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

J.Cèsar Rodríguez León de Santos
Enginyeria en Informàtica

Jordi Castellà Roca
Director del PFC

8 de gener de 2007

A la meva dona,
per la seva paciència
durant tots aquests anys.

T'estimo

RESUM DEL CONTINGUT

CONTINGUT

ÍNDEX D'IL·LUSTRACIONS

PRÒLEG

Aquest projecte, emmarcat en l'Àrea de Seguretat informàtica, pretén donar solució a un problema: la gestió segura dels historials mèdics dels pacients a través d'una xarxa de comunicacions. Part d'aquesta informació pot considerar-se confidencial, per la qual cosa no hauria de poder ser consultada per qualsevol metge, i menys encara per algú aliè.

Per a portar-ho a terme, s'hauran d'implementar un conjunt de protocols criptogràfics els quals garantiran els diferents requisits de seguretat necessaris: autenticitat, confidencialitat, integritat i no-repudi. Cada operació que el sistema ofereixi haurà de seguir un protocol de seguretat per tal de poder donar-se com a vàlida. La primera tasca a fer, doncs, serà precisament l'anàlisi i disseny d'aquests protocols.

Un cop assolit aquest punt, serà necessari definir l'estructura de les dades que s'intercanviaran les diferents aplicacions d'aquest sistema durant la realització dels diferents protocols. Aquesta estructura seguirà l'estàndard XML. Caldrà, doncs, determinar quina informació es transferirà durant l'execució dels protocols criptogràfics per construir l'estructura del fitxer XML de traspàs. Serà convenient assegurar que aquests documents estiguin ben formats -s'anomena documents "ben formats" (de l'anglès well formed) als documents que compleixen amb totes les definicions bàsiques de format i poden, per tant, ser analitzats correctament per qualsevol "parser" que compleixi amb la norma- i siguin vàlids, per la qual cosa es crearà també un fitxer DTD o definició de tipus de document a l'efecte.

La comunicació entre els diferents components també serà un tema que caldrà resoldre. Com que es vol que aquest sistema funcioni de manera remota, cal cercar una tecnologia que ens ho permeti, i donat que el llenguatge de programació que s'utilitzarà serà Java s'ha optat per la tecnologia RMI o Remote Method Invocation, una tecnologia purament Java la qual permet la invocació remota de mètodes.

La informació dels historials mèdics, dels pacients i dels metges, com a mínim, caldrà emmagatzemar-la en una base de dades per mantenir la seva persistència. L'anàlisi i el disseny d'aquesta base de dades serà la fase següent del PFC: es definiran quines dades cal guardar i de quin tipus seran, quines relacions existiran entre aquestes, etc.

Finalment, les interfícies gràfiques donaran accés a tota aquesta funcionalitat de manera còmoda per a l'usuari. La creació d'aquestes serà el darrer pas en la realització d'aquest projecte, donat que aquest no és un dels objectius principals del PFC.

Per tal de provar aquest sistema es crearà un joc de proves per tal de poder simular una execució i es donaran les pautes per a poder provar-lo.

CAPÍTOL 1 – Introducció

APARTAT 1.1 – Resum executiu

Justificació del PFC

Les xarxes de comunicació ens permeten accedir a un gran volum d'informació molt ràpidament sense la necessitat de desplaçar-nos, i amb independència de l'instant de temps. Aquests avantatges aporten un valor afegit més gran quan la persona que accedeix a la informació és un metge que consulta l'historial mèdic d'un pacient.

Les dades dels pacients han de ser protegides per tal que només siguin modificades pel personal qualificat, i han de ser accessibles únicament pel propi pacient o pel metge, garantint així la seva confidencialitat. Per a protegir les dades es faran servir diversos protocols criptogràfics els quals faran ús d'una parella de claus per a cada usuari i el seu corresponent certificat. Per tal de gestionar aquests certificats es fa servir una infraestructura de clau pública. Cada un dels serveis que oferirà el sistema ha de portar implementat algun d'aquests protocols de seguretat.

Aquest projecte implementarà un esquema criptogràfic que garantirà les necessitats de seguretat dels historials dels pacients, cobrint els següents aspectes:

- ➔ Confidencialitat: les dades de l'historial mèdic dels pacients s'han de protegir contra accessos no desitjats.
- ➔ Autenticitat: s'ha de provar que la informació emmagatzemada al sistema és autèntica.
- ➔ Integritat: les dades, un cop generades, no han de poder alterar-se furtivament.
- ➔ No repudi: si es produeix qualsevol canvi en la informació del sistema s'ha de poder identificar l'autor.

El resultat serà un sistema amb els següents components:

- ➔ Aplicació pacient: permetrà als pacients consultar els seus historials de visites.
- ➔ Aplicació metge: permetrà al metge consultar i modificar les dades dels historials dels pacients.
- ➔ Gestor central: mantindrà el magatzem de les dades del sistema i en controlarà la seva gestió. Permetrà donar d'alta els metges i els pacients.

Objectius del PFC

L'objectiu d'aquest projecte és implementar un esquema criptogràfic que garanteixi les necessitats de seguretat de les dades d'un historial mèdic el qual serà gestionat a través d'una xarxa de comunicacions convencional.

Abast del PFC

Es considera dins de l'abast d'aquest projecte la implementació dels serveis oferts i, principalment, el disseny i la implementació de l'esquema criptogràfic.

Serveis oferts

El sistema haurà d'oferir als diferents usuaris uns determinats serveis que els permetran accedir a la informació per consultar-la i/o afegir-ne de nova. Existiran restriccions d'accés a aquests serveis en funció del tipus d'usuari de què es tracta, és a dir, un usuari no podrà realitzar qualsevol acció.
A continuació s'enumeren els serveis que es podran utilitzar, tot i que hi haurà la possibilitat d'ampliar-los més endavant:

- ➔ Autenticar usuaris: un punt clau d'aquest projecte és portar a terme una autenticació segura pel que fa als usuaris del sistema, així com reconèixer quin tipus d'usuari és.
- ➔ Consultar historials: els pacients podran accedir a la informació del seu historial, mentre que els metges podran consultar l'historial de qualsevol pacient que tinguin assignat.
- ➔ Afegir visites: els metges podran així mateix introduir al sistema la informació relativa a les visites realitzades als seus pacients.
- ➔ Consultar llista de pacients: només els metges podran consultar la llista dels pacients assignats.

Per aspectes legals, els següents serveis no es podran afegir:

- ➔ Modificar dades: els metges seran els únics usuaris que podran modificar aquestes dades.
- ➔ Eliminar dades: els metges podran eliminar dades errònies dels historials dels pacients.

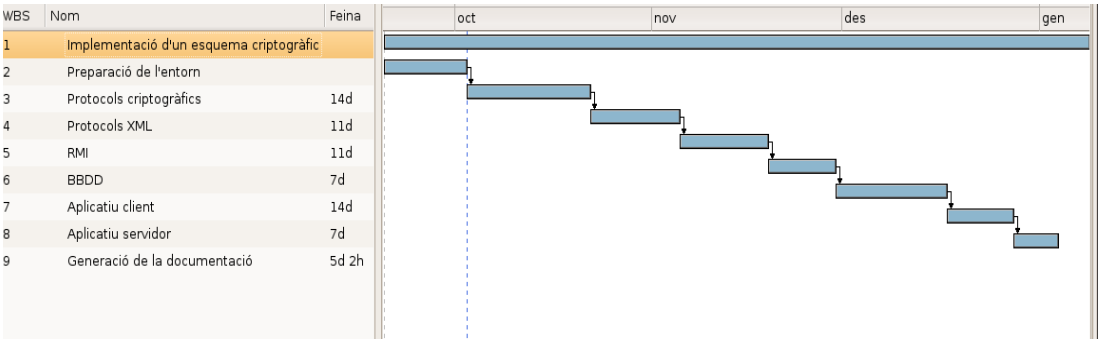
Enfocament i mètode seguit

En la realització d'aquest PFC intervenen diverses metodologies les quals s'han estudiat al llarg de la carrera: protocols criptogràfics de seguretat, intercanvi de dades mitjançant XML, comunicació remota entre els components amb tecnologia RMI, gestió de la informació amb l'SGBD MySQL i el llenguatge de programació Java.

Aquest projecte s'ha dividit en diverses fases, cada una d'elles corresponent a l'aplicació d'una de les tecnologies esmentades al paràgraf anterior. Cada nova fase pren com a punt de partida el resultat obtingut a la fase anterior, per la qual cosa es podria dir que el mètode seguit ha estat el disseny en cascada.

Planificació del projecte

El projecte tindrà una durada de 16 setmanes, durant les quals s'aniran assolint gradualment les diferents fases en què es divideix aquest. Es pot veure la distribució en el temps de les tasques de primer nivell en el diagrama de Gantt que mostra el calendari de treball previst:



Il·lustració 1: Diagrama de Gantt

WBS	Nom	Comença	Acaba	Feina	Durada
1	Implementació d'un esquema criptogràfic	set 20	gen 8		119d
2	Preparació de l'entorn	set 20	oct 2		14d
3	Protocols criptogràfics	oct 2	oct 22	14d	21d
4	Protocols XML	oct 22	nov 5	11d	15d
5	RMI	nov 5	nov 19	11d	15d
6	BBDD	nov 19	nov 29	7d	11d
7	Aplicatiu client	nov 29	des 17	14d	19d
8	Aplicatiu servidor	des 17	des 27	7d	11d
9	Generació de la documentació	des 27	gen 4	5d 2h	8d

Il·lustració 2: Durada de les tasques

En les taules següents es pot veure la relació d'activitats prevista:

Tasca	Protocols criptogràfics
Esforç	56 hores
Durada	20 dies

Tasca	Protocols XML
Esforç	44 hores
Durada	14 dies

Tasca	RMI
Esforç	44 hores
Durada	14 dies

Tasca	Bases de Dades
Esforç	28 hores
Durada	10 dies

Tasca	Aplicació client + metge
Esforç	56 hores
Durada	18 dies

Tasca	Aplicació servidor
Esforç	28 hores
Durada	10 dies

Tasca	Generació de la documentació
Esforç	22 hores
Durada	7 dies

En aquest projecte l'acabament de cada fase es correspondrà a una fita, per la qual cosa en tindrem 7: protocols criptogràfics, protocols XML, RMI, bases de dades, aplicació client, aplicació servidor i generació de la documentació.

En la realització d'aquest projecte només intervindran dues persones, l'autor fent funcions de Cap de projecte, d'Analista i de Programador, i el consultor del projecte, en Jordi Castellà Roca, fent funcions de Director del projecte.

Productes obtinguts

Com a resultat d'aquest PFC s'obtindrà un sistema que permetrà l'usuari efectuar determinades accions en funció del seu perfil, dividit en dos aplicatius diferents:

- ➔ Aplicatiu de l'usuari: permetrà al pacient o al metge executar les funcionalitats disponibles en funció del seu perfil.
- ➔ Aplicatiu del gestor: gestionarà la seguretat del sistema.

Organització de la memòria

El capítol següent mostra els passos que s'han seguit per a la preparació de l'entorn de treball. Al capítol 3 es detallen els protocols criptogràfics i s'identifiquen els diferents actors del sistema i la descripció dels casos d'ús que s'implementaran. També en aquest punt hi haurà el diccionari de termes del document. El capítol 4 correspon a la representació de les dades en format XML. La comunicació dels diferents components serà el tema tractat al capítol 5; aquesta comunicació es farà amb el mecanisme RMI. El capítol 6 parlarà de la gestió de la informació, on es dissenyarà la base de dades -taules, camps i tipus de dades- i l'accés a aquesta. Al capítol 7 es veuran les interfícies gràfiques del sistema.

Per acabar, el capítol 9 contindrà les conclusions, l'opinió personal i es parlarà del futur d'aquest projecte: quines millores es poden realitzar, etc. Els annexos es troben al capítol final, el número 10.

CAPÍTOL 2 – Preparació de l'entorn

APARTAT 2.1 – Necessitats

Per a portar a terme aquest PFC la primera cosa que cal fer és preparar l'entorn de treball i després generar els certificats digitals necessaris per a la implementació dels diferents protocols de seguretat. Tanmateix, les eines necessàries per la generació de la documentació i per la consulta de fonts d'informació externes les proporciona el propi sistema Linux -concretament la distribució UOC de l'Ubuntu- on s'ha desenvolupat aquest.

APARTAT 2.2 – Instal·lació de les eines

El llenguatge de programació emprat ha estat el Java, i més concretament la versió 1.5 del JDK de Sun. Per cobrir els aspectes de la seguretat s'ha utilitzat la llibreria criptogràfica pròpia del llenguatge per implementar els diferents protocols. Pel desenvolupament del PFC no s'ha fet servir cap IDE dels existents i s'ha optat per fer-ho íntegrament amb un editor de text genèric.

Per comunicar les aplicacions s'ha fet servir el mecanisme RMI de Java, ja incorporat en el JDK utilitzat, mentre que les dades que s'intercanvien aquestes durant la realització dels protocols estan estructurades en format XML. Per analitzar els fitxers resultants s'ha utilitzat l'eina XMLStarlet.

CAPÍTOL 3 – Esquema criptogràfic

APARTAT 3.1 – Introducció

Es disposa per començar a treballar de la documentació entregada pel Director del PFC en Jordi Castellà Roca i de les diferents eines i exemples subministrats alhora pel mateix.

APARTAT 3.2 – Protocols de seguretat

Requisits

La implementació dels protocols de seguretat requereixen una sèrie de funcions, les quals s'enumeren en aquesta llista:

- ➔ Signatura digital d'un missatge amb la clau asimètrica privada d'una entitat.
- ➔ Verificació de la signatura d'un missatge amb la clau pública d'una entitat.
- ➔ Xifratge d'un missatge amb la clau asimètrica pública d'una entitat.
- ➔ Desxifratge d'un missatge amb la clau asimètrica privada d'una entitat.
- ➔ Obtenció dels números aleatoris necessaris per a portar a terme els protocols de seguretat explicat més endavant.

Notació

En els punts següents es detallen els passos que segueixen els protocols de seguretat. Per entendre aquests, cal fixar una notació per facilitar la comprensió d'aquests:

- ➔ *M*: missatge
- ➔ *E*: entitat. Pot correspondre al gestor o a l'usuari
- ➔ *U*: usuari (pacient o metge)
- ➔ *G*: gestor del sistema
- ➔ *Pe*: clau pública de l'entitat E
- ➔ *Te*: clau privada de l'entitat E
- ➔ *Ne*: número aleatori generat per l'entitat E
- ➔ *Id_e*: identificador de l'entitat E
- ➔ *Se[M]*: signatura digital del missatge M amb la clau privada Te
- ➔ *Ve[M]*: verificació de la signatura digital del missatge M amb la clau pública Pe
- ➔ *Xe[M]*: xifratge del missatge M amb la clau pública Pe
- ➔ *De[M]*: desxifratge del missatge M amb la clau privada Te

També cal tenir en compte la diferència de notació entre *Ne* i *Ne'*: el primer indica el número aleatori generat per l'entitat E, mentre que el segon representa el número aleatori

de l'entitat E recuperat pel receptor del missatge; aquests dos números poden no ser iguals, la qual cosa significaria que hi ha hagut un intent de suplantació d'identitat.

Aquests quatre protocols són els que el sistema haurà d'implementar.

Protocol 1: consulta d'un historial

Tant un metge com un pacient poden consultar un historial mèdic: el metge tots els corresponents als dels seus pacients i aquests el seu propi. Els passos necessaris per tal que un usuari consulti de manera segura l'historial d'un pacient són aquests:

1. *U* realitza les operacions següents:
   a. Executa el **Procediment 1** passant-li com a paràmetre *Pu*. Obté com a resultat *Xg[Nu, Id_u]*.
   b. Envia *Xg[Nu, Id_u]* a *G*.
2. *G* realitza les operacions següents:
   a. Executa el **Procediment 2** passant-li com a paràmetre *Xg[Nu, Id_u]*. Obté com a resultat *Xu[Nu', Ng, Id_g]*.
   b. Envia *Xu[Nu', Ng, Id_g]* a *U*.
3. *U* realitza les operacions següents:
   a. Executa *Du[Xu[Nu', Ng, Id_g]]*. Obté com a resultat *Nu'*, *Ng* i *Id_g*.
   b. Recupera *Nu*, generat al pas 1.
   c. Si (*Nu* és igual a *Nu'*) aleshores:
      i. Executa *Xg[Ng, CONSULTA, Id_u]*. *CONSULTA* és una constant que indica que es vol consultar l'historial de l'usuari identificat amb *Id_u*.
      ii. Envia *Xg[Ng, CONSULTA, Id_u]* a *G*.
   d. Sinó retorna un error indicant que el missatge de l'usuari no ha estat validat correctament.
4. *G* realitza les operacions següents:
   a. Executa *Dg[Xg[Ng, CONSULTA, Id_u]]*. Obté com a resultat *Ng', CONSULTA, Id_u'*.
   b. Recupera *Ng*, generat al pas 2.
   c. Si (*Ng* és igual a *Ng'*) aleshores:
      i. Recupera *Id_u*, obtingut del missatge al pas 2.
      ii. Si (*Id_u* és igual a *Id_u'*) o bé ((*Id_u* correspon a un metge) i (*Id_u'* és un pacient de *Id_u*)) aleshores:
         A. Executa el **Procediment 3** passant-li com a paràmetre *Pu* i *Id_u*. Obté com a resultat *Xu[historial]*.
         B. Envia *Xu[historial]* a *U*.
      iii. Sinó retorna un error.
   d. Sinó retorna un error indicant que el missatge del gestor no ha estat validat correctament.
5. *U* realitza les operacions següents:
   a. Executa el **Procediment 4** passant-li com a paràmetre *Xu[historial]*. Obté com a resultat *historial*.
   b. Mostra *historial* per pantalla.

