusivament de la gestió XML[1].



Imatge 10 – XML, relació de components de l'aplicació.

Aquesta separació també permet realitzar modificacions sobre la implementació sense afectar a la resta de components, com podria ser un canvi de tecnologia de gestió de XML[1].

En el següent diagrama es representa l'estructura de classes que modela la gestió dels documents XML[1].



Imatge 11 – Relació de les classes de la gestió de documents XML.

4.5 Proves.

Per a la realització de les proves d'aquest apartat, s'han utilitzat les classes generades per a les proves de l'esquema criptogràfic.

Només ha calgut substituir el format de la missatgeria, de manera que l'intercanvi entre el client i el gestor consisteixi en els documents XML[1].

Un exemple de les classes de prova es pot trobar l'Apendix B d'aquest document.

5 Comunicacions entre components: RMI.

5.1 Introducció.

La comunicació remota entre els diferents components de l'aplicació és una part tant important com ho és l'esquema criptogràfic. Per aconseguir els objectius principals del Projecte, s'ha de garantir que les dades emmagatzemades no seran accessibles per ningú que no hi tingui accés, i que qui hi tingui accés, ho pugui fer des de qualsevol ubicació.

RMI[2] està incorporat en la distribució de Java[20] estàndard des de la versió 1.1.

Per simplificar la comunicació remota entre els diferents aplicatius, s'ha optat per la utilització de RMI[2], ja que ens evita una sobrecàrrega de feina i està plenament integrat amb Java[20], en front d'altres opcions d'ús més general com CORBA o ONC.

RMI[2] és el mecanisme que proporciona Java[20] per a la comunicació remota entre diferents components. En una comunicació RMI[2] es pot distingir entre el client remot, que és qui sol·licita el servei, i el servidor que proporciona el servei. Alhora, en un servidor hi poden haver-hi diferents instancies de classes servidores oferint serveis a diferents clients.

5.2 Comunicacions RMI.

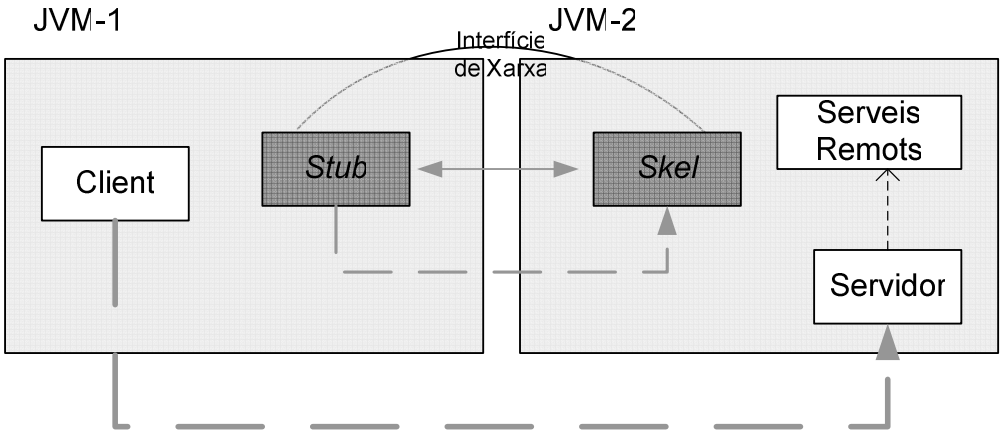
RMI[2] permet efectuar invocacions a objectes remots, enviar i rebre per un canal de comunicació els arguments i resultats, i tractar els errors com a excepcions Java[20]. Les invocacions es poden fer per referència, invocació de mètodes, o per valor, invocacions d'objectes complerts.

Per tal d'invocar un objecte/mètode, el client ha de conèixer-lo i localitzar-lo, i per això el servidor ha d'informar de les classes i mètodes disponibles. Per això s'utilitza un servei de directori, el registre RMI[2], *rmiregistry*.

Per que un objecte sigui remot, aquest ha d'implementar una interfície remota, i cadascun dels mètodes ha de ser capaç de gestionar excepcions del tipus *java.rmi.RemoteException*.

Per la seva part, el client necessita conèixer la interfície del objecte remot, i necessita conèixer el tipus del *stub* (referència que el servidor envia de l'objecte remot) que serà el representant local de l'objecte remot.

Les dades que es poden intercanviar objectes mitjançant RMI[2] són: qualsevol **tipus bàsic** de dades de Java[20], o objectes que implementin la interfície ***Serializable***.



Imatge 12 - Esquema de comunicacions RMI.

5.3 Comunicació RMI de les aplicacions.

Tal com s'ha definit anteriorment en aquest mateix document, com a premisses bàsiques de seguretat s'han establert:

- L'únic que podrà accedir a la base de dades serà el Gestor.
- Les claus privades no viatjaran mai d'un aplicatiu a un altra.
- Totes les dades viatjaran xifrades.

Per aconseguir això, s'ha implementat un objecte d'invocació remota que oferirà els serveis del Gestor. Els mètodes d'aquest objecte rebran les dades que necessitin, xifrades amb la clau pública del Gestor, i retornarà els resultats xifrats amb la clau pública del client (metge o pacient).

Amb aquest sistema, els clients només han de conèixer la ubicació del servidor i els mètodes remots que aquest ofereix.

Per aconseguir aquest objectius ha calgut definir una interfície per publicar els mètodes remots, ***IGestor***, una classe que els implementi, ***Gestor***, i un servidor que publiqui aquests mètodes, ***Servidor***.

5.3.1 IGestor i Gestor.

Com ja s'ha enunciat, ***IGestor*** és la interfície que publica els mètodes remots. El client ha de conèixer aquesta interfície per a poder utilitzar-la.

Gestor és la classe que implementa els mètodes remots, i per tant, serà la classe que accedirà a la base de dades, i gestionarà els documents XML[1] que s'intercanviaran amb el client.

5.3.2 Servidor.

Aquesta classe és l'encarregada de instanciar un *servidor*, una instancia de ***Gestor***, i fer-lo públic a la xarxa.

Haurà de conèixer el nom amb el que es farà públic el servei. En aquest cas el servei es diu ***PFCGestor***:

```
. . .
Gestor ges = new Gestor();
Naming.rebind("rmi://localhost:1099/PFCGestor", ges);
. . .
```

5.4 Implementació.

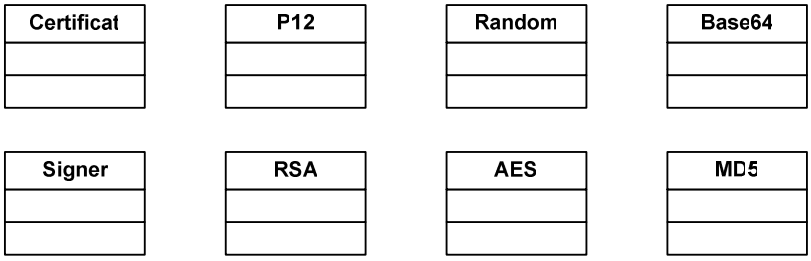
Respecte a l'estructura del projecte comentada fins ara, s'han ampliat les funcionalitats de les classes ***Gestor*** i ***Client*** per tal què ofereixin serveis remots i els utilitzin, respectivament, i s'han separat les classes que conformen l'esquema criptogràfic del client i del gestor de les classes que proporcionen els serveis de criptografia i d'accés remot, així s'augmenta la modularitat del sistema.

D'aquesta manera s'aconsegueixen aïllar la implementació de l'esquema criptogràfic, les classes de serveis criptogràfic i les classes que proporcionen accés remot.

Després d'aquesta divisió s'obtenen els següents components:

- Component *PFCEngines*, aquest component conté les classes que proporcionen els algoritmes criptogràfics necessaris per a implementar l'esquema criptogràfic.

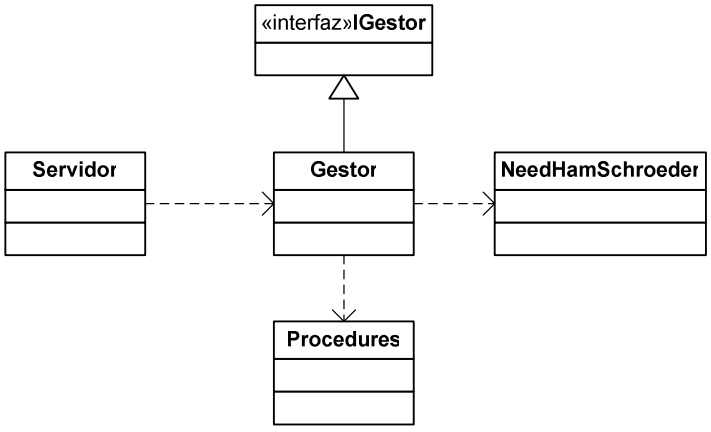
  El següent diagrama UML[21] mostra les classes d'aquest component.



Imatge 13- Component de serveis criptogràfics. PFCEngines.

- Component *PFCGestor*, aquest component conté les classes necessàries per a implementar la part de l'esquema criptogràfic corresponent al Gestor. També incorpora el servidor encarregat de publicar els serveis remots del Gestor.

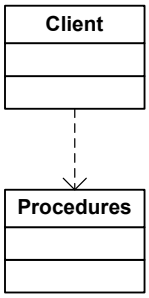
  El següent diagrama UML[21] mostra les classes d'aquest component.



Imatge 14 - Esquema criptogràfic del Gestor amb serveis remots. PFCGestor

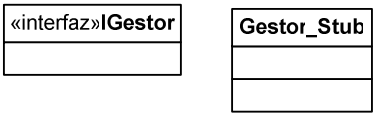
- Component PFCClient, aquest component conté les classes necessàries per a implementar la part de l'esquema criptogràfic corresponent al Client.

  El següent diagrama UML[21] mostra les classes d'aquest component.



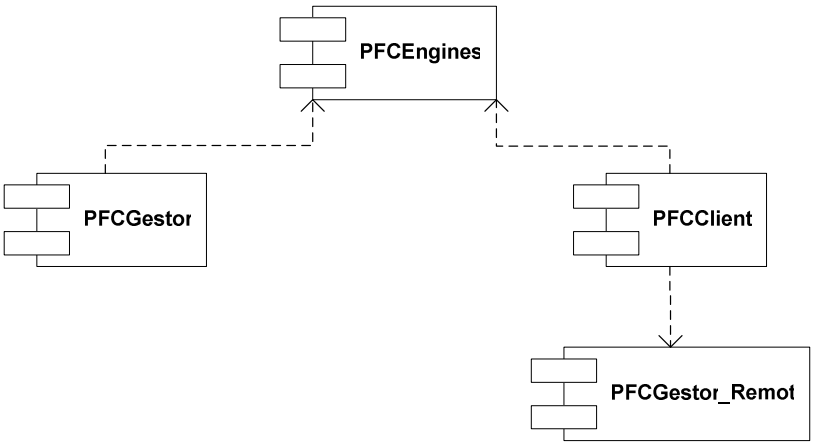
Imatge 15 - Esquema criptogràfic del Client. PFCClient

- Component PFCGestorRemot, per a fer accessible al client l'accés remot, aquest ha de conèixer la interfície, i ha de tenir accés a l'*stub* que encapsula la comunicació remota. Per això s'ha dissenyat un quart component que proporciona aquestes classes.



Imatge 16 - Classes client d'accés remot. PFCGestorRemot

Les relaciones entre els diferents components són les següents:



Imatge 17 – RMI, relació entre components de l'aplicació.

D'aquesta manera s'aconsegueix independitzar els diferents components de l'aplicació, de forma que es pugui modificar la implementació de cada component (p.e. canviar el proveïdor de serveis criptogràfics, o els parsers XML), sense afectar a la resta.

5.5 SSL sota RMI.

Alhora d'implementar el xifrat de les dades que es transmeten, es pot escollir fer-ho a nivell de d'aplicació, o nivell de capa de transport.

Amb la tecnologia actual, el xifrat a nivell de transport dona un gran rendiment, sobretot perquè es disposa de maquinari especialitzat que el porta a terme.

La implementació de RMI[2] que proporciona Java[20] permet efectuar comunicacions segures mitjançant SSL a nivell de comunicacions RMI[29], utilitzant *sockets*.

S'ha desestimat aqueta opció per augmentar la portabilitat de l'aplicació, si bé es cert que es milloraria el rendiment, si en algun moment es decidís utilitzar algun altra canal de comunicació diferent a RMI[2], les modificacions a realitzar serien més costoses.

5.6 Proves.

Per a la realització de les proves d'aquest apartat, s'han utilitzat les classes generades per a les proves de l'esquema criptogràfic i la integració amb XML[1].

Només ha calgut substituir la creació de l'objecte que referència al Gestor.

Un exemple de les classes de prova es pot trobar l'Apendix B d'aquest document.

6. Base de Dades.

6.1 Introducció.

Fins ara, per poder realitzar les proves, les dades corresponents a les diferents estructures de dades emprades (historials, visites, dades del metge...) s'emmagatzemaven en fitxers aïllats i sense cap relació entre ells.

La integració amb la base de dades permetrà emmagatzemar totes les dades de forma persistent i controlada, millorant el rendiment de l'aplicació *Gestor* i simplificant l'accés a les dades.

A més, els sistemes gestors de base de dades, d'ara endavant SGBD, proporcionen una sèrie de funcionalitats que ens proporcionaran valor afegit en quan a seguretat:

- **Control d'accés**: els SGBD defineixen l'accés a la informació a partir d'usuaris i rols, restringint l'accés a la informació en funció d'aquests.

- **Auditoria d'operacions**: els SGBD permeten enregistrar les diferents operacions sobre la base de dades, amb un ampli ventall d'informació sobre cada operació.

Com a SGBD s'utilitza MySQL[5]. Es tracta d'un SGBD de lliure distribució, disponible per a plataformes MS-Windows, Unix, Linux i MacOS, la qual cosa permet instal·lar l'aplicació en qualsevol d'aquestes plataformes, recordem que Java[20] també és multiplataforma.

6.2 Model de dades.

A continuació es mostra el model de dades dissenyat pel Projecte. Cal tenir en compte els següents punts:

- Un usuari de l'aplicació pot ser alhora metge i pacient.

- Tots els metges i pacients són usuaris.

- Les visites no estan vinculades a cap historial.

- Donat que es treballa amb estructures de dades XML[1], el que es desa a la base de dades són, directament, aquestes estructures XML[1]. Això permet aïllar el model de dades dels canvis en les estructures de l'aplicació.

