

Banc Electrònic d'Historials Clínic

Oliver Hernández Valls

Enginyeria Informàtica
Àrea de seguretat

Consultor: Jordi Castellà Roca

En agraïment a:

El meu tutor Jordi Castellà Roca
pel seu suport al llarg de tot el projecte.

La Sònia que, des de la distància i des del meu costat,
m'ha recolzat en tot moment.

La meva germana Ingird,
que m'ha sabut fer entendre la naturalesa d'un historial clínic,
alhora que me n'ha explicat els seus detalls tècnics.

Resum

El Banc Electrònic d'Historials Mèdics és un aplicatiu basat en l'arquitectura client-servidor que permet l'intercanvi segur de la informació continguda en els historials clínics dels pacients de la salut. Aquesta seguretat es garanteix mitjançant la criptografia de clau pública, tan per aconseguir la confidencialitat de les dades com per autenticar clients i servidor.

Es tracta d'un primer desenvolupament experimental on s'han implementat les funcionalitats bàsiques, corresponents a metges i a pacients, de la gestió d'historials clínics: l'alta, la consulta i l'ampliació. Per la normativa actual vigent, no es permet cap modificació que impliqui eliminar res del que ja s'hagi afegit a l'historial. D'altra banda, s'ha considerat l'eliminació d'historials del banc com una tasca de manteniment que no han de desenvolupar ni metges ni pacients, sinó el personal administrador del sistema.

Índex de contingut

I. Introducció.....	1
Punt de partida i aportació.....	1
Objectius.....	3
Enfocament i mètode seguit.....	4
Planificació del projecte.....	5
Productes obtinguts.....	6
Breu descripció dels altres capítols.....	7
Anàlisi.....	7
Seguretat.....	7
Disseny.....	7
Implementació.....	7
Configuració del programari.....	7
Manual.....	7
Treball Futur.....	7
Conclusions.....	7
II. Anàlisi.....	8
Documentació de requisits.....	8
Informació inicial: l'historial clínic.....	8
Objectius.....	8
Requisits legals.....	8
Característiques.....	9
Actors del sistema.....	10
Casos d'ús.....	11
Registre d'usuaris.....	11
Iniciar sessió en el sistema / identificació mútua entre usuari i sistema.....	11
Consulta docent d'un expedient.....	12
Inserció de dades a l'historial mèdic.....	12
Acabar sessió en el sistema.....	13
Gestió de la informació.....	14
Requisits de seguretat.....	16
III. Seguretat.....	18
Conceptes Generals.....	18
Autenticació.....	18
Autenticació del missatge.....	18
Autenticació de l'emissor i del receptor.....	19
Confidencialitat.....	19
Integritat.....	19
No repudi.....	19
Solucions de seguretat en el nostre sistema.....	21
IV. Disseny.....	24
Format dels missatges.....	24
Descripció dels protocols.....	26
Usuari.....	26
Gestor del Banc d'Historials.....	26
Usuari.....	26
Gestor del Banc d'Historials.....	26
Registre d'usuaris.....	26
Administrador.....	27
Gestor Banc d'Historials.....	27
Inserció d'un Historial.....	27

Docent.....	27
Gestor del Banc d'Historials.....	27
Inserció d'un Cas d'Ús.....	27
Docent.....	27
Gestor del Banc d'Historials.....	27
Inserció d'una Anotació.....	28
Docent.....	28
Gestor del Banc d'Historials.....	28
Consulta de l'Historial.....	28
Usuari de l'Historial.....	28
Gestor del Banc d'Historials.....	28
Usuari de l'Historial.....	28
Gestor del Banc d'Historials.....	28
Acabament de Sessió.....	29
Gestor del Banc d'Historials.....	29
Tecnologies utilitzades.....	30
Descomposició del programari en paquets.....	31
Diagrames de Col·laboració.....	33
Comunicació client-servidor.....	33
Diagrames de classes.....	35
Accions Client.....	35
Autenticadors.....	36
Criptografia.....	37
Persistència de les dades.....	39
V. Implementació.....	41
Interfície gràfica d'usuari.....	41
Tasques d'execució automàtica.....	42
VI. Configuració del programari.....	43
Sistema Operatiu.....	43
Entorn de desenvolupament.....	43
Documentació.....	44
Bastides.....	44
Instal·lació IAIK.....	44
Public Key Infrastructure.....	46
Configuració del Banc Electrònic d'Historials Clínics.....	46
SmartCard, i pkcs12 del gestor.....	46
Fitxers de configuració del Banc Electrònic d'Historials Clínics.....	47
Fitxers de configuració d'Hibernate.....	48
Creació de la base de dades.....	49
VII. Manual.....	51
Aplicatiu Client.....	51
Aplicatiu Servidor.....	52
VIII. Treball Futur.....	53
Corregir la comunicació per socket.....	53
Afegir actors al sistema.....	53
Ampliar el contingut de l'Historial Clínic.....	53
Validar Certificats.....	54
Auditories al sistema.....	54
IX. Conclusions.....	55
X. Glossari.....	57
XI. Bibliografia.....	58

I. Introducció

Punt de partida i aportació

Actualment, les tecnologies de la informació faciliten el treball en molts dels camps on es manega informació. És habitual ajudar-se d'aquestes tecnologies¹ per processar informació, per emmagatzemar-la, per transmetre-la i intercanviar-la, etc.

De totes maneres, hi ha alguns àmbits on les TI no s'aprofiten com caldria. És el cas de la sanitat, on la gestió de la informació és més crític que en altres àmbits, i on les TI són pràcticament inexistents. Una raó a la manca de TI en la sanitat, podria ser la privacitat de la informació, és a dir, l'obligació de protegir la informació sobre els pacients. Però les TI ja fa anys que disposen de sistemes segurs de maneig de dades. Fins i tot, un sector tan conservador com el de les notaries, on la informació que utilitzen també és molt sensible a la privacitat, s'ha vist obligat a informatitzar els seus arxius d'escriptures. I ho ha fet sense cap problemàtica.

L'objectiu d'aquest projecte, però, no és entendre les raons que fan que el sistema mèdic actual estigui encara basat en documents de paper. L'objectiu és aconseguir que la informació que forma part de l'historial de cada pacient no quedi dispersa pels CAPs, hospitals, clíniques i metges privats; o sigui, que qualsevol metge des de qualsevol punt del món on hi hagi una connexió a Internet i amb el programari adequat pugui consultar l'historial mèdic del pacient que està tractant en aquell moment de forma automàtica i instantània.

És evident que el projecte que s'està plantejant és molt ambiciós. En aquest projecte de final de carrera s'intentarà abordar només una part del que seria aquest super projecte. La part que serà objecte d'aquest estudi serà la seguretat en les comunicacions entre usuaris i dispensadors d'informació, i més concretament s'estudiarà la forma de garantir que la informació que s'intercanvia només la reben les parts autoritzades per fer-ho.

Per aconseguir-ho, s'ha desenvolupat un Banc Electrònic d'Historials Mèdics. Aquest banc és l'encarregat de concentrar la informació dels pacients. El banc permet crear historials, consultar-los i ampliar-los. Val a dir, que el banc el poden utilitzar tan pacients com metges, però els únics amb capacitat per crear i ampliar aquests historials són els metges.

A més a més, l'ús del Banc Electrònic d'Historials Mèdics implica l'actualització instantània dels historials, és a dir, que tan bon punt es faci una modificació a un historial aquesta modificació quedarà reflectida a l'historial i disponible a les següents consultes que es facin.

Aquest Banc Electrònic d'Historials Mèdics², utilitza la criptografia de clau pública per tal de garantir la confidencialitat de les dades. D'aquesta manera, els usuaris estan segurs que es connecten al BEHC, i el BEHC està segur que les peticions que rep

1 D'ara endavant TI.

2 D'ara endavant BEHC.

procedeixen d'usuaris autoritzats.

Una vegada finalitzat el projecte s'ha d'haver assolit un programari, entenent per programari tot el conjunt de documents, fitxers, gràfics, processos, etc.

Objectius

La informació continguda en els historials clínics està restringida per una sèrie de lleis que condicionen de manera severa l'accés a les dades de cada historial. Així doncs no es tracta d'un simple programari de gestió remota de dades. Perquè el programari sigui utilitzable, a la pràctica, haurà de garantir totes les normes referents als historials clínics.

Per tant els objectius d'aquest projecte són dos.

Primerament, el que volem és facilitar la gestió per part dels metges, i la consulta per part dels pacients, dels historials clínics des de qualsevol dispositiu que es pugui connectar a la xarxa Internet; sempre respectant les lleis referents als historials clínics.

En segon lloc, cal que la comunicació entre l'usuari i el banc d'historials sigui segura. És a dir que ha d'existir l'autenticació tan de l'usuari com del banc d'historials; mantenint sempre la confidencialitat de les dades transmeses.

Per assolir el primer objectiu, s'ha desenvolupat un aplicatiu basat en l'arquitectura client-servidor. Amb aquest tipus d'arquitectura pot haver-hi un número indeterminat de clients que fan peticions a un servidor que és el que 'centralitza' les dades i segons l'usuari l'hi permet o l'hi restringeix l'accés.

El segon objectiu es garanteix gràcies a l'ús de la criptografia de clau pública. Amb la criptografia de clau pública, podem garantir la confidencialitat a través del xifrat de les dades que enviem. També a través de la criptografia de clau pública podem garantir l'autenticació i la no revocació de les dades.

Enfocament i mètode seguit

A l'hora de desenvolupar aquest programari s'ha fet sempre pensant, per sobre de tot, en un producte de qualitat. Per aconseguir-ho, s'ha seguit una metodologia àmpliament provada i acceptada dins l'enginyeria del programari: es tracta del Rational Unified Process. Concretament, a la finalització d'aquest projecte no s'haurà arribat ni a la fi d'una primera iteració d'aquest procés. És a dir, s'haurà fet un anàlisi, un disseny, una implementació i un testeig funcional; però encara mancarà el testeig (per part d'usuaris reals) de la usabilitat de la interfície gràfica d'usuari i el rendiment a gran escala.

L'anàlisi ha consistit en la documentació de requisits de l'objecte central del projecte, l'historial clínic. Per aconseguir-ho s'ha entrevistat a dues persones relacionades amb l'àmbit sanitari, una infermera i un psicòleg.

Després de l'anàlisi, en el disseny i la implementació, s'ha seguit el model iteratiu de començar per peces petites de programari i anar-hi afegint funcionalitat de mica en mica. Finalment s'han unit les diferents parts del programari per formar l'aplicatiu BEHC.

Gràcies a que les diferents parts del programari s'han desenvolupat separatament s'ha pogut desenvolupar cada part amb independència de les altres. Això ha permès que el treball dedicat a cada component s'hagi fet de forma asíncrona en el temps.

D'altra banda alhora que el programari ha anat creixent també ho ha fet la documentació. Evidentment el disseny que s'ha hagut de fer per cada component s'ha fet sempre en un format reaprofitable per poder afegir-lo a la documentació. Així els gràfics que no s'han generat directament amb l'OpenOffice.org s'han fet amb eines que permeten exportar els gràfics com a imatges.

De cares a la documentació i al manteniment o ampliació de l'aplicatiu s'ha comentat al màxim el codi font. Juntament amb aquesta memòria s'hi pot trobar tota la documentació del codi font en format HTML³.

³Documentació generada per l'eina javadoc.

Planificació del projecte

El projecte s'ha planificat aprofitant, al màxim, el temps disponible. S'ha volgut aconseguir un desenvolupament de qualitat per damunt de l'obtenció d'una aplicació molt ampla però poc fiable. Consta de moltes fites per tal de tenir constància en tot moment de l'evolució del projecte i per facilitar-ne el seguiment.

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

- (1) [•] *Disseny dels protocols de seguretat.*
Una setmana, del tretze de Març fins al dinou de Març.
- (2) [•] *Anàlisi de l'aplicatiu.*
Una setmana, del vint de Març fins al vint-i-sis de Març.
- (3) [•] *Disseny de l'aplicatiu.*
Dues setmanes, del vint-i-set de Març fins al nou d'Abril.
- (4) [•] *Implementació dels protocols de seguretat.*
Dues setmanes, del deu d'Abril fins al vint-i-tres d'Abril.
- (5) [•] *Instal·lació del programari.*
Una setmana, del vint-i-quatre d'Abril fins al trenta d'Abril.
- (6) [•] *Comunicació remota amb Web Services (WS), serveis en xarxa.*
Una setmana, de l'u de Maig fins al set de Maig.
- (7) [•] *Disseny de les interfícies gràfiques.*
Una setmana, del vuit de Maig al catorze de Maig.
- (8) [•] *Implementació de l'aplicació.*
Dues setmanes, del quinze de Maig fins al vint-i-vuit de Maig.
- (9) [•] *Implementació de les interfícies.*
Dues setmanes, del vint-i-nou de Maig fins a l'onze de Juny.
- (10) [•] *Preparació de la presentació i integració de la documentació.*
Una setmana, del dotze de Juny fins al divuit de Juny.
- (11) [•] *Entrega.*
Dinou de Juny.

Productes obtinguts

Al finalitzar aquest projecte hem obtingut dos productes: el programari client i el programari servidor.

El programari client s'haurà d'executar des de les estacions de treball dels usuaris que vulguin fer ús del Banc Electrònic d'Historials Clínics. Aquest programari serà utilitzable per ambdós usuaris: metges i pacients. L'objectiu que té és permetre als usuaris interactuar amb el programari, comunicar-se amb el servidor d'històrics i gestionar les operacions criptogràfiques de seguretat.

El programari servidor s'executarà a la màquina servidora de les dades. Aquest programari haurà d'anar acompanyat d'un Sistema de Gestió de Bases de Dades que podrà variar de forma molt flexible gràcies a l'ús de la bastida, *framework*, *Hibernate*. En conjunt, el programari servidor s'encarregarà de rebre les peticions dels usuaris, filtrar-les segons si l'usuari que les fa té permisos per executar-les o no, dur a terme les operacions criptogràfiques necessàries per garantir la seguretat del sistema i garantir la persistència de les dades que gestiona.

Breu descripció dels altres capítols

Anàlisi

Documenta extensament la fase d'anàlisi del projecte. S'hi estudien la documentació de requisits, l'anàlisi dels actors del sistema i dels casos d'ús que aquests poden dur a terme.

Seguretat

Introdueix al lector en els conceptes de seguretat. Tot seguit continua centrant-se en els que influeixen directament a aquest projecte i en els atacs que pot patir l'aplicació. Per acabar donant-hi una proposta de solució.

Disseny

Detalla les solucions de disseny que s'han donat als diferents components del programari, així com també explica el funcionament d'aquests components. S'hi comenten les decisions arquitectòniques que s'han pres i es justifica l'ús de les diferents llibreries de les que s'ajuda el sistema.

Implementació

S'explica quines mesures s'han pres per aconseguir una interfície el màxim de clara i productiva possible, justificant cada decisió a través de diferents criteris d'usabilitat.

També es comenta l'script d'Ant utilitzat per generar automàticament tot l'arbre de directoris de l'aplicació.

Configuració del programari

Enumera i comenta el programari utilitzat per desenvolupar el BEHC. Alhora detalla com cal instal·lar les diferents llibreries que s'utilitzen i com cal configurar-les.

Tot seguit hi ha les explicacions de la instal·lació del BEHC, tan de la part client com de la part servidora, i de la seva configuració.

Finalment hi ha una referència a com s'ha de crear la base de dades en el SGBD escollit per efectuar la persistència de l'aplicació, MySql.

Manual

Com el propi nom indica, en aquest capítol s'hi fa una explicació del funcionament de l'aplicatiu client i de l'aplicatiu gestor.

Treball Futur

Aquest capítol és una guia de les possibles millores que s'han detectat al llarg del projecte. És clar que no són les úniques millores que es poden fer al projecte, senzillament són les que s'haguessin fet si hi hagués hagut més temps disponible.

Conclusions

A partir del producte final del projecte s'analitza l'assoliment d'objectius, la funcionalitat aconseguida i l'aportació que fa al sistema sanitari.

II. Anàlisi

Documentació de requisits

Informació inicial: l'historial clínic

Un historial mèdic és un document confidencial amb valor legal que conté el conjunt d'esdeveniments clínics que fan referència a una determinada persona.