Protocol 2: consulta dels pacients assignats a un metge

Els metges consulten quins pacients tenen assignats per poder accedir a l'historial d'algun d'aquests. El detall d'aquest protocol la podem veure a continuació:

1. *U* realitza les operacions següents:
   a. Executa el **Procediment 1** passant-li com a paràmetre *Pu*. Obté com a resultat *Xg[Nu, Id_u]*.
   b. Envia *Xg[Nu, Id_u]* a *G*.
2. *G* realitza les operacions següents:
   a. Executa el **Procediment 2** passant-li com a paràmetre *Xg[Nu, Id_u]*. Obté com a resultat *Xu[Nu', Ng, Id_g]*.
   b. Envia *Xu[Nu', Ng, Id_g]* a *U*.
3. *U* realitza les operacions següents:
   a. Executa *Du[Xu[Nu', Ng, Id_g]]*. Obté com a resultat *Nu'*, *Ng* i *Id_g*.
   b. Recupera *Nu*, generat al pas 1.
   c. Si (*Nu* és igual a *Nu'*) aleshores:
      i. Executa *Xg[Ng, LLISTA_PACIENTS]*. *LLISTA_PACIENTS* és una constant que indica que es vol consultar un llistat dels identificadors dels pacients assignats al metge identificat amb *Id_u*, enviat al pas 1.
      ii. Envia *Xg[Ng, CONSULTA]* a *G*.
   d. Sinó retorna un error indicant que el missatge de l'usuari no ha estat validat correctament.
4. *G* realitza les operacions següents:
   a. Executa *Dg[Xg[Ng, LLISTA_PACIENTS]]*. Obté com a resultat *Ng', LLISTA_PACIENTS*.
   b. Recupera *Ng*, generat al pas 2.
   c. Si (*Ng* és igual a *Ng'*) aleshores:
      i. Recupera *Id_u* obtingut del missatge al pas 2.
      ii. Si (*Id_u* correspon a un metge) aleshores:
         A. Executar el **Procediment 5** passant-li com a paràmetre *Pu* i *Id_u*. Obté com a resultat *Xu[{Id_u$_1$, ..., Id_u$_n$}, Sg[{Id_u$_1$, ..., Id_u$_n$}]]*.
         B. Envia *Xu[{Id_u$_1$, ..., Id_u$_n$}, Sg[{Id_u$_1$, ..., Id_u$_n$}]]* a *U*.
      iii. Sinó retorna un error indicant que el missatge del gestor no ha estat validat correctament.
5. *U* realitza les operacions següents:
   a. Executa el **Procediment 6** passant-li com a paràmetre *Xu[{Id_u$_1$, ..., Id_u$_n$}, Sg[{Id_u$_1$, ..., Id_u$_n$}]]*. Obté com a resultat *{Id_u$_1$, ..., Id_u$_n$}*.
   b. Mostra *{Id_u$_1$, ..., Id_u$_n$}* per pantalla.

Protocol 3: inserció de dades a l'historial de visites

L'operació d'inserció de dades a l'historial d'un pacient només la pot efectuar un metge, per la qual cosa aquesta condició serà una de les indispensables d'aquest protocol:

1. *U* realitza les operacions següents:
   a. Executa el **Procediment 1** passant-li com a paràmetre *Pu*. Obté com a resultat *Xg[Nu, Id_u]*.
   b. Envia *Xg[Nu, Id_u]* a *G*.

2. *G* realitza les operacions següents:
   a. Executa el **Procediment 2** passant-li com a paràmetre *Xg[Nu, Id_u]*. Obté com a resultat *Xu[Nu', Ng, Id_g]*.
   b. Envia *Xu[Nu', Ng, Id_g]* a *U*.
3. *U* realitza les operacions següents:
   a. Executa *Du[Xu[Nu', Ng, Id_g]]*. Obté com a resultat *Nu'*, *Ng* i *Id_g*.
   b. Recupera *Nu*, generat al pas 1.
   c. Si (*Nu* és igual a *Nu'*) aleshores:
      i. Obté les dades de la visita *visita*, la qual hauria d'incloure com a mínim l'identificador del pacient Id_u_p.
      ii. Executa *Su[visita]*.
      iii. Executa *Xg[Ng, INSERIR_VISITA, visita, Su[visita]]*. *INSERIR_VISITA* és una constant que indica que es vol afegir les dades de la visita a l'historial del pacient identificat amb Id_u_p.
      iv. Envia *Xg[Ng, INSERIR_VISITA, visita, Su[visita]]* a *G*.
   d. Sinó retorna un error indicant que el missatge de l'usuari no ha estat validat correctament.
4. *G* realitza les operacions següents:
   a. Executa *Dg[Xg[Ng, INSERIR_VISITA, visita, Su[visita]]]*. Obté com a resultat *Ng', INSERIR_VISITA, visita, Su[visita]*.
   b. Recupera *Ng*, generat al pas 2.
   c. Si (*Ng* és igual a *Ng'*) aleshores:
      i. Obté Id_u_p a partir de *visita*.
      ii. Recupera *Id_u* obtingut del missatge al pas 2.
      iii. Si (*Id_u* correspon a un metge) aleshores:
         A. Si (Id_u_p és un pacient de *Id_u*) aleshores:
            I. Si (*Vu[Su[visita]]* és correcte) aleshores:
               - Obté l'instant de temps actual *temps*.
               - Obté el número de sèrie *nserie* de la darrera visita de l'historial *historial* del pacient identificat amb Id_u_p.
               - Incrementa en una unitat *nserie*. Obté com a resultat *nserie+1*.
               - Executa *Sg[visita, Su[visita], temps, nserie+1]*.
               - Executa *Xg[visita, Su[visita]]*.
               - Executa *Sg[*Id_u_p*, nserie+1*].
               - Guarda *Xg[visita, Su[visita]]*, *nserie+1*, *temps*, *Sg[visita, Su[visita], temps, nserie+1]* i *Sg[*Id_u_p*, nserie+1]* a la base de dades.
            II. Sinó retorna un error indicant que la signatura de la visita no ha estat validada correctament.
         B. Sinó retorna un error indicant que el metge no té aquest pacient assignat.
      iv. Sinó retorna un error indicant que l'usuari no és un metge.
   d. Sinó retorna un missatge indicant que el missatge del gestor no ha estat validat correctament.

Protocol 4: autenticació d'usuaris

Aquest darrer protocol s'utilitza durant el procés de login dels usuaris. Permet al gestor validar que l'usuari que demana accedir al sistema és realment qui diu que és.

1. *U* realitza les operacions següents:
   a. Executa el **Procediment 1** passant-li com a paràmetre *Pu*. Obté com a resultat *Xg[Nu, Id_u]*.
   b. Envia *Xg[Nu, Id_u]* a *G*.
2. *G* realitza les operacions següents:
   a. Executa el **Procediment 2** passant-li com a paràmetre *Xg[Nu, Id_u]*. Obté com a resultat *Xu[Nu', Ng, Id_g]*.
   b. Envia *Xu[Nu', Ng, Id_g]* a *U*.
3. *U* realitza les operacions següents:
   a. Executa *Du[Xu[Nu', Ng, Id_g]]*. Obté com a resultat *Nu'*, *Ng* i *Id_g*.
   b. Executa *Xg[Ng]*.
   c. Envia *Xg[Ng]* a G.
4. *G* realitza les operacions següents:
   a. Executa *Dg[Xg[Ng]]*. Obté com a resultat *Ng'*.
   b. Recupera *Ng*, generat al pas 2.
   c. Si (*Ng* és igual a *Ng'*) aleshores:
      i. *G* i *U* estan autenticats bilateralment.
   d. Sinó retorna un error indicant que l'usuari no ha estat validat correctament.

Procediments principals

Per acabar aquest apartat, es descriuen a continuació els 6 procediments principals emprats en els diversos protocols anteriors:

- Procediment 1

  Aquest procediment l'executa l'usuari:
  1. Obté *Nu*.
  2. Executa *Xg[Nu, Id_u]*.
  3. Retorna *Xg[Nu, Id_u]*.

- Procediment 2

  El procediment 2 serà executat pel gestor.
  1. Rep com a paràmetre *[Xg[Nu, Id_u]]*.
  2. Executa *Dg[Xg[Nu, Id_u]]*. Obté com a resultat *Nu'* i *Id_u*.
  3. Guarda *Nu'* per utilitzar-lo més endavant.
  4. Recupera el certificat de l'usuari identificat per *Id_u* de la base de dades.
  5. Obté *Pu* a partir del certificat.
  6. Obté *Ng*.
  7. Guarda *Ng* per utilitzar-lo més endavant.
  8. Executa *Xu[Nu', Ng, Id_g]*.
  9. Retorna *Xu[Nu', Ng, Id_g]*.

- Procediment 3

  L'usuari executa el tercer procediment:
  1. Rep com a paràmetre *Pu* i *Id_u*.
  2. Recupera el historial *historial* de l'usuari identificat per *Id_u* de la base de dades.
  3. Executa *Dg[Xg[historial]]*. Obté com a resultat *historial*.
  4. Executa *Xu[historial]*.
  5. Retorna *Xu[historial]*.

- Procediment 4

  Per desxifrar un historial enviat pel gestor i verificar que és correcte es fa servir el procediment descrit a continuació:
  1. Rep com a paràmetre *Xu[historial]*.
  2. Executa *Du[Xu[historial]]*. Obté com a resultat *historial*.
  3. Per a cada entrada de l'historial *historial* fer:
     a. Executa *Vu[Su[visita]]*.
     b. Executa *Vg[Sg[visita, Su[visita], temps, nserie+1]]*.
     c. Executa *Vg[Sg[Id_u$_p$, nserie+1]]*.
     d. Verifica la seqüència.
  4. Retorna *historial*.

- Procediment 5

  El gestor, que serà qui executarà aquest procediment, recupera de la base de dades la llista dels pacients assignats a un metge determinat:
  1. Rep com a paràmetre *Pu* i *Id_u*.
  2. Recupera el llistat *{Id_u$_1$, ..., Id_u$_n$}* de tots els identificadors dels pacients assignats al metge identificat per *Id_u* de la base de dades.
  3. Executa *Sg[{Id_u$_1$, ..., Id_u$_n$}]*.
  4. Executa *Xu[{Id_u$_1$, ..., Id_u$_n$}, Sg[{Id_u$_1$, ..., Id_u$_n$}]]*.
  5. Retorna *Xu[{Id_u$_1$, ..., Id_u$_n$}, Sg[{Id_u$_1$, ..., Id_u$_n$}]]*.

- Procediment 6

  El metge utilitza aquest darrer procediment per obtenir la llista dels seus pacients i verificar que ha estat generada pel gestor:
  1. Rep com a paràmetre *Xu[{Id_u$_1$, ..., Id_u$_n$}, Sg[{Id_u$_1$, ..., Id_u$_n$}]]*.
  2. Executa *Du[Xu[{Id_u$_1$, ..., Id_u$_n$}, Sg[{Id_u$_1$, ..., Id_u$_n$}]]]*. Obté com a resultat *{Id_u$_1$, ..., Id_u$_n$}* i *Sg[{Id_u$_1$, ..., Id_u$_n$}]*.
  3. Si (*Vg[Sg[{Id_u$_1$, ..., Id_u$_n$}]]* és correcte) aleshores:
     i. Retorna *{Id_u$_1$, ..., Id_u$_n$}*.
  4. Sinó retorna un error indicant que la signatura de la visita no ha estat validada correctament.

APARTAT 3.3 - Generació del certificats digitals

Cada un dels protocols criptogràfics que s'han d'implementar necessiten que tots els actors disposin d'una parella de claus i del seu corresponent certificat. Per a gestionar aquests certificats s'ha emprat la infraestructura de clau pública. Donat que no s'ha fet servir cap llibreria criptogràfica específica -com per exemple IAIK o Bouncy Castle-, s'ha hagut de treballar amb unes estructures anomenades keystores, les quals representen magatzems on es guarden típicament les claus privades i els certificats corresponents, enlloc de fer-ho directament amb els fitxers PKCS, donada la limitació que el JDK de Java té a l'hora de tractar amb aquest tipus d'estructures -més concretament, la impossibilitat d'accedir a la clau privada del certificat-. Totes aquestes dades -claus, certificats i keystores dels actors- han estat obtingudes com s'indica a continuació en els procediments corresponents. Comentar només que s'ha fet servir la mateixa contrasenya per a tots i cada un dels certificats donat que es tracta d'un exemple pedagògic; en un cas real l'usuari seria qui donaria la seva pròpia contrasenya:

Procediment per obtenir el Keystore del Gestor

1. Generació dels magatzems de claus amb l'eina keytool.
   1.1. Creem el parell de claus pel Gestor:
   ```
   keytool -genkey -alias Gestor -keypass uoc0506 -keystore
   gestor.keystore -keyalg "RSA" -sigalg "SHA1withRSA"
   ```
   1.2. Creem la petició de certificat pel Gestor:
   ```
   keytool -certreq -alias Gestor -sigalg "SHA1withRSA" -file gestor.csr
   -keypass uoc0506 -keystore gestor.keystore -storepass uoc0506
   ```
2. Emissió dels certificats.
   2.1. Emetem el certificat del Gestor:
   ```
   openssl x509 -sha1 -req -in gestor.csr -days 180 -CA CA.crt -CAkey
   CA.key -CAcreateserial -out gestor.crt -outform DER
   ```
3. Importació dels certificats.
   3.1. Importem el certificat de la CA al keystore del Gestor:
   ```
   keytool -import -alias CA -file CA.crt -keystore gestor.keystore
   -trustcacerts -storepass uoc0506 -keypass uoc0506
   ```
   3.2. Importem el certificat del Gestor:
   ```
   keytool -import -alias Gestor -file gestor.crt -keystore
   gestor.keystore -keypass uoc0506 -storepass uoc0506
   ```

Procediment per obtenir el Keystore dels metges

1. Generació dels magatzems de claus amb l'eina keytool.
   1.1. Creem el parell de claus pel metge X:
   ```
   keytool -genkey -alias Metge -keypass uoc0506 -keystore X.keystore
   -keyalg "RSA" -sigalg "SHA1withRSA"
   ```
   1.2. Creem la petició de certificat pel metge X:
   ```
   keytool -certreq -alias Metge -sigalg "SHA1withRSA" -file X.csr
   -keypass uoc0506 -keystore X.keystore -storepass uoc0506
   ```
2. Emisió dels certificats.
   2.1. Emetem el certificat del metge X:
   ```
   openssl x509 -sha1 -req -in X.csr -days 180 -CA CA.crt -CAkey CA.key
   -CAcreateserial -out X.crt -outform DER
   ```

3. Importació dels certificats.
   3.1. Importem el certificat de la CA al keystore del metge X:
   ```
   keytool -import -alias CA -file CA.crt -keystore X.keystore
   -storepass uoc0506 -keypass uoc0506
   ```
   3.2. Importem el certificat del metge X:
   ```
   keytool -import -alias Metge -file X.crt -keystore X.keystore
   -keypass uoc0506 -storepass uoc0506
   ```

Procediment per obtenir el Keystore dels pacients

1. Generació dels magatzems de claus amb l'eina keytool.
   1.1. Creem el parell de claus pel pacient X:
   ```
   keytool -genkey -alias Pacient -keypass uoc0506 -keystore X.keystore
   -keyalg "RSA" -sigalg "SHA1withRSA"
   ```
   1.2. Creem la petició de certificat pel pacient X:
   ```
   keytool -certreq -alias Pacient -sigalg "SHA1withRSA" -file X.csr
   -keypass uoc0506 -keystore X.keystore -storepass uoc0506
   ```
2. Emisió dels certificats.
   2.1. Emetem el certificat del pacient X:
   ```
   openssl x509 -sha1 -req -in X.csr -days 180 -CA CA.crt -CAkey CA.key
   -CAcreateserial -out X.crt -outform DER
   ```
3. Importació dels certificats.
   3.1. Importem el certificat de la CA al keystore del pacient X:
   ```
   keytool -import -alias CA -file CA.crt -keystore X.keystore
   -storepass uoc0506 -keypass uoc0506
   ```
   3.2. Importem el certificat del pacient X:
   ```
   keytool -import -alias Pacient -file X.crt -keystore X.keystore
   -keypass uoc0506 -storepass uoc0506
   ```

APARTAT 3.4 – Documentació de requisits

Informació general

El sistema implementarà un esquema criptogràfic que garantirà les necessitats de seguretat d'un historial mèdic gestionat a través d'una xarxa de comunicacions. L'historial d'un pacient pot contenir informació molt variada, alguna de la qual pot ser considerada com a confidencial. Per aquesta raó, cada servei que ofereixi el sistema haurà d'especificar clarament quins requisits de seguretat acompleix: confidencialitat, autenticitat, integritat i no repudi. Per assegurar aquests requisits, es dissenyarà un protocol criptogràfic per cada acció o servei a desenvolupar. En aquest sistema només hi podran accedir aquells usuaris els quals disposin dels certificats i les claus privades corresponents.

Aquest sistema es composarà de dos components: una aplicació usuari -vàlida tant pel pacient com pel metge- i un gestor central. El primer el faran servir els pacients per accedir de forma segura al gestor del sistema i poder-hi fer les consultes i peticions permeses, i els metges per igualment accedir de forma segura al gestor del sistema i utilitzar les funcionalitats bàsiques que aquest li oferirà. Finalment, el gestor central serà el programari que gestionarà el magatzem dels historials mèdics dels pacients de forma centralitzada i controlarà l'accés a aquests.

Cada usuari s'autenticarà contra el gestor del sistema i només podrà accedir a aquest programari per a realitzar les seves tasques.

Guions dels actors

- Guió del pacient

  Els pacients han de poder realitzar següents accions que li són permeses:
  - ➔ Autenticar-se contra el gestor del sistema
  - ➔ Realitzar una consulta del seu historial
  - ➔ Abandonar de forma segura el sistema

- Guió del metge

  Els metges, per la seva banda, són els encarregats principals de mantenir i actualitzar els historials dels seus pacients. Per aquesta raó necessitaran:
  - ➔ Autenticar-se contra el gestor del sistema
  - ➔ Realitzar una consulta de la llista dels seus pacients
  - ➔ Realitzar una consulta de l'expedient d'un dels seus pacients
  - ➔ Introduir dades noves a l'historial d'un dels seus pacients
  - ➔ Abandonar de forma segura el sistema

- Guió del gestor del sistema

  És el programari central que gestiona el magatzem dels historials mèdics i que té les següents funcionalitats:
  - ➔ Autenticar als pacients i als metges que hi volen accedir
  - ➔ Acceptar les consultes d'ambdós actors
  - ➔ Guardar de forma segura els historials mèdics dels pacients
  - ➔ Verificar que les insercions als historials mèdics les han realitzat usuaris autoritzats
  - ➔ Permetre l'abandonament del sistema de forma segura als metges i pacients

Glossari

Pacient: Usuari del sistema. Utilitza l'aplicació pacient.
Metge: Usuari del sistema. Utilitza l'aplicació metge.
Gestor del sistema: Programari que permet donar d'alta metges i pacients i gestiona de forma segura els historials mèdics.
Aplicació pacient: Programari que permet a un pacient consultar les dades del seu historial i demanar un servei mèdic.
Aplicació metge: Programari que permet a un metge consultar i modificar les dades de l'historial d'un dels seus pacients i donar-ne d'alta nous pacients
Gestor central: Veure *Gestor del sistema*.
Autenticitat: Propietat que fa referència a la identificació. És el nexe d'unió entre la informació i l'emissor d'aquesta.