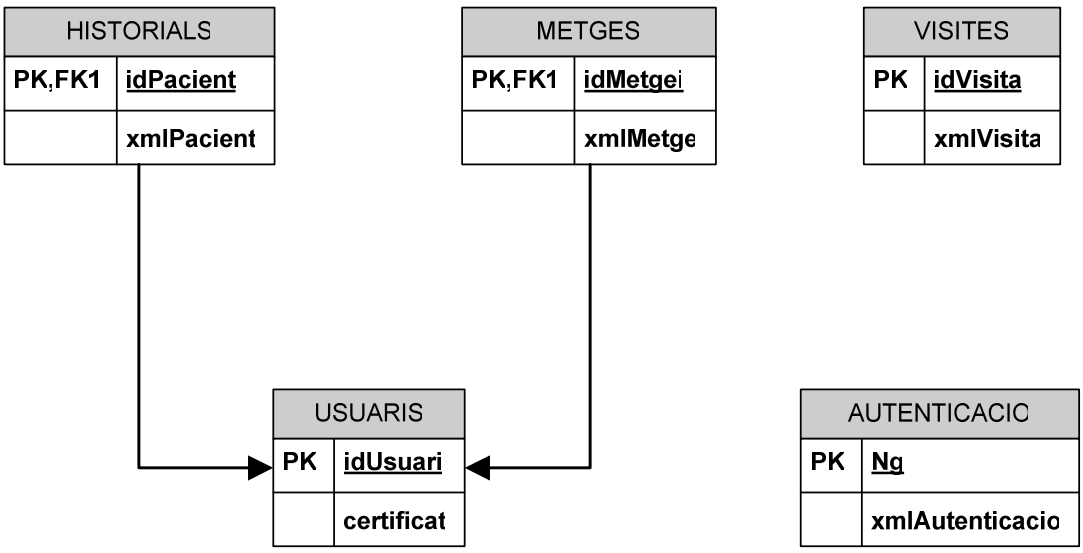
S'ha decidit desar els documents XML[1] a la base de dades, en lloc de les dades que aquests contenen, pels avantatges que això representa:

- Independitza el model de dades de les estructures XML utilitzades, així es pot variar el contingut dels documents XML[1] sense afectar el model de dades. Si es vol afegir o eliminar alguna dada, no caldrà modificar la base de dades ni els mètodes d'accés.

- Millora el rendiment de l'aplicació, ja que evita el temps de composició dels documents XML[1]. Desar les dades sense estructura implicaria més accessos a la base de dades i més pesats, a més de la posterior construcció de les estructures XML[1].

- Simplifica el codi de recuperació i inserció de dades a la base de dades.

La següent figura mostra el model de dades aplicat:



Imatge 18 - Model de dades.

A continuació es descriuran amb més detall cadascuna de les taules.

6.2.1 Taula *usuaris*.

Aquesta taula contindrà l'identificador d'un usuari i el seu certificat:

Camp	Tipus	Descripció.
***idUsuari** (PK)*	VARCHAR(32)	Número de seguretat social del usuari.
certificat	MEDIUMTEXT	Certificat X509 de l'usuari, emmagatzemat en format Base64.

Tota entitat amb accés a l'aplicació, metge o pacient, excepte el Gestor, haurà d'estar enregistrada dins de la taula ***usuaris***.

S'ha descarta el DNI com a clau, donat que es (*massa*) comú que existeixin DNIs repetits. Aquest fet es més difícil que passi amb el número de Seguretat Social.

L'objectiu principal d'aquesta taula és definir quins usuaris tenen accés a l'aplicació i proporcionar accés als certificats amb la clau pública de cadascun.

6.2.2 Taula *historials*.

Aquesta taula contindrà els historials mèdics dels usuaris enregistrats com a pacients:

Camp	Tipus	Descripció.
***idPacient** (PK)*	VARCHAR(32)	Número de seguretat social del pacient.
xmlHistorial	MEDIUMTEXT	Estructura XML[1] amb l'historial del pacient.

L'objectiu principal d'aquesta taula és emmagatzemar els historials dels pacients. Com ja s'ha comentat, existeix la restricció de que un pacient primer ha d'estar registrat com a usuari a la taula ***usuaris***.

6.2.3 Taula *metges*.

Aquesta taula contindrà les dades dels usuaris enregistrats com a metges:

Camp	Tipus	Descripció.
***idMetge** (PK)*	VARCHAR(32)	Número de seguretat social del metge.

Camp	Tipus	Descripció.
xmlMetge	MEDIUMTEXT	Estructura XML[1] amb les dades del metge.

L'objectiu principal d'aquesta taula és emmagatzemar les dades dels metges. A l'igual que en el cas dels pacients, existeix la restricció de que un metge primer ha d'estar registrat com a usuari a la taula ***usuaris***.

6.2.4 Taula *visites*.

Aquesta taula contindrà les dades de les visites dels pacients:

Camp	Tipus	Descripció.
idVisita *(PK)*	VARCHAR(32)	Identificador de la vista.
xmlVisita	MEDIUMTEXT	Estructura XML[1] amb les dades de la visita.

L'objectiu principal d'aquesta taula és emmagatzemar les dades de les visites. Les visites no estan vinculades amb els historials de forma evident, per aconseguir relacionar les dades de la visita amb les del seu historial corresponent, s'hauria de recuperar l'historial i disposar de la clau privada correcta per accedir a l'identificador de visita.

6.2.5 Taula *autenticació*.

Aquesta taula emmagatzemarà les dades corresponents a les sessions autenticades:

Camp	Tipus	Descripció.
Ng *(PK)*	VARCHAR(32)	Número aleatori generat pel Gestor.
xmlAutenticacio	MEDIUMTEXT	Estructura XML[1] amb les dades d'autenticació.

L'objectiu principal d'aquesta taula és emmagatzemar, de forma temporal, les dades d'autenticació de les diferents peticions. La clau d'accés a aquesta taula serà el número aleatori generat pel Gestor durant el procés d'autenticació.

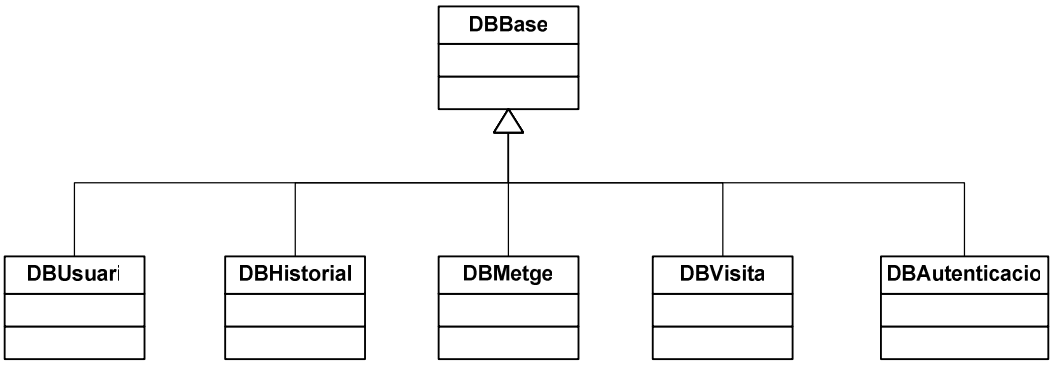
6.3 Implementació.

Per a la implementació de l'accés a la base de dades, s'ha seguit el mateix patró que per a la resta del Projecte, s'ha dissenyat un component específic i aïllat, que serà l'encarregat d'accedir a la base de dades.

Aquest component està format per classes especialitzades a gestionar cadascuna de les entitats definides en el model de dades. És a dir, per a cada entitat, hi ha una classe capaç de gestionar les operacions per a recuperar-la, desar-la o eliminar-la.

D'aquesta manera, si s'afegeix una entitat nova al model de dades, caldrà afegir al component la classe que modeli les operacions per a gestionar-la.

La següent figura mostra el diagrama UML[21] de les classes del component d'accés a la base de dades.



Imatge 19 – Relació de classes d'accés a base de dades.

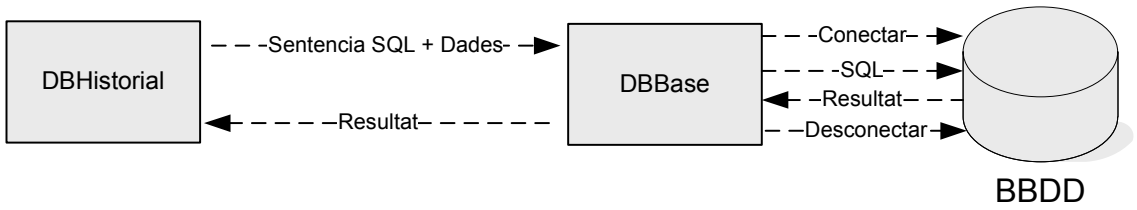
Com s'ha pogut veure en la definició del model de dades, totes les entitats segueixen el mateix model: un camp clau i un camp de dades. Això ha facilitat la implementació de la classe ***DBBase***, ja totes les sentencies de selecció, inserció i modificació de dades segueixen el mateix model.

La classe ***DBBase*** proporciona els mètodes necessaris per a establir una connexió amb la base de dades i alliberar-la, així com els mètodes per a executar sentències de selecció, inserció, modificació i eliminació de dades. També proporciona camps contenidors per a la clau i les dades corresponents.

La resta de classes se serveixen, mitjançant herència, de ***DBBase*** per accedir a les dades.

Cada classe coneix l'estructura de la taula corresponents, i les sentencies SQL[22] que cal executar per cada operació sobre les dades.

La següent imatge mostra com interactuen els diferents components per a executar una operació sobre la base de dades.



Imatge 20 - Execució d'operació sobre base de dades.

Les operacions que s'efectuen sobre la base de dades són atòmiques, és a dir, s'executaran d'una en una, no dins un bloc, no s'executaran dins de transaccions.

Cada cop que es vulgui executar una operació sobre la base de dades, s'obrirà i es tancarà una connexió. Això és així per evitar col·lapsar el *pool* de connexions de la base de dades.

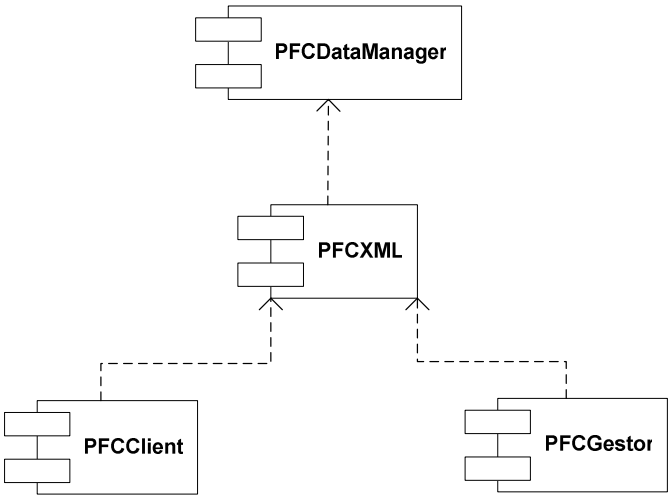
6.4 Integració amb contenidors XML.

Els contenidors XML[1] disposen de dos mètodes dissenyats per a la integració amb les classes que gestionen l'accés a la base de dades:

- *Mètode* ***load***: aquest mètode rep un *stream* Java[20], i és capaç de validar-lo i interpretar-lo per omplir les dades del contenidor.

- *Mètode* ***toXMLString***: aquest mètode bolca el contingut del contenidor en una cadena en format XML[1].

La integració d'aquests dos mètodes permet omplir els contenidors amb dades de la base de dades, i bolcar les dades del contenidor cap a la base de dades, i d'aquesta manera la resta de components de l'aplicació tenen accés a les dades.



Imatge 21 – Base de dades, relació entre components de l'aplicació.

Aquesta implementació aïlla la capa d' XML[1] de la capa d'accés a la base de dades, de fet, es pot omplir un document XML[1] amb qualsevol font que retorni un *stream*: un fitxer, un *socket*, un canal de comunicacions..., el que permetria modificar l'origen de dades de la base de dades a, per exemple, un canal de comunicacions.

Alhora, aïlla els consumidors dels documents XML[1] de la construcció i estructura d'aquests, de manera que només han de conèixer els atributs que necessiten, i el contenidor els hi proporcionarà.

Mantenint les interfícies, s'aconsegueix que els canvis en un dels components no afectin a la resta.

7. Interfície gràfica.

7.1 Introducció.

La interfície d'usuari és la part amb la qual l'usuari final interactuarà amb l'aplicació, per tant, és una de les parts més importants de qualsevol aplicació, ja que una interfície mal dissenyada pot acabar amb la paciència de l'usuari i dificultar la seva acceptació com a eina.

En el cas d'aquest Projecte, la interfície d'usuari no és un requeriment crític, el seu objectiu principal és oferir la possibilitat d'accedir a totes les operacions de l'esquema criptogràfic definit en aquest Projecte, és per això que s'ha optat per simplificar el màxim possible la interfície gràfica de l'aplicació.

La interfície proporcionada consta de tres diàlegs diferenciats: un diàleg principal des d'on es poden visualitzar els historials emmagatzemats, un segon diàleg per a visualitzar les dades de les visites emmagatzemades, y un tercer diàleg per a la identificació de l'usuari.

7.2 Llibreria gràfica utilitzada.

Per a la implementació de la interfície gràfica del Projecte, s'ha utilitzat la llibreria gràfica Swing[23] de Java, amb l'ajuda del *plug-in* d'Eclipse[3] *Visual Editor*[24], aquest *plug-in* facilita molt el disseny d'interfícies Java[20], a més d'accelerar-ne notablement el desenvolupament.

Swing[23] permet dissenyar interfícies per aplicacions Java[20] d'una forma ràpida i senzilla. És de lliure distribució, i està inclòs en la distribució de Java[20].

7.3 Aplicació Client.

Al capítol 8, dedicat a les proves, s'explica com iniciar l'aplicació *Client*.

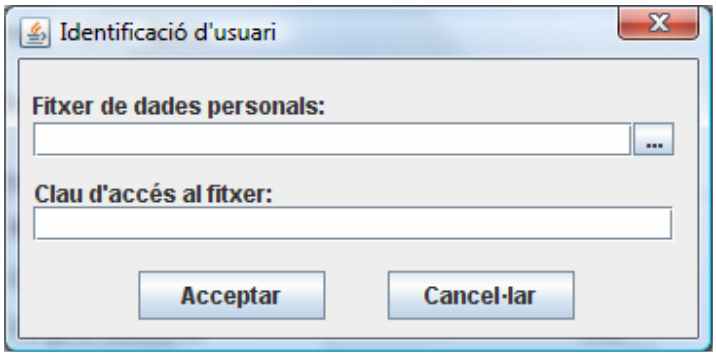
Com s'ha comentat durant la introducció d'aquest capítol, aquesta aplicació consta de tres diàlegs. A continuació es comentarà cadascun d'aquest diàlegs i la seva funció dins de l'aplicació.