Objectius

Els objectius d'un historial mèdic són:

- Avaluar l'estat de la persona.
- Ajudar a determinar els objectius i les prioritats dels tractaments que es fan.
- Seleccionar (decidir) correctament el mètode terapèutic a seguir en cada cas.
- Obtenir una base de dades útil per a la recerca.
- Aportar rigor tècnic a l'actuació professional sanitària.

Requisits legals

Legislació de la Història Clínica a Espanya:

- Llei General de Sanitat (14/1986, de 25 d'Abril): El pacient té dret a que quedi constància per escrit de tot el seu procés, en una història clínica, així com a rebre un Informe d'Alta, o bé Informe de Consulta Externa, al finalitzar la seva estada hospitalària. La Història Clínica s'identifica amb un número únic per cada pacient i s'ha d'emmagatzemar de forma centralitzada en un sol lloc.
- Reial Decret 63/1995 (BOE 10-2-95): El pacient té dret a la comunicació o entrega, a petició de l'interessat, d'un exemplar de la seva història clínica o de determinades dades contingudes en la mateixa, sense perjudici de la obligació de la seva conservació en el centre sanitari. El Sistema Públic ha de preservar i garantir la confidencialitat de les dades contingudes dins la Història Clínica. El pacient té dret a la confidencialitat de tota la informació relacionada amb el seu procés i amb la seva estada en les institucions sanitàries públiques o privades que col·laboren amb el sistema públic. L'accés a la Història Clínica sense autorització, en perjudici d'un tercer, està tipificat com a delictes greu i està castigat amb penes de presó. Igualment el professional que reveli o divulgui dades de la història clínica serà castigat amb les mateixes penes. La Història Clínica ha de ser un reflex veraç del curs de la malaltia. Les dades incloses entre els seus documents no es poden alterar, falsejar, ni simular, la qual cosa constituiria un delictes de falsedat documental que està penat pel propi Cos Legal.
- LOPD: Llei Orgànica de Protecció de Dades de caràcter personal. Llei Orgànica 15/1999, de 13 de desembre.
- Llei 41/2002, del 14 de novembre, reguladora de l'autonomia del pacient i dels drets i

obligacions en matèria d'informació i documentació clínica.

Com s'acaba de veure hi ha un conjunt de requisits legals que cal complir respecte a l'historial mèdic. Aquests requisits són intrínsecs a l'historial, alhora que són directrius per les persones que hi tinguin accés. En resum, cal:

- Garantir el dret de l'usuari a accedir a la seva història clínica i a obtenir-ne dades.
- Respectar la intimitat, la dignitat humana i les conviccions del pacient.
- Fer-lo disponible per l'usuari les 24 hores del dia, els set dies de la setmana.
- Arxivar-lo almenys deu anys després de la mort del pacient.

Característiques

Una vegada vistos els objectius de l'historial clínic i la normativa que l'envolta, es passarà a veure com ha de ser una història clínica. A grans trets un historial clínic es pot caracteritzar per les següents propietats:

- La història mèdica ha de ser única per cada pacient.
- Ha de ser confidencial.
- S'ajusta a un model normalitzat.
- Les dades enregistrades han de ser objectives, intel·ligibles i el més complertes possibles.

I és justament per donar garantia als requisits d'intimitat, de confidencialitat i de legalitat que té sentit la perspectiva de seguretat d'aquesta aplicació.

Es pretén crear una aplicació segura per garantir la intimitat del pacient. S'utilitzarà la criptografia de clau pública com a eina per garantir la confidencialitat i el valor legal d'aquest document.

Però a més de la seguretat, també cal dissenyar una aplicació funcional. No se'n farà res de tenir una aplicació molt segura si després a la pràctica és poc útil.

En aquest punt s'introdueix el text de dos paràgrafs extrets de la pàgina en xarxa de l'hospital Nostra Senyora de Maritxell, on queda clara la complexitat que comporta la gestió de dades sanitàries:

“En un servei d'urgències, per la seva natura, es poden recollir gran volums d'informació. Aquesta informació ha de ser recollida, processada i emmagatzemada amb molta rapidesa, el que ens obliga a un maneig molt efectiu i selectiu de la mateixa. És per això que és fonamental no automatitzar processos mal dissenyats, que impliquin un important esforç amb poc aprofitament”.

“L'objectiu d'una gestió efectiva de la informació és poder disposar d'aquesta amb finalitats de registre, administratives, docents i de recerca. Qualsevol gestió d'informació que no compleixi aquests quatre objectius es pot considerar no efectiva, i per tant no és aconsellable”.

És doncs, tenint en compte aquestes recomanacions, que s'ha pensat en la funcionalitat bàsica que ha de tenir el sistema.

Actors del sistema

Inicialment, s'identifiquen tres tipus d'usuaris del sistema segons la tasca que hi desenvolupen:

- El pacient o usuari, que és la persona a la qual pertany l'historial mèdic.
- El personal sanitari, format pel conjunt de persones que han dut a terme alguna acció sanitària sobre el pacient.
- L'administrador del sistema, que és l'encarregat de controlar la configuració i el bon funcionament de l'aplicació.

Tot i aquesta classificació inicial, el personal sanitari és un grup heterogeni. És a dir, que no tot el personal sanitari interactuarà de la mateixa manera en el programari. Cal distingir entre els diferents sub-grups que formen aquest tipus d'actor per parar atenció al diferent ús que cadascú d'ells fa d'un historial mèdic.

La tasca d'un metge consisteix en recollir l'anamnesi, dur a terme les exploracions físiques, fer l'orientació diagnòstica, estipular un tractament i descriure el curs clínic.

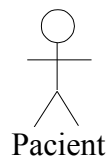
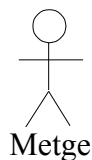
D'altra banda, el personal especialitzat (infermeria, fisioteràpia, etc.) també generarà documentació i s'incorporarà a l'àrea corresponent de l'expedient.

També es podria tenir en compte el personal administratiu, però en el nostre cas es considera que el personal administratiu no utilitzarà la nostra aplicació per a gestionar el calendari de visites. Es prefereix que aquest personal utilitzi alguna aplicació paral·lela per dos motius. En primer lloc és més segur separar físicament les dades de l'historial clínic de la gestió de visites, ja que, així s'aïlla al personal que té dret a consultar-lo per motius sanitaris. Aquest aïllament és bo, perquè disminueix els punts d'entrada a l'aplicació i això en millora la seguretat, ja que existiran menys llocs des d'on atacar al sistema. En segon lloc es té previst que la base de dades d'historials agafi unes dimensions molt grans i traient-ne la gestió de visites s'aconseguirà simplificar-la i reduir-la en gran mesura.

Donada la complexitat extra que suposa la implementació de la part corresponent als especialistes s'ha optat per deixar aquesta tasca com una possible ampliació en un treball futur.

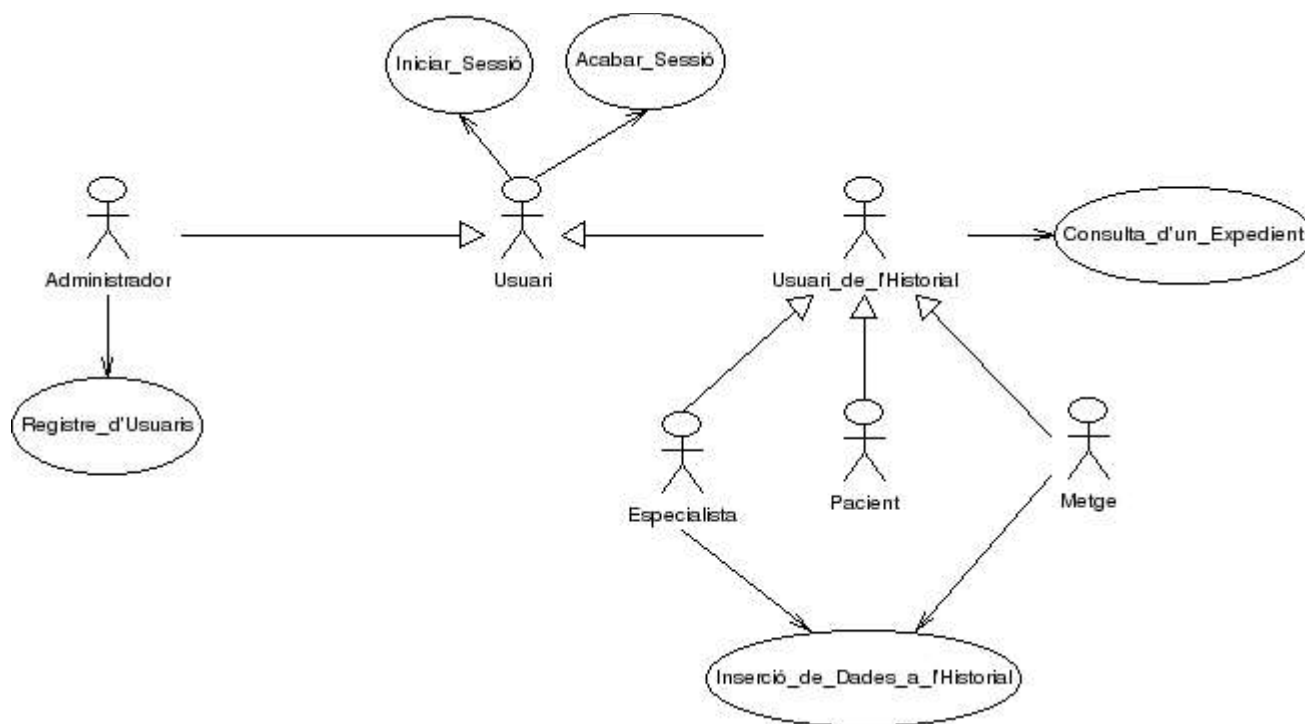
En resum hi haurà dos actors que interactuaran amb el nostre sistema:

- El propi **pacient**.
- El **metge** que l'atengui.



Casos d'ús

El conjunt d'actors anterior accedirà al sistema per desenvolupar-hi un conjunt de tasques propi. Vegem l'ús que cadascun d'aquests actors farà del sistema.



Registre d'usuaris

Cas d'ús: 1

Nom Cas d'ús: Registre d'usuaris.

Actors: Administrador del sistema.

Dades d'entrada: Dades de l'usuari que es vol donar d'alta.

Precondicions: La identitat de l'usuari que l'administrador dona d'alta s'haurà d'haver validat convenientment i l'administrador haurà d'haver iniciat una sessió.

Descripció: Perquè un usuari pugui fer ús del sistema caldrà que hi estigui registrat. El sistema necessita conèixer els usuaris que hi poden accedir per poder identificar als usuaris que hi volen obrir una sessió. L'administrador serà l'encarregat de donar d'alta als altres usuaris en el sistema.

Postcondicions: S'han introduït totes les dades necessàries perquè l'usuari interactiu lliurement amb el sistema.

Dades de sortida: ∅

Iniciar sessió en el sistema / identificació mútua entre usuari i sistema

Cas d'ús: 2

Nom Cas d'ús: Iniciar sessió.

Actors: Tots els usuaris dels sistema.

Dades d'entrada: ∅

Precondicions: ∅

Descripció: En aquest cas d'ús durem a terme dues tasques molt importants per la seguretat del sistema: la identificació i l'autenticació de l'usuari.

Qualsevol usuari que interactiu amb el sistema s'haurà d'obrir una sessió de forma segura (identificació+autenticació).

Passos: Donada la importància de saber qui està interactuant amb l'aplicació utilitzarem el protocol de Needham-Schroeder per fer l'autenticació.

Postcondicions: S'ha identificat l'usuari i s'ha demostrat que l'usuari és qui diu ser. Per tant s'haurà iniciat una sessió per aquest usuari.

Dades de sortida: ∅

Consulta docent d'un expedient

Cas d'ús: 3

Nom Cas d'ús: Consulta d'un expedient.

Actors: Metge, Especialista i Pacient.

Dades d'entrada: El sistema ja sap qui està fent la petició, però cal deixar constància de totes les persones que consulten l'expedient. Per tant l'entrada serà la petició de consulta signada.

Precondicions: L'usuari ha d'haver iniciat una sessió perquè estigui autenticat prèviament.

Descripció: Segons quina sigui la identitat de l'usuari tindrà associat un rol que li permetrà consultar l'historial o no, el rol també condiona si la consulta pot ser total o parcial. Els actors pacient, metge i especialista seran els únics que tindran accés a la consulta de l'expedient.

Passos:

- Validar la signatura de la petició.
- Desar la petició signada.
- Comprovar els permisos de consulta.
- Fer la consulta.
- Signar i xifrar l'historial.
- Retornar l'historial signat i xifrat.

Postcondicions: La petició de consulta signada es troba desada en el sistema.

Dades de sortida: Es retornaran les dades de l'historial mèdic del pacient sol·licitat.

Inserció de dades a l'historial mèdic

Cas d'ús: 4

Nom Cas d'ús: Inserció de dades.

Actors: Metge i Especialista.

Dades d'entrada: Agrupació d'afegits a l'historial signats individualment.

Precondicions: L'usuari s'ha d'haver autenticat prèviament en el sistema.

Descripció: Aquest cas d'ús ha de permetre l'ampliació de les dades de l'historial mèdic per part d'un metge o d'un especialista. Normalment l'usuari (metge o especialista) haurà consultat prèviament l'historial que vol modificar.

Passos:

-Validació de la signatura dels afegits.

-Desar els afegits signats.

-Actualitzar l'historial mèdic.

Postcondicions: El sistema ha desat els canvis a l'historial.

Dades de sortida: Rebut signat de les noves dades en el sistema.

Acabar sessió en el sistema

Cas d'ús: 5

Nom Cas d'ús: Acabament sessió.

Actors: Tots els usuaris.

Dades d'entrada: ∅

Precondicions: L'usuari s'ha d'haver autenticat prèviament en el sistema.

Descripció: Aquest cas d'ús consisteix en comunicar al sistema la fi del període d'utilització per part d'un usuari, és a dir donar per finalitzada la sessió. El tancament de la sessió s'assoleix de dues maneres. Després d'haver donat servei a una petició el sistema ha de donar per tancada la sessió, o bé si s'ha obert una sessió i l'usuari no ha fet cap petició en un període de temps determinat el servidor també ha de tancar la sessió per aquell usuari (timeout).

Passos:

-Al acabar l'execució d'un servei tancar la sessió.

-O bé, si s'exhaureix el temps d'espera de la petició també tancar la sessió.

Postcondicions: La sessió que hi havia oberta per un usuari es troba tancada o finalitzada.

Dades de sortida: ∅

Gestió de la informació

La peça central de la nostra aplicació és l'historial clínic. A grans trets l'historial clínic es divideix en tres parts: les dades personals, les dades mèdiques de les actuacions sobre el pacient i els annexos, que poden ser de molts tipus. D'aquestes tres parts les més denses són les dades mèdiques i els annexos. És obvi que les dades personals són poc canviants, tot al contrari que les dades mèdiques i els annexos, que reflecteixen l'evolució de l'estat de salut del pacient al llarg del temps. Així doncs aquests dos últims components de l'historial aniran creixent durant la vida del pacient.

Abans de continuar però, cal veure com és tècnicament un model d'història clínica:

- Anamnesi, que conté
 - les dades d'afiliació,
 - el document d'identitat,
 - nom i cognoms,
 - edat,
 - sexe,
 - lloc de naixement,
 - professió,
 - estat civil,
 - antecedents familiars,
 - antecedents personals,
 - antecedents patològics,
 - principal motiu de la consulta
- Exploració física.
- Orientació diagnòstica.
- Tractament.
- Curs Clínic.
- Documentació dels especialistes (infermeria, fisioteràpia, dietistes, ...).
- Annexes (radiografies, analítiques, autoritzacions d'operació, altes, ...).

En base a aquest model d'història es dissenya el nostre model abstracte. Els camps de dades personals són de mides estàndards, 25 caràcters pel nom i per cada cognom, l'edat és un numèric, el sexe un booleà, etc. En canvi la resta de camps són de longitud variable, sovint CLOBs i BLOBs per poder contenir fitxers de diferents tipus.

Tècnicament s'ha vist que l'anamnesi consta de les dades personals i de les dades de cada consulta. En el model de dades de l'aplicació es separarà l'anamnesi en dades personals i anamnesi en si mateixa. Aquesta anamnesi contindrà els resultats de l'interrogatori que el metge fa al pacient per determinar el motiu de la visita, simptomatologia, etc.

