Escenari segur client-servidor amb un proveïdor d'identitat i control d'accés extern

Memòria

Treball de Final de Carrera
U.O.C. - Enginyeria Tècnica d'Informàtica de Sistemes

Autor: *Sergi Collado i Figueras*
Consultor: *Carlos Ares Angulo*

12/01/2009

Aquest treball està dedicat a la Txell, la meva esposa, per la seva paciència i suport.

3. Resum

En aquest treball de fi de carrera s'intentarà implementar un escenari segur *client-servidor* que utilitzi un *proveïdor d'identitat* extern per efectuar les validacions relatives a la identitat i permisos associats d'un *client* a l'hora d'accedir a un *recurs*, allotjat en un *proveïdor de serveis*. D'aquesta forma, el *proveïdor de serveis* acabarà facilitant o no l'accés al *recurs* per part del *client*, però sense tenir cap dada que l'identifiqui, sinó exclusivament la *resolució* (petició acceptada o denegada) per part del *proveïdor d'identitat*.

Per tal d'implementar aquest escenari, es treballarà en camps com la modelització de dades seguint una infraestructura de criptografia de clau pública, i també s'intentarà proveir el producte amb un cert grau d'aplicació real mitjançant el fet d'utilitzar el *DNI electrònic* com a font dels certificats per a que els *clients* s'identifiquin i signin digitalment les seves peticions als recursos dels *proveïdors de serveis*.

Així doncs, serà necessari investigar les tecnologies necessàries per dur a terme aquest treball, i ésser capaç de portar-les a la pràctica de forma efectiva.

4. Índex

5. Memòria

5.1. Introducció

5.1.1. Justificació del TFC

El treball de fi de carrera (TFC) és una assignatura pensada per realitzar un treball de síntesi dels coneixements adquirits en altres assignatures de la carrera i que requereix posar-los en pràctica conjuntament en un treball concret. Normalment el TFC és un treball eminentment pràctic i vinculat a l'exercici professional de la informàtica, que mostra que s'han assolit els objectius d'aprenentatge en l'àmbit general de la Enginyeria Tècnica Informàtica.
El camp de la Criptografia és cada cop més necessari per tal de protegir les transaccions d'informació sensible que circulen a través de xarxes o sistemes informàtics. Actualment, cada cop és més necessari poder efectuar operacions, mitjançant canals segurs, que requereixen d'una autenticació robusta d'identitat des de qualsevol punt del planeta (possibles gràcies a Internet). Tant per poder garantir que certa informació no serà accessible per usuaris no autoritzats, com per poder garantir la legitimitat en emissors i receptors de la informació, esdevenen necessaris certs mecanismes criptogràfics.

Concretament, el camp de la criptografia de clau pública ha esdevingut una eina indispensable per poder implementar aquells mecanismes que fan possible la identificació inequívoca i irrefutable d'un individu o entitat.

Actualment, la societat ha efectuat un pas més en vers a la implementació d'aquests mecanismes d'identificació digital mitjançant la instauració del DNI electrònic. En aquest DNI es disposa d'un element electrònic que proveeix, mitjançant un lector adequat, del certificat d'identitat i de signatura digital de l'individu al que representa, possibilitant així una forma d'identificació prou robusta, que permet efectuar certes operacions que, fins ara, requerien de formes tradicionals d'identificació de individus.

La tecnologia Java, com a llenguatge de desenvolupament, juntament amb l'arquitectura J2EE esdevenen un estàndard en la implementació d'aplicacions web que permetin la interacció dels individus i els seus certificats amb les entitats proveïdores d'algun servei que requereixi d'aquest tipus d'identificació.

És per això que aquesta àrea de treball de final de carrera constitueix una proposta per a que l'estudiant s'introdueixi o aprofundeixi en el món del desenvolupament, mitjançant Java, d'aplicacions i serveis web que permetin la identificació d'usuaris mitjançant certificats, realitzant un treball pràctic amb tecnologies concretes i necessàries per tal d'efectuar un treball el màxim d'ajustat a la realitat: *Axis, SOAP, applets, Tomcat, etc*.