Com la majoria d'aplicacions, al inici es mostra una pantalla de benvinguda.

Imatge 22 - Pantalla de benvinguda de l'aplicació Client.

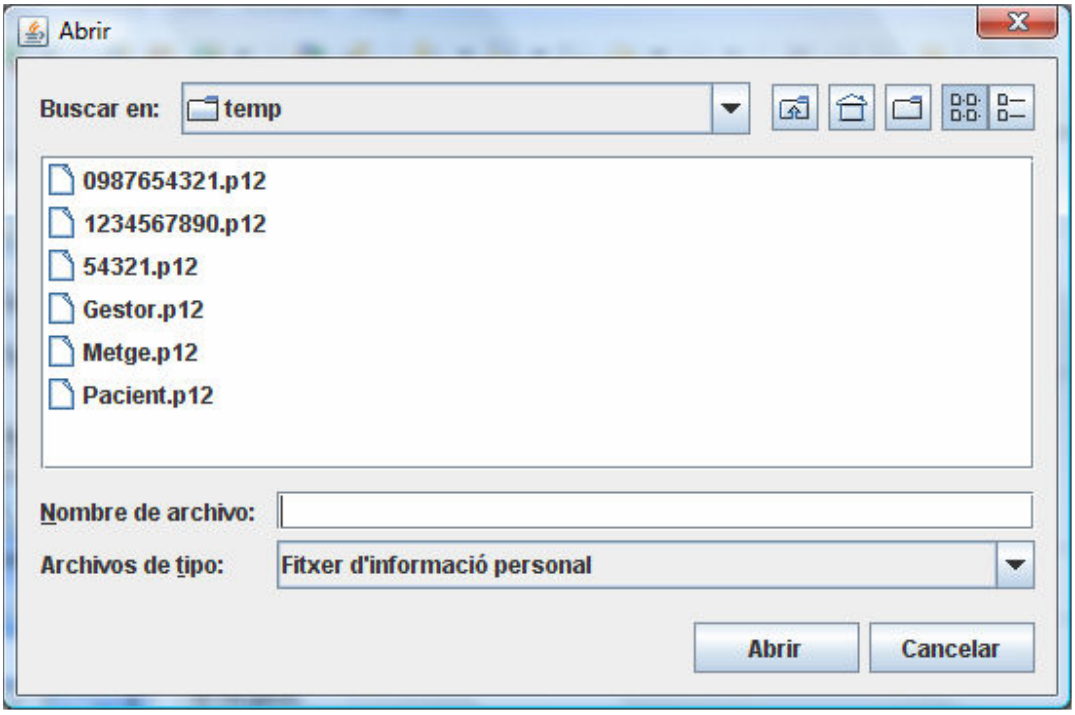
7.3.1 Diàleg d'identificació d'usuari.

Aquest diàleg és el primer que es mostra a l'iniciar l'aplicació, i permet al usuari *client* identificar-se dins d'aplicació. Per a efectuar la identificació, s'haurà d'indicar el fitxer de dades personals de l'usuari i la clau d'accés a aquest fitxer.



Imatge 23 - Diàleg d'identificació d'usuari.

La ubicació del fitxer de dades personals es pot indicar introduint-la a la caixa de text corresponents, o bé buscant-la utilitzant un diàleg de cerca de fitxers que s'obra prement sobre el botó '***...***'.

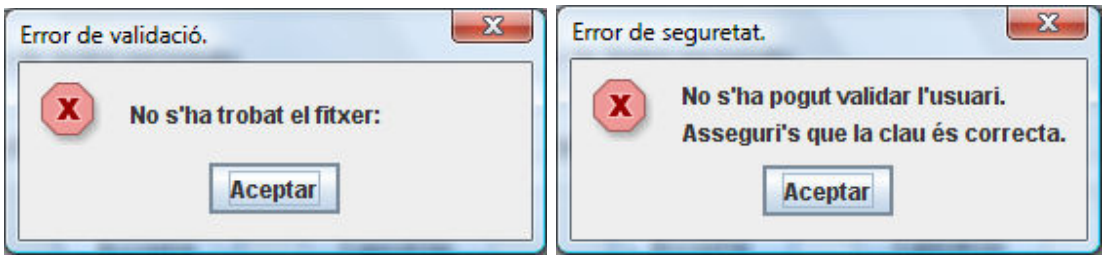


Imatge 24 - Diàleg de localització de fitxers.

La carpeta per defecte de cerca de fitxers d'informació personal s'indica mitjançant els paràmetres de configuració de l'aplicació, la llista completa d'aquests paràmetres es troba a l'Apèndix D.

Un cop indicat el fitxer de dades i la clau, prement el botó '***Acceptar***' es valida la identitat del usuari, comprovant que la clau introduïda correspon al fitxer de dades personals escollit, i es crea la sessió de l'aplicació.

En cas de produir-se algun error, es mostrarà un avís, per exemple:



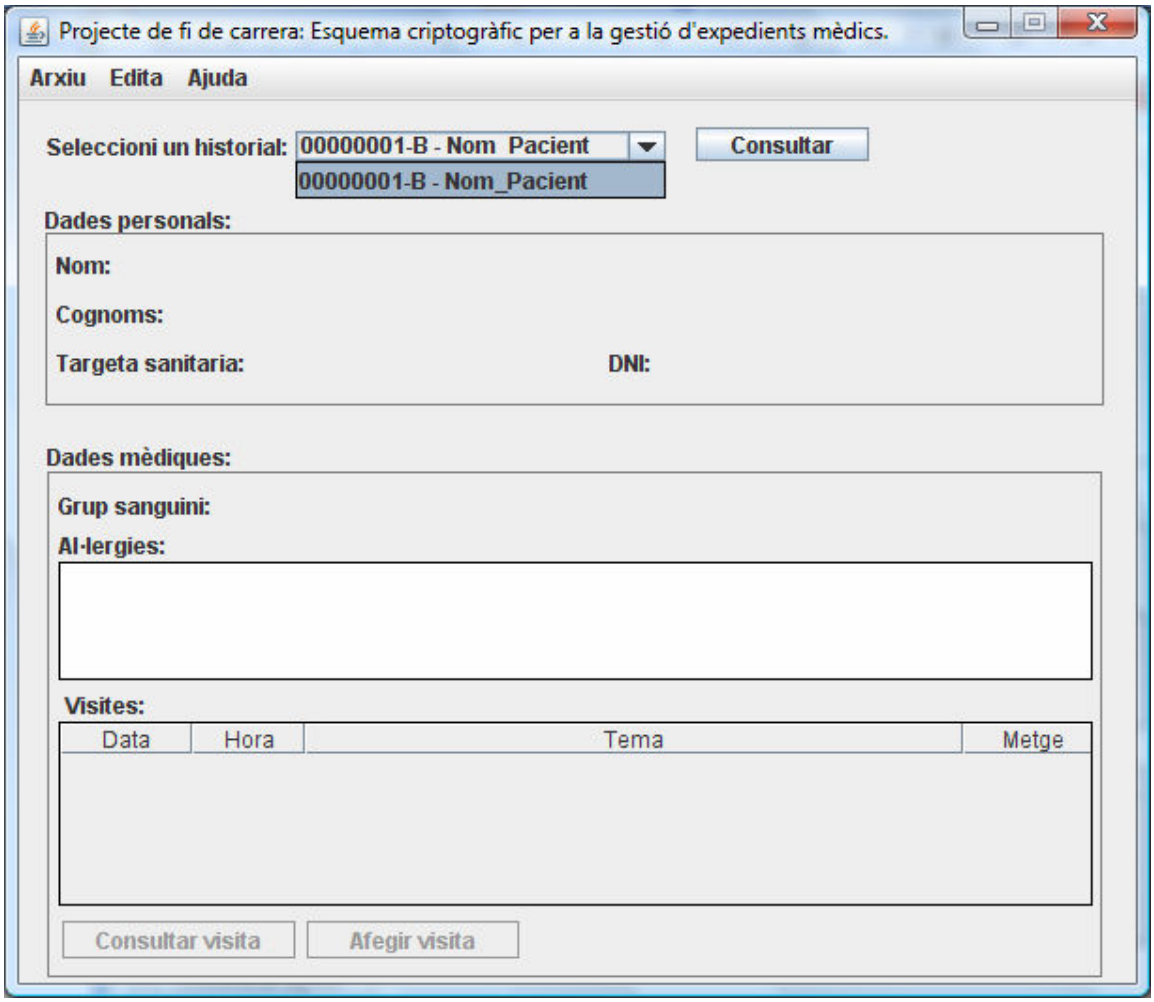
Imatge 25 - Missatges d'error d'autenticació.

7.3.2 Diàleg principal (historials).

Un cop identificat l'usuari, apareixerà el diàleg principal de l'aplicació. Des d'aquest diàleg es poden executar totes les operacions descrites en l'esquema criptogràfic del Projecte.

De forma automàtica, quan s'identifica un usuari, es carrega tota la llista d'historials mèdics que l'usuari pot consultar:

- → Si l'usuari **no és un metge**, la llista d'historials a consultar només tindrà un element, el mateix usuari identificat.
- → Si l'usuari **és un metge**, la llista d'historials a consultar contindrà tots els pacients als que tingui accés.

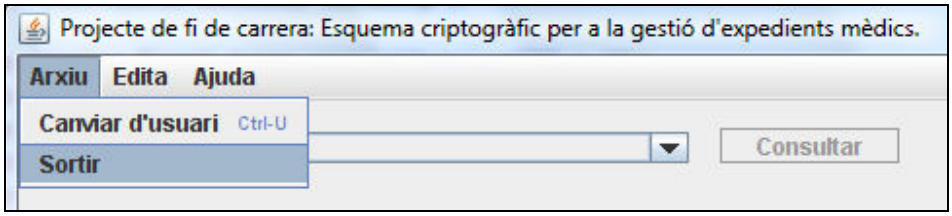


Imatge 26 - Diàleg principal de l'aplicació.

En l'anterior pantalla d'exemple només apareix l'usuari identificat, ja que l'usuari de l'aplicació no és un metge.

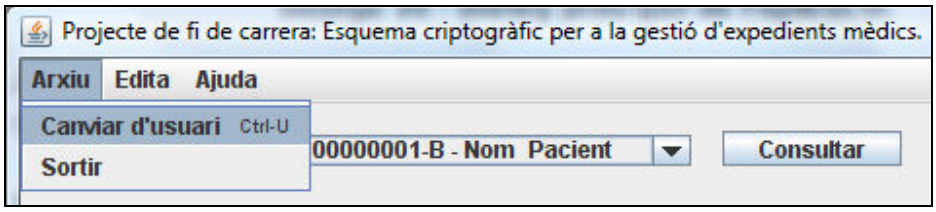
A partir d'aquesta pantalla, un usuari pot accedir a la resta d'operacions de l'aplicació:

- **Sortir de l'aplicació**: per sortir de l'aplicació es pot tancar el diàleg principal amb el botó del diàleg dedicat a tal efecte, o mitjançant la opció de menú ***Arxiu→Sotir***.



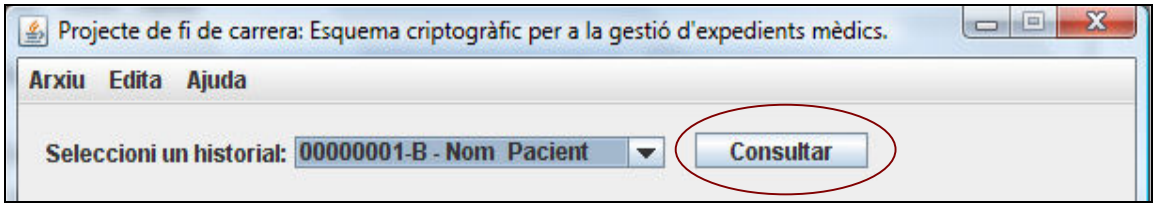
Imatge 27 - Opció de menú sortir de l'aplicació.

- **Canvi d'usuari identificat**: un usuari pot ser alhora un metge o un pacient, a la pràctica això és tradueix en que un usuari pot tenir diversos certificats. Mitjançant la opció de menú ***Arxiu→Canvi d'Usuari***, *CTRL+U*, es pot tornar a identificar un usuari a l'aplicació creant una nova sessió:



Imatge 28 - Opció de menú canvi d'usuari.

- **Consultar un historial**: seleccionant un historial de la llista d'historials, es pot accedir en mode consulta a les dades d'aquest. Per això, un cop seleccionat, s'ha de prémer el botó ***Consultar***.