Així doncs l'historial ideal estarà format per nou parts:

- Dades personals.
- Antecedents.
- Anamnesis de les consultes.
- Exploracions físiques.
- Orientació Diagnòstica.
- Tractaments.
- Curs Clínic.
- Documentació dels especialistes.
- Annexes.

Llevat de les dades personals i els antecedents totes les altres parts hauran estat introduïdes des d'alguna especialitat mèdica, o més concretament, faran referència a una especialitat mèdica.

La gestió dels annexes no s'ha inclòs, de moment, en el nostre programari, i queda pendent com a treball futur.

Requisits de seguretat

Es començarà aquest apartat enumerant els requeriments generals de seguretat en la transmissió de missatges entre dos ordinadors, client i servidor. Concretament es té:

- autenticació de client i servidor,
- autenticació del missatge,
- confidencialitat i
- no repudi.

Tots aquests requeriments de seguretat són els que s'han de garantir per complir els requisits que ens suposa intercanviar informació d'un historial mèdic.

Cal parar atenció a que quan es fa referència a l'autenticació en realitat es vol dir identificació i autenticació. Aquests dos conceptes acostumen a estar sempre molt relacionats, però són dues coses diferents. Una part queda identificada quan l'altra part té coneixement de qui és la primera part, d'altra banda una part queda autenticada quan dona una prova de que s'hi pot confiar. Així habitualment es pot dir que la identificació es fa amb un usuari i que l'autenticació es fa amb una clau. Però això no té perquè ser sempre així. També es poden trobar sistemes on només calgui autenticació, com ara una caixa forta. En una caixa forta s'introdueix una clau per demostrar que es té permís per accedir al seu interior. És a dir es fa l'autenticació, però en cap moment cal identificar-se. En el sistema que s'està desenvolupant quan es parla d'autenticació es vol dir identificació més autenticació.

La necessitat d'identificació ve donada per l'essència legal de l'historial mèdic. Cal tenir constància de qui és l'autor de cada afegit de l'historial.

Aquesta aplicació però, té més requeriments de seguretat que els esmentats abans. En primer lloc es definiran els conjunt de requeriments de seguretat de què es disposa. Ja s'han comentat les diferències entre identificació i autenticació, es continuarà ara amb la resta de conceptes de seguretat que afecten a aquest programari

La *confidencialitat* consisteix en mantenir secretes les dades que es comuniquen per a tota persona que no sigui el destinatari d'aquestes dades. Aquest requeriment de seguretat es pot garantir o bé mitjançant tecnologies del tipus ssl o tls, o bé mitjançant el xifrat de clau pública de les dades.

L'*autenticitat* pot fer referència al missatge o a l'originador del missatge. L'autenticitat de l'originador demostra qui envia el missatge, aquesta autenticitat la garantirem fent que l'usuari hagi de tenir una sessió iniciada per tal de comunicar-se amb el nostre aplicatiu. D'altra banda també cal garantir l'autenticitat del missatge, és a dir cal poder identificar de forma inequívoca qui l'ha generat, aquesta autenticitat s'assegura mitjançant la firma digital de clau pública.

La *integritat* garanteix que les dades rebudes pel destinatari són exactament les mateixes que les enviades per l'originador d'aquestes. Els possibles atacs a la integritat de les dades poden ser de tres tipus: modificació de les dades, introducció de dades i eliminació de dades. La modificació de les dades consisteix en canviar el contingut de les dades normalment per donar lloc a un missatge amb un significat diferent a l'original. La introducció de dades intercepta el missatge i hi afegeix més dades que les originalment enviades. L'eliminació de dades en canvi intercepta el missatge i n'elimina parts per donar lloc a un nou missatge mancat d'algunes parts. La integritat del missatge es garanteix mitjançant la signatura amb clau pública del missatge.

Garantia de *no repudi*, que permet al receptor d'un missatge demostrar de qui ha rebut aquell missatge. O el que és el mateix, impedeix al remitent d'un missatge negar haver creat i/o enviat cert missatge. Altra vegada s'utilitzarà la signatura digital de la

criptografia de clau pública per garantir el no repudi.

Vegem els requeriments de cada cas d'ús.

- CU1 - *Registre d'usuaris*

El registre d'usuaris ens ha de permetre introduir usuaris nous en el sistema de forma segura. No en traiem res de donar la màxima seguretat en les comunicacions entre el nostre servei i els usuaris si després es poden afegir i treure usuaris del sistema de forma maliciosa. Per tant haurà de complir els requisits de confidencialitat, d'autenticitat, integritat i no repudi.

- CU2 – *Iniciar sessió en el sistema*

La sessió serà l'element que ens permetrà garantir l'autenticitat del remitent d'un missatge. Això requerirà autenticitat, integritat i no repudi. Opcionalment, però també molt recomanablement, es pot demanar confidencialitat per aquest cas d'ús, a més si s'utilitza ssl/tls es pot delegar aquesta tasca a la capa inferior a la nostra aplicació.

- CU3 – *Consulta docent d'un expedient*

En aquest cas d'ús l'expedient viatjarà per la xarxa de manera que caldrà garantir al 100% tots els requisits abans esmentats. Identificació+autenticació del demandant, confidencialitat, autenticitat i integritat de l'historial, i no repudi del consultant. La idea és tenir una base de dades amb totes les consultes que s'han fet a cada expedient.

- CU4 – *Inserció de dades en un historial*

Aquest és un altre cas d'ús crític. Cal garantir que la inserció de dades la fa un docent autoritzat i que no es modifica cap dada de les ja existents a l'expedient. Per això cal identificar i autenticar el remitent i el receptor, garantir la confidencialitat i integritat de les dades transmeses per la xarxa i el no repudi per part del remitent.

- CU5 – *Acabar sessió*

Aquest cas d'ús s'ha d'executar automàticament quan s'ha acabat d'executar una petició en el servidor. Això vol dir que una sessió només permet l'execució d'una única petició de la part client.

III. Seguretat

Conceptes Generals

Com ja s'ha comentat els requisits de seguretat d'aquest aplicatiu són: identificació, autenticació, confidencialitat, integritat i no repudi. Vegem detalladament cadascun d'aquests conceptes.

Autenticació

L'autenticació implica dos tipus de requeriments de seguretat. El primer requeriment és l'autenticació de la identitat del creador del missatge. L'altre requeriment és l'autenticació de les identitats del remitent i del destinatari del missatge.

En entorns on els elements són de baixa confiança, el creador dels missatges no és sempre el mateix que el que el remitent. Per exemple, una vegada una part maliciosa ha robat un missatge amb autenticació creat pel remitent, aquesta part maliciosa pot reenviar el missatge a qui vulgui, fent creure als destinataris que és la part que ha autenticat el missatge. Així doncs, la distinció entre l'autenticació del creador i l'autenticació del remitent sí que és important.

L'autenticació del missatge garanteix que el missatge tramès no s'ha modificat durant la transmissió i que no s'ha modificat la identitat del remitent, i no garanteix res sobre qui ha enviat el missatge.

L'autenticació d'emissor/receptor garanteix que l'emissor i el receptor realment són qui diuen ser. L'emissor pot voler confirmar la identitat del receptor al que envia el missatge i, anàlogament, el receptor pot voler confirmar la identitat del remitent del missatge, i això no garanteix res sobre qui ha creat aquest missatge.

Queda clar doncs.

Autenticació del missatge

Per garantir l'autenticació del missatge es poden utilitzar dues tecnologies:

- *El codi d'autenticació del missatge, message authentication code (MAC).*

Els mac es calculen a partir d'una clau secreta compartida per emissor i receptor, com que es calcula més ràpidament que una signatura digital és útil en les transmissions de volums molt grans d'informació. Però, com que es genera a partir d'una clau compartida, només garanteix que el missatge l'ha generat o bé l'emissor o bé el receptor, però no aclareix quin dels dos ho ha fet.

- *La signatura digital.*

Com que la signatura digital es basa en la criptografia de clau pública el normal és que sigui més costosa de calcular que els MAC. Com a contrapartida com que emissor i receptor no comparteixen cap clau podem saber del cert quin dels dos és el creador d'un determinat missatge.

En el nostre sistema necessitem tenir constància de quina de les dues parts ha originat cada missatge així que s'utilitzarà la signatura digital com a sistema per autenticar

els missatges.

Autenticació de l'emissor i del receptor

També en aquest cas hi ha dues tecnologies per resoldre aquest tipus d'autenticació:

- *Autenticació per clau.*

Aquest és un mecanisme àmpliament utilitzat, concretament mitjançant un usuari i una clau es pot identificar la part client d'una aplicació. S'ha d'anar en compte que al enviar la clau aquesta viatgi xifrada perquè una tercera part no pugui conèixer-la.

- *Autenticació mitjançant criptografia de clau pública.*

Aquesta tecnologia permet identificar tant al client com al servidor mitjançant els certificats de clau pública de cadascun d'ells.

Per a la identificació i l'autenticació d'usuaris s'utilitzarà el protocol de Needham-Schroeder, basat en criptografia de clau pública. Concretament serà el protocol que s'executarà cada vegada que un usuari vulgui iniciar una sessió. La identificació és necessària per saber amb qui s'ha establert comunicació, mentre que l'autenticació serveix per decidir si realment podem confiar en aquest remitent. El protocol de Needham-Schroeder ens permet assolir aquests dos objectius i per tant serà el que utilitzarem per a fer la identificació i l'autenticació de les parts.

Una vegada client i servidor s'hagin identificat i autenticat caldrà garantir la confidencialitat, la integritat i el no repudi.

Confidencialitat

La confidencialitat s'aconsegueix xifrant les dades que viatgen entre client i servidor. Per fer això hi ha dues opcions, o bé és el propi sistema el que s'encarrega de xifrar els missatges que envia i després utilitza el sistema de comunicacions estàndard, o bé no es xifren els missatges a nivell d'aplicació i es delega aquesta tasca al nivell de transport mitjançant les tecnologies ssl o tls.

Integritat

La integritat fa referència al manteniment del contingut del missatge des de que l'envia l'emissor fins que el rep el destinatari. Això implica que el missatge que rep el destinatari no ha d'haver estat modificat, no s'hi ha d'haver afegit cap més dada de les que conté i tampoc s'han d'haver eliminat dades del seu contingut.

Al utilitzar la signatura del missatge com a mètode per la seva autenticació, la integritat del contingut del missatge també ens quedarà garantida. Això és així perquè la signatura es calcula a partir del contingut del missatge. Si es modifica el contingut del missatge la signatura deixa de ser vàlida i per tant sabem que s'ha vulnerat la integritat del missatge.

No repudi

Quan parlem de no repudi ens referim al fet que el remitent d'un missatge no pugui negar haver-lo enviat, alhora que tampoc pugui al·legar que el contingut del missatge no

sigui el que ell originalment va enviar.

Pel que fa al no repudi cal tenir en compte que limitar-se a signar un missatge no serveix de gaire. Encara hi ha atacs possibles al no repudi una vegada signat un document. La signatura per si sola no garanteix que el remitent realment sigui qui diu ser. D'aquesta manera, només amb la signatura la transmissió del missatge és vulnerable a diverses tècniques d'atac.

Cal garantir que cap emissor maliciós pugui més tard negar la creació i l'enviament d'un missatge.

Per exemple, tenim que una empresa A crea i envia una ordre de compra a una empresa B. L'empresa B processa l'ordre i genera la corresponent factura. En aquest punt l'empresa A no ha de poder negar el fet d'haver enviat l'ordre de compra, o el que és el mateix, no ha de poder repudiar-la.

Per satisfer el requeriment de no repudi es necessita tant l'autenticació del creador del missatge, com l'autenticació de l'emissor del missatge. La identificació del receptor no implica res sobre el no repudi, encara que és molt important en altres requeriments de seguretat.

Si només s'utilitza l'autenticació del missatge som vulnerables als atacs de reenviament. Per exemple, si una empresa A envia una ordre de compra signada digitalment a una empresa B i durant la transmissió una tercera empresa C maliciosa pot copiar el missatge, aquesta tercera empresa pot reenviar el missatge a B. Al rebre el missatge enviat per l'empresa maliciosa C, l'empresa B es pensa que el missatge l'ha enviat A, perquè així ho indica la signatura del missatge. D'altra banda també pot passar que l'empresa A, ara maliciosa, envii dues o més ordres de compra iguals i que després les vulgui repudiar justament dient que no li corresponen i que són origen d'un atac de reenviament, encara que realment qui hagi enviat totes les ordres hagi sigut l'empresa A.

D'altra banda, només amb l'autenticació de l'emissor del missatge no podem garantir que aquest no hagi estat modificat durant la transmissió. Qualsevol part pot reclamar sobre la integritat del missatge i qualsevol tercera part maliciosa pot modificar el missatge.

Solucions de seguretat en el nostre sistema

Donada la importància de la seguretat en el nostre aplicatiu es decideix desenvolupar els nostres propis protocols de seguretat per no haver de dependre de terceres parts i haver-nos de sotmetre a les seves polítiques i decisions. Amb això s'aconsegueixen cinc fites bàsiques:

1. Llibertat total en la presa de decisions.

Al no dependre de cap tercera part tenim llibertat total per decidir quines tecnologies volem utilitzar, quan les volem utilitzar, com les volem utilitzar i com les volem combinar. Aquesta llibertat es veu sovint limitada quan es decideix fer ús de productes de terceres parts.

Una característica importantíssima del programari de terceres parts és que les correccions, millores, etc. no depenen de nosaltres. Si es detecta un error, un forat de seguretat o un comportament incorrecte no podem decidir ni el grau de gravetat de la incidència, ni quina solució aplicar-hi, ni tant sols podem decidir si cal arranjar la incidència o no. Segons la majoria de contractes que ens lliguen amb la terceres parts no podem decidir ni reclamar res.

2. Especialització en seguretat del nostre equip (know how).

Donat que és el nostre equip qui dissenya i implementa els protocols a utilitzar, aquest equip tindrà un alt grau de coneixement de les teories de la protecció de la informació. També coneixerà les tecnologies existents, amb els punts febles i forts de cadascuna, les alternatives etc. Tot això ens donarà un avantatge competitiu davant de possibles atacs, correccions, canvis, i crisis en el sector.

3. Coneixement exacte de què s'està protegint i de què no s'està protegint.

Amb el control del disseny i la implementació podem tenir un major grau de coneixement del grau real de protecció que estem oferint. I el que és més important, sabem exactament, fins al més mínim detall, de quina manera estem garantint cada requeriment de seguretat.

4. Depenem de nosaltres mateixos.

Donat un moment de crisi podem tenir informació més acurada de les nostres possibilitats de resolució i del temps de resolució. Això ens ajudarà en gran mesura a la presa de decisions.

En aquest punt cal destacar que disposem de tots els elements de seguretat del nostre sistema de forma directa i per tant no tenim els retards que suposen les peticions a terceres parts, les possibles negociacions de solucions, etc.

5. Implica una gran metodologia tant en la teoria de la informació com en el procés d'enginyeria del programari.

Com a contrapartida de ser nosaltres mateixos els propietaris del *know how* haurem de mantenir una alta responsabilitat en dos camps.

D'una banda cal un estudi a fons de les teories de la protecció de la informació, de criptografia i de seguretat informàtica. D'altra banda cal estar al corrent dels diversos atacs que poden patir tant els sistemes informàtics com la informació que es processa. Alhora aquest equip ha de ser capaç de dissenyar, actualitzar i corregir els protocols segons els últims coneixements adquirits.

D'altra banda el desenvolupament del programari s'ha de fer d'una forma especialment estricta per poder garantir la seva màxima qualitat. Això implica tenir uns processos de desenvolupament, una configuració del programari, un manteniment, etc. també de qualitat.

Encara que sembli que aquest l'últim punt 5 sigui un punt que juga en contra de la nostra aplicació en realitat és un punt a favor. El seguiment d'unes bones pràctiques de seguretat i d'enginyeria del programari aporta qualitat i per tant no s'ha de tenir cap problema en adquirir aquestes responsabilitats.

En resum. S'ha pres la decisió de desenvolupar nosaltres mateixos els protocols de seguretat de la nostra aplicació i per fer-ho s'han utilitzat com a base en els cinc punts que acabem d'esmentar.