Confidencialitat: Propietat que assegura que només aquells que estan autoritzats tindran accés a la informació.
Integritat: Propietat que assegura la no alteració de la informació.
No repudi: Propietat que preserva que alguna de les parts negui algun compromís o acció presa amb anterioritat.

Casos d'ús

Dels guions dels diferents actors s'extreuen els següents casos d'ús:

- Cas d'ús número 1: “Entrar al sistema”

  - ➔ Resum de la funcionalitat: l'usuari entra al sistema.
  - ➔ Paper dins el treball de l'usuari: és un cas d'ús bàsic dins el treball dels actors.
  - ➔ Actors: **pacient**, **metge**.
  - ➔ Casos d'ús relacionats: Autenticar usuaris.
  - ➔ Precondició: l'usuari no pot accedir al sistema.
  - ➔ Postcondició: l'usuari ja pot accedir als diversos serveis oferts si el sistema li ha permès l'entrada.
  - ➔ Descripció detallada: l'usuari introdueix el seu nom d'usuari i la seva contrasenya i el sistema comprova aquestes dades, permetent-lo accedir o rebutjant-lo si s'escau.

- Cas d'ús número 2: “Abandonar el sistema”

  - ➔ Resum de la funcionalitat: l'usuari deixa de treballar amb el sistema.
  - ➔ Paper dins el treball de l'usuari: és un cas d'ús bàsic dins el treball dels actors.
  - ➔ Actors: **pacient**, **metge**.
  - ➔ Casos d'ús relacionats: cap.
  - ➔ Precondició: l'usuari es troba dins el sistema.
  - ➔ Postcondició: l'usuari ha sortit del sistema.
  - ➔ Descripció detallada: l'usuari decideix sortir del sistema i aquest gestiona aquesta sortida.

- Cas d'ús número 3: “Consultar historial”

  - ➔ Resum de la funcionalitat: l'usuari accedeix a les dades de l'historial mèdic.
  - ➔ Paper dins el treball de l'usuari: és un cas d'ús bàsic dins el treball dels actors.
  - ➔ Actors: **pacient**, **metge**.
  - ➔ Casos d'ús relacionats: cap.
  - ➔ Precondició: l'usuari es troba dins el sistema i es troba visualitzant la fitxa del **pacient** en qüestió.
  - ➔ Postcondició: la informació de l'historial del **pacient** es mostra per pantalla.
  - ➔ Descripció detallada: l'usuari -mentre es troba consultant les dades personal d'un **pacient**- accedeix a les dades de l'historial mèdic d'aquest.

- Cas d'ús número 4: “Consultar pacients”

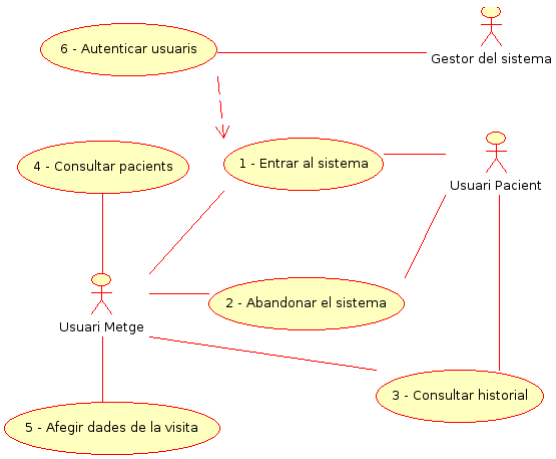
  - ➔ Resum de la funcionalitat: el **metge** realitza una consulta de la llista dels seus **pacients**.
  - ➔ Paper dins el treball de l'usuari: és un cas d'ús bàsic dins el treball del **metge**.
  - ➔ Actors: **metge**.
  - ➔ Casos d'ús relacionats: cap.
  - ➔ Precondició: el **metge** es troba dins el sistema.
  - ➔ Postcondició: la llista de **pacients** del **metge** es mostra per pantalla.
  - ➔ Descripció detallada: el **metge** vol una llista dels seus **pacients** i selecciona aquesta opció per obtenir-la per pantalla.

- Cas d'ús número 5: “Afegir dades de la visita”

  - ➔ Resum de la funcionalitat: el **metge** afegeix les dades de la visita a l'historial d'un **pacient**.
  - ➔ Paper dins el treball de l'usuari: és un cas d'ús bàsic dins el treball del **metge**.
  - ➔ Actors: **metge**.
  - ➔ Casos d'ús relacionats: cap.
  - ➔ Precondició: el **metge** ha realitzat una visita a un **pacient**.
  - ➔ Postcondició: l'historial mèdic del **pacient** ha estat actualitzat a la BD.
  - ➔ Descripció detallada: el **metge** introdueix les dades de la visita al sistema i guarda els canvis.

- Cas d'ús número 6: “Autenticar usuaris”

  - ➔ Resum de la funcionalitat: permet l'accés dels usuaris al sistema.
  - ➔ Paper dins el treball de l'usuari: és un cas d'ús bàsic dins el treball del **gestor del sistema**.
  - ➔ Actors: **gestor del sistema**.
  - ➔ Casos d'ús relacionats: Entrar al sistema.
  - ➔ Precondició: hi ha una petició d'accés al sistema i l'usuari no hi és a dins d'aquest.
  - ➔ Postcondició: l'usuari té permís per accedir al sistema si aquest existeix a la base de dades i no està donat de baixa, o no el té en cas contrari.
  - ➔ Descripció detallada: el **gestor del sistema** rep una petició d'entrada al sistema per part d'algun usuari i comprova si aquest existeix a la base de dades, no està donat de baixa i si es tracta d'un usuari vàlid.



Il·lustració 3: Diagrama dels casos d'ús

APARTAT 3.5 – Anàlisi

Descomposició del programari en paquets

Donada la senzillesa d'aquest PFC, només es tindrà un únic paquet que agruparà tots els casos d'ús. En un cas real, o en un desenvolupament més exhaustiu d'aquest projecte, es podria arribar a tenir tres paquets de programari: de seguretat (en aquest hi haurien tots els elements que tenen a veure amb la seguretat del sistema), de servei (contindria els elements propis del treball dins el sistema) i de manteniment (es correspondria als elements de manteniment dels actors i serveis especials).

Identificació de les classes d'anàlisi

Hi han dos tipus de classes en aquesta etapa: d'entitat i de control. A grans trets, les primeres modelen entitats o esdeveniments del món real i les de control fan referència a objectes interns del programari i no persistents. L'altre tipus de classe habitual, les classes de frontera (les quals representen la interfície d'usuari) no es tindran en compte perquè la interfície com a tal no es desenvoluparà fins a la darrera fase.

La metodologia que s'utilitzarà per identificar les classes d'entitat de l'anàlisi serà enumerar per a cada cas d'ús les entitats que hi han:

- Cas d'ús número 1: Entrar al sistema: *Usuari*.
- Cas d'ús número 2: Abandonar el sistema: *Usuari**.
- Cas d'ús número 3: Consultar historial: *HistorialVisites*, *Pacient*, *Usuari**.
- Cas d'ús número 4: Consultar pacients: *Metge**, *Pacient**.
- Cas d'ús número 5: Afegir dades de la visita: *Metge**, *Visita*, *Pacient**, *HistorialVisites**.
- Cas d'ús número 6: Autenticar usuaris: *Usuari**, *Gestor*.

D'entre totes aquestes possibles classes, s'eliminaran aquelles que no tenen sentit i ens restaran finalment les següents:

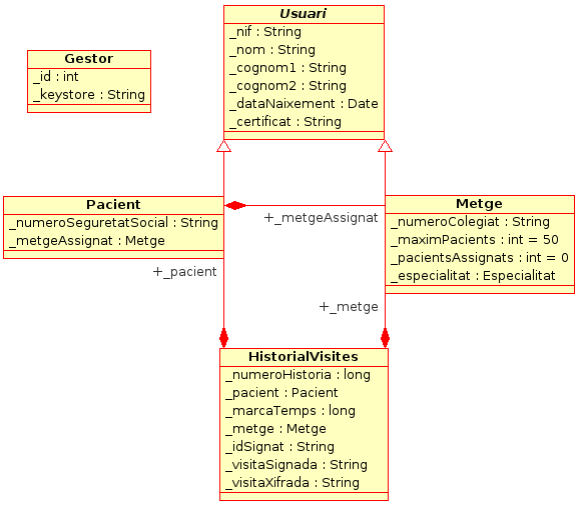
- *Usuari*: nif, certificat.
- *Metge*: numeroCol·legiat, pacientsAssignats, maximPacients.
- *Pacient*: numeroSeguretatSocial, metgeAssignat.
- *HistorialVisites*: numeroHistoria, pacient, data, metge, descripcio.
- *Gestor*: certificat.

La classe Usuari serà una superclasse que contindrà les classes Metge i Pacient. Per a cada usuari es guardarà el nif -que farà d'identificador-, i el certificat -el qual indicarà si es tracta d'un metge o d'un pacient- com a dades bàsiques. Per fer el model de dades més real, s'hi afegiran altres dades complementàries com el nom, l'adreça, el telèfon, el fax i el correu electrònic.

Per a cada metge es guardarà, també de forma complementària, el número de col·legiat -el qual podrà fer també d'identificador-, l'especialitat mèdica, el número màxim de pacients que pot tenir assignats, el número de pacients assignats actualment i l'horari per les visites; el camp que indica el número de pacients assignats serà un camp calculat que s'actualitzarà cada vegada que se li assigna un metge a un pacient. D'altra banda, pel pacient es guardarà el metge assignat i, addicionalment, el número de la Seguretat Social -el qual servirà també com a identificador-.

Els historials de les visites s'identificaran pel número d'història -el qual serà un número seqüencial autoincrementat-, i emmagatzemaran a més a més el nif del pacient, una marca de temps la qual indicarà el moment en què s'ha inserit la informació, l'identificador del metge que l'ha realitzat i la descripció d'aquesta.

Finalment, el Gestor simplement contindrà un identificador, el qual es correspondrà al hash del seu certificat, i el keystore -que no el certificat- el qual contindrà el certificat i la parella de claus.



Il·lustració 4: Diagrama de les classes d'anàlisi

Al diagrama anterior es poden veure les cinc classes que en un principi s'han determinat.

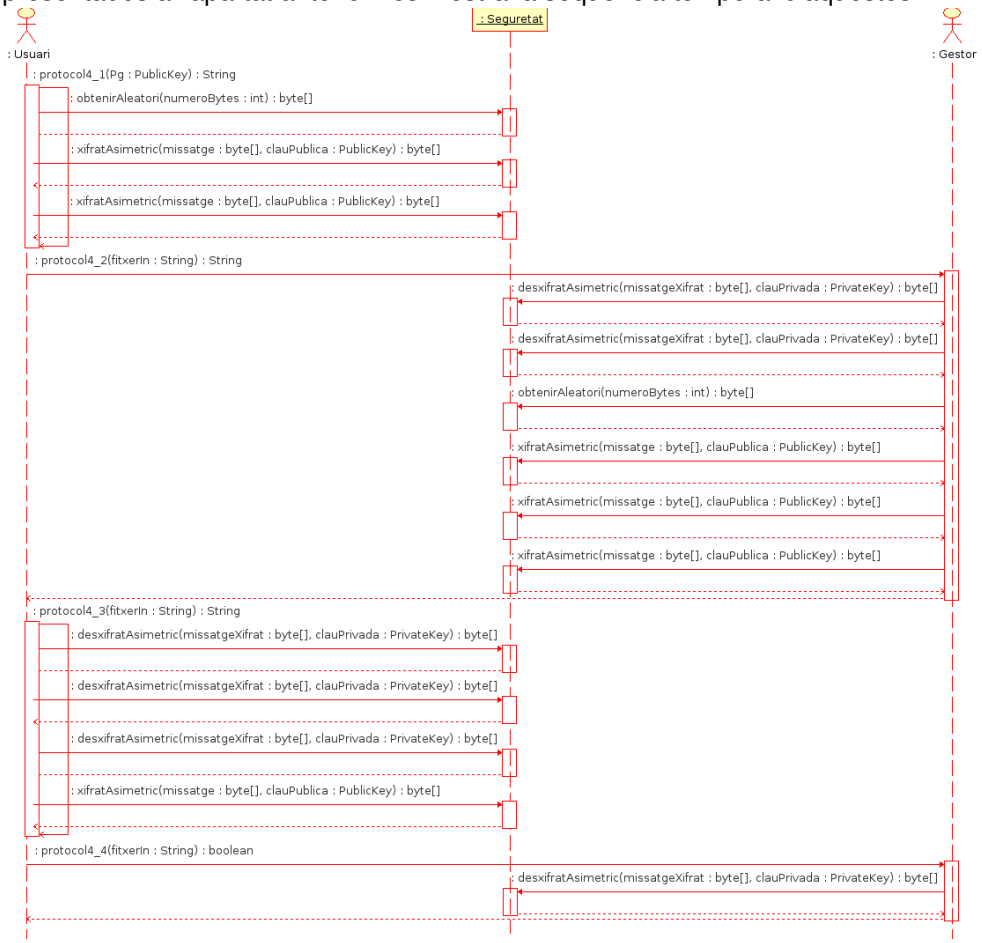
És important parar atenció a la multiplicitat, exclosa d'aquest diagrama per no embrutir-lo, i que a continuació s'explica més detingudament:

- ➔ Cada metge tindrà un número indeterminat de pacients assignats, mentre que cada pacient només en pot tenir com a molt un de metge.
- ➔ Cada entrada de l'historial de visites pertany a un únic pacient, i cada un d'aquests en en té com a mínim una entrada a l'historial -el dia en què es va donar d'alta-.
- ➔ Alhora, cada entrada de l'historial de visites ha estat introduït per un únic metge, mentre que cada metge en pot introduir un número indeterminat d'aquestes.

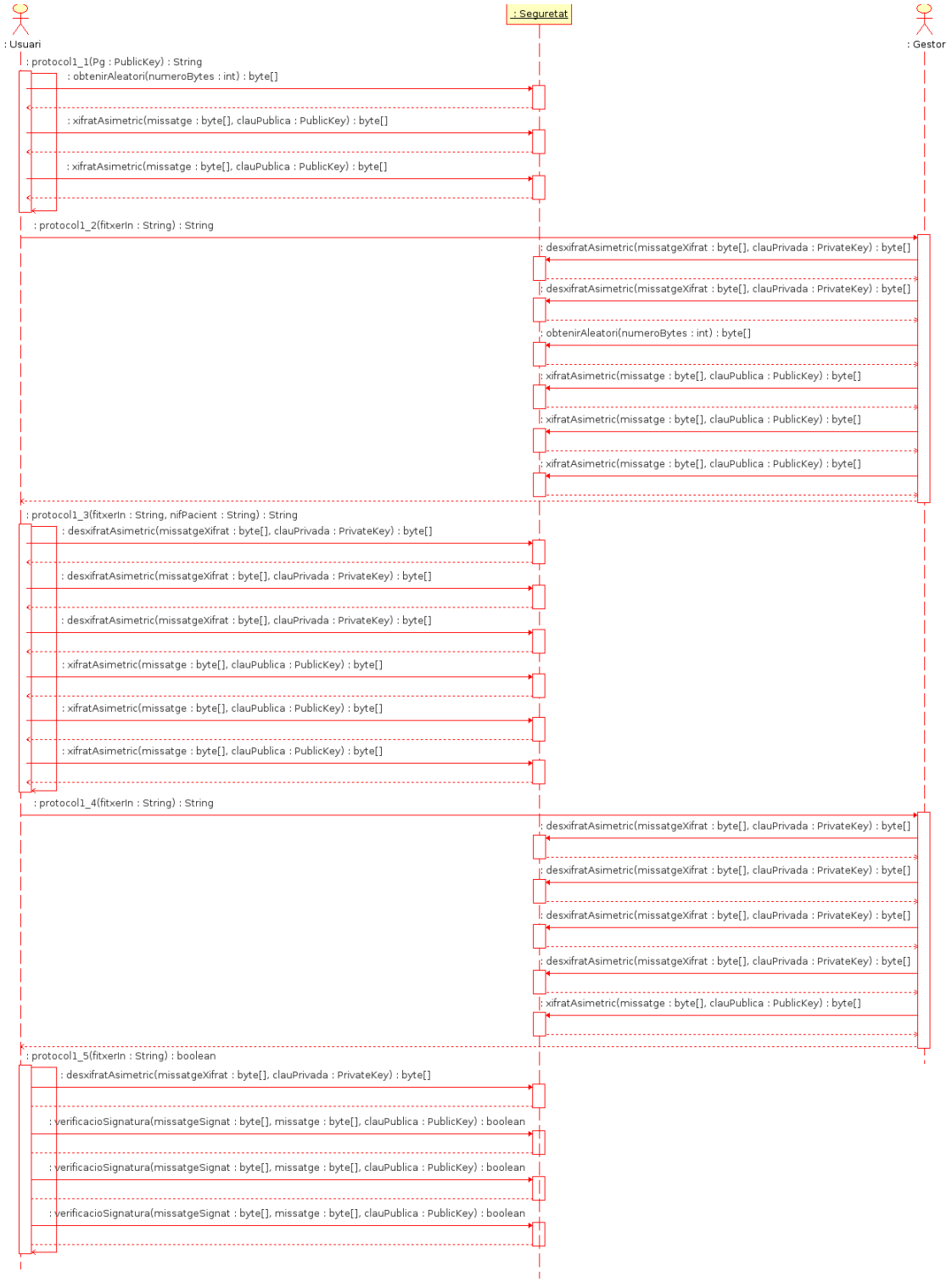
Respecte les classes de control, es considera que hi ha una classe que guardarà la informació dels certificats dels usuaris i que es dirà *Certificat*, i un altra a la qual s'ha de parar especial atenció donat que es tracta de la classe que implementarà els protocols de seguretat de l'aplicació. A aquesta classe se l'anomenarà *Seguretat*, i oferirà els serveis corresponents als requisits dels protocols de seguretat vistos a l'apartat corresponent.