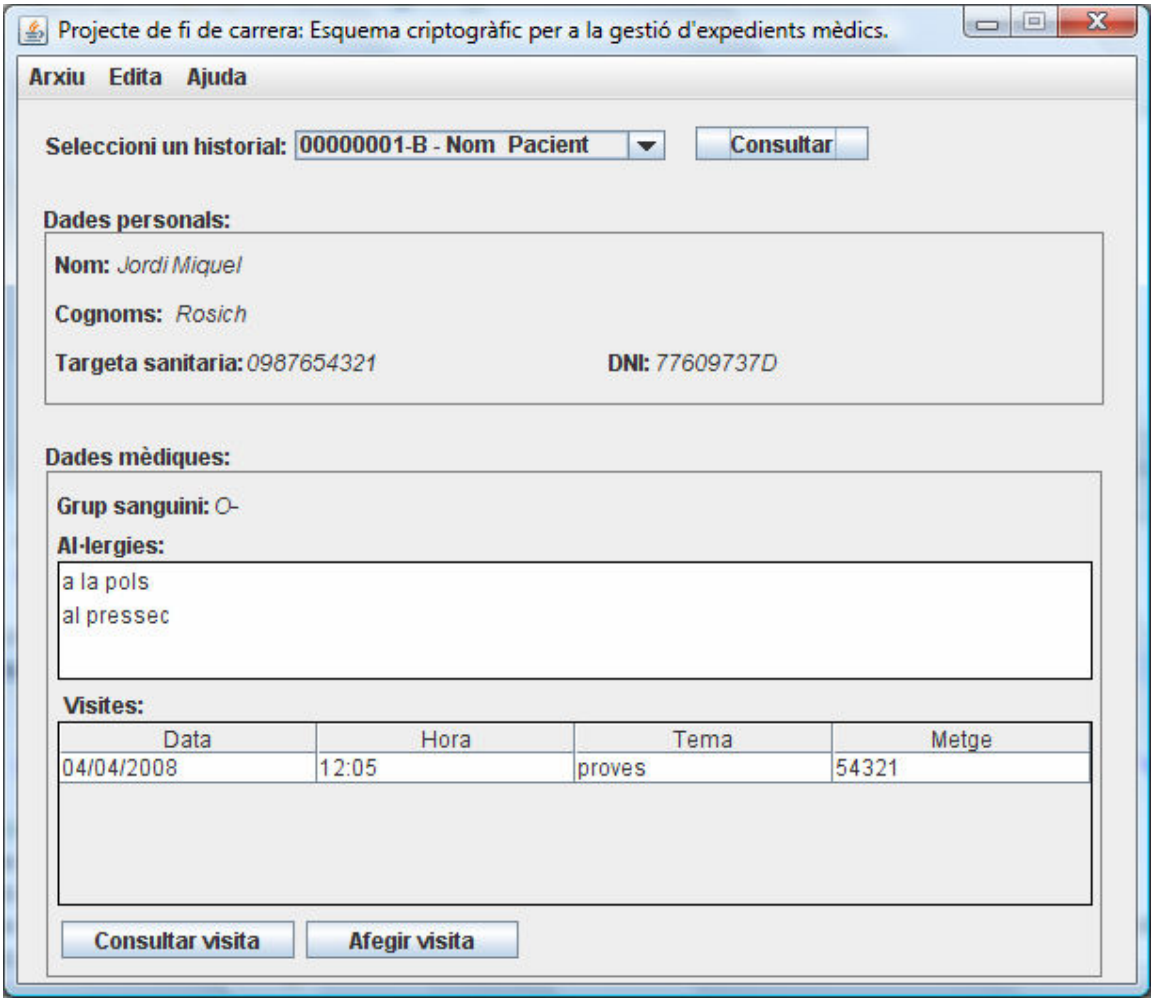


Imatge 29 - Selecció de l'historial a consultar.

Un cop s'ha recuperat l'historial, aquest es mostra als camps corresponents del diàleg principal de l'aplicació.

Un cop carregat l'historial seleccionat, es poden consultar les visites relacionades i afegir-ne de noves. Si no es carrega cap visita, el botó de consulta queda desactivat.

La següent imatge correspon a un historial carregat.

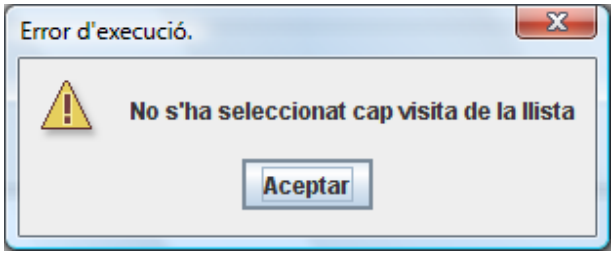


Imatge 30 - Diàleg principal amb l'historial carregat.

7.3.3 Diàleg de dades de visita.

L'accés a aquest diàleg pot ser per a consultar una visita, o per a donar-ne una d'alta.

Si s'intenta consultar una visita sense seleccionar-ne una de la llista:



Imatge 31 - Missatge d'avís de selecció de visita.

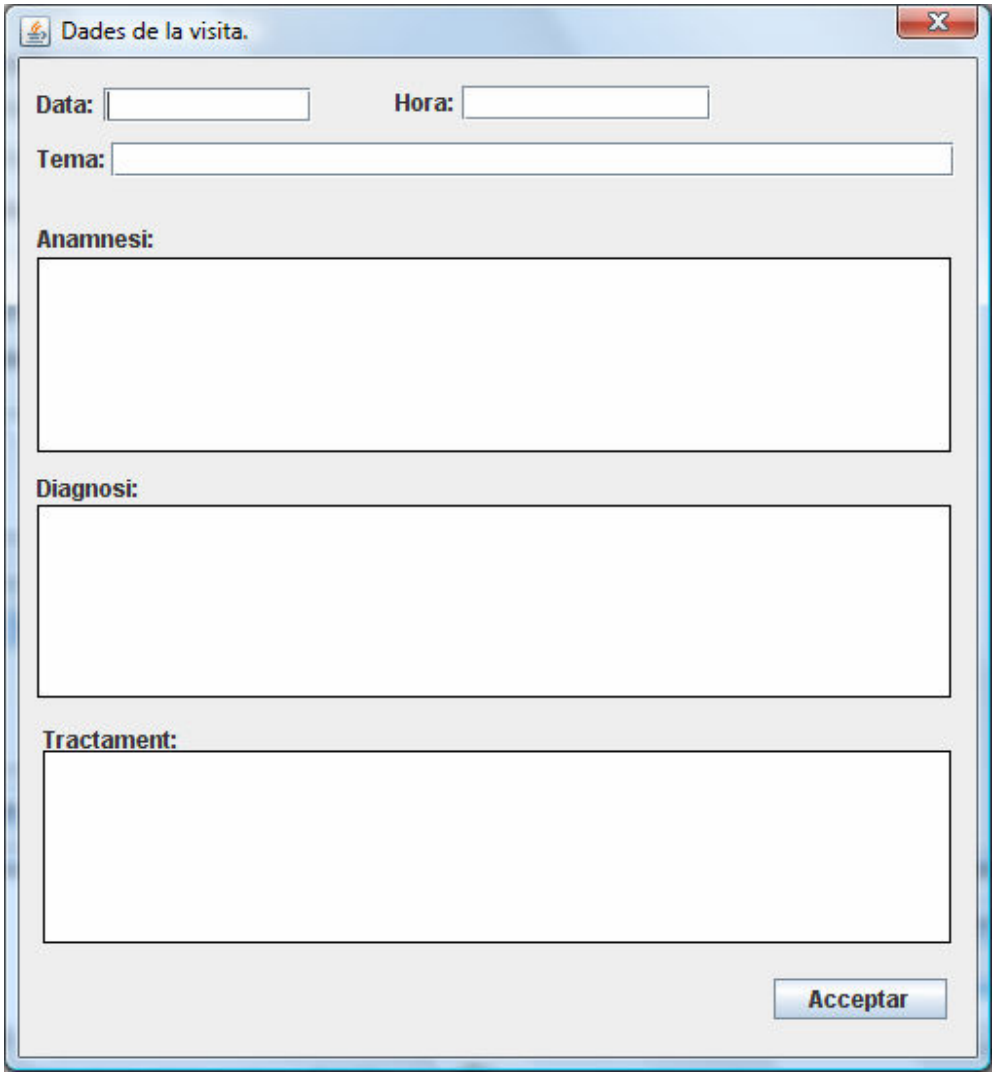
Un cop seleccionat un element de la llista de visites, es prem el botó '***Consultar visita***' que mostrarà el diàleg amb les dades de la visita seleccionada carregades.

Imatge 32 - Diàleg amb les dades d'una visita.

En mode consulta, els camps d'aquest diàleg no són editables, de forma que no es pugui modificar el contingut de les dades.

Si es vol donar d'alta una visita, s'ha de prémer el botó '***Afegir visita***' del diàleg principal. En aquest cas, apareix el mateix diàleg que en el cas d'una consulta de dades d'una visita, però sense informació assignada als camps i amb la possibilitat de modificar el contingut d'aquests.

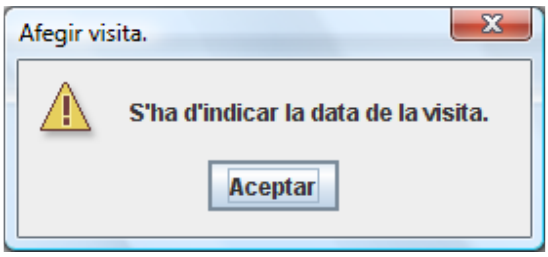
Cal recordar que només els metges podran afegir visites.



Imatge 33 - Diàleg per a la introducció de dades d'una visita.

Un cop completades les dades de la visita, es prem el botó '***Acceptar***', que comprovarà si totes les dades s'han introduït correctament, si es

supera aquesta comprovació s'afegirà la visita a l'historial. En cas de no superar-se al comprovació es mostrarà un missatge avisant l'error, per exemple, si no s'ha indicat la data de la visita:



Imatge 34 - Diàleg d'avís d'error en introducció de dades de visita.

Un cop afegida la visita, es refrescarà de forma automàtica la informació de l'historial que es visualitza al diàleg principal de l'aplicació.

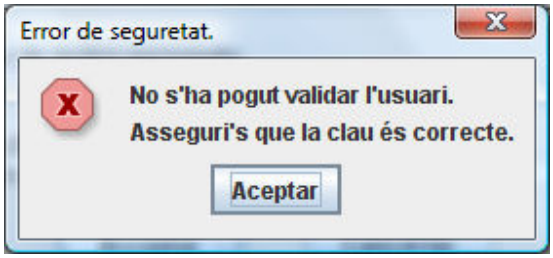
7.3.4 Missatges d'error.

Com s'ha pogut veure en els apartats anteriors, la interfície gràfica és l'encarregada en darrera instancia de gestionar els errors.

Quan es produeix un error, aquest es *fa pujar* per les diferents capes de l'aplicació, fins arribar a la interfície gràfica, on es captura i es mostra dins un diàleg d'error estàndard.

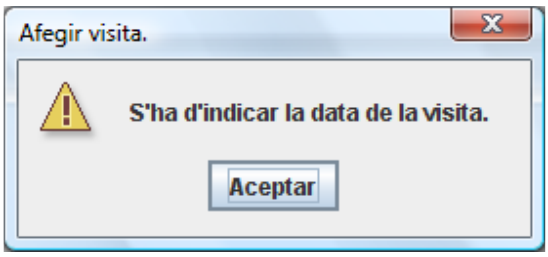
Es diferencien dos tipus de diàlegs d'error, en funció de la gravetat de l'error:

- **Missatge d'error**: es mostren quan es tracta d'un error greu que atura l'execució de l'acció en curs.



Imatge 35 - Exemple de missatge d'error.

- **Missatge d'avís**: es mostren quan es tracta d'un error que pot ser corregit i continuar l'execució de l'acció en curs. Per exemple:



Imatge 36 - Exemple de missatge d'avís.

Tots els missatges d'error i d'avís queden registrats a les traces de l'aplicació. La configuració d'aquestes traces s'explica a l'Apèndix E.

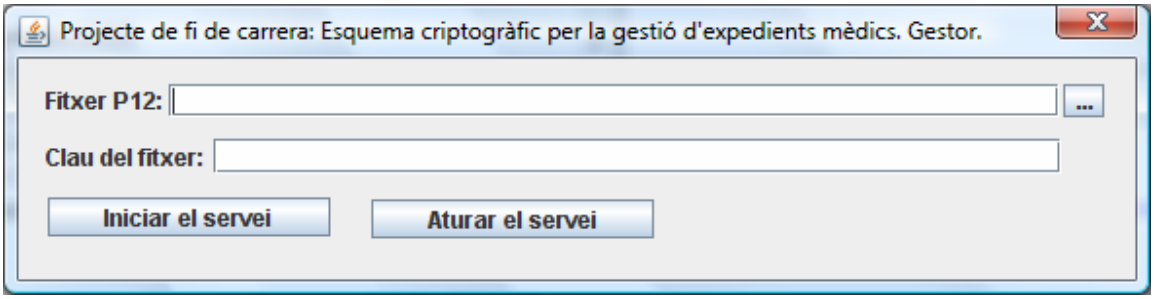
7.3.5 Inici i aturada de l'aplicació *Client*.

Per iniciar l'aplicació client, només cal executar el fitxer de comandes que s'adjunta en la distribució, ***PFCClient.bat***.

Com ja s'ha comentat en aquest capítol, per aturar l'aplicació client es pot utilitzar el botó del diàleg principal situat a la part superior dreta, o a partir de la opció de menú ***Arxiu→Sortir***.

7.4 Aplicació Gestor.

L'aplicació *Gestor* del Projecte funciona com un servei, per tant no té una interfície d'usuari complexa, consisteix en un únic diàleg que permet iniciar i aturar el servei.



Imatge 37 - Diàleg d'aturada i arrencada del servei remot.

No obstant, com a servei, també es permet realitzar una crida amb paràmetres, de manera que es pugui invocar de forma automàtica, i no calgui la interacció de cap persona per iniciar el servei.

Tots els missatges que l'aplicació *Gestor* genera s'emmagatzemen en fitxers de traces que es configuren mitjançant el fitxer de configuració, el detall d'aquest fitxer es pot veure a l'Apèndix D.

7.4.1 Inici i aturada de l'aplicació *Gestor*.

Es diferencien dos casos:

- **Inici assistit**: per iniciar l'aplicació que inicia el servei remot, només cal executar el fitxer de comandes que s'adjunta en la distribució, ***PFCGestor.bat***, sense cap argument.

- **Inici desatès**: com s'ha dit, aquest tipus d'inici està pensat per a ser inclòs en processos automàtics (*scripts*, treballs planificats...).

  Consisteix en invocar al fitxer de comandes ***PFCGestor.bat***, però aquest cop indicant la localització del fitxer .P12 i la seva clau:

  ```
  PFCGestor [Fitxer.p12] [Clau_Fitxer.p12]
  ```

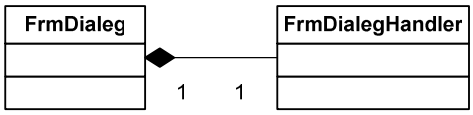
  Aquesta opció no és tan segura com l'anterior, la clau està en clar dins un fitxer de comandes, però pot ser útil i hauria d'estar disponible només per usuaris administradors.

7.7 Implementació.

Per la implementació de la interfície gràfica s'ha ampliat els components corresponents al *Client* i al *Gestor*, afegint els diàlegs de la interfície d'usuari i les classes per gestionar la lògica de la interfície.

S'ha intentat separar el màxim possible la presentació de les dades i la lògica de control de la interfície d'usuari.