Una vegada recordada la problemàtica dels requeriments de seguretat que ens afecten, vegem quines solucions s'han adoptat:

- *Identificació i autenticació dels usuaris.*

Per aconseguir la identificació i autenticació entre dos elements del sistema, normalment client i servidor, implementarem el conegut protocol de Needham-Schroeder. Aquest protocol s'executarà cada vegada que un dels elements vulgui intercanviar dades amb un l'altre, iniciant d'aquesta manera una sessió. Per obrir una sessió les dues parts s'hauran d'identificar i autenticar mitjançant l'esmentat protocol de Needham-Schroeder, a través seu crearan la sessió i l'hauran de mantenir secreta (sota la seva responsabilitat). La resta de comunicacions que es facin durant la conversa hauran de citar la sessió per garantir-ne la identificació i autenticació. Una vegada finalitzi la conversa s'haurà de tancar la sessió. Per estar apunt davant de possibles errors tècnics (com ara *timeouts*, etc) o atacs que deixin fora de servei una de les dues parts, quan l'altra part intenti comunicar-se amb la part inaccessible durant una conversa i no pugui, automàticament haurà d'invalidar la sessió. Caldrà doncs que iniciï una nova conversa.

- *Confidencialitat.*

Per garantir la confidencialitat s'utilitzarà la criptografia de clau pública per fer el xifrat de les dades. Amb aquesta tecnologia garantirem que ningú altre que el destinatari de la informació pugui desxifrar la informació enviada.

Per ser més precisos, podrà desxifrar la informació tot sistema que tingui accés a la clau privada que fa parella amb la clau pública que s'ha utilitzat per signar les dades.

Per dificultar els atacs per robatori de claus cada usuari disposarà d'una targeta de seguretat amb el seu pkcs12. Quan un usuari vulgui comunicar-se amb el servidor caldrà que connecti la seva targeta a l'ordinador per poder dur a terme tots els processos de seguretat. Una vegada l'usuari hagi acabat d'utilitzar l'aplicatiu haurà de retirar la targeta de l'ordinador i guardar-la en un lloc segur.

- *Integritat de les dades.*

La integritat de les dades també quedarà garantida mitjançant l'ús de la criptografia de claus públiques, en aquest cas mitjançant la signatura de les dades.

La signatura de clau pública es calcula a partir del text que s'ha de signar. Per un text només hi ha una signatura possible. Així quan signem un text amb la nostra clau privada es calcula una signatura única que garanteix que aquell text només es pot validar amb la nostra clau pública. Si algú modifica el text al validar-lo amb la nostra clau pública ens adonarem que signatura i text no es corresponen i per tant sabrem que el text que tenim no és el que originalment s'ha signat.

Com veiem ara succeeix al revés, es signaran les dades utilitzant la clau privada de l'usuari i es validarà la signatura mitjançant la clau pública d'aquest usuari. Per garantir que només l'usuari propietari de les dues claus ha pogut signar unes dades cal que només ell tingui accés a la seva clau privada i per tant també caldrà l'ús de targetes criptogràfiques de seguretat.

- *No repudi.*

Per garantir el no repudi es necessita un registre de les peticions que fan les parts. Es per això que cada missatge inclourà un identificador de missatge amb la sessió a què pertany, l'instant de temps en què s'ha creat i un número de seqüència. Així com actualment cal que cada docent deixi constància de la seva autoria en les anotacions que fa a cada historial mitjançant la signatura del document en paper, en la nostra aplicació també cal que els historials mantinguin un registre de l'autoria de cada nova inserció. A més, com que es tracta d'un document confidencial s'aprofitarà aquest registre per anotar-hi totes les consultes que es facin de l'historial, amb la informació de qui ha fet la consulta.

Tal com hem comentat abans es pot utilitzar ssl/tls per, una vegada les dues parts s'han identificat i autenticat, mantenir una comunicació. Aquesta és una bona solució si les dues parts es tenen confiança mútua, però no garanteix el no repudi.

Donat que per garantir el no repudi cal l'autenticació d'emissor i receptor i l'autenticació del missatge nosaltres resoldrem ambdues autenticacions de la següent forma:

- L'autenticació de les dues parts queda garantida a través de la sessió a través de la que s'enviaran els missatges que s'intercanvien. Aquesta sessió s'ha obert de forma segura al iniciar la conversa. Com que al iniciar la conversa emissor i receptor s'identifiquen i autèntiquen de forma segura aprofitem aquest procés per iniciar una sessió única que serà vàlida durant tota la conversa. Com que aquesta sessió només la saben emissor i receptor (i és responsabilitat seva mantenir-la secreta) el fet d'incloure-la en els missatges és senyal que el missatge s'ha creat durant aquella sessió. Aquí la sessió fa el paper de clau acordada entre les dues parts.
- L'autenticació del missatge la garantirem mitjançant la signatura digital feta amb criptografia de clau pública. Com veurem en el següent punt inclourem un timestamp i una seqüència en els missatges per evitar atacs de reenviament.

No cal dir que totes les converses es faran mitjançant comunicacions xifrades.

IV.Disseny

Format dels missatges

Els missatges que s'intercanviaran usuari i gestor estarà definit per dos tipus de document, les peticions, que fa l'usuari al gestor, i les respostes, que dona el gestor a l'usuari. Tot i això el format dels dos tipus de document serà sempre el mateix, i només variarà el tipus i les dades que continguin en cada cas.

Com a estàndard que s'és s'ha preferit utilitzar documents xml per a fer l'intercanvi de dades entre clients i servidor.

Vegis tot seguit el format de l'xml:

```
<behc>
  <capcalera>
    <tipus-msg>xxx</tipus-msg>
    <id-remitent>xxx</id-remitent>
    <timestamp>xxx</timestamp>
    <contexte>
      <pas>xx</pas>
      <nl>xxxx</nl>
      <timestamp1>xxxx</timestamp1>
      <n2>xxxx</n2>
      <timestamp2>xxxx</timestamp2>
    </contexte>
  </capcalera>
  <cos>
    <dades>
      ...
    </dades>
    </signatura>
    ###
    <signatura>
  </cos>
</behc>
```

En l'xml de petició tots els camps són obligatoris, mentre que en l'xml de resposta el camp *signatura* és opcional perquè pot haver-hi situacions en què una resposta no tingui valor legal i que per tant, si s'utilitza xifratge un cop identificats i autenticats els usuaris, no calgui signar-la.

La capçalera contindrà les dades referents al tipus de missatge que s'estigui enviant i a l'autenticació de les parts. Concretament les dades d'autenticació es troben dintre del *contexte* de la capçalera.

La marca⁴ *dades* contindrà en el seu interior les dades que una part vol comunicar a l'altra. Així doncs el seu format variarà segons quines siguin aquestes dades. Com que s'està utilitzant la bastida xStream per fer la serialització⁵ i la deserialització⁶ de les dades podem afirmar que el format de l'interior de la marca *dades* es generarà automàticament segons les dades que s'enviïn.

Pel que fa a la marca *signatura*, que com s'ha comentat és opcional en algun

4 En anglès *tag*.

5 D'objecte a xml.

6 D'xml a objecte.

casos, el seu contingut és estrictament binari.

Una vegada vist un esquema del contingut dels missatges de petició i de resposta es continuarà amb la descripció dels protocols.

Descripció dels protocols

En tots aquests protocols es té en compte que la comunicació entre remitent/destinatari es troba xifrada per algun sistema del tipus ssl/tls. Si Aquesta suposició no fos certa cada vegada abans de fer un enviament caldrà xifrar i desxifrar el missatge enviat.

És important matitzar el concepte de sessió en el context d'aquest aplicatiu. Tot i que habitualment una sessió permet l'execució de diverses peticions a la banda del servidor, el Banc Electrònic d'Historials Mèdics només permet l'execució d'una petició per cada sessió. Aleshores un es pot preguntar si té sentit utilitzar una sessió per només executar una sola petició. Però tot i que això últim sigui cert cal aclarir que per executar el protocol segur de Needham-Schroeder cal un intercanvi de com a mínim quatre missatges entre client i servidor, i és entre aquest intercanvi de missatges que es manté activa una sessió. La sessió s'inicia al rebre el primer missatge d'autenticació i perdura fins que ha acabat l'intercanvi de missatges que suposen l'autenticació i l'execució de la petició del client.

Usuari

1. Obtenir un valor de forma aleatòria, N_i .
2. Xifrar N_i i I_u , $E_g(N_i, I_u)$. On és l'identificador de l'usuari.
3. Enviar $E_g(N_i, I_u)$.

Gestor del Banc d'Historials

1. Desxifrar $E_g(N_i, I_u)$ amb S_g .
2. Obtenir el certificat de P_i amb I_u . A partir del certificat obtindrà P_u .
3. Obtenir un valor de forma aleatòria, N_g .
4. Xifrar N_i , N_g , I_g amb la clau pública P_u de P_u , $E_u(N_i, N_g, I_g)$.
5. Enviar $E_u(N_i, N_g, I_g)$.

Usuari

1. Desxifrar $E_u(N_i, N_g, I_g)$ amb la clau privada S_u .
2. Xifrar N_g amb la clau pública de G , $E_g(N_g)$.
3. Enviar $E_g(N_g)$.

Gestor del Banc d'Historials

1. Desxifrar $E_g(N_g)$ amb la clau privada S_g .
2. Si $N'_g=N_g$, g i u estan autenticats bilateralment i el gestor pot procedir a executar la petició de l'usuari. I, en cas de necessitat, respondre a l'usuari.
3. Tancar la Sessió.

Registre d'usuaris

Una vegada feta l'autenticació de client i servidor:

Administrador

1. Generar petició de registre d'usuari (PR).
2. Signar petició $S_{ad}(PR)$.
3. Enviar $S_{ad}(PR)$.

Gestor Banc d'Historials

1. Validar la signatura de la petició $P_{ad}(PR)$.
2. Desar $S_{ad}(PR)$.
3. Executar petició: desar les dades personals i la clau pública del nou usuari.
4. Enviar ACK.
5. Tancar la Sessió.

Inserció d'un Historial

Una vegada superada correctament l'autenticació entre client i servidor:

Docent

1. Generar petició d'inserció (PI).
2. Signar petició d'inserció $S_{dc}(PI)$.
3. Enviar $S_{dc}(PI)$ al Gestor.

Gestor del Banc d'Historials

1. Validar la signatura de la petició, $P_{dc}(PI)$.
2. Desar $S_{dc}(PI)$.
3. Executar petició-> desar insercions.
4. Enviar ACK.
5. Tancar la Sessió.

Inserció d'un Cas d'Clínic

Després d'haver superat correctament l'autenticació entre client i servidor:

Docent

4. Generar petició d'inserció (PI).
5. Signar petició d'inserció $S_{dc}(PI)$.
6. Enviar $S_{dc}(PI)$ al Gestor.

Gestor del Banc d'Historials

6. Validar la signatura de la petició, $P_{dc}(PI)$.
7. Desar $S_{dc}(PI)$.
8. Executar petició-> desar insercions.

9. Enviar ACK.
10. Tancar la Sessió.

Inserció d'una Anotació

Una vegada s'hagin autenticat client i servidor:

Docent

7. Generar petició d'inserció (PI).
8. Signar petició d'inserció $S_{dc}(PI)$.
9. Enviar $S_{dc}(PI)$ al Gestor.

Gestor del Banc d'Historials

11. Validar la signatura de la petició, $P_{dc}(PI)$.
12. Desar $S_{dc}(PI)$.
13. Executar petició-> desar insercions.
14. Enviar ACK.
15. Tancar la Sessió.

Consulta de l'Historial

Una vegada s'hagin autenticat mútuament client i servidor:

Usuari de l'Historial

1. Generar petició de consulta (PC).
2. Signar petició de consulta, $S_{uh}(PC)$.
3. Enviar $S_{uh}(PC)$.

Gestor del Banc d'Historials

1. Validar la signatura de la petició, $P_{uh}(PC)$.
2. Desar $S_{uh}(PC)$.
3. Executar petició -> recuperar l'història.
4. Signar l'història recuperat, $S_g(H)$.
5. Enviar $S_g(H)$.

Usuari de l'Historial

1. Validar la signatura de la resposta, $P_g(H)$.
2. Fer ús de l'història.

Gestor del Banc d'Historials

1. Tancar sessió

Acabament de Sessió

Gestor del Banc d'Historials

1. Una vegada acabada una petició es tancarà la sessió que hagi activat el client que s'hagi autenticat en aquella sessió.

Tecnologies utilitzades

L'ús de la criptografia de clau pública fa que el programari que els usuaris no puguin treballar amb *thin clients*. Això és així perquè la criptografia de clau pública requereix un càlcul intensiu del que no disposen aquests clients lleugers.

Per garantir al màxim l'ús del programari desenvolupat s'ha optat per l'ús de la tecnologia Java2. Tot i que la part servidora sempre es va tenir clar que es desenvoluparia amb Java, inicialment també es va tenir en compte la possibilitat de desenvolupar *rich clients* mitjançant tecnologia HTML i JavaScript, finalment aquesta possibilitat es va desestimar per la gran dificultat que suposava el xifratge i la signatura de les dades.

S'ha escollit doncs l'ús de Java2 tan a la part client com a la part servidor. Per al desenvolupament de la interfície gràfica d'usuari client s'ha optat per les tecnologies AWT/Swing/JFC incloses en el propi jdk de java, versió 1.5.

La comunicació entre client i servidor es fa a través de *sockets*, que permeten la transmissió d'xmls de forma fàcil. I, com és natural, aquesta informació s'enviarà estructurada en format xml.

Per la serialització i la deserialització de les dades al format xml s'ha aprofitat la bastida, o *framework*, XStream a la seva versió 1.1.3. Aquesta bastida és extremadament senzilla d'utilitzar, dinàmica i adaptable als canvis de format dels xmls.

S'ha escollit la bastida junit, versió 4.0, pel sistema de testeig. Així doncs, tots els tests unitaris estan integrats en aquesta bastida.

Pel que fa a la persistència de les dades en el programari servidor s'ha escollit el sistema de gestió de bases de dades, o SGBD, MySQL, ja que és un bon sgbd de codi obert que respecta l'estàndard SQL. Alhora també s'ha inclòs el *middleware* Hibernate, versió 3.0, per facilitar les operacions de persistència i consulta d'objectes.

Finalment i, pot ser, més important s'ha decidit utilitzar la llibreria criptogràfica IAIK. La raó d'aquesta decisió recau en la necessitat d'una gran quantitat de temps per implementar les llibreries necessàries per a estendre les llibreries criptogràfiques incorporades en el jdk de java 1.5. Aquestes últimes són les mínimes per a implementar una infraestructura de clau pública i la majoria d'algorismes que ofereixen són febles en comparació amb els existents en l'actualitat. Això significa que per tenir una base criptogràfica adequada cal estendre aquestes llibreries i justament això és el que fa el paquet IAIK. Preferiblement hagués estat millor integrar el desenvolupament d'aquestes llibreries en el propi projecte del Banc Electrònic d'Històrics Clínics, però només això ja implicaria molt més temps que el disponible per fer tot aquest projecte. Així doncs la justificació de l'ús de la llibreria IAIK queda clarament justificada.

Descomposició del programari en paquets

A l'hora de pensar l'aplicació s'ha tingut sempre al cap el fet que aquesta sigui el màxim de modular possible. Així s'aconsegueix que la implementació de cada part no condicioni a cap altra part. Això vol dir que sempre es podrà substituir un mòdul per un altre que supleixi millor la funcionalitat del primer. Només hi haurà la restricció d'haver de mantenir la mateixa interfície que el primer mòdul.

Alhora la separació en mòduls ajuda a fer un disseny menys monolític i per tant més simple. Justament, que una aplicació sigui el màxim de simple possible és una característica a desitjar perquè en millora la seva comprensió i el seu manteniment.

L'aplicatiu Banc Electrònic d'Historials Clínics constarà de dos sub-sistemes clarament diferenciats, l'aplicatiu client i l'aplicatiu servidor. Tots dos sistemes tenen tasques ben diferenciades, però això no ens ha d'impedir aconseguir un reaprofitament òptim del codi.