5.1.2. Experiència prèvia

Es parteix d'una experiència força bàsica en la programació d'entorns Java (únicament la obtinguda en l'assignatura "Criptografia") i interaccions amb serveis i aplicacions web. És per aquest motiu pel que cal posar especiar èmfasi a la necessitat d'una etapa inicial de presa de contacte i d'estudi de les tecnologies necessàries pel desenvolupament dels objectius proposats en el *TFC*.

5.1.3. Objectius del TFC

5.1.3.1. Objectius de l'assignatura

Dins dels coneixements adquirits en l'àmbit de les assignatures de la carrera, en el TFC es posa especial èmfasi en objectius genèrics concrets:

- Correcta aplicació de metodologies de treball.
- Anàlisi dels requeriments proposats (cercant informació i adquirint els coneixements de les tecnologies necessàries).
- Formulació d'objectius i definició de l'abast de la feina per l'elaboració d'un pla de treball.
- Aprenentatge en la configuració i ús de les tecnologies i components de software que intervenen en el treball.
- Documentació del disseny de manera formal.
- Implementació del producte final.
- Elaboració d'una memòria estructurada on es reculli la feina realitzada, així com les decisions preses, els resultats obtinguts i la documentació necessària per utilitzar el producte.
- Preparar una presentació que mostri de forma resumida i clara la feina realitzada.

I per les característiques de l'àrea del TFC es fa èmfasi en els següents objectius:

- Capturar els requisits que justifiquen la necessitat d'aplicar mecanismes de seguretat.
- Valorar la idoneïtat dels mecanismes escollits.

5.1.3.2. Abastament del projecte

L'abastament del projecte és el desenvolupament amb èxit d'un *escenari segur client-servidor amb un proveïdor d'identitat i control d'accés extern*.

S'implementaran dues entitats que interaccionin a la web i que representaran els rols de *proveïdor de servei* i *proveïdor d'identitat*. Caldrà implementar, doncs, l'escenari amb les garanties de seguretat que aporten les signatures digitals generades amb els certificats X.509, així com la resta d'elements d'una infraestructura de clau pública. Caldrà implementar també una tercera entitat, corresponent al *client*, capaç de connectar mitjançant un canal segur (per protegir les dades de la comunicació i per autenticar la identitat del servidor) per sol·licitar certs recursos dins la web que, en aquest cas, es tractaran de l'accés a una web de descàrrega de fitxers.

Aquesta implementació es durà a terme íntegrament en Java, utilitzant i investigant sobre els estris necessaris per tal d'assolir l'objectiu principal amb èxit.

- *Client:*

  Utilitza un navegador web per contactar amb els dos proveïdors. El component web que s'enviarà des del *proveïdor de servei* s'implementarà com un *applet Java* (executat en el navegador client) que haurà de ser capaç de disposar d'un certificat de reconeguda confiança per poder efectuar signatures (un dels certificats instal·lats en el navegador del client), així com comunicar-se mitjançant missatges *SOAP* amb el *proveïdor d'identitat*.

- *Proveïdor de servei:*

  Es tracta d'una aplicació de descàrregues de fitxers, que s'allotjarà en un servidor web i serà capaç de mantenir l'estat d'una sessió. Ha de poder establir connexions segures sense autenticar al *client* (li caldrà un *certificat digital* per tal de poder establir aquestes connexions, i tanmateix per generar signatures sobre les dades de la petició del client). També implementarà la lògica necessària per enviar a una *autoritat de segells de temps* la signatura per obtenir una garantia de la data de petició (per ser més concrets cal dir que en realitat s'utilitzarà un servei web, com el *servei DR de TrustedX*, que efectuarà la feina d'actualitzar les signatures). Finalment, ha de ser capaç de verificar les signatures que rebi del client signades pel *proveïdor d'identitat*.

- *Proveïdor d'identitat:*

  Per una banda es tractarà d'una aplicació web per a navegadors on els administradors (que hauran de poder autenticar-se d'alguna forma) disposaran d'un canal *SSL* segur per gestionar la informació de *proveïdors de serveis*, *recursos* i *clients*. Per tant, es disposarà d'un *certificat digital* al servidor web.