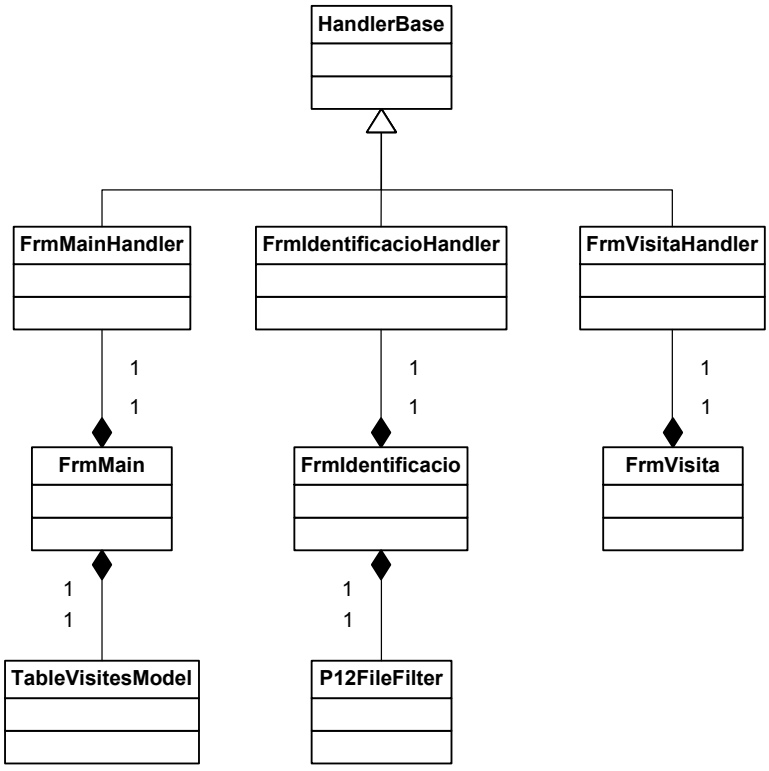
La següent figura mostra l'esquema seguit en la implementació de la classes:



Imatge 38 - Esquema de gestió de la interfície gràfica.

Tots els diàlegs de la interfície contenen únicament els controls Swing[23] i una referència al seu *handler*, que serà l'encarregat de gestionar la lògica del diàleg. Cada event en el diàleg tindrà el seu equivalent en el *handler*, i serà des del *handler* des d'on es cridarà a la resta d'elements de l'aplicació per tal d'executar les peticions de l'usuari.

El següent diagrama UML[21] mostra el conjunt sencer de classes de la interfície gràfica.



Imatge 39 - Relació de classes de la interfície gràfica.

S'ha implementat un model específic per a la gestió de la taula de visites, ***TableVisitesModel***. Aquest model implementa un contenidor específic per a les dades de les visites que es mostraran a la pantalla

principal, amb operacions per a afegir dades, obtenir valors i establir la capçalera de la taula.

De la mateixa manera, s'ha implementat un filtre de fitxers per a que el selector de fitxers només mostri els fitxers que en interessa, fitxers .P12. La classe que implementa aquest filtre és ***P12FileFilter***.

Per a la gestió de les traces, s'ha utilitzat Log4j[25], a l'Apèndix E es comenta amb més detall la seva configuració.

El sistema utilitzat ha consistit en crear a l'iniciar l'aplicació un objecte *static,* comú a tota l'aplicació. D'aquesta forma quan es produeix un error només cal recórrer a aquest objecte per a crear la traça.

8. Instal·lació i joc de proves.

8.1 Introducció.

En aquest capítol es mostra el joc de proves que es proporciona per comprovar el funcionament de les aplicacions desenvolupades.

En aquest capítol s'explica també la instal·lació i configuració de l'aplicació des de zero.

La generació dels certificats i parells de claus s'ha explicat amb detall al capítol 2 d'aquest mateix document.

8.2 Prerequisits.

Per el correcte funcionament de les aplicacions distribuïdes amb aquest projecte, és necessari que al maquinari on s'hagin d'executar el *Client* i/o el *Gestor*, s'ha de disposar d'una versió del JRE, *Java Run Time*, 1.6 o superior correctament instal·lada, amb les polítiques de seguretat del JCE canviades, cal les polítiques de seguretat instal·lades permetin utilitzar criptografia forta. El següent enllaç permet descarregar les darreres versions:

Apartat:
`Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 6`

URL:
`http://java.sun.com/javase/downloads/index.jsp`

En el cas del *Gestor* també s'ha de disposar d'una instal·lació de MySQL[5], versió 5.0 o superior.

8.3 Instal·lació i configuració del servei *Gestor*.

Per la instal·lació del servei *Gestor*, només cal descomprimir el fitxer ***Lliurament.zip***, que s'adjunta en la distribució del Projecte, l'Apèndix F detalla el contingut del distribució.

Aquest fitxer s'ha de descomprimir a la ubicació on es cregui convenient. Es crearà la següent estructura de directoris:

```
[Carpeta_Instal·lacio]\Lliurament\
                                  \bin
                                       \conf
                                       \lib
                                       \logs
                                       \resources
                                       \uoc
                                  \doc
                                  \pki
                                  \src
                                  \project
```

Dins la carpeta *conf* es poden trobar els fitxer de configuració de l'aplicació amb els valors per defecte:

- **gestor.properties**: on es defineixen els atributs del *Gestor*. Aquest arxiu es comenta amb més detall a l'Apèndix D.

- **log4j.properties**: on es defineixen els atributs de les traces. Aquest arxiu es comenta amb més detall a l'Apèndix E.

Dins la carpeta *resources* és la ubicació per defecte dels arxius P12 dels usuaris.

Dins la carpeta *logs* s'emmagatzemaran les traces que es vagin generant.

Dins la carpeta *lib* es poden trobar les llibreries que requereix el projecte.

Dins la carpeta *uoc* es pot trobar l'estructura de classes corresponent a l'aplicació *Gestor*.

8.3.1 Creació de la base de dades.

Dins la carpeta *resources*, es proporciona un *script* de MySQL que crea l'estructura de dades necessària per executar l'aplicació, així com l'usuari que utilitzarà el *Gestor* per accedir-hi.

Per executar aquest *script* cal executar la següent comanda des de la línea de comandes:

```
mysql –u[user_name] –p[password]  < creationScript.sql
```

, on **user_name** correspon a un usuari amb permís de creació de base de dades (p.e. *root*).

8.4 Inici del servei *Gestor*.

Per iniciar el servei *Gestor* només cal executar l'arxiu ***PFCGestor.bat***, que es troba a l'arrel de la instal·lació. Si es proporcionen paràmetres,

PFCGestor [Fitxer.p12] [Clau_Fitxer.p12]

, s'executarà el servei de forma directa. Si no es proporcionen paràmetres, es sol·licitaran a través d'un diàleg.

8.5 Instal·lació i configuració de l'aplicació *Client*.

Per la instal·lació del *Client*, només cal descomprimir el fitxer ***Lliurament.zip***, que s'adjunta en la distribució del Projecte, l'Apèndix F detalla el contingut del distribució.

Aquest fitxer s'ha de descomprimir a la ubicació on es cregui convenient. Es crearà la següent estructura de directoris:

```
[Carpeta_Instal·lacio]\Lliurament\
                                  \bin
                                       \conf
                                       \lib
                                       \logs
                                       \resources
                                       \uoc
                                  \doc
                                  \pki
                                  \src
                                  \project
```

Dins la carpeta *conf* es poden trobar els fitxer de configuració de l'aplicació amb els valors per defecte:

- **client.properties**: on es defineixen els atributs del *Client*. Aquest arxiu es comenta amb més detall a l'Apèndix D.

- **log4j.properties**: on es defineixen els atributs de les traces. Aquest arxiu es comenta amb més detall a l'Apèndix E.

Dins la carpeta *resources* és on s'ha de situar el certificat del *Gestor* **de forma obligatòria** i és la ubicació per defecte dels arxius P12 dels usuaris.

Dins la carpeta *logs* s'emmagatzemaran les traces que es vagin generant.

Dins la carpeta *lib* es poden trobar les llibreries que requereix el projecte.

Dins la carpeta *uoc* es pot trobar l'estructura de classes corresponent a l'aplicació *Client*.

8.6 Inici de l'aplicació *Client*.

Per iniciar l'aplicació client, només cal executar el fitxer de comandes ***PFCClient.bat***, que es troba a l'arrel de la instal·lació

8.7 Contingut del joc de proves.

El joc de proves conté la següent estructura:

- **Lliurament\PKI\Creat_DMM\claus**: conté les claus generades.
- **Lliurament\PKI\Creat_DMM\certificats**: conté els certificats generats.
- **Lliurament\PKI\Creat_DMM\certificatsAutosignats**: conté el certificat autosignat de la CA.
- **Lliurament\PKI\Creat_DMM\Peticions**: conté les peticions de certificats.
- **Lliurament\PKI\Creat_DMM\P12**: conté els fitxer P12 generats.

- **Lliurament\MySQL**: conté un *script* per omplir la base de dades amb historials i usuaris prèviament creats.

Es proporciona una clau, un certificat i un fitxer P12 pels següents actors:

- **Metge**: usuari metge client, amb els dos pacients assignats.
- **Pacient-1**: usuari pacient client.
- **Pacient-2**: usuari pacient client.
- **Gestor**: gestor del sistema.

La paraula clau per accedir a tots els certificats i fitxer P12 generats és: ***uoc0506***.

9. Treball futur.

9.1 Introducció.

Donat el termini de temps del que s'ha disposat, i de la complexitat que un Projecte d'aquest tipus comporta, han quedat molts aspectes per completar.

Aquest capítol pretén esmentar aquests aspectes, i suggerir millores sobre alguns dels punts desenvolupats.

9.2 Millores proposades.

A més de completar l'aplicació amb les funcionalitats bàsiques no desenvolupades: creació d'usuari, creació d'historial, eliminacions...; es proposen les següent millores:

Interfície de client Web.

La necessitat de tenir un programari específic per realitzar la connexió des del client pot presentar problemes alhora de la distribució d'actualitzacions, a més limita els punts d'accés a l'aplicació. Pot ser molt interessant desenvolupar un frontal web per a que realitzi les funcions de l'aplicació *Client*, basant les comunicacions en el protocol *HTTPS*, i utilitzant, per exemple, un *applet* per a executar els serveis criptogràfics des de la màquia client.

Aplicació multi-idioma.

Es podria avaluar la possibilitat de que l'aplicació *Client* pogués mostrar els missatges en diversos idiomes, en funció de, per exemple, la configuració local de la màquina.

Proveïdor de serveis criptogràfics configurable.

Igual que es permet definir de forma externa la connexió a la base de dades, la connexió remota i el format de traces, seria interessant poder escollir quin proveïdor de serveis criptogràfics es vol utilitzar. Ja que tots els proveïdors compleixen els estàndards i publiquen una sèrie constants

que identifiquen els diferents algoritmes (de xifrat, de resum, de signatura...) que proporcionen. Això permetria evolucionar de versió o canviar de proveïdor (IAIK, BouncyCastle...) sense afectar el codi de l'aplicació.

Compatibilitat amb DNIe.

En un futur pròxim, tots els ciutadans de l'estat disposaran de DNI electrònic, de manera que tindran del seu propi certificat digital i el podran portar arreu. Un aspecte a tenir en compte pot ser la capacitat de l'aplicació de llegir DNIe[26] o Smart Cards[27].

Xifrar el contingut de la base de dades.

Actualment la majoria de la informació de la base de dades està en *text clar*, encara que no es pugui relacionar directament amb els pacients, pot ser que una equivocació d'un metge alhora d'afegir dades, pugui revelar informació que hauria de ser confidencial. També podria ser que es pogués relacionar la informació a partir de la observació de les dades emmagatzemades. Seria interessant xifrar la informació abans de desar-la a la base de dades, és cert que la velocitat es veuria afectada, però l'objectiu principal del projecte és oferir seguretat de dades.

Crear un sistema d'autoria d'accés a la Base de Dades.

Els sistemes gestors de base de dades aporten eines per a realitzar auditories sobre les operacions realitzades sobre les dades (consultes, insercions, eliminacions...). Seria recomanable activar les eines, en aquest cas de MySQL[5], i definir polítiques d'auditoria per augmentar el nivell de seguretat i controlar els accessos que no es realitzin a través de l'aplicació desenvolupada.

Augmentar la portabilitat i escalabilitat.

La utilització de Java[20] ja assegura la portabilitat entre plataformes i una bona escalabilitat, en el referent a les aplicacions *Client* i *Gestor*. Però un punt a tenir en compte és la capacitat d'emmagatzemament del sistema gestor de base de dades, i la possibilitat de migrar-lo en algun moment.

Una ampliació a tenir en compte podria consistir afegir una capa d'accés a dades que independitzi l'aplicació del sistema gestor de base de dades i en faciliti la portabilitat, per exemple, Hibernate[28].

10. Conclusions.

Arribats aquest punt, és hora de fer un balanç de la tasca desenvolupada, sobre la feina requerida i la feina presentada.

En aquest capítol es demostra com s'han assolit els requisits sol·licitats a l'enunciat del Projecte.

Els dos objectius principals del Projecte consistien en proporcionar un sistema de gestió segura d'historials mèdics, i que aquesta gestió es pogués desenvolupar a través d'una xarxa de comunicacions. Ambdós objectius s'han aconseguit mitjançant la implementació d'un esquema criptogràfic de gestió d'historials mèdics i la utilització de la tecnologia RMI[2] per a la gestió de la comunicació.

A continuació es detalla el nivell de compliment dels dos objectius principals, i de la resta de línies, objectius *menors*, indicades a l'enunciat.

10.1 Esquema criptogràfic.

Tal com es demanava a l'enunciat, s'ha aconseguit que l'esquema criptogràfic compleixi les següents propietats:

- **Confidencialitat**: les dades són secretes, no hi ha lligams directes de la informació sensible amb els seus propietaris. La informació que proporciona aquests lligams està xifrada, i només és accessible per usuaris autoritzats.

- **Autenticitat**: totes les operacions estan precedides d'una autenticació del usuari que les efectua, es comprova la identitat d'aquest mitjançant els certificats de que es disposa.

- **Integritat**: no es permet manipular les dades sensibles un cop introduïdes. Aquestes dades es desen signades per verificar que no s'han modificat.

- **No repudi**: els metges no poden negar que han incorporat dades a un expedient, les visites incorporen la signatura del metge.

- **Classificació de la informació**: no tota la informació té les mateixes restriccions d'accés. Cal diferenciar els diferents nivells de privadesa.

10.2 Comunicació remota.

Per donar suport a la comunicació entre les diferents parts del Projecte a través d'una xarxa de comunicacions, s'ha utilitzat RMI[2]. La utilització de RMI[2] era un dels requeriments principals de l'enunciat.

La alternativa a RMI[2] més viable hagués estat la utilització de comunicacions HTTP.

10.3 Representació i emmagatzemament de dades.

Les estructures de dades utilitzades en el Projecte estan formades per documents XML[1].

Aquesta representació simplifica molt la transmissió entre els diferents components i, a més, permet que en un futur es puguin connectar altres aplicacions, ja que es basa en un estàndard àmpliament acceptat.