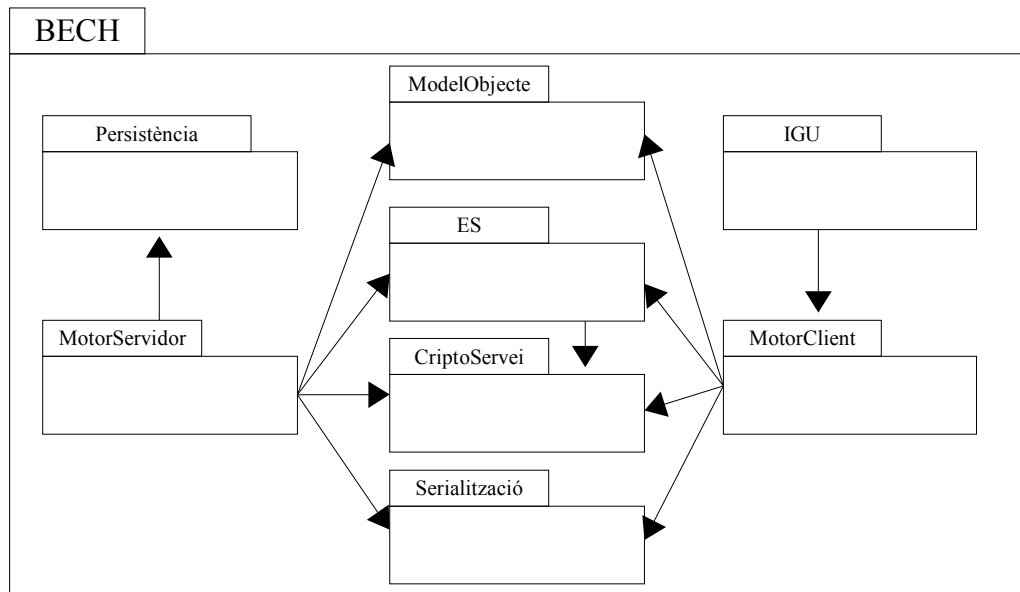
Comencem pel sistema client. Si volem un sistema modular cal separar la interfície gràfica del motor encarregat de dur a terme les operacions amb el servidor central. Per tant ja tenim dos paquets, un paquet que conté la IGU que utilitzarà l'usuari per comunicar-se amb l'aplicatiu, i un altre paquet que conté el MotorClient que s'encarregarà d'enviar les peticions provinents de la IGU al servidor.

En segon lloc caldrà un paquet de comunicacions entre la part client i la part servidor, anomenarem ES, acrònim d'entrada/sortida, a aquest paquet. Serà un paquet que durà a terme les comunicacions. Això vol dir que serà en aquest paquet on implementarem tots els protocols de comunicacions.

En tercer lloc el sistema servidor haurà de disposar d'un motor propi per processar les peticions que li arriben del client. Aquest motor es trobarà en el paquet MotorServidor, el qual accedirà a la base de dades mitjançant el servei de persistència, paquet Persistència.

Finalment tant el client com el servidor tindran unes necessitats comunes que empaquetarem en els següents paquets: ModelObjecte, Serialitzador i CriptoServei. A l'igual que l'ES, tots aquests paquets seran compartits tant pel client com pel servidor.

Vegem doncs els mòduls que formaran part de l'aplicatiu i les interrelacions que existiran entre ells:



Pel sistema de serialització farem ús de la bastida, o *framework*, XStream. Aquesta bastida ens facilitarà en gran mesura la tasca de serialitzar i de deserialitzar els objectes java a xml. La potència d'aquesta bastida que encapsulem dins del paquet Serialització està en que amb una sola línia de codi podrem serialitzar o deserialitzar els objectes i els xml.

Pel que fa al sistema de persistència també ens aprofitarem d'una altra bastida molt popular dins el món java. Es tracta d'Hibernate. Amb aquest *middleware* també simplifiquem molt la interacció entre la nostra aplicació i la base de dades.

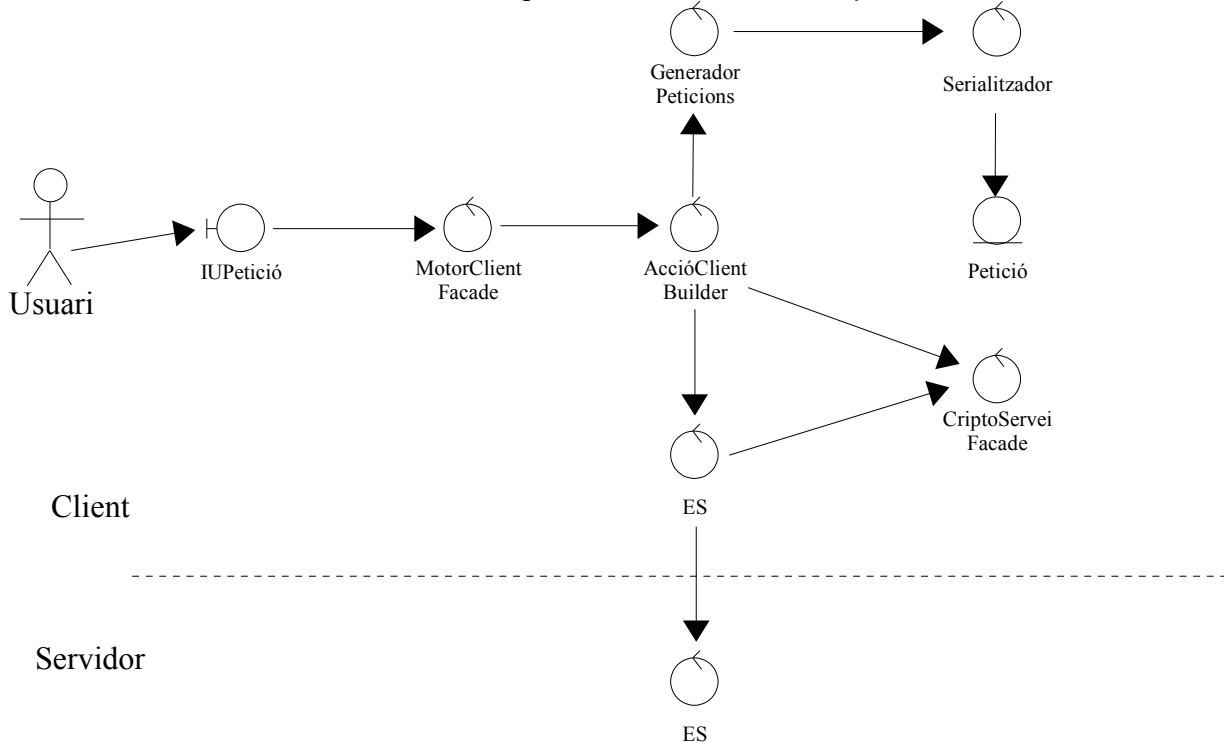
Cal parar atenció a que el paquet ES té els coneixements sobre les comunicacions segures, però no coneix res sobre la criptografia. És per això que els protocols els implementa ES, mentre que els algorismes de criptografia es troben dins el CriptoServei.

Diagrames de Col·laboració

Comunicació client-servidor

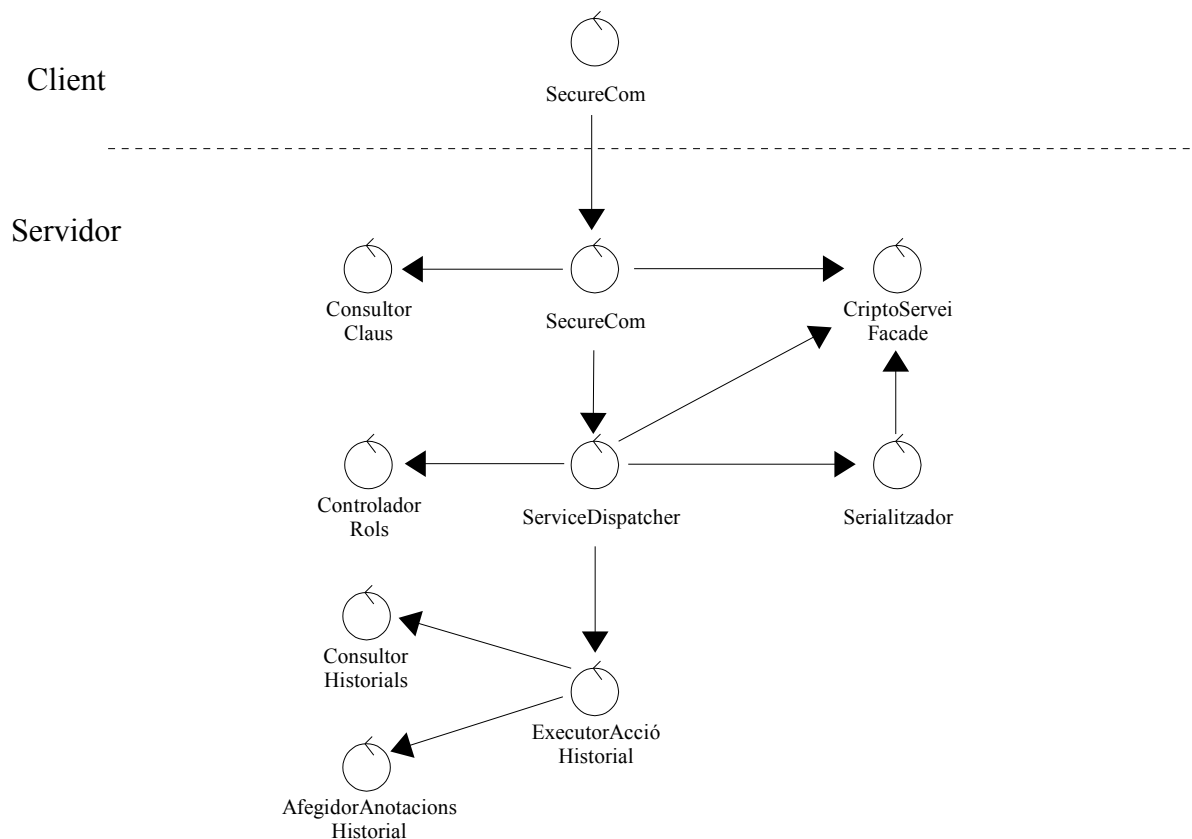
Donada l'especificació dels casos d'ús passem a mostrar els diagrames de classes corresponents a una petició del client cap al servidor.

Comencem doncs amb un diagrama de col·laboració pel sistema client:



1. Com podem veure l'actor Usuari inicia la petició dirigint-se a la classe frontera corresponent.
2. Aquesta classe passa la consulta al MotorClientFacade, que és una implementació del patró Façana. Aquesta façana disposarà d'un mètode per cada possible petició que el sistema pugui servir.
3. Donada la petició de l'actor la façana s'encarregarà de passar-la a l'AccióClient corresponent, és a dir que existirà una subclasse AccióClient per cada acció que el client pugui dur a terme en el sistema.
4. Concretament AccióClient instanciarà la petició sol·licitada mitjançant el GeneradorPeticions, el qual s'encarrega de la instanciació de les peticions que es poden fer al servidor.
5. Les dades retornades pel GeneradorPeticions ja estaran serialitzades, i a punt per enviar. La serialització es farà mitjançant el Serialitzador, que rep l'objecte petició i el converteix en un xml de text pla.
6. Tot seguit, una vegada l'AccióClient ja disposi de l'xml de la petició serialitzada, es farà ús del CriptoServei per signar la petició.
7. Finalment només faltará enviar l'xml signat a través del sistema d'entrada/sortida.

Vegem ara com processa les peticions que li arriben el sistema servidor:



EI

servei d'entrada/sortida del servidor és l'encarregat de rebre les peticions que els Usuaris li fan a través del sistema client. Una vegada rebuda la petició els passos que s'executen en el servidor són:

1. Recepció del missatge mitjançant el pertinent protocol.
2. Enviament de la petició al ServiceDispatcher.
3. Cerca de les claus de l'usuari que ha signat la petició per poder-la validar.
4. Validació de la signatura mitjançant el CriptoServei.
5. Control dels permisos que té l'usuari per executar la petició.
6. Si l'usuari passa el control el ServiceDispatcher instanciarà l'objecte encarregat d'executar l'acció demanada. Aquest objecte serà una subclasse d'ExecutorAccióHistorial.
7. Finalment l'ExecutorAccióHistorial, que és una implementació del patró Expert, durà a terme les accions que calgui segons quina petició s'hagi fet.

El desenvolupament a més baix nivell de totes aquestes interrelacions el farem en el moment de fer el disseny de cada part.

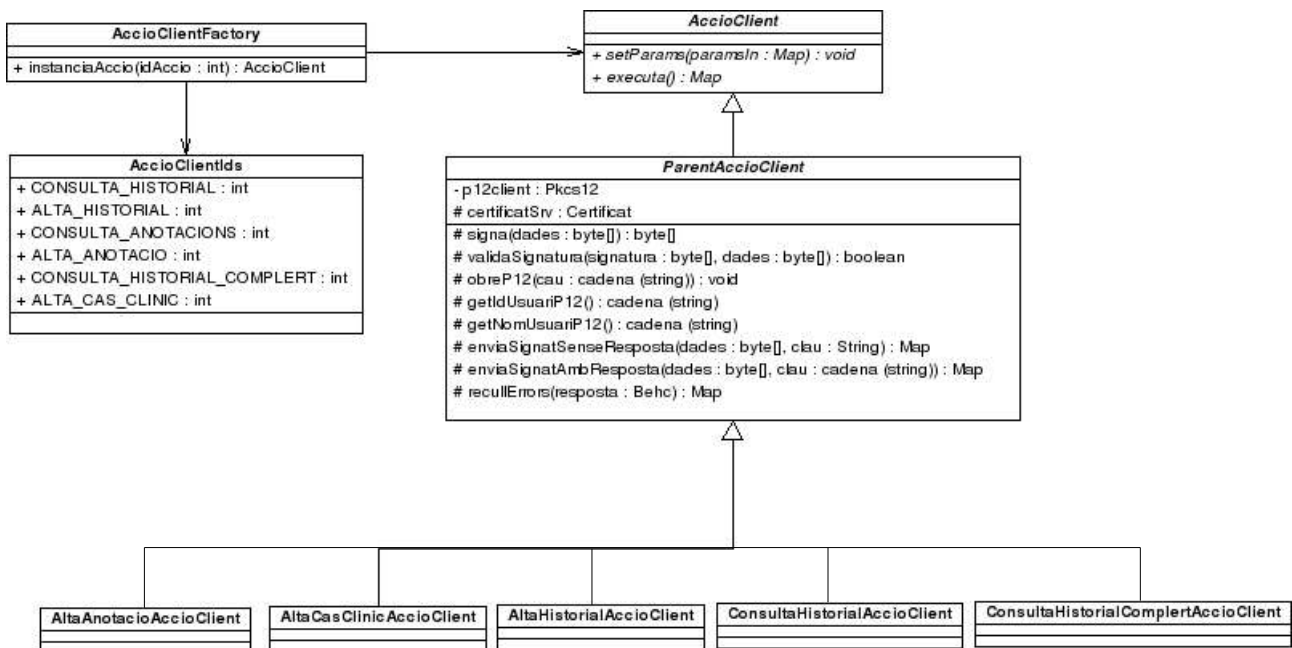
Diagrames de classes

Accions Client

Vist de forma estricta AccióClient és una herència de classes clàssica, amb una interfície que en defineix les operacions, una classe pare que implementa els mètodes comuns a totes les AccionsClient i amb el conjunt de subclasses especialitzades en cada servei.

Però el que estem fent en realitat és aplicar el patró Command. És a dir, mitjançant la interfície AccióClient estem definint un contracte comú d'execució per a totes les AccionsClient.

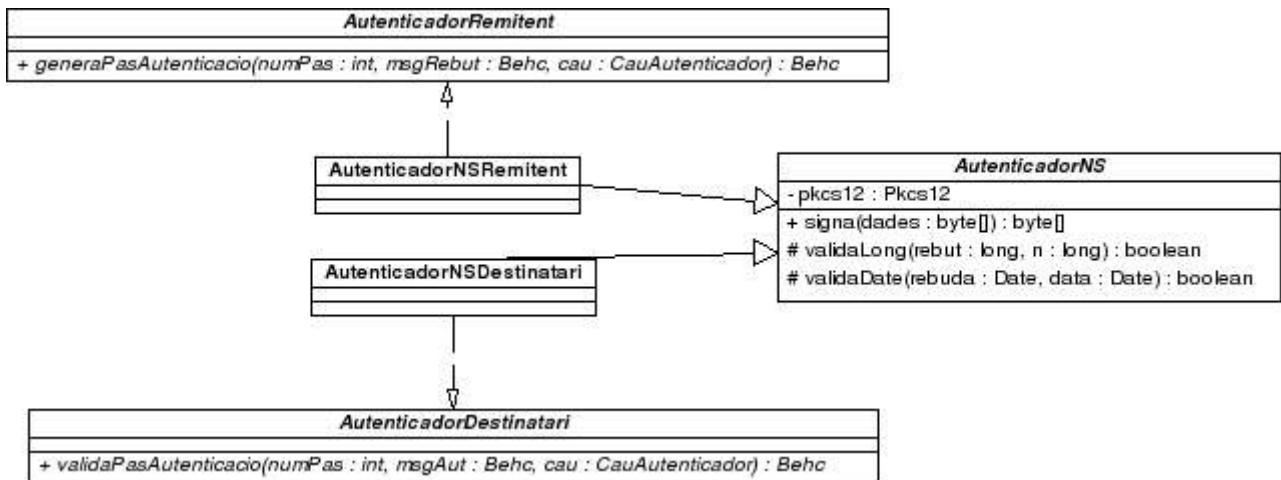
D'altra banda s'ha pensat en una factoria per a instanciar els diferents tipus d'accions de forma transparent a la resta del programari.