Diagrames de seqüències

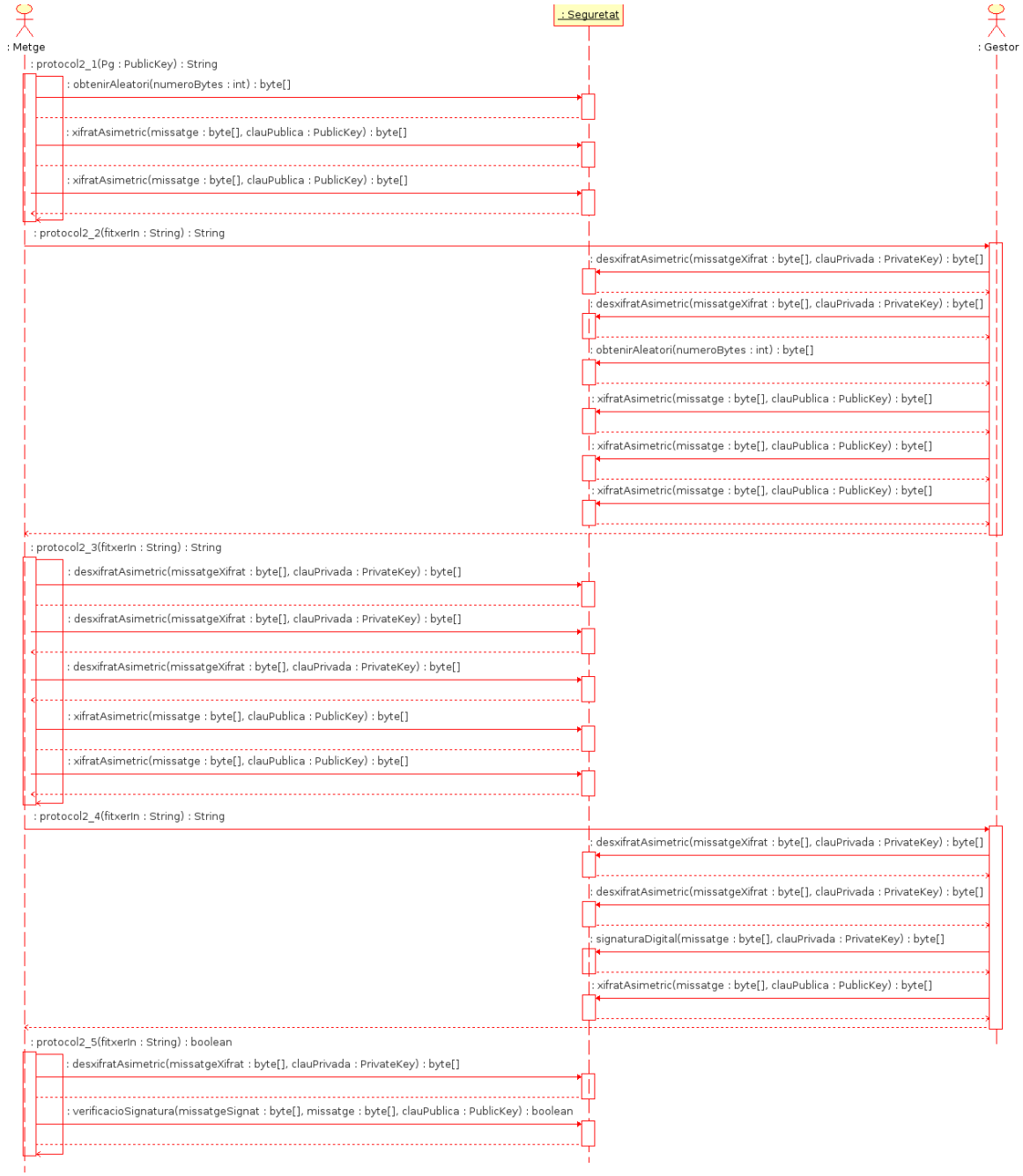
En aquest apartat es descriu la interacció de les operacions existents entre les classes presentades a l'apartat anterior i es mostra la seqüència temporal d'aquestes.



Il·lustració 5: Diagrama de seqüències del protocol d'autenticació



Il·lustració 6: Diagrama de seqüències del protocol de consulta d'un historial

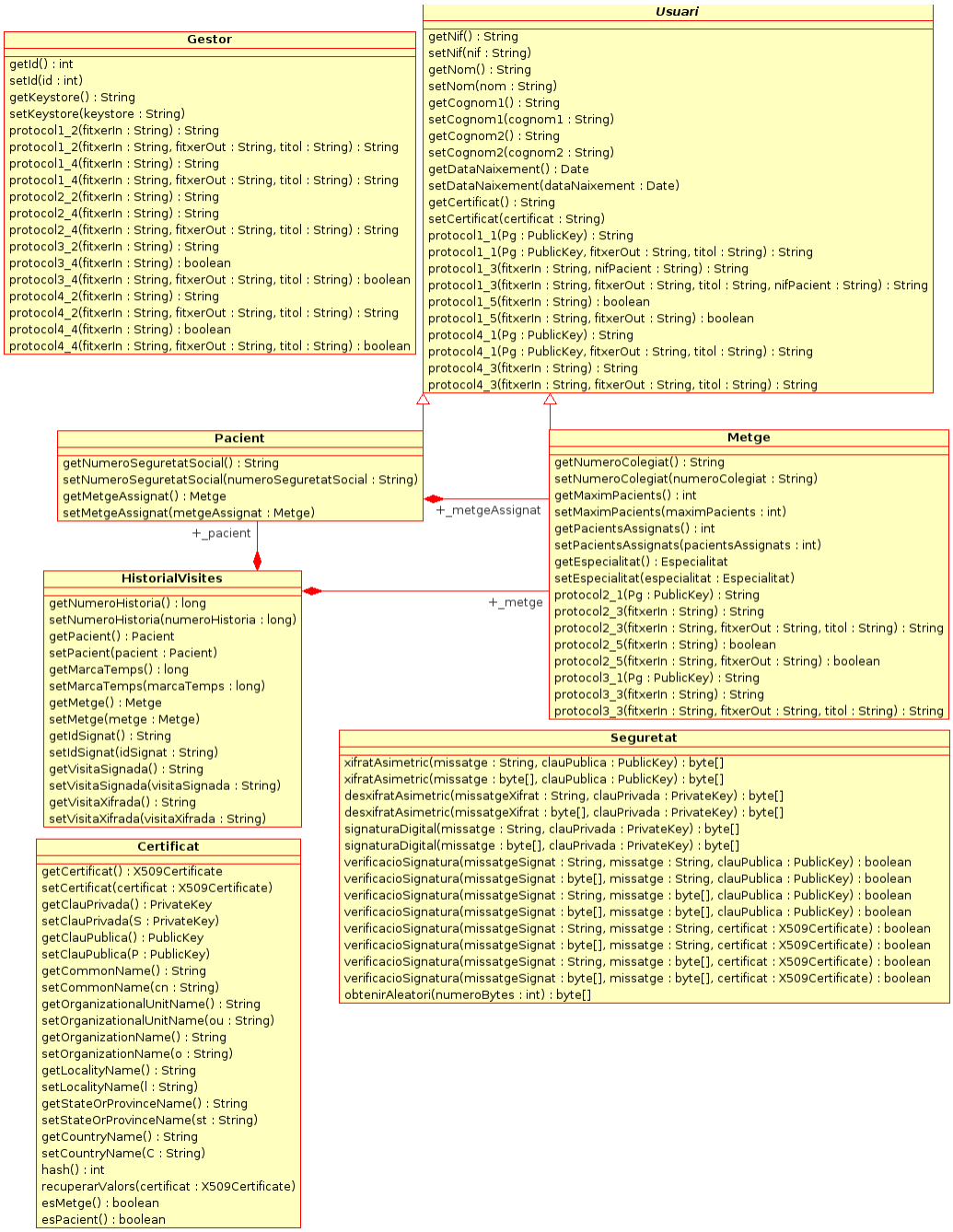


Il·lustració 7:Diagrama de seqüències del protocol de consulta dels pacients assignats a un metge

Il·lustració 8: Diagrama de seqüències del protocol d'inserció de dades a l'historial de visites

Diagrama estàtic de l'anàlisi

El següent diagrama és una ampliació de l'anterior en el qual s'han afegit les classes de control *Certificat* i *Seguretat*, a banda de mostrar les operacions per a totes i cada una d'aquestes:



Il·lustració 9: Diagrama estàtic de l'anàlisi

APARTAT 3.6 – Disseny

A l'hora de dissenyar els protocols de seguretat va aparèixer una complicació: les funcions criptogràfiques emprades per a xifrar i signar els missatges contenen caràcters no imprimibles, la qual cosa donada molts problemes a l'hora d'intercanviar les dades.

Per tal de solucionar-lo s'ha hagut de treballar amb una classe de control especial la qual codifica i decodifica un conjunt de bytes en format Base64, transformació la qual elimina aquests caràcters que provocaven els problemes. Aquesta nova classe s'ha obtingut d'internet, és de codi obert i ha estat implementada per Christian d'Heureuse. Ofereix mètodes per codificar dades en Base64 i altres per decodificar-los.

CAPÍTOL 4 – Representació de dades: XML

APARTAT 4.1 – Introducció

Història

L'XML prové d'un llenguatge inventat per IBM en els anys setanta, anomenat GML (General Markup Language), que va sorgir per la necessitat que tenia l'empresa d'emmagatzemar grans quantitats d'informació. Aquest llenguatge va agradar a l'ISO, pel que el 1986 van treballar per normalitzar-ho, creant SGML (Estàndard General Markup Language), capaç d'adaptar-se a un gran ventall de problemes. A partir d'ell s'han creat altres sistemes per emmagatzemar informació.

L'any 1989 es va crear el llenguatge HTML que va ser utilitzat per a la WWW. HTML ha anat creixent d'una manera descontrolada, no complint tots els requisits que demanava la societat global d'Internet, malgrat els esforços del W3C de posar ordre i establir regles i etiquetes per a la seva estandardització. Aquesta entitat va començar el 1998 el desenvolupament de l'XML, en el que encara continua. Aquest llenguatge és utilitzat per molts altres llenguatges de programació.

L'XML es va crear perquè complís diversos objectius:

- ➔ Que fos idèntic a l'hora de servir, rebre, i processar la informació del HTML per aprofitar tota la tecnologia implantada d'aquest.
- ➔ Que fos normal i concís des del punt de vista de les dades i la manera de guardar-los.
- ➔ Que fos extensible, perquè el puguin utilitzar en tots els camps del coneixement.
- ➔ Que fos fàcil de llegir i editar i implementar.
- ➔ Que fos fàcil d'implantar, programar i aplicar els diferents sistemes.

Ús del model

Els avantatges d'aquest format són aquests:

- ➔ Comunicació de dades. Si la informació es transfereix en l'XML qualsevol aplicació podria escriure un document de text pla amb les dades que estava manejant en format XML i una altra aplicació rebre aquesta informació i treballar amb ella.
- ➔ Migració de dades. Si treballem en format XML seria molt senzill moure dades d'una base de dades a una altra.
- ➔ Aplicacions web. Amb l'XML hi ha una sola aplicació que maneja les dades i per a cada navegador podem tenir un full d'estil o similar per aplicar-li l'estil adequat.

A primera vista, un document l'XML és similar als documents HTML, però realment no són iguals, hi ha una diferència principal: un document l'XML conté dades que s'autodefineixen

exclusivament, mentre que un document HTML conté dades mal definides, barrejades amb elements de format. En l'XML se separa el contingut de la presentació de forma total.

Igual com el HTML, es basa en documents de text pla en els quals s'utilitzen etiquetes per delimitar els elements d'un document. Tanmateix, l'XML defineix aquestes etiquetes en funció del tipus de dades que està descrivint i no de l'aparença final que tindran en pantalla o a la còpia impresa; a més de permetre definir noves etiquetes i ampliar les existents.

Un exemple d'estructura d'un fitxer XML podria ser el següent:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE MISSATGE SYSTEM "missatge.dtd">
<missatge>
        <remitent>
                <nom>J.Cèsar Rodríguez</nom>
                <mail>jrodriguezle@uoc.edu</mail>
        </remitent>
        <destinatari>
                <nom>Jordi Castellà</nom>
                <mail>jcastellar@uoc.edu</mail>
        </destinatari>
        <assumpte>Salutació</assumpte>
        <text>
                <paràgraf>Hola, Jordi. Estic treballant en la memòria.</paràgraf>
        </text>
</missatge>
```

Els documents XML han d'estar ben formats. S'anomena documents "ben formats" (de l'anglès well formed) als documents que compleixen amb totes les definicions bàsiques de format i poden, per tant, ser analitzats correctament per qualsevol "parser" que compleixi amb la norma. Se separa això del concepte de validesa, que implica que no solament el document està ben format sinó que també la seva estructura es correspon amb la definida en un document extern (expressada com a DTD o com a Xschema).

- ➔ Els documents han de seguir una estructura estrictament jeràrquica amb el qual respecta a les etiquetes que delimiten els seus elements. Una etiqueta ha d'estar correctament inclosa en una altra, és a dir, les etiquetes han d'estar correctament niades. Els elements amb contingut han d'estar correctament tancats.
- ➔ L'XML permet elements sense contingut però l'etiqueta ha de ser: < element_sense_contingut />
- ➔ Els documents XML només permeten un element arrel del qual tots els altres siguin part, és a dir, només pot tenir un element inicial.
- ➔ Els valors atributs en l'XML sempre han d'estar tancats entre cometes simples o dobles.
- ➔ L'XML és sensible a majúscules i minúscules. Existeix un conjunt de caràcters anomenats espais en blanc (espais, tabuladors, retorns de carro, salts de línia) que els processadors XML tracten de forma diferent en el marcat XML.
- ➔ És necessari assignar noms a les estructures, tipus d'elements, entitats, elements particulars, etc. En l'XML els noms tenen alguna característica en comú.
- ➔ Les construccions com a etiquetes, referències d'entitat i declaracions es denominen marques; són parts del document que el processador XML espera

entendre. La resta del document entre marques són les dades entendibles per les persones.

Crear una definició DTD és com crear el nostre propi llenguatge de marcat, per a una aplicació específica. La DTD defineix els tipus d'elements, atributs i entitats permeses, i pot expressar algunes limitacions per combinar-los. Els documents XML que s'ajusten al seu DTD es denominen vàlids. Que un document sigui bé format no vol dir que sigui vàlid, un document ben format simplement és aquell que respecta l'estructura i sintaxis definides per l'especificació de l'XML. Un document ben format pot ser vàlid si compleix les regles d'una DTD determinada.

APARTAT 4.2 – Anàlisi

Durant l'execució dels protocols criptogràfics les dades que s'intercanvien els diferents aplicatius tenen una estructura XML. Per tant, a cada pas de cada un dels protocols hi ha un fitxer XML. Després de revisar les dades que s'intercanvien a cada pas, informació extreta de la declaració dels propis protocols, s'obté com a resultat una definició de fitxers molt similar entre tots ells, amb dos elements destacats: *Document*, el qual conté les dades concretes, i *Signatura*, element final de cada fitxer que serveix per validar el fitxer en el seu conjunt, encara que en aquest PFC no s'hagi implementat aquesta opció.

APARTAT 4.3 – Disseny

Un cop revisada l'estructura XML necessària, aquesta s'ha simplificat i s'ha reduït al màxim la seva complexitat, afavorint alhora el seu tractament. Per a portar-ho a terme, s'ha fet un primer esbós de l'esquema DTD final el qual servirà per validar els diferents fitxers XML.

Si s'analitza l'estructura anterior, es pot obtenir un llistat dels diferents elements necessaris i, agrupant aquests, resulta la següent relació d'elements:

```
document  : obligatori
        titol          : obligatori
        validacio      : opcional
                aleatoriUsuari : opcional
                aleatoriGestor : opcional
                identificador  : opcional
        accio (valor)  : opcional
                aleatoriGestor : obligatori
                identificador  : opcional
                dadesVisita    : opcional
                visitaSignada  : opcional
        llistaPacients : opcional
                pacientsDeMetge       : obligatori
                pacientsDeMetgeSignat : obligatori
        historial      : opcional
                identificador : obligatori
```

visites : obligatori
visita : 0 o mes
numeroHistoria : obligatori
data : obligatori
metge : obligatori
idSignat : obligatori
visitaSignada : obligatori
visitaXifrada : obligatori
signatura : obligatori

A l'annex corresponent es poden veure tant l'estructura del fitxer DTD com les pròpies estructures XML resultants, així com les descripcions d'aquestes mateixes estructures.

Per a tractar aquests fitxers s'ha creat una classe de control anomenada fitxerXML la qual pemet per una banda crear un fitxer XML i per l'altra llegir-lo.

CAPÍTOL 5 – Comunicació de components: RMI

APARTAT 5.1 – Introducció

RMI (Remote Method Invocation) és un mecanisme ofert a Java per invocar un mètode remotament. En ser RMI part estàndard de l'entorn d'execució Java usar-lo proveeix un mecanisme simple en una aplicació distribuïda que només necessita comunicar servidors codificats per a Java. Si es requereix comunicar-se amb altres tecnologies s'ha d'usar CORBA o SOAP en lloc de RMI.

En estar específicament dissenyat per a Java RMI pot donar-se el luxe de ser molt amigable per als programadors, proveint passatge per referència d'objectes (cosa que no fa SOAP), "collita d'escombraries" distribuïda i passatge de tipus arbitraris (funcionalitat no proveïda per CORBA).

Per mitjà de RMI, un programa Java pot exportar un objecte. A partir d'aquesta operació aquest objecte està disponible a la xarxa, esperant connexions en un port TCP. Un client pot llavors connectar-se i invocar mètodes. La invocació consisteix en el *marshaling* dels paràmetres (utilitzant la funcionalitat de *serialització* que proveeix Java), després se segueix amb la invocació del mètode (cosa que succeeix en el servidor). Mentre això succeeix el client es queda esperant per una resposta. Una vegada que acaba l'execució el valor de retorn (si n'hi ha) és serialitzat i enviat al client. El codi client rep aquest valor com si la invocació hagués estat local.

Per implementar un servidor RMI s'ha de crear una interfície que enumeri les funcions proveïdes per l'objecte a ser exportat per la xarxa. Aquesta interfície és compartida entre client i servidor. Del costat servidor és necessari implementar aquesta interfície amb una classe que proveeixi la funcionalitat. Del costat client, aquest accedeix a la mateixa interfície, però sota hi ha un *stub* que comença a la crida via la xarxa. Fins a versions recents de Java era necessari generar manualment el *stub* per mitjà d'un compilador especial anomenat *rmic*.

El costat client i el costat servidor hauran de tenir ambdós accés a la definició de les classes utilitzades per a l'intercanvi de paràmetres i de valors de retorn. Per a això Java té la capacitat de dinàmicament obtenir les classes que es necessitin des d'un servidor.

APARTAT 5.2 – Anàlisi

Per a fer que un objecte sigui remot s'ha de declarar com a implementador de la interfície *remot*. Com que la invocació remota pot fallar, cada mètode de la interfície declara també l'excepció *java.rmi.RemoteException* a la secció *throws*.

És necessari un mecanisme per a localitzar objectes remots, i s'anomena registre RMI o *rmiregistry*. Els passos per crear una comunicació RMI entre un client i un servidor són:

- ➔ Definir una interfície la qual contingui la declaració de les funcions proveïdes.
- ➔ Crear un objecte el qual contingui la implementació d'aquesta funcionalitat. Aquest serà l'objecte remot (el servidor).
- ➔ Crear un objecte el qual localitzi l'objecte servidor, obtingui a canvi un *stub* dinàmic i invoqui algun dels mètodes oferts. Aquest objecte serà l'objecte local (el client).

Les classes que necessiten comunicar-se són Gestor i Usuari (i les subclasses Metge i Pacient). La classe Usuari serà qui iniciarà el diàleg amb el Gestor. En un anàlisi inicial, es modificarien aquestes classes per adaptar-les a les necessitats del RMI, això és, crear una classe que declara les funcions i modificar l'actual per tal que implementi aquestes.

La decisió que s'ha pres és una altra: s'han creat tres classes noves per a cada classe existent: la primera per iniciar el servei, la segona per declarar les funcions proveïdes i una tercera que els implementa. Aquesta darrera efectua crides als mètodes de la classe a la qual pertany, evitant així haver de modificar les classes ja existents.

Tanmateix, s'ha integrat en la classe que implementa els mètodes de l'usuari les crides a les funcions pròpies de la classe Metge -la classe Pacient, en aquest projecte, no té cap mètode propi-. D'aquesta manera s'ha aconseguit tenir un servidor i un únic client.

APARTAT 5.3 – Disseny

La classe Usuari

Respecte la classe Usuari, les funcions que ha de proveir són bàsicament les corresponents als diferents passos dels protocols de seguretat. Com ja s'ha comentat, s'ha aprofitat aquesta classe per a proveir les funcionalitats de les seves subclasses i d'aquesta forma obtenir un únic client RMI. Per a portar-ho a terme, s'ha de conèixer el perfil de l'usuari que es comunica amb el Gestor, donat que hi han mètodes que són exclusius de la subclasse Metge; aquesta informació s'aconsegueix a través del certificat del propi usuari, on l'atribut *OrganizationalUnitName* indica a quina mena d'usuari pertany el certificat. La classe *Certificat* compta amb els mètodes necessaris per a identificar el perfil dels usuaris.

Respecte aquest punt, s'han gestionat els errors que es podrien produir si un pacient intentés executar una acció exclusiva d'un metge, mostrant un missatge d'error en el cas en què aquesta circumstància es produís.

La classe que fa les funcions de servidor de l'objecte Usuari crea una instància d'aquest i la retorna per tal que el client la pugui localitzar. Aquest client, un cop ha obtingut el *stub*, ja pot invocar els diferents mètodes declarats a la interfície.

Revisant el codi del servidor de l'objecte Usuari es pot observar que hi han dos constructors, un amb paràmetres i un altre sense. La sintaxi d'ambdós només es diferencia en la crida a l'objecte *ImplementacioUsuari*, en la qual un li passa com a paràmetres el nif i la contrasenya de l'usuari i l'altre no -se suposa que ja es coneix-. En aquest codi es pot

veure també que per localitzar el servidor es necessita conèixer una sèrie de paràmetres (el nom del host i el port utilitzats) els quals es recullen d'un fitxer de configuració.

De la part del servidor també cal la implementació de les funcions proveïdes per l'objecte; aquest codi també compta amb dos constructors, un amb paràmetres (el nif i la contrasenya de l'usuari) i un altre sense. Per al segon constructor existeix el mètode *IniciarUsuari* el qual crea els objectes necessaris (Usuari i Metge o Pacient).

Es pot comprovar com les funcions proveïdes criden els mètodes de les classes subjacents (ja sigui Usuari o Metge), aprofitant d'aquesta manera el codi ja existent.

La interfície compartida entre client i servidor declara les funcions que es proveeixen, i totes aquestes es corresponen, a excepció del darrer mètode, als passos intermedis dels diferents protocols criptogràfics.

El client necessita localitzar el servidor que proveeix les funcions, per la qual cosa ha de conèixer també els paràmetres de connexió necessaris de la mateixa manera com ho fa el servidor a l'hora de generar la instància.

La classe Gestor

Tot el que s'ha dit al punt anterior sobre el disseny RMI de la classe Usuari es pot aplicar a aquest objecte. La diferència principal és que aquesta classe és la quina té accés a la base de dades i, per aquesta raó, qui recupera les dades de la resta d'objectes. Per tant, conté mètodes els quals recuperen la informació dels usuaris (metges i pacients).

El Gestor estarà sempre pendent que el client en demani mètodes, per la qual cosa no té cap objecte client com a tal, sinó que només hi han el servidor -que crea la instància-, la interfície -que declara les funcions- i la implementació -que les implementa-.

Novament s'ha aprofitat el codi ja existent i la implementació de les funcions proveïdes pel Gestor efectua crides als mètodes de la classe subjacent.

CAPÍTOL 6 – Gestió de la informació: Base de Dades

APARTAT 6.1 – Introducció

El gestor de Bases de Dades utilitzat en el desenvolupament d'aquest PFC és MySQL.

Història

El llenguatge de consulta estructurat SQL va ser comercialitzat per primera vegada el 1981 per IBM, el qual va ser presentat a ANSI i des d'aquest llavors ha estat considerat com un estàndard per a les bases de dades relacionals. Des de 1986, l'estàndard SQL ha aparegut en diferents versions com per exemple: SQL:92, SQL:99, SQL:2003. MySQL és una idea originària de l'empresa de codi obert MySQL AB establerta inicialment a Suècia el 1995 i els fundadors de la qual són David Axmark, Allan Larsson, i Michael "Monty" Widenius. L'objectiu que persegueix aquesta empresa consisteix que MySQL compleixi l'estàndard SQL, però sense sacrificar velocitat, fiabilitat o usabilitat.

Michael Widenius en la dècada dels 90 va tractar d'usar mSQL per connectar les taules usant rutines de baix nivell ISAM, tanmateix, mSQL no era ràpid i flexible per a les seves necessitats. Això el va portar a crear una API SQL denominada MySQL per a bases de dades molt similar a la de mSQL però més portable.

La procedència del nom de MySQL no és clara. Per més de 10 anys, les eines han mantingut el prefix My. També, es creu que té relació amb el nom de la filla del cofundador Monty Widenius que es diu My.

D'altra banda, el nom del dofí de MySQL és Sakila i va ser seleccionat pels fundadors de MySQL AB en el concurs "Name the Dolphin". Aquest nom va ser enviat per Ambrose Twebaze, un desenvolupador d'OpenSource Africà, derivat de l'idioma SiSwate, l'idioma local de Swaziland i correspon al nom d'una ciutat a Arusha, Tanzània, a prop d'Uganda, la ciutat origen d'Ambrose.

MySQL és un sistema de gestió de base de dades, multifil i multiusuari amb més de sis milions d'instal·lacions. MySQL AB desenvolupa MySQL com a programari lliure en un esquema de llicenciament dual. D'una banda ho ofereix sota la GNU GPL, però, empreses que vulguin incorporar-lo en productes privatius poden comprar a l'empresa una llicència que els permeti aquest ús. Està desenvolupat majoritàriament en ANSI C.

Al contrari de projectes com l'Apache, on el programari és desenvolupat per una comunitat pública, i el copyright del codi és en poder de l'autor individual, MySQL pertany i està patrocinat per una empresa privada, que té el copyright de la major part del codi. Això és el que possibilita l'esquema de llicenciament anteriorment esmentat. A més de la venda de

llicències privatives, la companyia ofereix suport i serveis. Per a les seves operacions contracten treballadors al voltant del món que col·laboren via Internet.

APARTAT 6.2 – Anàlisi

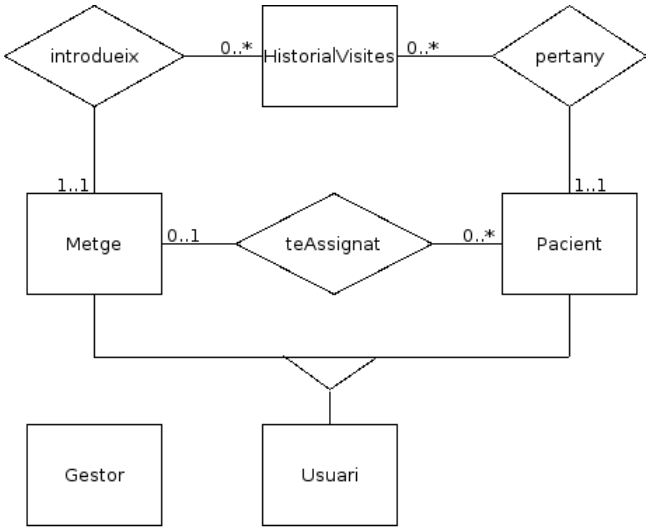
Disseny de la persistència

Totes les classes d'entitat mostrades anteriorment són persistents. Per accedir a les seves dades serà necessària la creació d'una classe especial la qual s'encarregarà de gestionar les dades que es guarden a la base de dades. Aquesta classe de control s'anomena *gestorDades*, i ofereix els mètodes necessaris per tal d'accedir a la base de dades i emmagatzemar o recuperar la informació demanada.

Obtenció de l'estructura de la base de dades relacional

- Pas del model estàtic al model ER

  Les classes del diagrama estàtic de l'anàlisi es corresponen en aquest model a entitats unides per una associació. Cada entitat té un atribut el qual indica la cardinalitat existent entre aquestes. Les relacions entre aquestes associacions es mostren en el diagrama Entitat-Relació següent:



Il·lustració 10: Diagrama ER

- Pas del model ER al model relacional

  Com que no existeix cap relació de la forma N:N, a cada entitat li correspon una taula de la base de dades. El nom de les taules es mantindrà en relació a l'entitat relacionada.

APARTAT 6.3 – Disseny

Model de Base de Dades

Tot seguit es mostrarà l'estructura de les diferents taules que conformen aquest model de dades, i s'indicaran per a cada una d'elles els camps corresponents als atributs de les classes les quals modelen.

Taula Usuari	
Camp	***Restricció***
nif	PK
nom	
cognom1	
cognom2	
dataNaixement	
certificat	

Taula Metge	
Camp	***Restricció***
nif	PK, FK
numeroColegiat	U
maximPacients	
pacientsAssignats	
especialitat	FK

Taula Gestor	
Camp	***Restricció***
id	PK
keystore	

Taula HistorialVisites	
Camp	***Restricció***
numeroHistoria	PK
pacient	FK
marcaTemps	
metge	FK
idSignat	
visitaSignada	
visitaXifrada	

Taula Pacient	
Camp	***Restricció***
nif	PK, FK
numeroSeguretatSocial	U
metgeAssignat	FK

Taula Especialitat	
Camp	***Restricció***
id	PK
descripcio	

Les restriccions PK indiquen que l'atribut és clau primària de la taula. FK significa que són claus forànies, les quals fan referència a altres camps d'altres taules. Les restriccions marcades amb la lletra U signifiquen que són valors únics de la taula, per la qual cosa es poden fer servir d'índex de cerques.

En aquest model es pot veure la introducció d'una nova taula, *Especialitat*, fins el moment desconeguda. El motiu és que des d'un primer moment es va pretendre fer un disseny de les dades flexible per poder adaptar-se a qualsevol necessitat. Això representava un considerable augment de la complexitat en el seu tractament, però els beneficis que comporta aquesta decisió són més que acceptables. Per motius de llegibilitat no s'ha fet esment -fins arribats a aquest punt- de cap entitat tret de les ja aparegudes al llarg d'aquest document. A l'apartat en que es parla de les possibles millores futures es podrà observar el diagrama inicial que es va realitzar en el seu dia.

La taula esmentada ha fet aparició en aquest moment per donar sentit a l'atribut *Especialitat* de l'objecte *Metge*. La gestió d'aquesta, així com la de la resta dels objectes no apareguts al llarg del desenvolupament d'aquest PFC, no ha estat implementada.

Descripció de les taules

D'una manera més formal, a continuació es podrà veure la descripció de les taules a les quals es feia referència al punt anterior:

Taula Usuari				
Nom	***Tipus de dades***	***Nul***	***Valor de defecte***	***Restricció***
nif	char (9)	NO		PK
nom	varchar (100)	NO		
cognom1	varchar (100)	NO		
cognom2	varchar (100)	NO		
dataNaixement	date	NO		
certificat	blob	SI		

Pel que fa al MySQL, el tipus de dades *BLOB* és una cadena de bytes d'una longitud màxima de 65.535 bytes ($2^{16} – 1$). D'altra banda, el tipus de dades *smallint unsigned* que es troba a la descripció de la taula següent representa un nombre enter el rang del qual va des de 0 fins a 65.535 ($2^{16} – 1$).

Taula Metge				
Nom	***Tipus de dades***	***Nul***	***Valor de defecte***	***Restricció***
nif	char (9)	NO		PK, FK
numeroColegiat	char (8)	NO		U
maximPacients	smallint unsigned	NO	50	
pacientsAssignats	smallint unsigned	NO	0	
especialitat	smallint unsigned	NO		FK
FK(nif) – El camp *nif* fa referència al camp *nif* de la taula *Usuari*				
FK(especialitat) – El camp *especialitat* fa referència al camp *id* de la taula *Especialitat*				

Taula Pacient				
Nom	***Tipus de dades***	***Nul***	***Valor de defecte***	***Restricció***
nif	char (9)	NO		PK, FK
numeroSeguretatSocial	char (11)	NO		U
metgeAssignat	char (9)	SI		FK
FK(nif) – El camp *nif* fa referència al camp *nif* de la taula *Usuari*				
FK(metgeAssignat) – El camp *metgeAssignat* fa referència al camp *nif* de la taula *Metge*				

Taula HistorialVisites				
Nom	***Tipus de dades***	***Nul***	***Valor de defecte***	***Restricció***
numeroHistoria	int unsigned	NO		PK
pacient	char (9)	NO		FK
marcaTemps	int unsigned	NO		
metge	char (9)	NO		FK
idSignat	mediumblob	NO		
visitaSignada	longblob	NO		
visitaXifrada	longblob	NO		
FK(pacient) – El camp *pacient* fa referència al camp *nif* de la taula *Pacient*				