Les mateixes estructures que s'utilitzen per a la transmissió de dades, són les estructures utilitza el *Gestor* per emmagatzemar les dades dels historials.

El *Gestor* utilitza un sistema gestor de base de dades per emmagatzemar les dades (historials, visites, certificats d'usuaris...), concretament MySQL[5].

10.4 Interfície de d'usuaris.

La interfície d'usuaris s'ha intentat fer el més senzilla possible, en part perquè no s'ha disposat de temps suficient per a dedicar-li.

La interfície desenvolupada, encara que senzilla, dóna suport suficient per a la utilització de les operacions incloses a l'esquema criptogràfic, i, com a darrera capa de l'aplicació, gestiona els errors i les traces generades.

10.5 Opinió personal.

El Projecte m'ha servit per a consolidar bona part dels coneixements adquirits en les assignatures de comerç electrònic, seguretat i criptografia. M'ha proporcionat la oportunitat d'aplicar una sèrie de coneixements que em poden ser de gran utilitat, sobretot en aquests moments on la utilització d'Internet creix dia a dia, i les aplicacions que requereixen elevats nivells de seguretat són cada vegada més habituals.

Personalment, d'aquest Projecte, em quedo en els coneixements obtinguts d'aprofundir en els diferents temes de criptografia aplicats, per sobre dels aspectes purament *tècnics*, com el llenguatge utilitzat o la tecnologia de suport a les comunicacions.

El que més m'ha costat a estat canviar la forma d'afrontar la construcció del Projecte (m'agrada començar sempre pel model de dades), però he de reconèixer que el sistema de disseny i implementació incremental proposat s'adapta perfectament al Projecte, i permet abordar les diferents fases gairebé com a desenvolupaments independents.

Bibliografia

[1] World Wide Web Consortium (2008). *Extensible Markup Language (XML)*. Recuperat el 5 de març de 2008, de http://www.w3.org/XML

[2] Sun Microsystems (2008). *Java Remote Method Invocation (Java RMI).* Recuperat el 5 de març de 2008, de http://java.sun.com/products/jdk/rmi/.

[3] Eclipse.org (2008). *Eclipse universal tool platform*. Recuperat el 5 de març de 2008, de http://www.eclipse.org

[4] Institute for Applied Information Processing and Communication (2008). *IAIK*. Recuperat el 5 de març de 2008, de http://jce.iaik.tugraz.at/aboutus/index.php

[5] Sun Microsystems (2008). *The MySQL database server*. Recuperat el 5 de març de 2008, de http://www.mysql.com/documentation/index.html

[6] IETF (1999). *RFC 2560 – X.509 Internet Public Key Infraestructure Online Certificate Status Protocol – OCSP*. Recuperat el 21 de març de 2008, de http://www.ietf.org/rfc/rfc2560.txt

[7] RSA Laboratories. *PKCS#12: Personal Information Exchange Syntax Standard*. Recuperat el 21 de març de 2008, de http://www.rsasecurity.com/rsalabs/node.asp?id=2138

[8] IETF (1999). *RFC 2510 - X.509 Public Key Infraestructure Certificate Management Protocols*. Recuperat el 21 de març de 2008, de http://www.ietf.org/rfc/rfc2510.txt

[9] UOC 2003 - Apunts de *Comerç Electrònic – Mòdul 2 Seguretat en el comerç electrònic.*

[10] UOC 2003 - Apunts de *Criptografia.*

[11] jdom.org. *JDOM*. Recuperat el 24 de març de 2008, de http://www.jdom.org.

[12] R. Needham i M. Schroeder. *Using Encryption for Authentication in Large Networks of Computers - Communications of the ACM* pp.393-399, 1978.

[13] DIMACS.edu. *Needham-Schroeder Public Key Protocol*. Recuperat el 24 de març de 2008, de http://dimacs.rutgers.edu/Workshops/Security/program2/boyd/node14.html.

[14] Wikipedia.org. *Base64*. Recuperat el 24 de març de 2008 de http://es.wikipedia.org/wiki/Base64

[15] IETF Tools .*The Base16, Base32 and Base64 Data Encodings*. Recuperat el 24 de març de 2008 de http://tools.ietf.org/html/rfc4648.

[16] RSA Laboratories. *PKCS #1: RSA Cryptography Standard*. Recuperat el 24 de març de 2008 de http://www.rsa.com/rsalabs/node.asp?id=2125.

[17] John J.G. Savard (2005). *The Advanced Encryption Standard (Rijndael)*. Recuperat el 25 de març de 2008 de http://www.quadibloc.com/crypto/co040401.htm.

[18] The OpenSSL Project. *About the OpenSSL Project*. Recuperat el 25 de març de 2008 de http://www.openssl.org/about/.

[19] IETF.org (1992). *The MD5 Message-Digest algorithm.* Recuperat el 26 de març de 2008 de http://www.ietf.org/rfc/rfc1321.txt.

[20] Sun Microsystems (2008). *The Source for Java Developers*. Recuperat el 25 d'abril de 2008 de http://java.sun.com/.

[21] Object Management Group (2008). *Unified Modeling Language*. Recuperat el 26 d'abril de 2008 de http://www.uml.org/.

[22] Wikipedia.org (2008). *SQL*. Recuperat el 5 de maig de 2008 de http://en.wikipedia.org/wiki/SQL.

[23] Java Swing (2008). *History Of Swing*. Recuperat el 23 de maig de 2008 de http://www.javaswing.net/history-of-swing.html.

[24] Eclipse.org (2008). *Visual Editor Project*. Recuperat el 23 de maig de 2008 de http://www.eclipse.org/vep/WebContent/main.php.

[25] Apache Software Foundation (2007). *Apache Logging Services: Apache Log4j*. Recuperat el 29 de maig de 2008 de http://logging.apache.org/log4j/index.html

[26] Ministerio de Interior (2008). *DNI Electrónico*. Recuperat el 3 de juny de 2008 de http://www.dnielectronico.es/

[27] Smart Card Alliance (2008). *Smart Card Alliance*. Recuperat el 3 de juny de 2008 de http://www.smartcardalliance.org/

[28] Hibernate (2008). *Relational persistence for Java and .NET*. Recuperat el 4 de juny de 2008 de http://www.hibernate.org

[29] Java en castellano (2005). *RMI mano a mano con SSL*. Recuperat el 7 de juny de 2008 de http://www.programacion.net/java/articulo/joa_rmissl

Apèndixs

S'adjunten a continuació les notes, comentaris i exemples que complementen la memòria.

- **Apèndix A**: Glossari.
- **Apèndix B**: Jocs de proves.
- **Apèndix C**: Scripts MySQL.
- **Apèndix D**: Fitxers de configuració.
- **Apèndix E**: Traces del sistema.
- **Apèndix F**: Contingut de la distribució del Projecte.

Apèndix A: Glossari.

Aplicatiu: Conjunt de programes informàtics utilitzats per a desenvolupar tasques específiques en un ordinador. En el cas d'aquest Projecte, el programari client del Pacient/Metge i el programari del Gestor.

Autoritat de certificació: Entitat de confiança, responsable d'emetre i revocar els certificats digitals utilitzats en sistemes de signatura electrònica i criptografia de clau pública.

Base de Dades: conjunt de dades estructurat i persistent que pertany a un mateix context, normalment associades a un programari que les consulta i mante. En el nostre cas, conjunt de dades utilitzat per emmagatzemar la informació referent als pacients i els seus historials.

Base 64: Sistema de numeració posicional que té com a base 64, que és la potencia de 2 més gran, 2^6, que es pot fer servir utilitzant els caràcters imprimibles del codi ASCII, utilitzat bàsicament per la codificació de contingut binari en comunicacions basades en text. En el nostre cas s'utilitza per a la transmissió de certificats, claus i contingut signat o xifrat.

Certificat digital: Document digital mitjançant el qual un tercer de confiança, ***autoritat de certificació***, garanteix la vinculació entre una identitat física o jurídica i la seva ***clau pública***.

Clau (criptogràfica): Seqüència de caràcters mitjançant la qual s'especifica la transformació de la informació en clar a informació xifrada, es diu que una clau controla la operació d'un algoritme criptogràfic.

Clau privada: ***Clau criptogràfica*** utilitzada en sistemes de criptografia asimètrica. La clau privada d'un usuari és aquella que només és coneguda per aquest, s'utilitza per a xifrar i signar informació de forma que només sigui possible desxifrar-la amb la ***clau pública***, garantint d'aquesta manera la identitat del emissor.

Clau pública: ***Clau criptogràfica*** utilitzada en sistemes de criptografia asimètrica. La clau pública d'un usuari és aquella que es coneguda per la resta d'usuaris del sistema, i és utilitzada per xifrar informació que només va dirigida al propietari de la clau.

Client-Servidor: Relació establerta entre dues entitats de programari i/o maquinari, de forma que el *servidor* ofereix un recurs físic o un servei, i el *client* l'utilitza i en treu avantatge.

DTD: Sigles de ***D**ocument **T**ype **D**efinition*, proporciona la definició de l'estructura d'un document ***XML***. Permet validar un document.

IAIK: ***Proveïdor de serveis criptogràfics*n** utilitzat en aquest Projecte.

Interfície: Dispositiu de maquinari o element de programari, que permet la connexió entre diferents elements d'un sistema informàtic, publicant les diferents funcions o característiques d'un element o dispositiu.

Java: Llenguatge de programació orientat a objectes de propòsit general, multi-plataforma, interpretat i distribuït. Desenvolupat per *Sun Microsystems*.

JDOM: Sigles de ***J**ava **D**ocument **O**bject **M**odel*. Solució complerta basada en ***Java*** per accedir i manipular el contingut de documents ***XML***.

Maquina Virtual Java (JVM): Model específic de màquina virtual que accepta un tipus de llenguatge intermig específic de ***Java***, anomenat *bytecode*, que representa el conjunt d'instruccions d'un llenguatge basat en una pila i implementa una capa orientada a la seguretat.

MySQL: Sistema Gestor de Base de Dades de lliure distribució utilitzat en aquest projecte.

Número (pseudo)aleatori: Número generat sense que el usuari tingui coneixement del procés. Un número pseudo-aleatori és aquell generat per un procés informàtic, de forma que no tingui cap patró evident.

PKI: Sigles de ***P**ublic **K**ey **I**nfraestructure*, combinació d'elements de programari, maquinari i procediments de seguretat, que permetrà oferir amb garanties operacions criptogràfiques. També s'utilitza per a referir-se a l'autoritat de certificació i a la resta de components d'un sistema de ***clau pública***.

Plug-In: Aplicació informàtica que interactua amb una altra aplicació per tal d'aportar-li una funció o utilitat específica, a mode d'ampliació de funcionalitat.

Proveïdor de serveis criptogràfics: Element de programari, o maquinari en entorns d'alta disponibilitat, encarregat de proporcionar els elements i serveis necessaris per a executar operacions criptogràfiques: xifrat, desxifrat, signatura...

RMI: Sigles de ***R****emote* ***M****ethod* ***I****nvocation*, extensió del Java que permet l'execució de codi remot, allotjat en una altra màquina virtual, la qual alhora pot estar en una altra màquina física. En aquest Projecte s'utilitza per a les comunicacions entre el client i el servidor.

Signatura: Mecanisme de xifrat utilitzat per autenticar un missatge. S'utilitza la ***clau privada*** per generar la signatura d'un missatge. La signatura s'adjunta al missatge dins un document específic que permet comprovar posteriorment la validesa del missatge.

Sobre digital: Tècnica de xifrat híbrida, combina el ***xifrat simètric*** i el ***xifrat asimètric***, que aconsegueix treure partit dels avantatges dels del dos sistemes. El missatge es xifra amb un sistema de ***xifrat simètric***, i la ***clau*** es xifra utilitzant ***xifrat asimètric***, el resultat és un document que s'envia al destinatari. L'avantatge principal d'aquest sistema és la reducció de temps de procés en front d'un ***xifrat asimètric***.

UML: Sigles de ***U****nified* ***M****odeling* ***L****anguage*, és el llenguatge de modelat de sistemes de programari més utilitzat, utilitzat per visualitzar, especificar, construir i documentar un sistema de programari.

Xifrat asimètric: Mètode de xifrat que utilitza dues claus diferents, una clau per xifrar i una clau per desxifrar, normalment s'utilitza la ***clau pública*** per xifrar, i la ***clau privada*** per desxifrar.

Xifrat simètric: Mètode de xifrat que utilitza una única clau per xifrar i desxifrar la informació.

XML: Sigles de *E****x****tensible* ***M****arkup* ***L****anguage*, metallenguatge d'etiquetes extensible, desenvolupat pel *World Wide Web Consortium*. Consisteix en una adaptació del *SGML*, que permet definir una gramàtica específica per a cada cas d'ús. En el cas d'aquest Projecte, utilitzat per a representar les estructures de dades emmagatzemades a la base de dades, i facilitar-ne l'intercanvi entre els ***aplicatius***.

Apèndix B: Jocs de proves.

Proves de l'esquema criptogràfic integrat amb els documents XML.

Per a les proves de l'esquema criptogràfic s'ha generat una classe per cadascun dels protocols que s'han dissenyat, excepte pel protocol d'autenticació que està inclòs en tota la resta.

A continuació s'adjunta el codi corresponent al joc de proves del Protocol-1.

Les dades corresponents als certificats i fitxers .P12 es carreguen directament del sistema d'arxius degut a que, en aquest punt de la implementació, encara no s'ha implementat la base de dades i les aplicacions de client i gestor.

Les dades que s'intercanvien els diferents components són totes del tipus ***byte[]***, això és així perquè corresponen als documents XML xifrats. Excepte en el darrer cas que correspon al document que obté el client per mostrar les dades.

Procés d'autenticació.

```
private void run(){
	Client cl = null;
	Gestor ges = null;
	NeedhamSchroeder nh = null;
	Certificate cer = null;
	byte[] buffer = null;
	String ni = "";
	XMLHistorial historial = null;

	try{
		// Inici del procés d'autenticació
		// Es carrega el certificate el GESTOR amb la seva clau pública.
		cer = new Certificate();
		cer.load(new FileInputStream(
			"C:\\Dades\\UOC\\PFC\\PKI\\Certificats\\Gestor_DER.crt"));

		// Es crea un objecte gestor del procès d'autenticació. Com que
		// carrega el fitxer .P12, serveix per identificar l'usuari.
		nh = new NeedhamSchroeder("Pacient", "uoc0506");

		// S'executa el procès client de l'autenticació.
		buffer = nh.client("Pacient", cer);
		ni = nh.getNi();

		System.out.println("Inici del procés d'autenticació del PACIENT
				pel Protocol-1.");
```

```
            // El gestor respón al procés d'autenticació.
            // L'objecte creat carrega el fitxer .P12 del Gestor.
            nh = new NeedhamSchroeder("Gestor", "uoc0506");
            buffer = nh.server(buffer);

            System.out.println("El gestor accepta la sol·licitud
                              d'autentiació i retorna la resposta al PACIENT.");
```

Aquí finalitza el procés d'autenticació comú a tots els protocols. En cas de produir-se un error en el procés, es generaria una excepció.