  Per una altra banda, el *proveïdor d'identitat* també serà un servei web *SOAP* sense estat per respondre peticions de consulta d'accés a un servei per part d'un *client*. El servei web haurà de poder verificar les signatures de les altres dues parts sobre les dades de la petició, tenint en compte que la signatura del *proveïdor de servei* inclourà un *segell de temps* que ha de verificar (mitjançant el *servei DSV de TrustredX*). En quant a la signatura del *client*, caldrà validar també l'estat del certificat. En tots els casos cal tenir en compte que el certificat del signant sigui de confiança, és a dir, que es confia en la *CA emissora* (que signa el certificat del signant).

  Per poder generar una resposta, el servei haurà d'accedir a la informació de *clients* i *serveis* que l'administrador hagi donat d'alta, cercant els signants de la petició a partir dels certificats, utilitzant el seu propi certificat en retornar aquesta resposta signada.

El *proveïdor d'identitat* disposarà d'un nivell més de control dels accessos dels clients en base a la comprovació irrefutable de l'hora en que s'ha efectuat la petició al *proveïdor de servei*, i les hores a les quals el client pot tenir accés als recursos. Per tal de dur-ho a terme s'utilitzaran els *segells de temps* emesos per una autoritat *PKI* de tipus *TSA pública* i la verificació dels mateixos mitjançant un servei web, públic, de verificació (mitjançant els serveis webs públics comentats anteriorment).

Pel que fa als *certificats digitals*, totes tres entitats disposaran de certificats digitals X.509 per poder realitzar les signatures sobre les dades i poder efectuar les connexions *SSL* (en el cas dels dos servidors).

El certificat del *proveïdor d'identitat* es generarà mitjançant alguna eina de generació de certificats (*OpenSSL* ó *KeyTool*) i no serà necessària la existència de cap mecanisme de validació del mateix, ja que el *proveïdor de servei* únicament verificarà les signatures i comprovarà si han estat emeses per algú de confiança (haurà de comprovar si es autoritats emissores dels certificats son de confiança). El certificat del *proveïdor de servei* complirà els mateixos requisits que l'anterior, si bé podrà generar signatures que passaran per una *autoritat de segells de temps* externa.

Finalment, el certificat del client correspondrà al certificat digital del *DNI electrònic*. S'adquirirà un *lector de targetes* i un *DNI electrònic* per tal de poder dur a terme la implementació i les probes del producte, i serà requisit indispensable per la utilització de l'escenari final, ja que les

validacions de certificats s'efectuaran mitjançant el protocol *OCSP* cap a una autoritat de validació pública.

5.1.3.3. Objectius personals

S'espera assolir satisfactòriament les competències necessàries en els diferents aspectes que tracta aquest TFC:

- D'una banda, s'espera assolir el bagatge adequat per tal d'aprendre a organitzar les diferents fases d'elaboració d'un projecte i la posada en pràctica, fent èmfasi en les tasques de planificació, d'anàlisi i de disseny.

- D'altra banda, s'espera obtenir una certa experiència inicial en l'aplicació de la criptografia en un cas realista, així com en les eines (Java i llibreries utilitzades) que s'hauran necessitat.

- I finalment també s'espera que tot aquest bagatge assolit serveixi com a punt inicial en una possible reorientació professional en vers el camp de la seguretat informàtica o, en el seu defecte, una satisfactòria realització personal en aconseguir la fita de finalitzar el TFC amb èxit.

5.1.4. Enfocament i mètode seguit

5.1.4.1. Consideracions sobre la tecnologia a utilitzar

A efectes de planificació, caldrà tenir en compte que inicialment hi ha un seguit de tecnologies que cal escollir i amb les que també cal començar a familiaritzar-se per tal de dur a bon port la implementació del producte amb tots els seus requisits:

- *Plataforma de programació:*

  S'utilitzarà el software *Eclipse* com a plataforma de desenvolupament Java, en les seves versions sobre sistemes Windows i Linux indistintament. Les funcionalitats addicionals que puguin enllaçar aquesta plataforma amb alguns aspectes utilitzats en la implementació del producte, mitjançant *plugins*, caldrà estudiar-los sobre la marxa i el propi curs del desenvolupament.