Autenticadors

En aquest programari existeixen dos tipus d'autenticadors, el de la part que s'està autenticant, anomenat AutenticadorRemitent, i el de la part a la que s'està autenticant, anomenat AutenticadorDestinatari. Cadascun d'aquests dos autenticadors té els seus mètodes de funcionament, sigui quin sigui l'algorisme que s'utilitzi per autenticar-se. Altra vegada s'està fent ús del patró Command, mitjançant les dues interfícies anteriors.

Com que l'algorisme escollit per a dur a terme l'autenticació entre dues parts és el de Needham-Schroeder hi ha dues classes que implementen les respectives interfícies d'autenticació segons el funcionament d'aquest algorisme. Per generalització dels mètodes comuns als autenticadors per Needham-Schroeder sorgeix una classe pare anomenada AutenticadorNS.



Criptografia

El paquet de classes criptogràfiques és una mica més complex que els anteriors. El fet d'utilitzar les llibreries IAIK ens ha estalviat la implementació de tots els algorismes criptogràfics, però el que no ha fet ha sigut simplificar l'ús de tots aquests algorismes. És per això que ha calgut desenvolupar aquest servei de criptografia. Aprofitant això a l'hora de fer el disseny d'aquest servei s'ha separat l'ús d'aquestes llibreries de la pròpia interfície del servei. D'aquesta manera si en un futur es volen canviar les llibreries IAIK per unes altres, si es volen desenvolupar unes llibreries pròpies o si IAIK modifica l'ús de les seves llibreries es podrà fer sense cap repercussió per la resta del codi del programari.

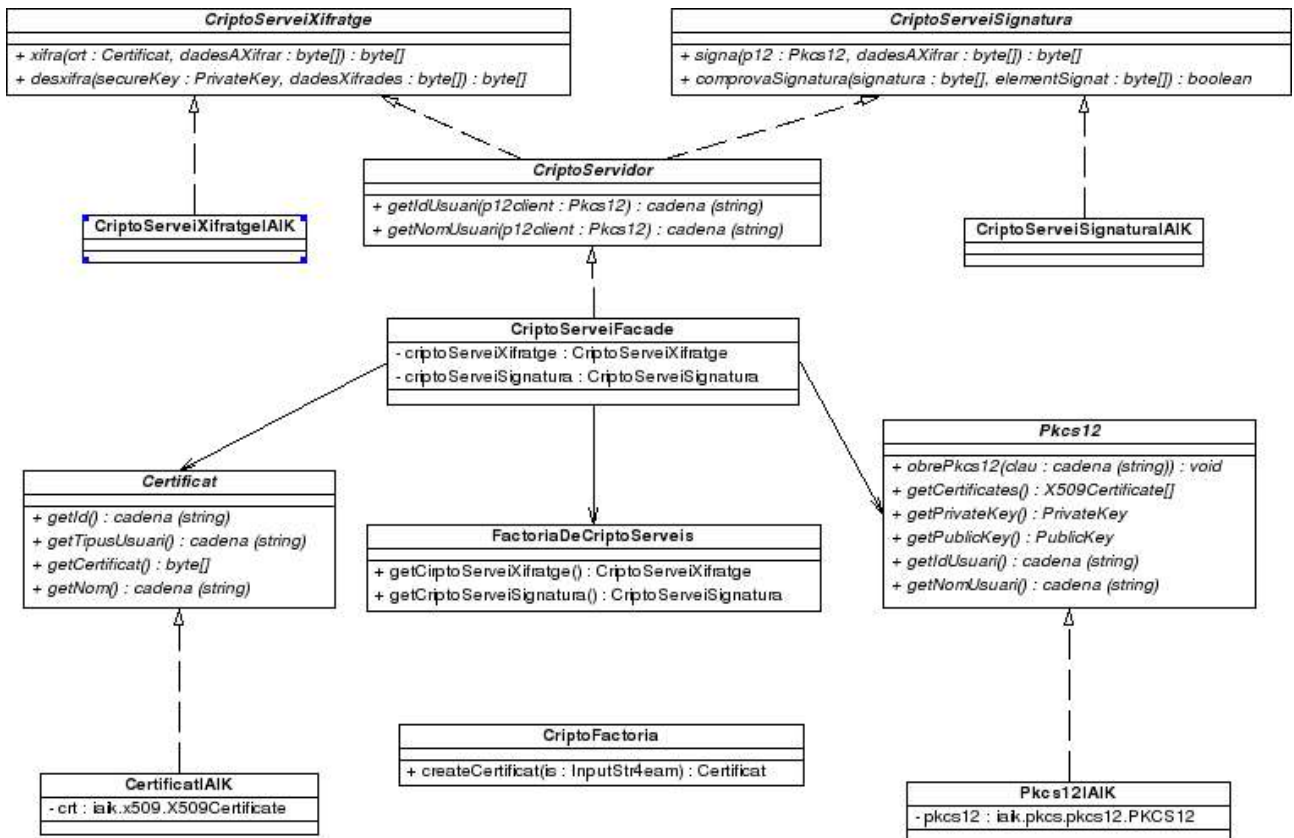
Com a element central del servei de criptografia hi ha la interfície complerta de serveis que ofereix aquest paquet, el *CriptoServidor*. Aquesta interfície actua com a façana perquè la resta del programari pugui accedir als serveis de criptografia de forma ordenada. I és el resultat de concentrar en una mateixa interfície els serveis criptogràfics de xifratge i els serveis criptogràfics de signatura.

Alhora que a *CriptoServidor* es fa ús del patró *Facade* per donar accessibilitat a tots els serveis que ofereix el paquet de criptografia a la resta de l'aplicatiu, aquest mateix paquet també utilitza el patró *Facade* per amagar darrera de les respectives interfícies les implementacions del servei de xifratge i del servei de signatura.

Per aïllar la implementació que es fa de cada servei existeix la *FactoriaDeCriptoServeis*, que decidirà quina és la implementació que cal de cada servei i la retornarà. Això evitarà que hagem de prendre aquesta decisió a cada punt del codi on calgui instanciar un servei, facilitant així el manteniment de tot el codi. D'aquesta manera el funcionament de la façana és molt senzill d'entendre: *CriptoServidor*, que és la implementació de la façana del paquet de criptografia, farà ús de la factoria per instanciar el servei que calgui utilitzar en cada moment i delegar-li la petició rebuda.

D'altra banda dins el paquet de criptografia també existeixen classes que encapsulen elements de les llibreries IAIK. La missió d'aquestes classes és simplificar el maneig d'aquests elements i altra vegada separar la implementació que en fa IAIK del concepte que representa cada element. L'abstracció de cada element criptogràfic l'expressen les interfícies *Pkcs12* i *Certificat*, i la concreció d'aquests elements la fan *Pkcs12IAIK* i *CertificatIAIK* respectivament.

Per llegir els certificats des d'un disc o qualsevol altre sistema d'emmagatzematge també s'ha pensat en afegir a aquest disseny una *CriptoFactoria* que instància un *Certificat* llegint d'un *InputStream*.



Persistència de les dades

El model de dades ens ha de permetre fer el manteniment de les dades personals de cada pacient alhora que també ha de permetre poder fer un seguiment dels casos clínics que hagi tingut el pacient al llarg del temps. D'altra banda ha de contenir les dades dels usuaris de l'aplicació.

Així doncs hi haurà l'entitat Usuari que estarà indexada per l'identificador de l'usuari i que contindrà el certificat vàlid per aquell usuari. Opcionalment també pot contenir altres dades associades a l'usuari.

Respecte a la part més sanitària d'aquest model, les dades menys canviants es desaran a l'entitat Historial. Aquesta entitat a més d'emmagatzemar les dades personals del pacient també contindrà els antecedents, tan familiars, com personals, com patològics, del pacient. L'identificador únic d'un historial pot ser el número d'afiliat del pacient i també pot ser el número del document nacional d'identitat. Com que aquest programari està pensat per donar servei en el context de la sanitat s'ha decidit utilitzar el número d'afiliat del pacient com a clau primària de l'entitat Historial. De totes maneres l'Historial també contindrà la informació del DNI, alhora que s'establirà una restricció UNIQUE a aquesta columna.

Cada vegada que un pacient tingui símptomes de trobar-se malament anirà a veure a un metge i aquest obrirà un nou CasClínic per aquell pacient. A aquest CasClínic el metge que tracti al pacient hi anirà afegint anotacions que correspondran a l'anamnesi, al tractament i al curs clínic. A mesura que s'avança en el tractament d'un pacient aquest pot passar per diferents metges i tots aquests metges podran anar afegint al CasClínic les anotacions que els hi corresponguin.

El CasClínic quedarà únicament identificat per un id que es generarà automàticament. A més, també tindrà un nom que haurà d'indicar el metge que l'obri. Per relacionar-lo amb el pacient al que fa referència tindrà una columna que contindrà el número d'afiliat d'aquest pacient. Aquest número és la clau forana a l'entitat Historial.

Com s'acava d'explicar, cada vegada que un metge vulgui documentar algun fet relacionat amb un CasClínic haurà d'afegir-hi una Anotació. A l'igual que els CasosClínics, una Anotació quedarà identificada únicament per un id que es generarà automàticament. Per saber a quin CasClínic correspon cada Anotació també tindrà una clau forana cap al corresponent CasClínic.

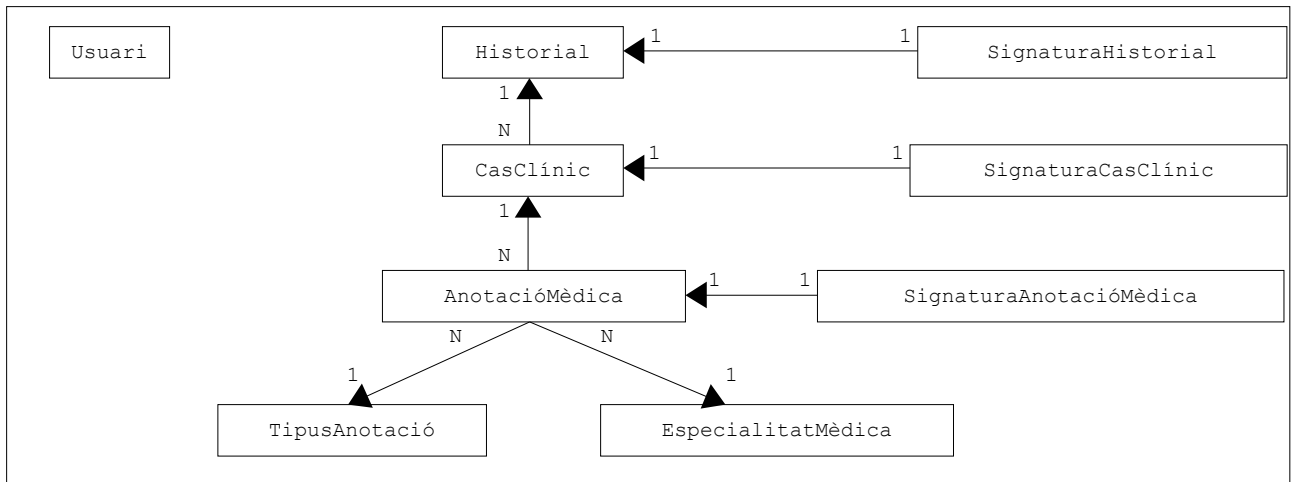
Depenent de què s'estigui documentat en una Anotació aquesta podrà ser d'un tipus o d'un altre. La classificació de tots els tipus d'Anotació possibles la contindrà l'entitat TipusAnotació. Alhora una anotació pot pertànyer a una especialitat mèdica o a una altra, és per això que també classificarem les Anotacions segons una EspecialitatMèdica.

Com es pot deduir per les claus foranes de les tres entitats anteriors entre elles hi haurà dues associacions d'un a molts. Un Historial té molts CasosClínics i un CasClínic té moltes Anotacions.

A més a més de les dades estrictament mèdiques també cal que el model de dades permeti desar-hi les dades de seguretat. Així que caldrà una taula on desar-hi les signatures de les peticions d'inserció que facin els metges. Tan Historial, com CasClínic, com Anotació tindran la seva corresponent taula on dipositar-hi les signatures de les altes que s'hi vagin fent. La decisió d'utilitzar una taula a part per desar-hi les signatures es fa per motius d'eficiència. Com que les signatures s'han de desar en columnes del tipus BLOB i aquestes són molt grans sempre és més convenient utilitzar una taula a part per

tenir-hi els BLOBs i així no penalitzar les operacions a la taula de la que pengen els BLOBs.

Vegem-ho:



V. Implementació

Interfície gràfica d'usuari

A l'hora d'implementar la interfície gràfica d'usuari s'ha preparat un *look and feel* propi. Aquest *look and feel* ha estat pensat per facilitar al màxim la usabilitat de l'aplicació. Els colors escollits sempre intenten centrar l'atenció de l'usuari sobre el que l'aplicació li està comunicant.

Així doncs s'ha escollit per a tots els elements de la interfície, excepte els missatges d'error, l'ús d'un fons blanc que eviti distraccions.

Els panells de treball s'han delimitat amb marcs d'un color blau clar. Aquest color difús no distreu a l'usuari del contingut dels panells. Al contrari, l'ajuda a delimitar els límits de la seva zona de treball.

S'ha optat per l'ús de finestres emergents en dues situacions.

La primera situació es dona quan l'usuari ha d'entrar una dada puntual a l'aplicatiu. Són casos en que no val la pena canviar el formulari de l'àrea de treball i on les dades que cal entrar no estan directament relacionades amb el formulari que l'usuari té obert en aquell moment. Com a exemple podem citar la finestra d'entrada de la clau del pkcs12 o la finestra on s'ha d'introduir el nom d'un nou cas clínic.

La segona situació es dona quan l'usuari demana veure les dades associades amb algun element del formulari obert en aquell moment a l'àrea de treball i aquestes dades són prou denses per requerir un formulari propi, però l'element de l'àrea de treball s'ha de mantenir obert perquè l'usuari hi pugui continuar interactuant. Un exemple d'aquest cas és la finestra que mostra la informació d'un cas clínic i de les seves anotacions.

Els errors s'han classificat en dos tipus, errors de validació i errors d'execució.

Els errors de validació avisen a l'usuari de la incorrecció d'alguna de les dades que ha entrat i van apareguren dinàmicament en el propi formulari on hi ha l'error.

D'altra banda els errors d'execució fan referència a excepcions que s'han produït durant l'execució del codi de l'aplicació. Aquest tipus d'errors s'acostumen a mostrar en una finestra pròpia per cridar l'atenció de l'usuari sobre el fet. En cas que no es pugui mostrar l'error d'execució en una nova finestra el que es fa és eliminar el contingut de l'àrea de treball i mostrar-hi allí l'error.

Sigui quin sigui el tipus de l'error que s'hagi produït aquest sempre es mostrarà en un panell amb el fons d'un color vermell, magenta, i amb el text de color negre. Amb aquests colors tant diferents del *look and feel* de la resta de l'aplicació s'espera que l'usuari vegi clarament que hi ha alguna cosa que requereix la seva atenció. El fet de mostrar els errors d'execució en una finestra pròpia es fa perquè es considera que aquest tipus d'error requereixen més l'atenció de l'usuari i el fet que s'obri espontàniament una finestra davant de l'usuari fa que ràpidament aquest l'hi pari atenció.