A continuació es mostra la traça del procés d'autenticació:

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

Inici del procés d'autenticació del PACIENT pel Protocol-1.
El gestor accepta la sol·licitud d'autentiació i retorna la resposta al PACIENT.
```

Sol·licitud de servei.

El següent pas consisteix en la sol·licitud de l'operació, en aquest punt es comproven les dades del procés d'autenticació, si no són correctes no s'executa la petició. Com a exemple es mostra la sol·licitud d'historial.

```
//Solicitud de servei: obtenció d'historial.

// Es crea una instancia del client. Aquest objecte conté els mètodes
// que necessitarà un metge o pacient.
cl = new Client("Pacient", "uoc0506",
                "C:\\Dades\\UOC\\PFC\\PKI\\Certificats\\Gestor_DER.crt");

// Sol·licitud de servei: solicitar les dades generals de "Pacient".
buffer = cl.solicitarDadesGenerals("Pacient", buffer, ni);

System.out.println("El client ha generat la petició de servei: obtenció de
                   l'historial.");
```

Tots els mètodes de sol·licitud de servei tenen la mateixa estructura, desxifren les dades rebudes i generen la petició.

```
// Desxifra les dades rebudes i carrega un contenidor de dades d'autenticació.
auth = loadAuth(pData, P12Container);

// Es comprova l'autenticitat de les dades d'autenticació rebudes del gestor.
if(auth.getNi().equals(pNi)){
      // Obtenir la clau pública del Gestor.
      managerCert = new Certificate();
      managerCert.load(new FileInputStream(mCertPath));
```

```
        // Creació petició a Gestor.
        // La petició va encapsulada dins un contenidor XML.
        peticio = new XMLPeticio();
        peticio.setNg(auth.getNg());

        peticio.setOperacio(Constants.OP_CONSULTA_DADES_GENERALS);
                            peticio.addParametre(pIdUsuari);

        // Xifrat de la petició.
        engine = new RSA();

        retValue = engine.encrypt(peticio.toString().getBytes(),
                                  managerCert.getCertificate());
}else{
        throw new SecurityException("error de seguretat -  el missatge obtingut
                                     no és correcte");
}
```

Les traces de la operació són les següents:

```
El client ha generat la petició de servei: obtenció de l'historial.
```

En cas d'error, concretament el número aleatori generat no s'ha comprovat correctament.

```
Error durant l'execució del Protocol-1
error de seguretat -  el missatge obtingut no és correcte
```

Obtenció del servei.

En el següent pas el gestor obté la informació sol·licitada, prèvia validació de les dades d'autenticació.

```
// Obtenció de l'historial per part del gestor, s'envien les dades
// generades per la petició de servei del client.
ges = new Gestor("Gestor", "uoc0506");
buffer = ges.recuperaHistorial(buffer);
```

El gestor rep les dades i comprova l'autenticació del client amb les dades desades en la base de dades durant el procés d'autenticació.

```
...
// Desxifrar les dades de la petició rebudes.
engine = new RSA();
peticio = new XMLPeticio();

peticio.load(new ByteArrayInputStream(engine.decrypt(pData,
                                      P12Container.getPrivateKey())));

// Recuperar dades d'autenticació de la base de dades.
authProc = new NeedhamSchroeder();
userIdBBDD = authProc.getAuthParms(peticio.getNg());
```

```
// Verificació d'autenticació del usuari.
// Si no es recupera l'usuari, es pot considerar que el procés
// d'autenticació no és correcte.
if (userIdBBDD != null){
   . . .
}else{
      throw new SecurityException("error de seguretat -  el missatge
                                    obtingut no és correcte");
}
. . .
```

Els resultats possibles d'aquesta operació poden ser: les dades sol·licitades, o un error, ja sigui de l'execució de la darrera part del protocol, o bé del procés l'autenticació.

En cas de que l'operació s'executi de forma correcta.

```
El client ha generat la petició de servei: obtenció de l'historial.
El gestor ha recuperat l'historial del pacient.
```

En cas de que es produeixi un error durant la validació de dades.

```
El client ha generat la petició de servei: obtenció de l'historial.
Error durant l'execució del Protocol-1
error de seguretat - usuari sense permisos per aquesta consulta
```

El darrer pas consisteix en tractar les dades rebudes del gestor per a que el client les mostri: desxifrat i comprovació d'integritat de dades.

```
// El client obté l'historial i el desxifra.
XMLHistorial historial = null;
historial = cl.obtenirHistorial(buffer);

System.out.println("Historial obtingut i desxifrat correctament:\n\n");
System.out.println(historial.toString());
```

El resultat obtingut:

```
Historial obtingut i desxifrat correctament:

<historial>
      <nom>Pepe</nom>
      <cognoms>Rubianes</cognoms>ç
      <numTargetaSanitaria>1234567890</numTargetaSanitaria>
      <dni>77690737A</dni>
      <grupSanguini>O-</grupSanguini>
      <alergies>
            <alergia>al polen</alergia>
            <alergia>a la penicilina</alergia>
      </alergies>
```

```
      <certificat>Y7NCVnMz6knakOsMf9lP9Yv8zOTei2Y9Riyo36oHO5tdDRg0W3sF5
8Gta5cD4CZD...
</historial>
```

Els programes de test de cada protocol estan inclosos en el codi font del projecte.

Proves de comunicacions RMI.

Per a les proves de l'esquema criptogràfic s'han aprofitat les classes generades per a les proves d'integració entre l'esquema criptogràfic i XML[1], substituint la creació de l'objecte *Gestor*.

Les dades que s'intercanvien els diferents components són totes del tipus ***byte[]***, per tant, no hi ha cap problema alhora de transferir els objectes mitjançant RMI[2], ja que es tracta d'un tipus bàsic de Java[20].

Per aquest exemple s'utilitzarà el Protocol-2.

Procés d'autenticació.

El procés d'autenticació és el mateix que s'ha descrit en el cas de prova, la diferencia rau en que, per crear la instancia corresponent al Gestor, s'utilitza RMI[2].

```
private void run(){
       Client cl = null;
       IGestor ges = null;
       Certificate cer = null;
       byte[] auth = null;
       XMLVisita visita = null;

       try{
              cl = new Client("Pacient", "uoc0506",
                     "C:\\Dades\\UOC\\PFC\\PKI\\Certificats\\Gestor_DER.crt");

              cer = new Certificate();
              cer.load(new FileInputStream(
                     "C:\\Dades\\UOC\\PFC\\PKI\\Certificats\\Gestor_DER.crt"));

              // Pas 1 - Inici del procés d'autenticació
              auth = cl.authClient("Pacient", cer);

              System.out.println(
               "Inici del procés d'autenticació del PACIENT pel Protocol-2.");

              // Pas 2 - El gestor respón al procés d'autenticació.
              ges = (IGestor)Naming.lookup("rmi://localhost:1099/PFCGestor");
```

```
            auth = ges.authGestor(auth);

            System.out.println("El gestor accepta la sol·licitud
d'autenticació i retorna la resposta al PACIENT.");
```

Aquí finalitza el procés d'autenticació comú a tots els protocols. En cas de produir-se un error en el procés, es generaria una excepció.

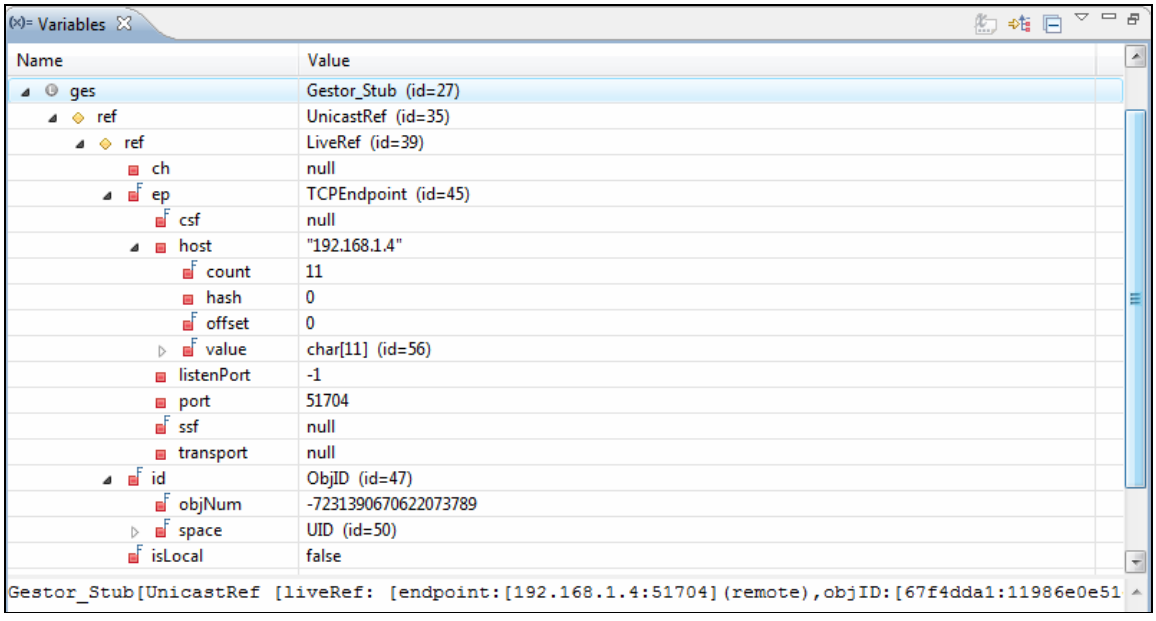
A continuació es mostra la traça del procés d'autenticació:

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

Inici del procés d'autenticació del PACIENT pel Protocol-2.
El gestor accepta la sol·licitud d'autentiació i retorna la resposta al PACIENT.
```

Un cop instanciat l'objecte remot la resta de crides és exactament igual que si es tractés d'un objecte local.

En la següent captura de pantalla es mostra com la variable instanciada de forma remota pertany al tipus *Gestor_Stub*.



Imatge 40 - Inspecció mitjançant Eclipse de la variable remota.

Apèndix C: Scripts MySQL.

El següent *script* crea l'esquema de dades MySQL[5], l'usuari amb el que es realitzaran les operacions, i les taules i les seves relacions.

```
CREATE DATABASE pfchistorials;

// --------------------------
// Creació de usuaris.
// --------------------------
CREATE USER pfcgestor IDENTIFIED BY 'pfcgestor';

GRANT SELECT, INSERT, UPDATE, DELETE ON 'pfchistorials'.* TO
'pfcGestor'@'%'


// --------------------------
// Creació de taules.
// --------------------------

// Taula de dades d'autenticació.
CREATE TABLE `pfchistorials`.`autenticacio` (
  `Ng` VARCHAR(32) NOT NULL DEFAULT '',
  `xmlAutenticacio` VARCHAR(2000) NOT NULL DEFAULT '',
  PRIMARY KEY (`Ng`)
)
ENGINE = InnoDB;


// Taula d'usuaris.
CREATE TABLE `pfchistorials`.`Usuaris` (
  `idUsuari` VARCHAR(32) NOT NULL DEFAULT '',
  `certificat` MEDIUMTEXT NOT NULL,
  PRIMARY KEY (`idUsuari`)
)
ENGINE = InnoDB;

// Taula de visites.
CREATE TABLE `pfchistorials`.`visites` (
  `idVisita` VARCHAR(32) NOT NULL DEFAULT '',
  `xmlVisita` MEDIUMTEXT NOT NULL,
  PRIMARY KEY (`idVisita`)
)
ENGINE = InnoDB;

// Taula de metges.
CREATE TABLE `pfchistorials`.`metges` (
  `idMetge` VARCHAR(32) NOT NULL DEFAULT '',
  `xmlMetge` MEDIUMTEXT NOT NULL,
  PRIMARY KEY (`idMetge`)
)
ENGINE = InnoDB;
```

```
// Taula d'historials.
CREATE TABLE `pfchistorials`.`historials` (
  `idPacient` VARCHAR(32) NOT NULL DEFAULT '',
  `xmlHistorial` MEDIUMTEXT NOT NULL,
  PRIMARY KEY (`idPacient`)
)
ENGINE = InnoDB;

// --------------------------
// Creació de constraints.
// --------------------------

// Un metge ha de ser un usuari del sistema.
ALTER TABLE `pfchistorials`.`metges` ADD CONSTRAINT `FK_METGE_USUARI`
FOREIGN KEY `FK_METGE_USUARI` (`idMetge`)
    REFERENCES `usuaris` (`idUsuari`)
    ON DELETE RESTRICT
    ON UPDATE RESTRICT;

// Un historial ha de pertànyer a un usuari.
ALTER     TABLE     `pfchistorials`.`historials`     ADD     CONSTRAINT
`FK_HISTORIAL_USUARI` FOREIGN KEY `FK_HISTORIAL_USUARI` (`idPacient`)
    REFERENCES `usuaris` (`idUsuari`)
    ON DELETE RESTRICT
    ON UPDATE RESTRICT;
```

Aquest *script* està inclòs amb el codi de provés que s'adjunta amb aquesta memòria.

Apèndix D: Fitxers de configuració.

En aquest apèndix es mostren dos exemples dels fitxers de configuració de l'aplicació, un per al *Client* i un altre per al *Gestor*.

Fitxer de configuració del Client.

Dins d'aquest fitxer s'especifiquen tots els paràmetres que necessita l'aplicació *Client* per a funcionar correctament.

Aquest fitxer es pot trobar dins de la carpeta *conf* de la distribució de l'aplicació corresponent al Client.