FK(metge) – El camp *metge* fa referència al camp *nif* de la taula *Metge*				

En la descripció de la taula anterior apareixen dos tipus de dades nous que cal remarcar: el tipus de dades *int unsigned* representa un nombre enter el rang del qual va des de 0 fins a 4.294.967.295 ($2^{32} – 1$). D'altra banda, el tipus de dades *MEDIUMBLOB* és una cadena de bytes d'una longitud màxima de 16.777.215 bytes ($2^{24} – 1$), mentre que la longitud màxima del tipus de dades LONGBLOB és de 4.294.967.295 ($2^{32} – 1$).

Taula Gestor				
Nom	***Tipus de dades***	***Nul***	***Valor de defecte***	***Restricció***
id	int unsigned	NO		PK
keystore	blob	SI		

Taula Especialitat				
Nom	***Tipus de dades***	***Nul***	***Valor de defecte***	***Restricció***
id	smallint unsigned	NO	autoincrementat	PK
descripcio	varchar (100)	SI		

Com a nota aclaratòria, cal dir que els camps *certificat* i *keystore* de les taules *Usuari* i *Gestor* respectivament poden tenir valor NUL perquè no està implementada la funcionalitat d'altes de registres, per la qual cosa el certificat -o el keystore, en el cas del Gestor- no es pot inserir directament; per efectuar-lo s'ha d'executar l'aplicatiu Gestor amb un joc de paràmetres determinats -indicats al capítol *Joc de proves*-. En un cas real, el mateix procés d'alta de nous registres s'encarregaria d'inserir el valor corresponent a aquests camps, fet el qual permetria canviar aquesta restricció de la Base de Dades.

Accés a les dades

Com ja s'ha comentat anteriorment, existirà una classe de control la qual s'encarregarà de gestionar l'accés a les dades, anomenada *gestorDades*. Aquesta classe serà instanciada i accedida únicament des de l'objecte Gestor.

Per flexibilitzar l'accés a la Base de Dades s'ha emprat un fitxer de configuració el qual emmagatzema tots els paràmetres necessaris per a efectuar la connexió amb aquesta.

CAPÍTOL 7 – Interfícies gràfiques

APARTAT 7.1 – Anàlisi

Introducció

Per a la creació de les interfícies gràfiques s'ha emprat Swing. Swing és una biblioteca gràfica per a Java que forma parteix de les Java Foundation Classes (JFC). Inclou widgets per a interfície gràfica d'usuari tals com a caixes de text, botons, desplegables i taules.

Les Internet Foundation Classes (IFC) eren una biblioteca gràfica per al llenguatge de programació Java desenvolupada originalment per Netscape i que es va publicar el 1996.

Des dels seus inicis l'entorn Java ja comptava amb una biblioteca de components gràfics coneguda com a AWT. Aquesta biblioteca estava concebuda com una API estandarditzada que permetia utilitzar els components natius de cada sistema operatiu. Llavors una aplicació Java corrent a Windows usaria el botó estàndard de Windows i una aplicació corrent en UNIX usaria el botó estàndard de Motif. A la pràctica aquesta tecnologia no va funcionar:

- ➔ En dependre fortament dels components natius del sistema operatiu el programador AWT estava confinat a un mínim denominador comú entre ells. És a dir que sol es disposen en AWT de les funcionalitats presents en tots els sistemes operatius.
- ➔ El comportament dels controls varia molt de sistema a sistema i es torna molt difícil construir aplicacions portables. Va ser per això que l'eslògan de Java "ho escrigui una vegada, l'executi en tots costats" va ser parodiat com "ho escrigui una vegada, ho provi en tots costats".

En canvi, els components d'IFC eren mostrats i controlats directament per codi Java independent de la plataforma. Dels Esmentats components es diu amb freqüència que són components lleugers, ja que no requereixen reservar recursos natius del sistema de finestres del sistema operatiu. A més en estar enterament desenvolupat a Java augmenta la seva portabilitat assegurant un comportament idèntic en diferents plataformes.

En 1997, Sun Microsystems i Netscape Communications Corporation van anunciar la seva intenció de combinar IFC amb altres tecnologies de les Java Foundation Classes. A més dels components lleugers subministrats originalment per l'IFC, Swing va introduir un mecanisme que permetia que l'aspecte de cada component d'una aplicació pogués canviar sense introduir canvis substancials en el codi de l'aplicació. La introducció de suport ensamblable per a l'aspecte va permetre a Swing emular l'aparença dels components natius mantenint els avantatges de la independència de la plataforma.

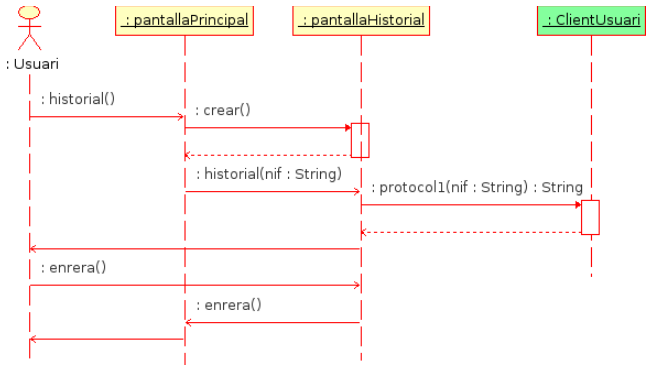
Disseny de la interfície d'usuari

A l'hora de realitzar el disseny s'ha volgut reutilitzar al màxim tots els components. Per aquesta raó s'ha afegit un mètode a la interfície RMI de l'usuari que permet conèixer el perfil d'aquest -si és metge o pacient-; d'aquesta manera només ha calgut fer una única interfície gràfica, habilitant en funció del tipus d'usuari unes o unes altres funcionalitats.

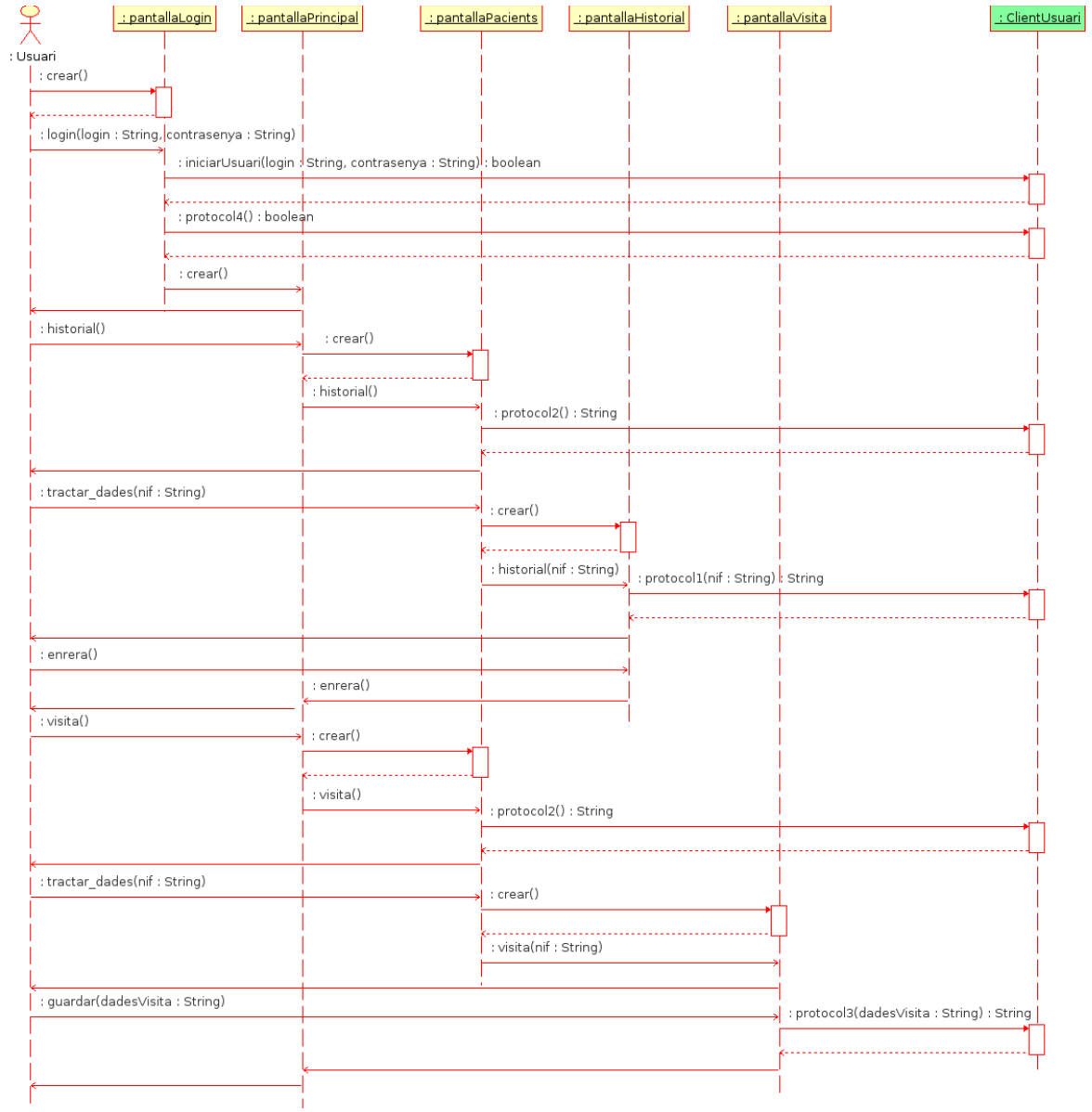
Diagrames de seqüència

Els diagrames de seqüències posteriors mostren el flux de les dades considerant les interfícies del sistema. S'hi ha representat -ressaltada d'un altre color- la classe *ClientUsuari*, corresponent al client RMI, el qual interactua amb la implementació RMI de l'usuari.

El diagrama corresponent al pacient mostra la seqüència de l'única operació (a banda del login) que pot realitzar, i que difereix lleugerament amb la mateixa operació del metge donat que no cal escollir el pacient:



Il·lustració 11: Diagrama de seqüències - pacient



Il·lustració 12: Diagrama de seqüències - metge

APARTAT 7.2 – Aplicatiu de l'usuari

S'ha elaborat un únic aplicatiu pels dos possibles perfils d'usuari, reutilitzant el codi i reduint la complexitat del sistema. Per a tal efecte, s'ha hagut d'implementar una funció RMI la qual en una primera instància no era necessària, i que ens permet conèixer el tipus d'usuari que ha iniciat el programa.

L'única diferència entre la interfície del metge i la del pacient és les operacions que poden realitzar l'un l'altre: el primer pot inserir les dades d'una visita i consultar l'historial dels

pacients que té assignats, mentre que el darrer només li és permès consultar el seu propi historial.

Val a dir que no s'ha fet un disseny de la interfície d'acord al que demanaria el mercat, donat que les dades que mostra són molt bàsiques -no apareix el nom de l'usuari, ni es poden consultar les seves dades, etc.-. Cal recordar que no és aquest l'objectiu principal d'aquest PFC, per la qual cosa s'ha preferit no ampliar innecessàriament la seva complexitat.

Al capítol següent corresponent al joc de proves es podran veure unes captures de pantalla del procés d'execució per part primer d'un metge i posteriorment d'un pacient, on es podrà apreciar la diferència existent entre ambdues interfícies -encara que a priori el programa sigui el mateix-.

APARTAT 7.3 – Aplicatiu del gestor

El gestor com a tal no necessita cap interfície, donat que només respon a les peticions que l'usuari li fa. En un estadi més elaborat es podrien gestionar els missatges que es generen durant la comunicació RMI per poder analitzar possibles problemes i corregir-los apropiadament, però tampoc és aquest l'objectiu que es persegueix.

CAPÍTOL 8 – Conclusions

APARTAT 8.1 – Repàs general

Com a resultat final d'aquest projecte, s'ha implementat un sistema remot el qual gestiona de manera segura els historials mèdics dels pacients a través d'una xarxa de comunicacions convencional. Aquesta fita s'ha pogut aconseguir gràcies, en part, al mètode de treball utilitzat que ha permès concloure cada fase individualment i aprofitar-la per a la següent.

A continuació es valorarà cada una d'aquestes fases i es demostrarà que tot allò que es va demanar a l'enunciat d'aquest PFC s'ha assolit de forma satisfactòria.

Esquema criptogràfic

S'ha implementat l'esquema criptogràfic demanat, compost pels 4 protocols necessaris per a garantir les necessitats de seguretat d'un historial mèdic que viatja per una xarxa de comunicacions exposada a tot tipus d'accés il·lícit.

El protocol d'autenticació garanteix la identitat de l'usuari que vol accedir a les dades de l'historial mèdic. Aquest protocol és emprat a l'inici de l'execució -fase de login-.

El protocol de consulta de la llista dels pacients assignats a un metge permet a aquests garantir que només ells podran accedir a aquesta informació, assolint així el requisit de confidencialitat de les dades.

El protocol d'inserció de dades a l'historial del pacient assegura la integritat d'aquestes, donat que només el metge pot efectuar aquesta tasca.

Finalment, el protocol de consulta de les dades de l'historial d'un pacient permet assolir el requisit d'autenticitat de la informació, la qual cosa significa que si s'ha validat correctament aquest protocol es pot assegurar que les dades les quals s'està consultant han estat inserides realment per qui diu que ho ha fet.

Representació de les dades: XML

El sistema emprat per a representar la informació que passa d'una aplicació a una altra -de client a servidor o a l'inrevés- es basa en l'estàndard XML. L'estructura definida per a cada un dels fitxers necessaris ha estat validada per la DTD creada a l'efecte, la qual cosa indica que aquests fitxers estan ben formats. Per assegurar aquest punt s'ha fet servir una eina

anomenada *XMLStarlet* la qual verificava si realment el fitxer XML acomplia l'esquema indicat al fitxer de definició del tipus de document (DTD).

Comunicació de components: RMI

La comunicació entre les diferents aplicacions està gestionada pel protocol estàndard RMI, la qual cosa era una fita important dins la feina demanada en aquest PFC. La generació dels *skeletons* i dels *stubs* fa possible que les diferents parts que composen aquest sistema en comuniquin entre elles i es passin informació l'una a l'altra, informació en format XML segons s'ha pogut veure al punt anterior.

Gestió de la informació: Base de Dades

Com la immensa majoria d'aplicacions existents, la informació implicada en aquest sistema cal guardar-la en una base de dades i gestionar-la mitjançant alguna entitat de control. L'estudi de les necessitats del sistema ha portat a la generació de les taules dins el MySQL i a la creació d'una classe de control per tal de realitzar els accessos a aquesta informació a través d'un únic punt.

Interfícies gràfiques

Pel que fa a les interfícies gràfiques s'ha utilitzat la llibreria Swing de Java, una extensió del AWT (Kit d'Eines de Manegament Abstracte de Finestres) incorporada en Java 2, i no s'ha fet servir cap eina visual per a crear les diferents pantalles. S'ha buscat la facilitat d'ús més que no pas una interfície real, però l'experiència ha estat molt gratificant alhora que intensa. Les eines actuals cert és que faciliten la feina, però abstreuen molts aspectes interns del llenguatge que són força interessants de conèixer.

APARTAT 8.2 – Opinió personal

Durant tot el temps en què ha durat aquest PFC he pogut treballar diferents aspectes estudiats al llarg d'aquest segon cicle d'Enginyeria Informàtica. He anat aplicant els meus coneixements sobre les distintes matèries, aprofundint en aquells apartats en què era necessari arribar més lluny del que en un principi havia après, la qual cosa m'ha reportat un increment notable en els meus coneixements.

Fase a fase he anat analitzant la problemàtica plantejada i resolent-la sovint amb l'ajuda del material de les assignatures cursades, permetent-me refrescar la memòria. També m'he recolzat bastant en la informació obtinguda en Internet, sobretot pel que fa referència al llenguatge Java i a l'ús de les diferents classes emprades en aquest projecte: *java.rmi*, *org.dom*, *java.security*, etc.

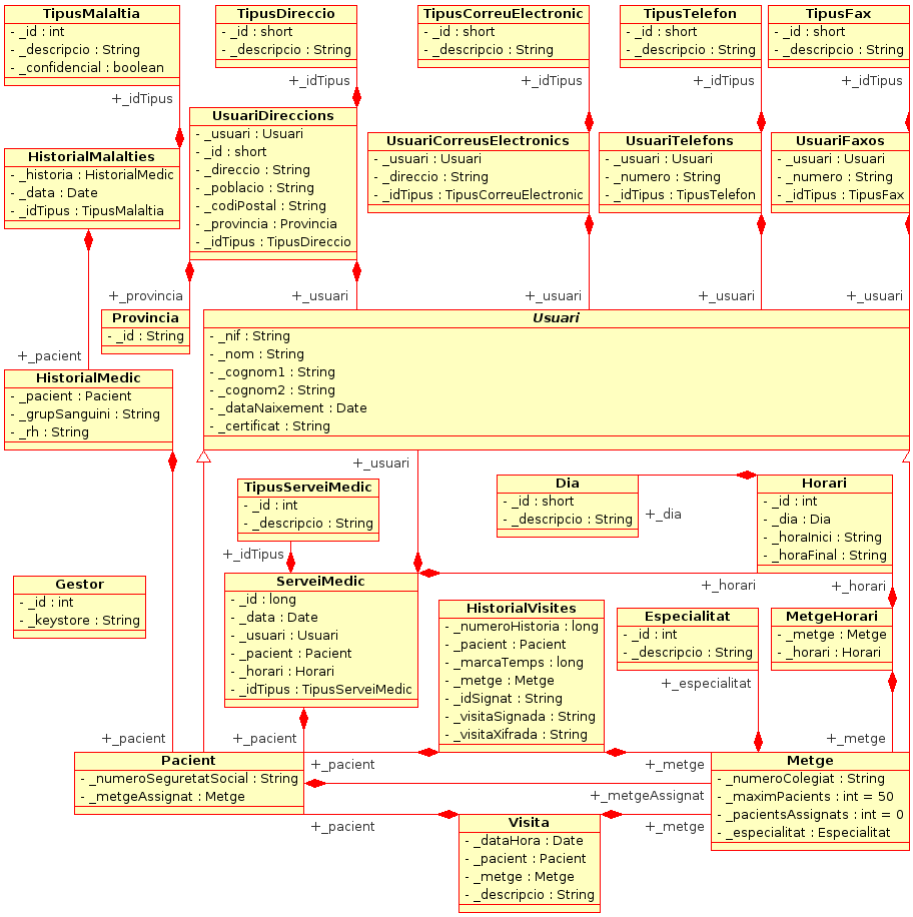
Crec que aquest PFC ha fet la funció pel qual està pensat: aprofundir en els coneixements adquirits durant la carrera i anar encara més enllà, enriquint-me aquests.

Tot i així, la realització d'aquest projecte m'ha portat molta més feina del que en un principi havia previst, la qual cosa ha repercutit negativament en la meva vida privada, havent de robar hores a la son i de sacrificar-me -i sacrificar la meva dona i la meva filla- molts caps de setmana i festes de guardar. Com a mostra, un botó: si aquests dies algú pregunta a la meva filla per mi li dirà que *està a l'ordinador*.

APARTAT 8.3 – Millores futures

Aquest projecte dóna peu a múltiples i variades possibles ampliacions i/o millores, d'entre les quals es podrien seleccionar les següents:

- Gestionar usuaris: alta, baixa i modificació dels usuaris al sistema. L'administrador podrà donar d'alta els metges, i aquests podran fer el mateix amb els seus pacients. Els metges podran donar-los de baixa de la seva llista de pacients, però no desapareixerà del sistema -simplement deixarà d'estar assignat a cap metge-.
- Demanar hora de visita o d'un determinat servei mèdic -radiografia, revisió ginecològica, etc-: els pacients podran demanar hora de visita amb el seu metge; el sistema proposarà una i l'usuari podrà acceptar-la o rebutjar-la si no està d'acord.
- Consultar serveis concertats: els pacients podran consultar quins serveis i quines hores de visita té concertades, i podrà anul·lar-los i/o demanar aplaçar-los per més endavant; el sistema, en aquest cas, haurà de proposar una nova hora de visita.



Il·lustració 13: Diagrama ampliat de les classes d'anàlisi

A tall personal, en un principi havia realitzat un disseny del sistema molt complet i ric -i possiblement millorable-, allunyat del que es demanava realment. La manca de temps em va fer desistir d'aquest disseny i centrar-me en el més indispensable, però el diagrama de les classes d'anàlisi ja el tenia fet, tal i com es pot veure a la il·lustració anterior.

CAPÍTOL 9 - Joc de proves

APARTAT 9.1 – Introducció

Per a la realització de les proves unitàries s'ha creat un joc de proves amb les dades necessàries per a efectuar aquestes.

Fins arribar a la fase de la Base de Dades, les proves s'efectuaven amb missatges imbricats directament al codi, la qual cosa va permetre validar l'execució dels protocols de seguretat, el posterior intercanvi de dades en format XML i la comunicació mitjançant RMI entre les dues aplicacions.

Un cop efectuat l'anàlisi disseny de la persistència de les dades, es van crear diversos fitxers per a efectuar les proves: un script per generar les taules de la Base de Dades i un altre per inserir les dades fictícies.

Donat que en les primeres proves sempre apareixen problemes, es van crear també els respectius scripts de buidat de les taules i d'eliminació d'aquestes, per d'aquesta manera començar a treballar sempre amb un entorn de dades conegut.

APARTAT 9.2 – Instal·lació

Per executar els aplicatius d'aquest PFC cal tenir instal·lat el JDK de Java i el gestor de Base de Dades MySQL. L'entorn de treball de proves compta amb la versió 1.5.0_9 del JDK de Sun i el MySQL versió 5.0.22; el sistema operatiu és la distribució de Linux Ubuntu en la seva versió 6.06.

La manera en què s'han executat els scripts al MySQL ha estat la tècnica de copiar i enganxar el text. Primer cal crear la Base de Dades amb la comanda *CREATE DATABASE <nom_de_la_base_de_dades>*. Un cop fet aquest pas, cal situar-se en aquesta amb la comanda *USE <nom_de_la_base_de_dades>*. Si a continuació s'executa la comanda *SHOW TABLES* el MySQL retornarà la llista de taules actuals existents en aquesta base da dades -la qual cosa, la primera vegada, hauria de retornar el missatge *0 rows* indicant que no hi ha cap taula-. En aquest punt es copia el script de creació de les taules i s'executa, obtenint d'aquesta forma les taules necessàries per a poder treballar. Per a no començar de zero, cal inserir uns registres de prova. Aquesta tasca es realitza copiant el script d'inserció de dades a les taules.

Com a darrer pas cal afegir els certificats dels usuaris i el keystore del gestor als registres corresponents. Per a tal efecte es va crear una opció al programa del Gestor la qual permet realitzar aquest pas, d'altra banda necessari doncs no s'ha implementat l'alta d'usuaris. La sintaxi que cal emprar és:

`java Gestor 0 <contrasenya_gestor>` on *<contrasenya_gestor>* és ***uoc0506***

Si tot ha anat correctament, ja es pot començar a efectuar les proves dels aplicatius.

APARTAT 9.3 – Funcionament

Per executar el sistema primer cal registrar els servidors RMI de comunicació. Això es farà amb el script ***servidors***, executant la següent instrucció:

```
bash servidors start
```

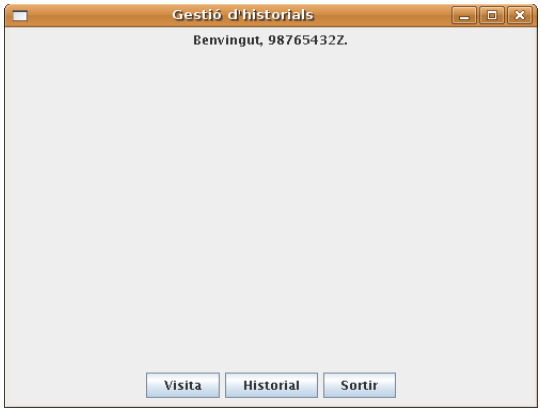
El contingut del script es pot veure a l'apartat corresponent dels annexos. Un cop iniciats els servidors, ja es pot iniciar l'aplicatiu de l'usuari amb la comanda:

```
java IUsuari
```

La pantalla que apareix és la següent:

Il·lustració 14: Pantalla de login

A continuació s'introdueix l'identificador (nif) de l'usuari i la seva contrasenya i el sistema intenta validar-lo. En cas negatiu, es retorna un missatge d'error i es permet tornar a intentar-ho. En aquest exemple el nif serà ***98765432Z*** i la contrasenya ***uoc0506***. Aquests són els valors identificatius d'un metge. les opcions que es mostren a la pantalla que li apareix a l'usuari són **Visita** -per inserir les dades d'una visita-, **Historial** -per consultar el seu historial- i **Sortir** -per finalitzar l'execució del programa-.

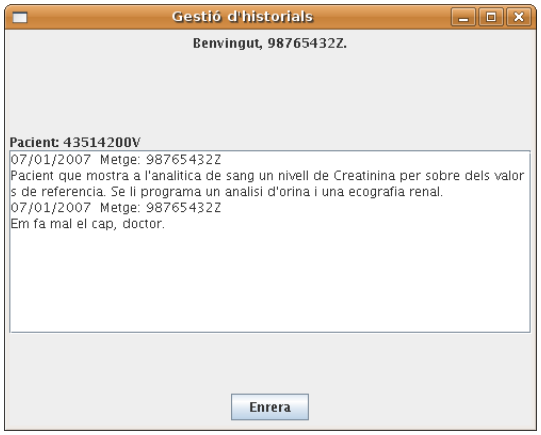


Il·lustració 15: Pantalla principal del metge

Qualsevol de les dues accions que escolleixi l'usuari necessitarà de l'elecció del pacient sobre el qual operar. En la interfície del metge es podria haver mostrat la llista dels seus pacients assignats en la pantalla principal, però s'ha deixat com a pas posterior per ressaltar més clarament l'execució dels protocols. L'usuari ara escollirà revisar l'historial d'un pacient, per la qual cosa després de pitjar el botó corresponent la pantalla que li apareix és la següent:

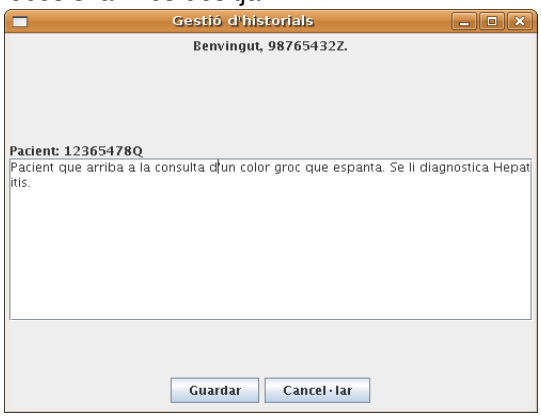
Il·lustració 16: Pantalla dels pacients del metge

S'escull el darrer pacient d'aquesta llista i s'obté el següent resultat:



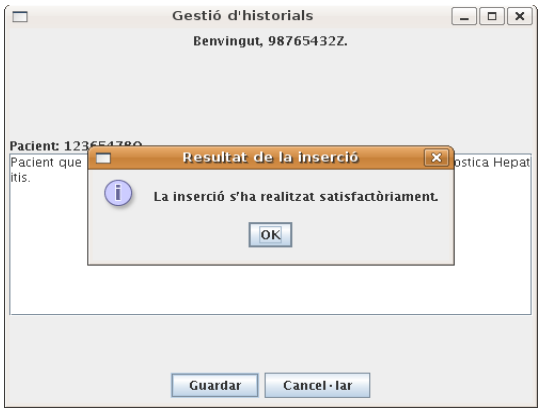
Il·lustració 17: Pantalla de l'historial del pacient

Pitjant el botó **Enrera** es torna a la pàgina principal del metge. Aquí ara es podrà escollir inserir les dades d'una visita, per la qual cosa es pitja el botó **Visita** i s'accedeix novament a la llista dels pacients assignats al metge -l'actual usuari-. Seleccionant l'altre usuari s'arriba a la pantalla on es podrà escriure la descripció de la visita mèdica i guardar-la o cancel·lar aquest procés si així es desitja.



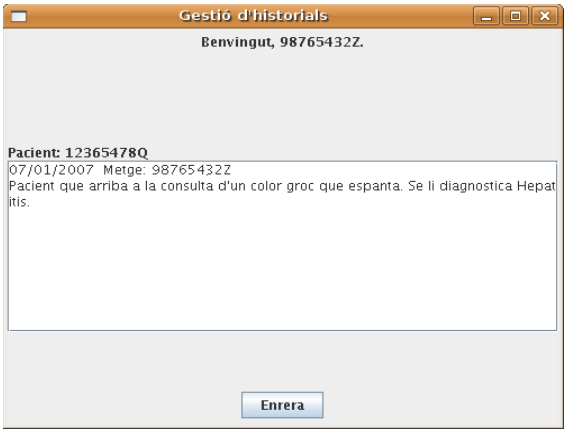
Il·lustració 18: Pantalla d'inserció de visita

Un cop introduïdes les dades, es pot pitjar el botó **Guardar** i, si tot ha anat correctament, el sistema retorna un missatge de confirmació -feedback- a l'usuari i es retorna novament a la pantalla principal.



Il·lustració 19: Missatge de confirmació

Si es torna a consultar l'historial d'aquest mateix pacient es pot comprovar com efectivament hi ha inserida una nova visita.



Il·lustració 20: Pantalla de revisió de la inserció

Per acabar, es tornarà a la pantalla principal amb el botó **Enrera** i es pitjarà el botó **Sortir**, finalitzant l'execució del programa. A continuació es tornarà a escriure a la còmsola la comanda per iniciar el programa:

```
java IUsuari
```

Ara però s'introduirà ***43514200V*** com a login i ***uoc0506*** com a contrasenya. Donat que es tracta d'un pacient, la pantalla que apareix tot seguit és una mica diferent a la del metge: només té els botons per consultar historial i per sortir:

Il·lustració 21: Pantalla principal del pacient

Si s'escull visualitzar l'historial, el resultat obtingut és el següent:

Il·lustració 22: Pantalla de l'historial propi

L'usuari no pot fer cap operació més, i només li resta sortir del programa.

Cal tenir en compte

És necessari comprovar que els diversos fitxers de configuració tenen els valors vàlids. D'una banda, s'ha de comprovar que els valors de la base de dades i l'usuari i la contrasenya d'accés existents al fitxer *configuracioBaseDades.xml* són correctes. També cal assegurar que els fitxers locals dels usuaris es troben a la carpeta que indica el fitxer *configuracioKeystores.xml* (el camí que s'indica sempre és relatiu al directori de treball on s'executa l'aplicatiu *IUsuari*). Com a pas final, s'han de repassar els paràmetres relatius a a comunicació RMI confirmant que els valors del fitxer *configuracioRMI.xml* es corresponen als indicats al script ***servidors***.

CAPÍTOL 10 – Annexos

APARTAT 10.1 – Fitxers adjunts

Codi font

Adjunt a aquesta memòria hi ha el codi font de tots els fitxers que composen els aplicatius creats en aquest PFC, així com les classes i els keystores emprats. A continuació es mostra la llista d'aquests fitxers font i una breu descripció de cada un d'ells:

- Base64Coder.java

  Codifica i/o descodifica en Base64 els caràcters no imprimibles que es generen durant el procés de xifrat i signat de les dades.

- Certificat.java

  Emmagatzema les dades referents als certificats dels usuaris i del gestor, des de les claus privada i pública fins el propi certificat i els seus atributs, un dels quals ens permet diferenciar el perfil de l'usuari (metge o pacient).

- ClientUsuari.java

  És el programa del costat del client en la comunicació remota. Inicia la comunicació i efectua les crides a les funcions proveïdes per les interfícies de servidor.

- configuracioBaseDades.java

  Petit programa que s'ocupa de recuperar els paràmetres de configuració corresponents a la Base de Dades.

- configuracioKeystores.java

  Ídem pel que fa a la localització física dels fitxers magatzems de claus i certificats. El camí que s'hi indica és relatiu al de l'executable.

- configuracioRMI.java

  Ídem pel que fa a la comunicació remota mitjançant RMI.

- Especialitat.java

  Classe que implementa l'objecte del mateix nom, i que representa les possibles especialitats mèdiques dels diferents metges. Aquests darrers tenen un camp que fa referència a aquesta taula.

- fitxerXML.java

  Gestiona l'accés als fitxers XML dels protocols. Construeix i recupera els fitxers i permet accedir als seus elements.

- gestorDades.java

  S'encarrega d'accedir a la Base de Dades. Guarda i recupera la informació d'aquesta. Només s'instancia des de l'objecte Gestor.

- Gestor.java

  Centre del sistema, gestiona els accessos segurs als historials i en permet als usuaris el seu ús.

- HistorialVisites.java

  És el codi de l'objecte historial pròpiament dit.

- ImplementacioGestor.java

  És l'objecte que implementa la funcionalitat proveïda per l'objecte Gestor; es correspon a l'objecte remot, al servidor.

- ImplementacioUsuari.java

  Ídem pel que fa a l'objecte Usuari. Remarcar que en aquest codi s'han unificat els objectes Usuari, Metge i Pacient, facilitant la seva gestió i reutilitzant codi.

- InterficieGestor.java

  Defineix la interfície que conté la funcionalitat proveïda per l'objecte Gestor.

- InterficieUsuari.java

  Ídem pel que fa a l'objecte Usuari.

- IUsuari.java

  Codi de la interfície gràfica, el qual interactua amb l'objecte ClientUsuari per a respondre les demandes dels usuaris.

- Metge.java

  Classe corresponent a l'objecte Metge, subclasse de Usuari.

- Pacient.java

  Ídem pel que fa a la subclasse Pacient. En aquest PFC no té cap mètode propi.

- Seguretat.java

  Gestiona totes les funcionalitats criptogràfiques i de seguretat: el xifrat i desxifrat, la signatura i la seva validació, i la generació de números aleatoris segurs.

- ServidorGestor.java

  Programa que exportar l'objecte Gestor i el posa disponible a la xarxa, per tal que un client s'hi connecti i invoqui els seus mètodes mètodes.

- ServidorUsuari.java

  Ídem pel que a a l'objecte Usuari.

- Usuari.java

  Superclasse que agrupa els objectes Metge i Pacient. Classe principal del sistema, donat que ella és qui inicia la comunicació remota per respondre a la petició d'un servei.

- Visita.java

  Gestiona les dades corresponents a la visita del pacient, en el pas entre la recuperació d'aquesta informació des de la interfície gràfica fins a la seva inserció a la base de dades.

Script ***servidors***

El script que inicia els servidors RMI del Gestor i de l'Usuari té la següent sintaxi:

```
#!/bin/bash
case "$1" in start)

rmic -classpath . -d . ImplementacioGestor
rmic -classpath . -d . ImplementacioUsuari
rmiregistry 2006 &
java ServidorGestor uoc0506 &
java ServidorUsuari &
```

```
echo "Servers up"
ps

;;
stop)

echo "Stopping servers"

pid1=`ps ax | grep "java ServidorGestor" | head -c 5`
pid2=`ps ax | grep "java ServidorUsuari" | head -c 5`

kill -9  $pid1
kill -9  $pid2

killall rmiregistry
ps

;;
*)

echo "Usage: "$0" start|stop"

esac

exit 0
```

Fitxers de configuració

Existeixen tres fitxers de configuració, els quals guarden la informació següent:

- *configuracioKeystore.xml*: directori on es troba el keystore local de l'usuari
- *configuracioBaseDades.xml*: paràmetres de connexió amb la base de dades:
  - Nom de la base de dades
  - Usuari autoritzat
  - Contrasenya (en clar)
- *configuracioRMI.xml*: paràmetres de connexió amb el servidor RMI:
  - Nom del host
  - Número de port

Els fitxers estan en format XML, i per a cada un d'aquests s'ha creat un fitxer DTD que els valida. A continuació es pot veure el contingut de cada un d'aquests fitxers, juntament amb la seva definició de tipus de document:

- configuracioKeystores.dtd

```
<!-- J.Cesar Rodriguez (jrodriguezle@uoc.edu) PFC - UOC 2006-07/01 -->
<!-- Esquema DTD del fitxer XML del directori dels keystores locals dels
                usuaris -->
<!ELEMENT directoriKeystores (#PCDATA)>
```

- configuracioKeystores.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!-- J.Cesar Rodriguez (jrodriguezle@uoc.edu) PFC - UOC 2006-07/01 -->
<!DOCTYPE directoriKeystores SYSTEM "configuracioKeystores.dtd">
<directoriKeystores>../keystores</directoriKeystores>
```

- configuracioBaseDades.dtd

```
<!-- J.Cesar Rodriguez (jrodriguezle@uoc.edu) PFC - UOC 2006-07/01 -->
<!-- Esquema DTD del fitxer XML dels parametres de configuracio de
        connexio amb la base de dades -->
<!ELEMENT baseDades (mySQL)>
<!-- Elements de baseDades:
        mySQL : obligatori
                conte els elements usuari i contrasenya
                te l'atribut nom
-->
        <!ELEMENT mySQL (usuari, contrasenya)>
<!-- Elements de mySQL:
        usuari      : obligatori
        contrasenya  : obligatori
-->
        <!ATTLIST mySQL nom CDATA #REQUIRED>
<!-- Atributs de mySQL:
        nom         : obligatori
-->
        <!ELEMENT usuari (#PCDATA)>
        <!ELEMENT contrasenya (#PCDATA)>
```

- configuracioBaseDades.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!-- J.Cesar Rodriguez (jrodriguezle@uoc.edu) PFC - UOC 2006-07/01 -->
<!DOCTYPE baseDades SYSTEM "configuracioBaseDades.dtd">
<baseDades>
        <mySQL nom = "PFC">
                <usuari>jcesar</usuari>
                <contrasenya>sacchetti</contrasenya>
        </mySQL>
</baseDades>
```

- configuracioRMI.dtd

```
<!-- J.Cesar Rodriguez (jrodriguezle@uoc.edu) PFC - UOC 2006-07/01 -->
<!-- Esquema DTD del fitxer XML de configuracio dels parametres RMI -->
<!ELEMENT rmi (host, port)>
<!-- Elements de rmi:
        host  : obligatori
```

```
        port  : obligatori
-->
        <!ELEMENT host (#PCDATA)>
        <!ELEMENT port (#PCDATA)>
```

- configuracioRMI.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!-- J.Cesar Rodriguez (jrodriguezle@uoc.edu) PFC - UOC 2006-07/01 -->
<!DOCTYPE rmi SYSTEM "configuracioRMI.dtd">
<rmi>
        <host>localhost</host>
        <port>2006</port>
</rmi>
```

De la mateixa manera, existeix una classe de control per a cada un d'aquests fitxers la qual permet recuperar el seu contingut.