- *Servidor d'aplicacions i serveis web:*

  La tecnologia que s'utilitzarà com a *servidor d'aplicacions* i de *serveis web* serà la proveïda per un servidor web *Apache Tomcat* (v. 6) per desplegar els components web. Aquest servidor es configurarà o bé sobre una plataforma *Linux* (*Ubuntu 8.04*) o bé sobre *Windows* (Vista 64bits). Caldrà estudiar el funcionament i configuració bàsics, així com els exemples de desplegament de components que s'inclouen en la instal·lació del *Apache Tomcat*. La funcionalitat SOAP necessària s'aconseguirà important en els projectes creats les llibreries Axis facilitades per l'assistent de creació de serveis web incorporat al software *Eclipse*.

- *Servidor de bases de dades:*

  Per tal de proveir persistència en les dades sobre els permisos i els horaris d'accés als recursos caldrà configurar un servidor *MySQL 5.1.*

- *Client:*

Tal com s'ha esmentat anteriorment, les funcionalitats d'accés al *proveïdor d'identitat* per part del client s'implementaran mitjançant un *applet Java* que s'executarà en el navegador. Caldrà efectuar un estudi sobre el funcionament general dels *applets Java* mitjançant la creació d'exemples senzills, així com la creació d'interfícies d'usuari simples mitjançant la llibreria *Swing* (doncs caldrà presentar a l'usuari una llista per tal de seleccionar el certificat a utilitzar). També caldrà investigar i estudiar la forma en que es signen els *applets* per tal d'adquirir suficients privilegis i poder accedir al magatzem de certificats del navegador (també adquirirà privilegis per connectar amb servidors diferents), estudiant també la forma en que *Java 6* incorpora funcions per accedir als *magatzems de claus natius de Windows* o als *PKCS#11* (en cas de *Linux*) i utilitzar els certificats del DNI electrònic. Finalment també serà necessari estudiar la forma en que es construiran i signaran digitalment els missatges que s'intercanviaran, mitjançant el protocol *SOAP*, amb el *proveïdor d'identitat* (definir quina serà la estructura del missatge *XML*, decidir si la signatura serà *ENVELOPED*, *ENVELOPING* o *DETACHED*, etc).

- *Servei web del proveïdor d'identitat:*

Un dels components del proveïdor d'identitat serà un servei web sense estat que haurà de ser capaç d'establir comunicació mitjançant missatges *SOAP*. Per efectuar aquesta tasca es començarà amb l'estudi de les API's *Axis*, *Axis2*, *JAX-WS* i *JAX-RPC* (aquest estudi també caldrà començar-lo per aplicar-lo a l'*applet Java*) i decidir quin s'ajusta millor als requeriments, tant de funcionalitat com de recursos de temps i corba d'aprenentatge, però anteriorment caldrà efectuar probes més senzilles per tal d'agafar familiaritat amb els *servlets* i *serveis web sense estat*. Tanmateix, de la mateixa forma que en l'estudi de les funcionalitats de l'*applet Java*, també caldrà estudiar la forma en la que es signen els missatges que s'enviaran a través del protocol *SOAP* i la forma en la que es verifiquen les signatures. Caldrà estudiar a fons el protocol *OCSP* i efectuar probes de validació amb un DNI electrònic autèntic (i potser amb algun que no ho sigui per tal de poder contemplar el comportament davant intents il·legítims d'accés). També caldrà estudiar la verificació de *segells de temps*. Finalment, caldrà estudiar l'aspecte de *persistència en Java* mitjançant l'API de *JDBC*, o bé algun *framework* tipus *Hibernate* (en aquest cas només es necessitaran efectuar operacions de consulta).

- *Aplicació web del proveïdor d'identitat:*

L'altre component del *proveïdor d'identitat* serà una aplicació web que proveeixi d'una interfície per a que els administradors es puguin identificar. En aquest sentit caldrà estudiar quina és la millor forma d'autenticació web dels administradors, així com la forma en que caldrà configurar el servidor d'aplicacions per tal d'establir el canal *SSL* amb el certificat del proveïdor d'identitat. Una aproximació inicial passarà per estudiar l'ús de *servlets* i *JSP* (havent descartat el *framework Spring* i els *JSF*, donada la complexitat inicial que suposen alguns dels conceptes que cal dominar per desenvolupar satisfactòriament en aquests components) per implementar l'aplicació web. En aquest cas, també caldrà prestar atenció a la *persistència* ja que l'aplicació efectuarà modificacions, consultes i eliminacions sobre les dades del servidor de bases de dades.