Els missatges que mostren els errors procuren assolir dos objectius, informar a

l'usuari de quina ha estat la causa que ha provocat l'error i, sempre que sigui possible, suggerir-li una possible solució. Aquesta fita sempre suposa invertir certa part del temps del projecte, però, tot i que en molts de casos es prefereix dedicar aquest temps a d'altres tasques, l'ambició de produir una aplicació que realment sigui de qualitat ha fet que es prefereixi no passar per alt aquesta tasca.

Tasques d'execució automàtica

Per poder facilitar la generació dels components client i servidor que finalment s'hauran d'executar s'ha implementat una tasca que funciona amb el programari Ant. Aquesta tasca consisteix en un fitxer anomenat `build.xml` on s'hi defineixen tots els passos que s'han de seguir per obtenir els dos components que formen el BEHC.

Si executem el programari Ant amb aquest fitxer s'aniran executant tots els passos que s'hi indiquen i s'obtidran els components esperats.

Per executar-lo només cal tenir instal·lat el programari Ant a la seva versió actual i des del directori on es troba el fitxer `build.xml` fer la crida

```
alumne$ant
```

això farà que l'usuari alumne executi les ordres del `build.xml`.

Però abans d'executar res cal informar correctament les propietats `orig.dir` i `build.dir`. A la primera hi informarem el directori on es troben els projectes del BEHC i a la propietat `build.dir` hi informarem el directori on volem que es generi l'estructura de directoris necessària per dur a terme la construcció del projecte.

Concretament aquesta tasca automàtica fa els següents passos:

- Recull el codi font del projecte i el copia al directori `src`.
- Copia les llibreries del projecte al directori on hi haurà l'aplicatiu, `bin/app`.
- Compila el codi font de `src` i desa el codi objecte a `bin/classes`.
- Genera dos fitxers. El corresponent a l'aplicació client anomenat `pfc-client.jar` i el corresponent a l'aplicació servidora anomenat `pfc-servidor.jar`.
- Finalment a partir del codi font del directori `src` genera el javadoc al directori `doc/javadoc`.

Gràcies a aquesta tasca ens serà molt fàcil mantenir la documentació que conté el programari al dia, alhora que organitzarem amb facilitat el codi de les dues aplicacions.

VI. Configuració del programari

Sistema Operatiu

Donades les necessitats de mobilitat del desenvolupador i donada l'arquitectura multiplataforma de l'entorn de desenvolupament Java s'ha optat per treballar en els sistemes operatius següents:

1. Principalment SuSE Linux versió 9.2.
2. En segon lloc Mac OS X versió 1.4.6.
3. Ocasionalment també s'ha fet ús del sistema Microsoft Windows XP amb l'SP2.

Entorn de desenvolupament

En qualsevol cas l'entorn on s'ha desenvolupat el programari ha sigut el mateix, un jdk versió 1.5 i l'entorn de desenvolupament integrat Eclipse 3.1.

Donada la mida petita del projecte s'ha obviat l'ús d'un repositori i s'ha treballat sense mantenir un control de versions de cap fitxer del programari.

Pel que fa al jdk caldrà instal·lar-li les extensions de criptografia de sun anomenades Java Cryptography Extensions (JCE), senzillament col·locant-les en el directori "security" de les llibreries del jre.

Dins l'àrea de treball d'Eclipse, o *workspace*, s'ha subdividit el projecte en subprojectes per facilitar-ne el desenvolupament. En total s'han necessitat sis subprojectes per organitzar correctament tot el codi. Alhora aquests subprojectes també tenen el codi organitzat en directoris segons la naturalesa d'aquest.

1. PFCcommon. Conté les classes comunes per l'aplicatiu servidor i per l'aplicatiu client i està organitzat de la següent manera:
 - "es". Entrada/Sortida. Codi de recepció i enviament de dades.
 - "common". Codi comú a tot l'aplicatiu i no pertanyent a cap paquet de l'arquitectura.
 - "criptografia". Com el seu nom indica conté el codi de gestió criptogràfica.
 - "modeldades". Conté els objectes que contenen les dades que s'intercanvien la part client i la part servidora. Aquests objectes són els que es serialitzen a xml. No contenen cap lògica i segueixen un patró que s'anomena de moltes maneres diferents: *data object*, *pojo*, *value object*.
2. PFCigu. Conté la interfície gràfica d'usuari. Únicament s'organitza en un sol directori anomenat "igu".

3. PFClibs. Conté les llibreries del projecte, cap línia de codi font.
4. PFCmotorClient. Conté la part de l'aplicatiu client amb tota la lògica de negoci.
 - “es”. Entrada/Sortida. Codi de recepció i enviament de dades.
 - “motorClient”. Conté la lògica amb tot el codi que duu a terme la comunicació de les accions sol·licitades des de la interfície gràfica amb el programari servidor.
5. PFCservidor. Conté codi que pertany exclusivament al programari servidor.
 - “es”. Entrada/Sortida. Codi de recepció i enviament de dades.
 - “persistència”. Conté els *data objects* que persisteixen i el codi que manega el *framework* Hibernate.
 - “motorServidor”. Conté els serveis que executen les accions que ha demanat l'usuari, ja siguin consultes o altes.
6. PFCtest. Conté els testos unitaris de diferents parts del programari.
 - “es”. Testos sobre el contingut dels missatges, sobre el *framework* xStream, sobre els autenticadors i sobre les connexions.
 - “criptografia”. Testeja els dos serveis que ofereix el paquet de criptografia, el xifratge i desxifratge, i la signatura, amb la corresponent validació.

Documentació

Per dur a terme la documentació s'ha treballat amb el programari OpenOffice.org 1.1.0 i per a generar els diagrames UML l'Umbrello 1.1.1 que es troba integrat en el sistema de finestres KDE de Linux.

Donat l'ampli suport que existeix en totes les plataformes pel format de documents pdf s'ha aprofitat la capacitat de l'OpenOffice.org per a generar directament la documentació en aquest format.

Bastides

En el capítol de disseny del programari ja es justifica perquè s'ha escollit cada bastida així que aquí només es llistaran les bastides o *frameworks* utilitzats pel Banc Electrònic d'Historials Mèdics.

- Per a l'accés a la base de dades Hibernate 3.
- Per serialitzar i deserialitzar objectes XStream 1.1.3.
- Llibreries criptogràfiques IAIK last release.

Instal·lació IAIK

Els passos per instal·lar IAIK són els següents:

1. descarregar l'última versió del JDK de SUN i instal·lar-lo. Cal vigilar que no s'utilitzi la màquina virtual de MS al compilar o executar.
2. descarregar l'última versió d'IAIK. Cal registrar-se, però no suposa cap cost. Si voleu descarregueu únicament l'arxiu iaik_jec_full.jar, que és la versió completa signada.
3. descarregar les polítiques de seguretat de java que permeten emprar qualsevol longitud de clau, Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 5.0 RC
4. (Windows) Copiar l'arxiu iaik_jce_full.jar als directoris:
 - c:\Archivos de Programa\Java\jdk1.5.0\jre\lib\ext
 - c:\Archivos de Programa\Java\jre1.5.0\lib\ext
5. (Linux) Copiar l'arxiu iaik_jce_full.jar al directori:
 - \$JAVA_HOME/jre/lib/ext
6. (Windows) dins de l'arxiu jce_policy-1.5.0-beta2.zip hi ha els arxius:
 - local_policy.jar
 - US_export_policy.jarCopiar-los a:
 - c:\Archivos de Programa\Java\jdk1.5.0\jre\lib\security
 - c:\Archivos de Programa\Java\jre1.5.0\lib\security
7. (Linux) dins de l'arxiu jce_policy-1.5.0-beta2.zip hi ha els arxius:
 - local_policy.jar
 - US_export_policy.jarCopiar-los a:
 - \$JAVA_HOME/jre/lib/security

8. Compilar i executar.

Public Key Infrastructure

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

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

Una PKI consta d'una autoritat de certificació notada amb CA, aquestes sigles corresponen al terme en anglès *Certification Authority*. Un altre component de la PKI són les autoritats de registre, notades amb les sigles RA (*Registry Authority*). Quan un usuari vol obtenir un certificat normalment realitza els passos següents. En un primer pas crea una parella de claus i realitza una petició de certificat mitjançant una RA. La RA valida la identitat de l'usuari que ha demanat el certificat i envia la petició a la CA.

La CA rep les peticions de les RA i emet els certificats. La clau privada de la CA és una peça d'informació molt sensible, i per això està en un entorn amb un alt nivell de seguretat.

En el nostre cas s'ha construït una petita PKI seguint els següents passos:

1. Generar la parella de claus de la CA (2048 bits).
2. Generar un certificat autosignat amb la parella de claus de la CA. Aquest serà el certificat de la CA.
3. Generar una parella de claus per l'usuari. La longitud de les claus pot ser 1024 bits.
4. Emetre una petició de certificat.
5. Emetre el certificat.
6. Generar l'arxiu PKCS12 que contindrà la parella de claus de l'usuari, el certificat de l'usuari, i el certificat de la CA.
7. Repetir els passos de 3 a 6 per generar el pkcs12 del gestor del sistema.

El resultat de la creació de la PKI es pot veure en el directori "pki" del projecte.

Configuració del Banc Electrònic d'Historials Clínics

Per tal que el programari funcioni correctament cal fer una preparació mínima del sistema on s'executarà.

SmartCard, i pkcs12 del gestor

Teòricament el client del Banc Electrònic d'Historials Clínics hauria de delegar a una SmartCard la signatura i el xifrat de dades, així es garanteix que en cap moment la

clau privada de l'usuari pot ésser copiada per cap codi maliciós. Al no disposar d'aquest dispositiu per la realització d'aquest projecte el que s'ha fet és llegir el pkcs12 de l'usuari i el certificat del gestor d'una hipotètica SmartCard, que en realitat és un directori del nostre sistema.

El servidor del Banc Electrònic d'Historials Clínics també necessita tenir el pkcs12 del gestor en un directori del sistema. Aquest directori és el que s'indiqui a la propietat "path" del fitxer de configuració serverP12.properties.

Així que cal tenir, i si no es pot crear, un directori que contingui:

- El pkcs12 de l'usuari en un fitxer anomenat usuari.p12.
- El certificat del gestor en un fitxer anomenat gestor.crt.
- Com que s'acostuma a executar l'aplicació client i l'aplicació servidor en el mateix ordinador també es pot posar el pkcs12 del gestor en aquest directori.

Fitxers de configuració del Banc Electrònic d'Historials Clínics

Fins aquí arriba la configuració essencial del sistema. Opcionalment existeixen una sèrie de fitxers de configuració que permeten modificar diferents característiques de l'aplicació.

Entrada/Sortida

Fitxer: edu.uoc.pfc.seguretat.es.servidor.p12loader.serverP12.properties.

Propietats:

path= indica on es troba el pkcs12 del gestor.

Fitxer: edu.uoc.pfc.seguretat.es.sockets.socket.properties

Propietats:

server.adress= adreça on el client ha de connectar amb el servidor.

server.port= port on el client ha de connectar amb l'aplicació servidora.

Interfície Gràfica d'Usuari

Fitxer: edu.uoc.pfc.seguretat.igu.aplic.elements.esp_mediques.ca_ES.properties

Propietats:

Enumeració de les especialitats mèdiques que reconeix l'aplicació.

Motor Servidor

Fitxer: edu.uoc.pfc.seguretat.motorservidor.servicedispatcher.services-roles.properties

Propietats:

En aquest fitxer s'hi defineixen les els rols que poden executar cada servei del servidor. Les claus de cada propietat són el nom de cada servei i els valors de cada servei són els rols que el poden executar.

Fitxers de configuració d'Hibernate

El programari intermediari, o *middleware*, utilitzat per accedir a la base de dades també disposa d'un fitxer de configuració on se li han d'indicar certs paràmetres.

Aquest fitxer de configuració s'ha de trobar al *classpath* d'execució de l'aplicació, per això es troba a l'arrel del codi de l'aplicació. Així que es pot trobar una còpia del mateix fitxer al directori *src*, una altra al directori *bin*, i una altra al directori *project/PFCservidor/persistencia*.

Per simplificar l'explicació es mostra tot seguit un exemple de fitxer de configuració i a continuació es fa l'explicació de la configuració que s'ha dut a terme, de manera que fer configuracions diferents es converteix en una tasca trivial.

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

    <session-factory>

        <!-- Database connection settings -->
        <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.url">jdbc:mysql://localhost/db_behc?user=gestorbehc</property>
        <property name="connection.username">gestorbehc</property>
        <property name="connection.password">behc</property>

        <!-- JDBC connection pool (use the built-in) -->
        <property name="connection.pool_size">1</property>

        <!-- SQL dialect -->
        <property name="dialect">org.hibernate.dialect.MySQLInnoDBDialect</property>

        <!-- Enable Hibernate's automatic session context management -->
        <property name="current_session_context_class">thread</property>

        <!-- Disable the second-level cache -->
        <property name="cache.provider_class">org.hibernate.cache.NoCacheProvider</property>

        <!-- Echo all executed SQL to stdout -->
        <property name="show_sql">>true</property>

        <!-- Drop and re-create the database schema on startup -->
        <!-- property name="hbm2ddl.auto">create</property -->

        <mapping resource="edu/uoc/pfc/seguretad/persistencia/beans/UsuariDo.hbm.xml"/>
        <mapping resource="edu/uoc/pfc/seguretad/persistencia/beans/SignaturaHistorialDo.hbm.xml"/>
        <mapping resource="edu/uoc/pfc/seguretad/persistencia/beans/HistorialDo.hbm.xml"/>
        <mapping resource="edu/uoc/pfc/seguretad/persistencia/beans/SignaturaCasClinicDo.hbm.xml"/>
        <mapping resource="edu/uoc/pfc/seguretad/persistencia/beans/CasClinicDo.hbm.xml"/>
        <mapping resource="edu/uoc/pfc/seguretad/persistencia/beans/AnotacioDo.hbm.xml"/>
        <mapping resource="edu/uoc/pfc/seguretad/persistencia/beans/SignaturaAnotacioDo.hbm.xml"/>

    </session-factory>

</hibernate-configuration>
```

El primer que es pot observar és la configuració de la connexió a la base de dades que ha d'utilitzar Hibernate.

Els paràmetres que s'hi indiquen són els següents:

- *connection.driver_class* defineix el driver a utilitzar per connectar amb el SGBD, a l'exemple MySQL.
- *connection.url* defineix la url a on accedir a la base de dades.

- *connection.username* és l'usuari amb què es vol connectar a l'SGBD.
- *connection.password* és la clau de l'usuari per autenticar-lo davant l'SGBD.

El següent paràmetre que ens interessa és el que informa a Hibernate de quin dialecte SQL cal que utilitzi per crear les sentències que envia a l'SGBD. A l'exemple s'hi ha indicat el dialecte de l'SGBD MySql.

Com que no es vol que sigui Hibernate qui s'encarregui de crear i mantenir l'esquema de la base de dades segons els objectes que manipula s'ha comentat la propietat *hbm2ddl.auto*.

Finalment hi ha el llistat del conjunt de fitxers que defineixen la correspondència entre cada objecte que ha de persistir i el model de dades de la base de dades.

Creació de la base de dades

Per poder desar i consultar dades al SGBD MySql cal crear la base de dades de l'aplicació.

El primer que s'ha de fer és entrar en el SGBD amb un usuari amb prous privilegis per crear una base de dades nova i un usuari que pugui inserir i consultar en aquesta base de dades.

Una vegada s'hagi entrat en el SGBD es començarà per crear la base de dades. Per fer això cal executar la sentència

```
CREATE DATABASE bd_behc;
```

Tot seguit cal crear l'usuari que utilitzarà l'aplicació per fer la persistència de les dades. La sentència que utilitzarem és

```
CREAE USER gestorbehc IDENTIFIED BY behc;
```

on s'ha creat el nou usuari *gestorbehc*. Alhora s'ha assignat la clau d'autenticació *behc* a aquest usuari.

Una vegada creat aquest usuari se li han de donar el privilegis necessaris perquè pugui fer altes i consultes a la base de dades *bd_behc*. Com a mesura de seguretat no se li donaran més privilegis a aquest usuari perquè així no es pugui utilitzar per manipular les dades mitjançant actualitzacions o esborrats. La sentència sql que s'utilitzarà és