El contingut del fitxer ***client.properties*** és el següent:

```
// Rutes de localització de recursos.
certpath = C:\\Dades\\UOC\\PFC\\PKI\\Certificats\\temp\\Gestor_DER.crt
P12FilesPath = c:\\Dades\\UOC\\PFC\\PKI\\Certificats\\temp\\

// Localització del servei remot.
host = localhost
port = 1099
servei = PFCGestor

// Nivell de traça. Level.DEBUG < Level.INFO < Level.WARN < Level.ERROR <
Level.FATAL  Level.ALL o Level.OFF.
nivellTrace = INFO
```

A continuació s'explica el significat de cada paràmetre:

- **certpath**: indica la ruta on l'aplicació anirà a buscar el certificats del gestor amb la seva clau pública.
- **P12FilesPath**: indica la ruta on l'aplicació anirà a buscar els fitxers P12 del client.
- **host**: nom o adreça IP del servidor on resideix el servei remot corresponent al Gestor.
- **port**: port d'accés a aquest servei remot, per defecte **1099**.
- **servei**: nom del servei remot.
- **nivellTrace**: indica el nivell de traça per defecte amb el que treballarà l'aplicació. Com més elevat sigui aquest nivell, pitjor

rendiment tindrà l'aplicació. Es recomana el nivell **WARN** per poder registrar els errors que es puguin produir, sense afectar el rendiment.

Fitxer de configuració del Gestor.

Dins d'aquest fitxer s'especifiquen tots els paràmetres que necessita l'aplicació *Gestor* per a funcionar correctament.

Aquest fitxer es pot trobar dins de la carpeta *conf* de la distribució de l'aplicació corresponent al Gestor.

El contingut del fitxer ***gestor.properties*** és el següent:

```
// Rutes de localització de recursos.
P12FilesPath = c:\\Dades\\UOC\\PFC\\PKI\\Certificats\\temp\\

// Dades del servei.
port = 1099
servei = PFCGestor

// Nivell de traça. Level.DEBUG < Level.INFO < Level.WARN < Level.ERROR <
Level.FATAL  Level.ALL o Level.OFF.
nivellTrace = INFO

// Connexio a base de dades.
hostBBDD = localhost
portBBDD = 3306
baseDades = pfchistorials
usuariBaseDades = pfcGestor
clauBaseDades = pfcGestor
```

A continuació s'explica el significat de cada paràmetre:

- **certpath**: indica la ruta on l'aplicació anirà a buscar el certificats del gestor amb la seva clau pública.
- **P12FilesPath**: indica la ruta on l'aplicació anirà a buscar el fitxer P12 del gestor.
- **port**: port per on s'oferirà el servei remot, per defecte **1099**.
- **servei**: nom del servei remot.
- **nivellTrace**: indica el nivell de traça per defecte amb el que treballarà l'aplicació. Com més elevat sigui aquest nivell, pitjor rendiment tindrà l'aplicació. Es recomana el nivell **WARN** per poder registrar els errors que es puguin produir, sense afectar el rendiment.

- **hostBBDD**: nom o adreça IP de la màquina que allotgi el servei de base de dades.
- **portBBDD**: port de connexió al servei de base de dades.
- **baseDades**: nom de la base de dades on es desaran les dades de l'aplicació.
- **usuariBaseDades**: usuari que s'utilitzarà per la connexió a la base de dades.
- **clauBaseDades**: paraula clau de l'usuari de la connexió a la base de dades.

Apèndix E: Traces del sistema.

Tant l'aplicació del Client com l'aplicació del Gestor, estan dotades d'un sistema de traces pensat per:

- **Auditar les operacions**: es permet deixar un rastre de les operacions que es van realitzant, de manera que es puguin obtenir dades de l'ús de l'aplicació.

- **Facilitar el seguiment d'errors**: com en totes les aplicacions, en algun moment es produirà algun error, i en el cas d'aquest Projecte amb serveis remots i connexions a bases de dades, les possibilitats són més elevades. Les traces permetran enregistrar els errors per poder disposar de dades que permetin solucionar-los.

Per la implementació d'aquestes traces s'ha utilitzat la llibreria Log4j[25], en la seva versió 1.2.8. Aquesta llibreria permet generar traces amb diferents formats i nivells de granularitat, amb un rendiment molt elevat. La seva configuració mitjançant un fitxer extern permet modificar el nivell de traca i el format en moment d'execució.

Fitxer de configuració de traces de Client.

El fitxer de configuració de traces del client està ubicat a la carpeta *conf* de la distribució de l'aplicació.

El contingut d'aquest fitxer de configuració de traces, ***log4j.properties***, és el següent:

```
log4j.rootCategory=ALL, Default
log4j.appender.Default=org.apache.log4j.RollingFileAppender
log4j.appender.Default.file=.\\logs\\client.log
log4j.appender.Default.MaxFileSize=2MB
log4j.appender.Default.layout=org.apache.log4j.PatternLayout
log4j.appender.Default.layout.ConversionPattern = [%d{DATE}] - [%p] -
%C{1}.%M - %m\n
log4j.appender.Default.append=true
log4j.appender.Default.ImmediateFlush=true
```

Fitxer de configuració de traces de Gestor.

El fitxer de configuració de traces del Gestor està ubicat a la carpeta *conf* de la distribució de l'aplicació corresponent al Gestor.

El contingut d'aquest fitxer de configuració de traces, ***log4j.properties***, és el següent:

```
log4j.rootCategory=ALL, Default
log4j.appender.Default=org.apache.log4j.RollingFileAppender
log4j.appender.Default.file=.\\logs\\gestor.log
log4j.appender.Default.MaxFileSize=2MB
log4j.appender.Default.layout=org.apache.log4j.PatternLayout
log4j.appender.Default.layout.ConversionPattern = [%d{DATE}] - [%p] -
%C{1}.%M - %m\n
log4j.appender.Default.append=true
log4j.appender.Default.ImmediateFlush=true
```

No s'entrarà en detall a la configuració dels fitxers de configuració de traces presentats, ens remetrem a la bibliografia de Log4j[25] per a tal efecte.

Només aclarir que la configuració per defecte aportada, crea unes traces a un fitxer, *.\logs\gestor.log*, que com a màxim tindrà una mida de 2 megabytes i serà circular (quan s'assoleixi la mida, es sobreescriurà des de l'inici).

Exemples de traces.

El format aplicat a les traces serà el següent:

```
[dd mmm yyyy hh:MM:ss,SSS] – [NIVELL] – Modul.metode – missatge
```

En cas de produir-se un error, es bolcarà a la traça la pila de crides que ha generat l'error.

A continuació es presenten les traces que es generen per al *Client* i per al *Gestor* per a la següent seqüència d'accions:

- Identificació d'usuari.
- Consulta d'historial.
- Consulta de visita de l'historial.
- Consulta d'historial inexistent.

El nivell de traça indicat per aquest exemple és *INFO*, cal recordar que el nivell òptim per a una execució normal és *WARN*.

La següent traça correspon al *Client*, donat el volum que es genera es presenta resumida:

```
[01 jun 2008 13:03:51,557] - [INFO] - FrmMainHandler.initialize - Configuració
carregada.
[01 jun 2008 13:04:05,781] - [INFO] - FrmIdentificacioHandler.validaFitxerClau - Inici
de validacio usuari.
[01 jun 2008 13:04:06,402] - [INFO] - FrmIdentificacioHandler.validaFitxerClau - Usuari
validat.
:
:
[01 jun 2008 13:04:06,549] - [INFO] - FrmMainHandler.fillLlistaHistorials - Inici de
recuperacio de la llisa de historials consultables.
[01 jun 2008 13:04:06,549] - [INFO] - FrmMainHandler.fillLlistaHistorials - El client es
un PACIENT.
[01 jun 2008 13:04:06,551] - [INFO] - FrmMainHandler.fillLlistaHistorials - Llista
d'historials carregada
:
:
[01 jun 2008 13:04:09,800] - [INFO] - FrmMainHandler.obtenirHistorial - Inici de
consulta d'historial.
[01 jun 2008 13:04:09,801] - [INFO] - FrmMainHandler.obtenirHistorial - Solicitud
d'execucio del protocol de recuperacio d'hitorial.
[01 jun 2008 13:04:50,210] - [INFO] - FrmMainHandler.executeProtocol1 - Inici del proces
d'autenticacio.
[01 jun 2008 13:04:51,929] - [INFO] - FrmMainHandler.executeProtocol1 - Gestor accepta i
respon al proces d'autenticacio.
[01 jun 2008 13:04:52,836] - [INFO] - FrmMainHandler.executeProtocol1 - El client
solicita la informacio d'un historial.
[01 jun 2008 13:04:53,527] - [INFO] - FrmMainHandler.executeProtocol1 - El gestor ha
recuperar la informació del historial.
[01 jun 2008 13:04:54,756] - [INFO] - FrmMainHandler.executeProtocol1 - Historial
recuperat i desxifrat.
[01 jun 2008 13:04:56,583] - [INFO] - FrmMainHandler.obtenirHistorial - Fi d'execucio
del protocol de recuperacio d'historial.
[01 jun 2008 13:04:56,584] - [INFO] - FrmMainHandler.obtenirHistorial - Omplint
formulari de dades d'historial.
[01 jun 2008 13:04:56,587] - [INFO] - FrmMainHandler.obtenirHistorial - Fi de consulta
d'historial.
:
:
[01 jun 2008 13:05:59,431] - [INFO] - FrmVisitaHandler.initialize - Peticio de protocol
de recuperacio de dades de visita.
[01 jun 2008 13:05:59,436] - [INFO] - FrmVisitaHandler.executeProtocol2 - Client inicia
la autenticacio.
[01 jun 2008 13:05:59,736] - [INFO] - FrmVisitaHandler.executeProtocol2 - Gestor respon
a la autenticacio.
[01 jun 2008 13:06:06,854] - [INFO] - FrmVisitaHandler.executeProtocol2 - Client
solicita servei.
[01 jun 2008 13:06:07,690] - [INFO] - FrmVisitaHandler.executeProtocol2 - Gestor retorna
dades solicitades.
[01 jun 2008 13:06:07,733] - [INFO] - FrmVisitaHandler.executeProtocol2 - Client
desxifra i valida les dades rebudes.
[01 jun 2008 13:06:07,734] - [INFO] - FrmVisitaHandler.initialize - Protocol executat.
[01 jun 2008 13:06:07,734] - [INFO] - FrmVisitaHandler.initialize - Volcat dades per
pantalla.
[01 jun 2008 13:06:07,736] - [INFO] - FrmVisitaHandler.initialize - Fi recuperacio dades
visita
:
:
[01 jun 2008 13:06:57,417] - [INFO] - FrmMainHandler.executeProtocol1 - El client
solicita la informacio d'un historial.
[01 jun 2008 13:06:59,573] - [ERROR] - FrmMainHandler.obtenirHistorial - No s'ha pogut
recuperar l'historial seleccionat.

java.lang.SecurityException: error de seguretat - usuari sense permisos per aquesta
consulta
       at uoc.pfc.dmm.gestor.Gestor.recuperaHistorial(Gestor.java:162)
       at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
       at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
       at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
:
:
```

La següent traça correspon al *Gestor*:

```
[01 jun 2008 13:03:05,628] - [INFO] - Gestor.<init> - Gestor inicialitzat correctament.
[01 jun 2008 13:04:52,839] - [INFO] - Gestor.recuperaHistorial - Inici de la recuperació
d'historial
[01 jun 2008 13:04:52,866] - [INFO] - Gestor.recuperaHistorial - Carregats certificat i
claus del gestor.
[01 jun 2008 13:04:52,881] - [INFO] - Gestor.recuperaHistorial - Recuperades dades de
petició de l'usuari.
[01 jun 2008 13:04:52,993] - [INFO] - Gestor.recuperaHistorial - Usuari peticionari:
0987654321
[01 jun 2008 13:04:52,994] - [INFO] - Gestor.recuperaHistorial - Usuari peticionari
valid.
[01 jun 2008 13:04:52,994] - [INFO] - Gestor.recuperaHistorial - Usuari autoritzat a
accedir a les dades solicitades.
[01 jun 2008 13:04:53,391] - [INFO] - Gestor.recuperaHistorial - Dades del historial
recuperades.
[01 jun 2008 13:04:53,526] - [INFO] - Gestor.recuperaHistorial - Fi de la recuperacio de
l'historial.
[01 jun 2008 13:06:06,856] - [INFO] - Gestor.recuperaVisita - Inici recuperació de
visita.
[01 jun 2008 13:06:06,886] - [INFO] - Gestor.recuperaVisita - Carregats certificat i
claus del gestor.
[01 jun 2008 13:06:06,904] - [INFO] - Gestor.recuperaVisita - Recuperades dades de
petició de l'usuari.
[01 jun 2008 13:06:07,030] - [INFO] - Gestor.recuperaVisita - Usuari peticionari:
0987654321
[01 jun 2008 13:06:07,031] - [INFO] - Gestor.recuperaVisita - Usuari peticionari valid.
[01 jun 2008 13:06:07,210] - [INFO] - Gestor.recuperaVisita - Usuari autoritzat a
accedir a les dades solicitades.
[01 jun 2008 13:06:07,543] - [INFO] - Gestor.recuperaVisita - Dades de la visita
recuperades.
[01 jun 2008 13:06:07,689] - [INFO] - Gestor.recuperaVisita - Fi de la recuperacio de la
visita.
[01 jun 2008 13:06:57,418] - [INFO] - Gestor.recuperaHistorial - Inici de la recuperació
d'historial
[01 jun 2008 13:06:57,438] - [INFO] - Gestor.recuperaHistorial - Carregats certificat i
claus del gestor.
[01 jun 2008 13:06:57,452] - [INFO] - Gestor.recuperaHistorial - Recuperades dades de
petició de l'usuari.
[01 jun 2008 13:06:57,572] - [INFO] - Gestor.recuperaHistorial - Usuari peticionari:
0987654321
[01 jun 2008 13:06:57,573] - [INFO] - Gestor.recuperaHistorial - Usuari peticionari
valid.
[01 jun 2008 13:06:57,679] - [WARN] - Gestor.recuperaHistorial - error de seguretat -
usuari sense permisos per aquesta consulta: 0987654321
```

Apèndix F: Contingut de la distribució del Projecte.

A continuació es detalla l'estructura dels fitxers que s'adjunten amb aquesta memòria.

EL lliurament del conté la següent estructura de carpetes:

- **\bin**: conté les classes compilades, les llibreries necessàries per l'execució i els fitxers *.bat* necessaris per a executar les aplicacions del Projecte:
  - **PFCGestor.bat** per iniciar el *Gestor*.
  - **PFCClient.bat** per iniciar el *Client*.

- **\doc**: conté la memòria i la presentació del Projecte, així com el JavaDoc generat.

- **\pki**: conté l'estructura de PKI generada per a l'aplicació d'aquest Projecte.

- **\src**: conté totes les fonts dels Projecte.

- **\project**: conté una exportació del IDE Eclipse on s'ha desenvolupat el Projecte.