- *Aplicació web del proveïdor de serveis:*

Esdevindrà una *aplicació de descàrrega de fitxers*, que també facilitarà al client el component necessari (l'*applet Java*) per contactar amb el proveïdor de serveis. Aquest *applet Java* el facilitarà cada vegada que el client sol·liciti l'accés a un recurs de la secció de

descàrrega de fitxers. Tanmateix, caldrà estudiar la plataforma *TrustedX* per tal d'afegir correctament el *segell de temps* necessari (quan es produeix una sol•licitud) per a que el *proveïdor d'identitat* controli l'accés en front a la franja horària permesa. En cas que l'accés estigui permès, aquesta aplicació proveirà una plataforma de descàrrega de fitxers, pel recurs sol·licitat, per una via protegida (*SSL*), la qual cosa requerirà de la recerca i estudi sobre com dur-ho a terme d'una forma viable.

Com a última consideració a esmentar, cal tenir en compte que l'entorn de desenvolupament constarà d'un únic equip on hi residiran tant els *servidors d'aplicacions* i *serveis web* com de *bases de dades*, efectuant tot el desplegament sobre les mateixes instàncies de software. Però durant la fase de proves, i gràcies a la possibilitat de *virtualitzar* diferents màquines, es procurarà simular un context realista, on cada component del producte final (*client*, *proveïdor de servei*, i binomi *proveïdor d'identitat* més *servidor de base de dades*) es desplegarà en un element de xarxa independent. Eventualment fins i tot el *servidor de bases de dades* podria estar en un equip diferent, però aquest fet esdevindria irrellevant en el nostre cas, ja que qui necessita accedir-hi és únicament el *proveïdor d'identitat*, per la qual cosa es pot optar per instal·lar ambdós elements en el mateix dispositiu i mantenir el caire realista.

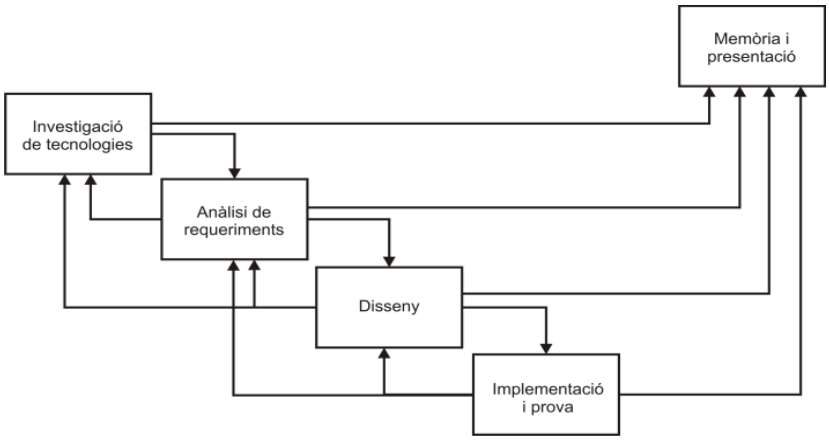
5.1.5. Planificació del projecte

5.1.5.1. Plantejament

Aquest projecte consta de molts aspectes tècnics sobre els que caldrà efectuar una extensa recerca d'informació i d'exemples, i que caldrà provar fins obtenir el resultat desitjat o, fins i tot en algun cas, la decisió final sobre si certa tecnologia és factible: per exemple, si *JDBC* esdevé insuficient o massa limitat per la implementació de la persistència necessària en alguns mòduls del producte és un fet que només es podrà conèixer un cop realitzades les proves i estudis pertinents (analitzant exemples o consultant manuals i documentació) i, en aquest cas, caldria emprendre una via alternativa (*Hibernate*) que també requeriria d'un nou temps d'estudi i de proves.

És per això que la part de recerca guanya un pes específic important en la planificació del treball (ja que els temps de recerca no son despreciables) i caldrà dur-la a terme, de forma paral·lela a la resta de fases, des del principi. Un cop "resolt" un aspecte tècnic es podrà implementar en la part corresponent del producte.