```
GRANT INSERT, SELECT ON db_behc.* TO gestorbehc;
```

En aquest punt l'entorn de l'SGBD ja es troba preparat per poder instanciar la base de dades de l'aplicació.

Els guions sql, tot i que no hi corresponen estrictament, s'han desat al directori doc de la memòria. Encara que també es poden trobar a la seva localització original dins el directori project de el memòria. Concretament dins el codi del PFCservidor, al directori database hi ha tres fitxers:

1. dadesInicials.sql
2. foreignKeys.sql
3. modelDades.sql

Com es pot intuir pel seu propi nom dins modelDades.sql s'hi pot trobar el model de dades relacional de l'aplicació, o el que és el mateix, les sentències sql de creació de la base de dades.

Dins el fitxer foreignKeys.sql s'hi pot trobar el conjunt de sentències sql que estableixen les restriccions i associacions existents entre les diferents entitats del model relacional.

Finalment en el fitxer dadesInicials.sql s'hi poden trobar el conjunt de dades que cal que ja estiguin informades quan l'aplicació comenci a funcionar per primera vegada. En realitat aquest fitxer també és un guió sql consistent en un conjunt de sentències insert que van inserint les dades a les taules que toqui.

Quan s'hagin executat aquests tres fitxers, en aquest mateix ordre, la base de dades ja estarà a punt per començar a ser utilitzada pel Banc Electrònic d'Historials Clínics.

VII. Manual

Aplicatiu Client

Al arrancar l'aplicatiu client apareix una finestra splash on es presenta l'aplicació, allí caldrà indicar-hi quin és el directori que simula ser la smart card.

Una vegada comprovat el contingut del directori smart card apareixerà la finestra de l'aplicació. Aquesta finestra es distribueix en tres parts. La combo box de selecció de rol, el panell d'opcions i el panell de treball.

A dalt a l'esquerra s'hi troba una combo box on l'usuari escollirà el rol amb el que vol treballar. Una vegada escollit el rol es mostraran les accions que es poden dur a terme.

A dalt a la dreta, justament al costat de la combo box de selecció de rol, hi ha el panell amb les opcions que es poden dur a terme. Aquestes aniran canviant segons el rol que s'estigui exercint en cada moment. L'usuari accedirà a les seves funcionalitats a través de les opcions, que seran mostrades mitjançant botons.

Tota la funcionalitat de l'aplicació client es durà a terme a l'àrea de treball. Aquesta consisteix en un panell on l'usuari interactuarà amb l'aplicatiu per dur a terme les accions que desitgi.

Per cada acció que vulgui fer un usuari se li demanarà la clau del pkcs12. Això és així per dos motius de seguretat. En primer lloc si es mantingués la clau en memòria durant tota la sessió s'estarien donant facilitats a possibles programes d'espionatge que podrien escombrar la memòria de l'ordinador i robar la clau, nosaltres mantenim la clau en memòria en mínim temps possible. En segon lloc és habitual que un usuari, tan metge com pacient, deixi el seu lloc de treball temporalment i que per tant quedi l'aplicació oberta i accessible a terceres persones. Fent que s'hagi d'entrar la clau del pkcs12 per cada acció que vulguem fer estem evitant que hi hagi intrusos que puguin utilitzar l'aplicació en absència de l'usuari. En definitiva s'ha preferit sacrificar una mica d'agilitat per millorar la seguretat.

L'usuari metge podrà dur a terme l'alta de nous historials i podrà fer consultes de les dades personals del pacient i podrà fer consultes de tot l'historial complet d'un pacient. Una vegada consultat un historial complet també serà possible obrir-hi casos clínics o bé afegir anotacions a aquests casos clínics.

L'usuari pacient només podrà consultar el seu propi historial clínic.

Per poder iniciar una acció només caldrà que es polsi el botó corresponent del panell d'opcions. Això farà que a l'àrea de treball aparegui el formulari perquè s'hi entrin les dades necessàries per poder dur a terme l'acció. En aquest punt l'usuari haurà d'entrar les dades demanades en el formulari i, abaix de tot del formulari, prémer al botó corresponent per iniciar l'acció.

En qualsevol cas, tan per l'usuari metge com per l'usuari pacient, s'han pensat unes interfícies molt clares i autoexplicatives, on cada camp d'entrada de dades té la seva

corresponent etiqueta.

També s'ha fet un gran esforç perquè els errors siguin entenedors per l'usuari i sempre que ha sigut possible s'ha procurat que suggereixin la seva solució.

Aplicatiu Servidor

L'aplicatiu servidor només requereix la interacció de l'usuari per la seva arrancada, com a servidor que és.

Com que l'aplicatiu servidor utilitza un sistema de gestió de bases de dades perquè funcioni correctament també cal que el sistema de gestió de bases de dades estigui engegat.

Per fer l'alta de nous usuaris existeix una classe al codi font de persistència de la part del servidor que permet donar d'alta a un nou usuari en el sistema i alhora carregar el seu certificat. Concretament es fa l'alta a partir de les dades del certificat de l'usuari que es vol donar d'alta, no cal afegir cap més dada ja que totes les necessàries s'han de trobar informades en el propi certificat.

Aquesta classe s'anomena *edu.uoc.pfc.seguretat.utilitats.AltaUsuari* i requereix la localització del certificat de l'usuari que es vol afegir a la base de dades com a únic paràmetre. La classe inserirà el certificat i les dades que contingui a la base de dades per tal de poder validar les signatures dels missatges que els usuaris enviïn. És molt simple executar aquesta classe des d'un entorn integrat de desenvolupament com ara l'Eclipse, però també és senzill executar-la des de la línia de comandos.

VIII. Treball Futur

Corregir la comunicació per socket

En cas que intervingui una màquina amb el sistema operatiu Windows en un dels costats de la comunicació client-servidor es produeix un problema de comprensió de dades per part de les llibreries criptogràfiques detectat a última hora. L'aplicatiu funciona perfectament si s'executa en un ordinador amb sistema operatiu Linux o Mac OS-X. Com que aquest error no s'ha detectat fins a l'últim moment no s'ha pogut corregir.

Afegir actors al sistema

Com ja s'ha comentat existeix la possibilitat d'ampliar el programari afegint-li les funcionalitats corresponents a l'actor especialista. S'entén per especialista a la persona que du a terme alguna acció sanitària sobre el pacient i que no és metge, com per exemple el infermers, el fisioterapeutes, els dietistes, etc.



Concretament la feina s'hauria de fer a la interfície gràfica d'usuari perquè a la part servidor només associant el rol especialista als serveis als que s'hi vulgui donar permís ja n'hi ha prou.

Ampliar el contingut de l'Historial Clínic

Una altra suggerència que es fa per a un treball futur és afegir la gestió dels annexes de l'Historial Clínic. Aquesta tasca pot suposar força feina segons com s'enfoqui.

Una suggerència és obligar a que tots els annexes siguin fitxers d'imatges d'un tipus determinat. Això que sembla una restricció és, en realitat, una opció força flexible ja que permetria incloure a l'història radiografies, tot tipus de documents escanejats, tacs, i d'altres resultat d'anàlisis i proves.

Tot i la flexibilitat de l'opció anterior s'ha de tenir en compte que hi ha resultats que s'obtenen en formats multimèdia. Per exemple ecografies, telecomandaments, trànsits

intestinals, etc.

Validar Certificats

Cal comprovar que els certificats amb què es firmen els missatges i es xifren les dades siguin vàlids, no estiguin caducats i que tampoc no estiguin revocats.

Alhora també queda pendent la validació de la cadena d'autorització dels certificats, és a dir cal comprovar que l'arbre d'autoritzacions del que penja un certificat arriba a una autoritat certificadora reconeguda pel Ban Electrònic d'Historials Mèdics.

Auditories al sistema

Per tal de poder dur a terme auditories del sistema caldran poques modificacions. Ja s'estan desant les peticions d'alta de les diferents parts de l'historial, per poder tenir un control complet de totes les accions que es fan sobre cada historial només caldrà afegir una taula on desar-hi aquestes peticions de consulta.

Per raons d'eficiència s'utilitza una taula per desar-hi cada CLOB, això vol dir que tindrem una taula d'auditoria per les consultes a la taula d'historials, una altra taula d'auditoria per les consultes a la taula de casos clínics i finalment una altra taula d'auditoria per les consultes a la taula d'anotacions. Si s'ampliés el programari per permetre el treball amb dades annexes als historials aleshores caldrien dues taules més d'auditoria a més de la d'annexes. En una d'aquestes taules s'hi desarien les de signatures d'alta i a l'altra les peticions de consulta.

IX. Conclusions

A l'hora de desenvolupar aquest programari s'ha fet sempre pensant, per sobre de tot, en un producte de qualitat. Per aconseguir-ho, s'ha seguit una metodologia àmpliament provada i acceptada dins l'enginyeria del programari: es tracta del Rational Unified Process.

Una vegada finalitzat el projecte es pot afirmar que s'han aconseguit finalitzar convenientment les fases que el composaven dins les dates previstes.

L'anàlisi ha consistit en la documentació de forma àmplia del format dels historials, del seu contingut i de les lleis que hi fan referència. A partir d'aquí, s'ha pensat en els casos d'ús dels actors que tindrà el sistema, en el model de dades i en el disseny de la interfície. Alhora, s'ha preparat una configuració del programari que faciliti al màxim el desenvolupament de l'aplicatiu.

En el disseny, s'ha partit de la informació generada durant l'anàlisi. S'ha adaptat el model de dades al llenguatge Sql, concretament del dialecte de MySql; s'ha estudiat un desenvolupament client-servidor que doni resposta als requisits dels casos d'ús i, a partir d'aquesta arquitectura pel projecte, s'han dissenyat les classes que formaran el programari client i les que formaran el programari servidor.

La implementació s'ha fet basant-se en el disseny anterior. Pel que fa al programari servidor, s'ha utilitzat el sistema dels testos unitaris que la bastida, o *framework*, *jUnit*, ha ajudat a implementar.

Finalment, s'ha testejat el funcionament general de l'aplicatiu a través de varis usuaris voluntaris.

El resultat del projecte és un aplicatiu que permet a l'usuari fer un treball productiu alhora que altament segur.

Si recordem els objectius que s'havien plantejat al principi de tot del projecte i els contrastem amb el resultat de tot el desenvolupament veurem que s'han aconseguit amb escreix aquests objectius.

El primer objectiu que era permetre la gestió dels historials des de qualsevol ordinador amb accés a Internet s'ha assolit ja que mitjançant l'aplicatiu client l'usuari pot connectar-se al servidor a través d'Internet i fer-li peticions de consulta i d'alta sobre un historial concret. El programari BEHC també respecta les lleis referents als historials clínics perquè no permet fer alteracions als historial si no és per afegir-hi més dades, mai es poden modificar les dades que ja conté l'historial. D'altra banda, com que es desen les signatures de totes les peticions d'alta que es fan a l'aplicació, sempre es pot saber qui és l'autor de cada part de l'historial.

Pel que fa al segon objectiu també podem afirmar que s'ha assolit. La comunicació entre l'usuari i el banc d'historials ha de ser segura. I ho és, perquè BEHC no permet que hi hagi usuaris que facin peticions de cap tipus al servidor si no s'han autenticat abans i a

més l'intercanvi de dades entre usuari i servidor és sempre xifrada.

L'autenticació s'aconsegueix perquè el servidor no executa cap servei fins que no està completament segur que l'usuari que fa la petició és qui diu ser, sempre segons els protocols criptogràfics.

La confidencialitat de les dades s'aconsegueix gràcies al fet que aquestes viatgen xifrades per la xarxa i com que s'utilitza criptografia de clau pública amb claus de 2048kB els missatges queden blindats de cares als possibles atacs de subtracció d'informació.

El resultat de tot el procés que ha suposat aquest projecte és un prototip d'una aplicació que ajuda al maneig d'informació dins de l'àmbit sanitari. Aquest prototip demostra clarament que les TI es poden integrar en l'àmbit de la gestió de la informació sanitària sense que això suposi cap renúncia a les lleis que fan referència a aquesta informació. I no només això sinó que al integrar-s'hi poden ajudar a millorar la gestió que s'està fent actualment d'aquesta informació. En aquests moments la informació dels historials clínics es troba dispersa per les diferents institucions, fins i tot un mateix historial s'acostuma a trobar dispers per les diferents institucions sanitàries.

El que permet una aplicació com el BEHC és centralitzar la gestió de la informació de cada historial clínic, facilitant la consulta de la totalitat d'aquesta informació i donant-hi accés als usuaris que tinguin dret a consultar-la.

En el capítol "Treball Futur" es donen una sèrie de suggerències de les millores que més a curt termini es poden anar aplicant al projecte. Però aquesta llista no s'acaba aquí. Perquè un projecte com aquest sigui plenament funcional calen moltes més ampliacions i millores.

En resum, podem dir que aquest projecte és un primer maó cap a la creació d'una solució tecnològica als actuals sistemes de maneig de la informació dels historials clínics que provoquen l'actual dispersió de les dades i lentitud de funcionament del sistema públic.

X.Glossari

Anamnesi: Interrogatori a que el metge sotmet al pacient per tal de descobrir símptomes, antecedents, intervencions i tota mena de dades clíniques del pacient.

Annexes: Conjunt de documents que formen part de l'expedient, però que no són ni dades personals, ni anamnesi, ni exploracions físiques, ni diagnòstics, ni tractaments, ni cursos clínics, ni documentació d'especialistes. Com ara tot tipus de radiografies, anàlisis, consentiments d'operació, etc.

Curs Clínic: Seguiment documentat que es fa d'un tractament per deixar constància de l'evolució del pacient durant el tractament.

Documentació dels especialistes: Anotacions que els especialistes (infermeria, fisioteràpia, etc.) fan a l'expedient.

Exploració física: Consisteix en la inspecció física del pacient per determinar-ne les constants vitals i l'estat dels diferents sistemes i aparells. Les tècniques utilitzades són observació, palpació, percussió i auscultació.

Història clínica: "La història clínica és un conjunt de documents sorgits de la relació entre el metge i el pacient, o en terminologia actual entre un usuari i el centre d'atenció sanitària. La història clínica és l'únic document vàlid des del punt de vista clínic i legal. A més de les dades clíniques que tinguin relació amb la situació del pacient, el seu procés evolutiu, tractament i recuperació, la història clínica no es limita a ser una narració o exposició dels fets simplement si no que inclou judicis, documents, procediments, informacions i consentiment del pacient. És un document que es va fent amb el temps, documentant fonamentalment la relació metge-pacient." Enciclopèdia lliure universal en espanyol [<http://enciclopedia.us?uc.es>].

Orientació Diagnòstica: Determinació d'una malaltia que el metge fa segons les dades i l'exploració del malalt.

Tractament: Conjunt d'actuacions que es fan sobre un pacient a partir d'un diagnòstic.

XI. Bibliografia

Enciclopèdia lliure universal en espanyol [<http://enciclopedia.us?uc.es>]

Pàgina en xarxa de l'hospital Nostra Senyora de Maritxell

es.wikipedia.org entrada: historia clínica

Ley General de Sanidad

http://www.juridicas.com/base_datos/Admin/l14-1986.html

Housley, R., "Cryptographic Message Syntax", RFC 2630, Juny 1999.

Myers, M., Adams, C., Solo, D. and D. Kemp, "Internet X.509 Certificate Request Message Format", RFC 2511, Març 1999.

B. Kaliski, "PKCS 3: Diffie-Hellman Key Agreement v1.4"

H. Prafullchandra, J. Schaad, "Diffie-Hellman Proof-of-Possession Algorithms", Work in Progress.

Krawczyk, H., Bellare, M. and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, Febrer 1997.

Kaliski, B., "PKCS #1: RSA Encryption, Version 1.5", RFC 2313, Març 1998.

Kaliski, B., "PKCS #7: Cryptographic Message Syntax v1.5", RFC 2315, Octubre 1997.

RSA Laboratories, "PKCS#8: Private-Key Information Syntax Standard, Version 1.2", Novembre 1, 1993.

Kaliski, B., "PKCS #10: Certification Request Syntax v1.5", RFC 2314, Octubre 1997.

Housley, R., Ford, W., Polk, W. and D. Solo "Internet X.509 Public Key Infrastructure Certificate and CRL Profile", RFC 2459, Gener 1999.

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, Març 1997.

Dusse, S., Hoffman, P., Ramsdell, B., Lundblade, L. and L. Repka, "S/MIME Version 2 Message Specification", RFC 2311, Març 1998.

Ramsdell, B., "S/MIME Version 3 Message Specification", RFC 2633, Juny 1999.

Rescorla, E., "Diffie-Hellman Key Agreement Method", RFC 2631, Juny 1999.