Fitxer DTD

L'estructura del fitxer DTD que valida els fitxers XML d'intercanvi de dades durant l'execució dels protocols resta com segueix:

```
<!-- J.Cesar Rodriguez (jrodriguezle@uoc.edu) PFC - UOC 2006-07/01 -->
<!ELEMENT documentSignat (document, signatura)>
<!-- Elements de documentSignat:
        document  : obligatori
                conte els elements titol, validacio, accio, llistaPacients i historial
        signatura : obligatori
-->
        <!ELEMENT document (titol, validacio?, accio?, llistaPacients?, historial?)>
<!-- Elements de document:
        titol         : obligatori
        validacio     : opcional
                conte els elements aleatoriUsuari, aleatoriGestor, identificador
        accio         : opcional
                conte els elements aleatoriGestor, identificador, dadesVisita i
                visitaSignada
                te l'atribut valor
        llistaPacients : opcional
                conte els elements pacientsDeMetge i pacientsDeMetgeSignat
        historial     : opcional
                conte els elements identificador i visites
-->
                <!ELEMENT titol (#PCDATA)>
                <!ELEMENT comentari (#PCDATA)>
                <!ELEMENT validacio (aleatoriUsuari?, aleatoriGestor?,
                                        identificador?)>
```

```
<!-- Elements de validacio:
        aleatoriUsuari : opcional
        aleatoriGestor : opcional
        identificador  : opcional
-->
                <!ELEMENT accio (aleatoriGestor, identificador?, dadesVisita?,
                                        visitaSignada?)>
<!-- Elements d'accio:
        aleatoriGestor : obligatori
        identificador  : opcional
        dadesVisita    : opcional
        visitaSignada  : opcional
-->
                <!ATTLIST accio valor CDATA #REQUIRED>
<!-- Atributs d'accio:
        valor : obligatori
-->
                <!ELEMENT llistaPacients (pacientsDeMetge,
                                                pacientsDeMetgeSignat)>
<!-- Elements de llistaPacients:
        pacientsDeMetge      : obligatori
        pacientsDeMetgeSignat : obligatori
-->
                <!ELEMENT historial (identificador, visites)>
<!-- Elements d'historial:
        identificador : obligatori
        visites       : obligatori
                conte l'element visita
-->
<!-- Elements de visites:
        visita : 0 o mes
                conte els elements numeroHistoria, data, metge, idSignat,
                visitaSignada i visitaXifrada
-->
                <!ELEMENT visita (numeroHistoria, data, metge, idSignat,
                                        visitaSignada, visitaXifrada)>
<!-- Elements de visita:
        numeroHistoria : obligatori
        data           : obligatori
        metge          : obligatori
        idSignat       : obligatori
        visitaSignada  : obligatori
        visitaXifrada  : obligatori
-->
                <!ELEMENT aleatoriGestor (#PCDATA)>
                <!ELEMENT aleatoriUsuari (#PCDATA)>
                <!ELEMENT dadesVisita (#PCDATA)>
                <!ELEMENT data (#PCDATA)>
                <!ELEMENT identificador (#PCDATA)>
                <!ELEMENT idSignat (#PCDATA)>
```

```
                <!ELEMENT metge (#PCDATA)>
                <!ELEMENT numeroHistoria (#PCDATA)>
                <!ELEMENT pacientsDeMetge (#PCDATA)>
                <!ELEMENT pacientsDeMetgeSignat (#PCDATA)>
                <!ELEMENT visitaSignada (#PCDATA)>
                <!ELEMENT visitaXifrada (#PCDATA)>
        <!ELEMENT signatura (#PCDATA)>
```

Fitxers XML

Els protocols es comuniquen durant la seva execució mitjançant fitxers XML, les estructures de les quals es mostren a continuació:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DocumentSignat SYSTEM "./esquema.dtd">
<DocumentSignat>
        <Document>
                <Titol>Protocol 1, pas 1</Titol>
                <Comentari />
                <validacio>
                        <aleatoriUsuari>valor</aleatoriUsuari>
                        <identificador>valor</identificador>
                </validacio>
        </Document>
        <Signatura>signatura</Signatura>
</DocumentSignat>

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DocumentSignat SYSTEM "./esquema.dtd">
<DocumentSignat>
        <Document>
                <Titol>Protocol 1, pas 2</Titol>
                <Comentari />
                <validacio>
                        <aleatoriUsuari>valor</aleatoriUsuari>
                        <aleatoriGestor>valor</aleatoriGestor>
                        <identificador>valor</identificador>
                </validacio>
        </Document>
        <Signatura>signatura</Signatura>
</DocumentSignat>

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DocumentSignat SYSTEM "./esquema.dtd">
<DocumentSignat>
        <Document>
                <Titol>Protocol 1, pas 3</Titol>
                <Comentari />
                <accio valor=valor_accio>
```

```
                        <aleatoriGestor>valor</aleatoriGestor>
                        <identificador>valor</identificador>
                </accio>
        </Document>
        <Signatura>signatura</Signatura>
</DocumentSignat>

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DocumentSignat SYSTEM "./esquema.dtd">
<DocumentSignat>
        <Document>
                <Titol>Protocol 1, pas 4</Titol>
                <Comentari />
                <historial>
                        <identificador>valor</identificador>
                        <visites>
                                <visita>
                                        <numeroHistoria>valor</numeroHistoria>
                                        <data>valor</data>
                                        <metge>valor</metge>
                                        <idSignat>valor</idSignat>
                                        <visitaSignada>valor</visitaSignada>
                                        <visitaXifrada>valor</visitaXifrada>
                                </visita>
                        </visites>
                </historial>
        </Document>
        <Signatura>signatura</Signatura>
</DocumentSignat>

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DocumentSignat SYSTEM "./esquema.dtd">
<DocumentSignat>
        <Document>
                <Titol>Protocol 2, pas 3</Titol>
                <Comentari />
                <accio valor=valor_accio>
                        <aleatoriGestor>valor</aleatoriGestor>
                </accio>
        </Document>
        <Signatura>signatura</Signatura>
</DocumentSignat>

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DocumentSignat SYSTEM "./esquema.dtd">
<DocumentSignat>
        <Document>
                <Titol>Protocol 2, pas 4</Titol>
                <Comentari />
                <llistaPacients>
```

```
                        <pacientsDeMetge>valor</pacientsDeMetge>
                        <pacientsDeMetgeSignat>valor</pacientsDeMetgeSignat>
                    </llistaPacients>
            </Document>
            <Signatura>signatura</Signatura>
    </DocumentSignat>

    <?xml version="1.0" encoding="UTF-8"?>
    <!DOCTYPE DocumentSignat SYSTEM "./esquema.dtd">
    <DocumentSignat>
            <Document>
                    <Titol>Protocol 3, pas 3</Titol>
                    <Comentari />
                    <accio valor=valor_accio>
                            <aleatoriGestor>valor</aleatoriGestor>
                            <visita>valor</visita>
                            <visitaSignada>valor</visitaSignada>
                    </accio>
            </Document>
            <Signatura>signatura</Signatura>
    </DocumentSignat>

    <?xml version="1.0" encoding="UTF-8"?>
    <!DOCTYPE DocumentSignat SYSTEM "./esquema.dtd">
    <DocumentSignat>
            <Document>
                    <Titol>Protocol 4, pas 3</Titol>
                    <Comentari />
                    <validacio>
                            <aleatoriGestor>valor</aleatoriGestor>
                    </validacio>
            </Document>
            <Signatura>signatura</Signatura>
    </DocumentSignat>
```

APARTAT 10.2 – Glossari de termes

ANSI: l'Institut Nacional Nord-Americà d'Estàndards (ANSI, per les seves sigles en anglès: American National Standards Institute) és una organització sense ànim de lucre que supervisa el desenvolupament d'estàndards per a productes, serveis, processos i sistemes als Estats Units. ANSI és membre de l'Organització Internacional per a l'Estandardització (ISO) i de la Comissió Electrotécnica Internacional (International Electrotechnical Commission, IEC). L'organització també coordina estàndards del país nord-americà amb estàndards internacionals, de manera que els productes de l'esmentat país puguin usar-se a tot el món.
Aquesta organització aprova estàndards que s'obtenen com a fruit del desenvolupament de temptatives d'estàndards per part d'altres organitzacions, agències governamentals, companyies i altres entitats. Aquests estàndards

asseguren que les característiques i les prestacions dels productes són consistents, és a dir, que la gent usi els esmentats productes en els mateixos termes i que aquesta categoria de productes es vegi afectada per les mateixes proves de validesa i qualitat.
ANSI acredita a organitzacions que realitzen certificacions de productes o de personal d'acord amb els requisits definits en els estàndards internacionals. Els programes d'acreditació ANSI es regeixen d'acord a directrius internacionals quant a la verificació governamental i a la revisió de les validacions.

Bouncy Castle: el paquet criptogràfic Bouncy Castle és un implementació en Java d'algorismes criptogràfics, de codi obert i desenvolupada per l'anomenada Legion of the Bouncy Castle.

CORBA: en computació, CORBA (Common Object Request Broker Architecture - arquitectura comuna de mitjancers en peticions a objectes), és un estàndard que estableix una plataforma de desenvolupament de sistemes distribuïts facilitant la invocació de mètodes remots sota un paradigma orientat a objectes.
CORBA va ser definit i està controlat per l'Object Management Group (OMG) que defineix les APIs, el protocol de comunicacions i els mecanismes necessaris per permetre la interoperabilitat entre diferents aplicacions escrites en diferents llenguatges i executades en diferents plataformes, la qual cosa és fonamental en computació distribuïda.
En un sentit general CORBA "embolica" el codi escrit en un altre llenguatge en un paquet que conté informació addicional sobre les capacitats del codi que conté, i sobre com cridar als seus mètodes. Els objectes que resulten poden llavors ser invocats des d'un altre programa (o objecte CORBA) des de la xarxa. En aquest sentit CORBA es pot considerar com un format de documentació llegible per la màquina, similar a un arxiu de capçaleres però amb més informació.
CORBA utilitza un llenguatge de definició d'interfícies (IDL) per especificar els interfícies amb els serveis que els objectes oferiran. CORBA pot especificar a partir d'aquest IDL la interfície a un llenguatge determinat, descrivint com els tipus de dada CORBA han de ser utilitzats en les implementacions del client i del servidor. Implementacions estàndard existeixen per a Ada, C, C++, Smalltalk, Java i Python. Hi ha també implementacions per a Perl i TCL.
En compilar una interfície en IDL es genera codi per al client i el servidor (l'implementador de l'objecte). El codi del client serveix per poder realitzar les crides a mètodes remots. És el conegut com a stub, el qual inclou un proxy (representant) de l'objecte remot en el costat del client. El codi generat per al servidor consisteix en uns skeletons (esquelets) que el desenvolupador ha d'omplir per implementar els mètodes de l'objecte.
CORBA és més que una especificació multiplataforma, també defineix serveis habitualment necessaris com a seguretat i transaccions.

DTD: sigles de Document Type Definition. La definició de tipus de document (DTD) és una descripció d'estructura i sintaxi d'un document XML o SGML. La seva funció bàsica és la descripció del format de dades, per usar un format comú i mantenir la consistència entre tots els documents que utilitzin la mateixa DTD. D'aquesta forma, els esmentats documents, poden ser validats, coneixen l'estructura dels elements i la descripció de les dades que implica cada document, i poden a més compartir la mateixa descripció i forma de validació dins d'un grup de treball que

usa el mateix tipus d'informació. La DTD és una definició, en un document SGML o l'XML, que especifica restriccions en l'estructura i sintaxi del mateix. La DTD es pot incloure dins de l'arxiu del document, però normalment s'emmagatzema en un fitxer ASCII de text separat. La sintaxi de les DTDs per a SGML i l'XML és similar però no idèntica. La definició d'una DTD especifica la sintaxi d'una aplicació de SGML o XML, que pot ser un estàndard àmpliament utilitzat com a XHTML o una aplicació local.

GNU GPL: la GNU GPL (General Public License o llicència pública general) és una llicència creada per la Free Software Foundation a mitjan els 80, i està orientada principalment a protegir la lliure distribució, modificació i ús de programari. El seu propòsit és declarar que el programari cobert per aquesta llicència és programari lliure i protegir-lo d'intents d'apropiació que restringeixin aquestes llibertats als usuaris.

Existeixen diverses llicències "germanes" de la GPL, com la llicència de documentació lliure GNU (GFDL) que cobreix els articles de la Wikipedia, l'Open Audio License, per a treballs musicals, etcètera, i altres menys restrictives, com la MGPL, o la LGPL (Lesser General Public License o Library General Public License), que permeten l'enllaç dinàmic d'aplicacions lliures a aplicacions no lliures.

IAIK: és un conjunt d'APIs i implementacions de funcionalitats criptogràfiques, incloent-hi funcions hash, codis d'autenticació de missatge, simètric, asimètric, fluxos i gestió de claus, certificats i encriptació de bloc. Complementa la funcionalitat de seguretat del JDK estàndard.

ISO: l'Organització Internacional per a l'Estandardització (ISO) és una organització internacional no governamental, composta per representants dels organismes de normalització (ONs) nacionals, que produeix normes internacionals industrials i comercials. Les esmentades normes es coneixen com|com a normes ISO i la seva finalitat és la coordinació de les normes nacionals, d'acord amb l'Acta Final de l'Organització Mundial del Comerç, per tal de facilitar el comerç, facilitar l'intercanvi d'informació i contribuir amb uns estàndards comuns per al desenvolupament i transferència de tecnologies. ISO no és un acrònim; prové del grec *iso*, que significa igual. És un error comú el pensar que ISO significa International Standards Organization, o una cosa similar; en anglès el seu nom és International Organization for Standardization, mentre que en francès es denomina Organisation Internationale de Normalisation; l'ús de l'acrònim conduiria a noms diferents: IOS en anglès i OIN en francès, pel que els fundadors de l'organització van elegir ISO com la forma curta i universal del seu nom.

Motif: és una llibreria per a la creació d'entorns gràfics sota X Window System en sistemes Unix. Motif és també un estàndard de la indústria sota el codi IEEE 1295.

Motif és utilitzat com a infraestructura de l'ambient d'escriptori CDE.

Molts desenvolupadors argumenten que s'ha tornat obsolet en comparació amb altres llibreries com a GTK o Qt, però continua sent utilitzat per diversos sistemes antics.

SOAP: sigles de Simple Object Access Protocol, SOAP és un protocol estàndard creat per Microsoft, IBM i d'altres, està actualment sota l'auspici de la W3C que defineix com dos objectes en diferents processos poden comunicar-se per mitjà d'intercanvi de dades XML. SOAP és un dels protocols utilitzats en els serveis web.
A diferència de DCOM i CORBA, que són binaris, SOAP usa el codi font en l'XML. Això és un avantatge ja que facilita la seva lectura per part d'humans, però també és un inconvenient ja que els missatges resultants són més llargs. L'intercanvi de missatges es realitza mitjançant tecnologia de components. El terme Object en el nom significa que s'adhereix al paradigma de la programació orientada a objectes.
SOAP és un marc extensible i descentralitzat que permet treballar sobre múltiples piles de protocols de xarxes informàtiques. Els procediments de crides remotes poden ser modelats en la forma de diversos missatges SOAP interactuant entre si.
SOAP funciona sobre qualsevol protocol d'Internet, generalment HTTP, que és l'únic homologat pel W3C. SOAP té com a base XML, amb un disseny, que compleix el patró Capçalera-Desenvolupament de disseny de programari, com molts altres dissenys, verbigràcia HTML. La capçalera Header és opcional i conté metadades sobre enrutamiento (routing), seguretat o transaccions. El desenvolupament Body conté la informació principal, que es coneix com a càrrega útil (payload). La càrrega útil s'acull a un XML Schema propi.

W3C: el World Wide Web Consortium, W3C abreujat, és un consorci internacional que produeix estàndards per a la World Wide Web (o Teranyina Mundial). Està dirigida per Tim Berners-Lee, el creador original d'URL (Uniform Resource Locator, Localitzador Uniforme de Recursos), HTTP (HyperText Transfer Protocol, Protocol de Transferència d'HiperText) i HTML (Llenguatge de Marcat d'HiperText) que són les principals tecnologies sobre el que es basa la Web. Creada l'1 d'octubre de 1994 pel mateix Tim Berners-Lee en el MIT, actual seu central del consorci. Unint-se posteriorment l'abril de 1995 INRIA a França, reemplaçat per l'ERCIM en el 2003 com a l'hoste europeu del consorci i Universitat de Keio (Shonan Fujisawa Campus) al Japó el setembre de 1996 com hoste asiàtic. Aquests organismes administren el consorci que està integrat per:

- ➔ Membres del W3C. A setembre de 2006 comptava amb 417 membres (llista completa)
- ➔ Equip W3C (W3C Team) 65 investigadors i experts a través del món (Directori)
- ➔ Oficines W3C (W3C Offices). Centres regionals establerts a Alemanya i Àustria (oficina cojunta), Austràlia, Benelux (oficina conjunta), Xina, Corea del Sud, Espanya, Finlàndia, Grècia, Hong Kong, Hongria, Índia, Israel, Itàlia, Marroc, Suècia i Regne Unit i Irlanda (oficina conjunta) (Oficines W3C)

L'oficina espanyola del W3C, establerta en el 2003, aquesta allotjada per la Fundació CTIC al Parc Científic Tecnològic de Gijón (Principat d'Astúries).

XMLStarlet: és un conjunt d'utilitats de línia d'ordres que es poden fer servir per transformar, interrogar, validar i editar documents i fitxers XML utilitzant un joc reduït de comandes de manera similar com ho fan amb fitxers de text plans les comandes UNIX grep, sed, awk, diff, patch, join, etc.

XSchema: és un llenguatge d'esquema utilitzat per descriure l'estructura i les restriccions dels continguts dels documents l'XML d'una forma molt precisa, més enllà de les normes sintàctiques imposades pel propi llenguatge XML. S'aconsegueix així, una percepció del tipus de document amb un nivell alt d'abstracció. El World Wide Web Consortium (W3C) va començar a treballar a XML Schema el 1998. La primera versió es va convertir en una recomanació oficial el maig de 2001. Una segona edició revisada està disponible des d'octubre de 2004.

APARTAT 10.3 – Bibliografia

Universitat Oberta de Catalunya. (2004). *Arquitectura de sistemes distribuïts: XP03/11068/02253* (2a ed.). Barcelona: Fundació per a la Universitat Oberta de Catalunya.

Universitat Oberta de Catalunya. (2001). *Enginyeria del programari III: XP01/11003/00021* (1a ed.). Barcelona: Fundació per a la Universitat Oberta de Catalunya.

Universitat Oberta de Catalunya. (2004). *Seguretat en xarxes de computadors: XP03/05070/02091* (1a ed.). Barcelona: Fundació per a la Universitat Oberta de Catalunya.

Universitat Oberta de Catalunya. (2002). *Comerç electrònic: XP01/11034/01037* (1a ed.). Barcelona: Fundació per a la Universitat Oberta de Catalunya.

Universitat Oberta de Catalunya. (1999). *Criptografia: P1/05024S* (1a ed.). Barcelona: RBA Realizaciones Editoriales, S.L..

Ceballos, Fco. Javier. (2000). *Java 2: Curso de programación* (1a ed.). Madrid: RA-MA Editorial.

Lemay, L., Cadenhead, R. (1999). *Aprendiendo Java 2 en 21 días*. México: Prentice-Hall.

Universitat de Girona. (2006). Com citar documents. Recuperat el 29 de novembre de 2006, des de http://biblioteca.udg.es/serveis/guies/Cites/Com_citar.asp.

Sun Microsystems, Inc. (2006). *Java 2 Platform Standard Edition 5.0: API Specification*. Recuperat entre el setembre i el desembre de 2006, des de http://java.sun.com/j2se/1.5.0/docs/api/index.html.

Hunter, J., McLaughlin, B. (2004). *JDOM v1.0: API Specification*. Recuperat entre el novembre i el desembre de 2006, des de http://www.jdom.org/docs/apidocs/index.html.

Wikimedia Foundation, Inc. (2006). *Wikipedia: la enciclopedia libre*. Recuperat el desembre de 2006, des de http://es.wikipedia.org/wiki/Portada.