La resta de fases que conformaran la totalitat del TFC es duran a terme seguint un cicle de vida clàssic en cascada, amb certa retroalimentació en els casos que faci falta, donat que és la millor manera de plantejar un projecte com aquest amb uns recursos limitats de personal (1 única persona).



5.1.5.2. Planificació

Un primer enfocament a la planificació del TFC ens porta a diferenciar les següents fases:

ID	Nombre	Fecha de inicio	Fecha de fin	Duración	Antecesores
0	TFC - Seguretat Informàtica	18/09/08	20/01/09	124	
1	Fase d'arrancada	18/09/08	11/10/08	23	
5	Establiment de l'abastament del projecte	18/09/08	20/09/08	2	
6	Definició del pla de treball inicial del projecte	20/09/08	25/09/08	5	5
7	PAC1 - Pla de treball inicial [FITA]	25/09/08	26/09/08	1	6
8	Revisió del pla inicial	26/09/08	30/09/08	4	7
9	Definició del pla de treball definitiu	30/09/08	10/10/08	10	8
10	PAC2 - Pla de treball definitiu [FITA]	10/10/08	11/10/08	1	9
2	Fase de disseny	11/10/08	6/11/08	26	1
11	Ampliació de l'analisi del projecte	11/10/08	17/10/08	6	
12	Disseny de l'arquitectura del projecte	17/10/08	21/10/08	4	11
13	Disseny dels casos d'ús	21/10/08	27/10/08	6	12
14	Disseny de la persistència (BD)	27/10/08	1/11/08	5	13
15	Disseny de la interfície gràfica	1/11/08	5/11/08	4	14
16	PAC3 - Disseny del producte [FITA]	5/11/08	6/11/08	1	15
3	Fase d'implementació parcial	6/11/08	5/12/08	29	2
17	Preparació de l'entorn de programació i pre-producció	6/11/08	8/11/08	2	
18	Generació de codi	8/11/08	30/11/08	22	17
19	Realització de proves	30/11/08	2/12/08	2	18
20	Revisió de documentació d'usuari	2/12/08	4/12/08	2	19
21	PAC4 - Implementació parcial del producte [FITA]	4/12/08	5/12/08	1	20
4	Fase d'implementació final i redacció de la memòria	5/12/08	13/01/09	39	3
23	Revisió de codi i finalització d'aspectes pendents	5/12/08	12/01/09	38	
24	Síntesi de la documentació generada: cloenda de la memòria	5/12/08	12/01/09	38	23
25	LLIURAMENT - Memòria i producte [FITA]	12/01/09	13/01/09	1	24
22	Fase de tancament	13/01/09	20/01/09	7	4
26	Realització de la presentació del projecte	13/01/09	19/01/09	6	
27	LLIURAMENT - Presentació [FITA]	19/01/09	20/01/09	1	26

Per realitzar aquesta planificació s'ha utilitzat l'eina gratuïta *GanttProject* (amb funcionalitats similars al producte *MS-Project,* en quant a disseny de diagrames). Cal tenir en compte que aquest software tracta la *"data final"* com no inclusiva, i per tant el projecte tindria la seva finalització el dia 19/01/2009 enlloc del 20/01/2009 que es mostra en el diagrama. Cal tenir aquesta consideració en totes les dates que apareixen (com a referència, hom pot fixar-se en les *fites* que, tot i durar 1 dia, es marquen com a que finalitzen l'endemà de començar).

- Arrancada:

  [Del 18 de Setembre al 10 d'Octubre de 2008]
  Aquesta fase contempla tant el pla de treball inicial (fita PAC1) com el definitiu (fita PAC2) i les tasques de planificació i presa de contacte inicial amb les tecnologies involucrades en el TFC. Tanmateix també contempla una primera fase d'anàlisi corresponent al contingut esperat en l'entrega de la PAC2.

- Disseny:

  [de l'11 d'Octubre al 5 de Novembre de 2008]
  Aquesta fase contempla totes les tasques per efectuar la definició formal del funcionament del producte (definició de casos d'us, dels actors, la comunicació, el disseny de la BD, les interfícies gràfiques, etc). Coincidirà completament amb el període destinat a la fita PAC3. Inicialment caldrà refinar la tasca inicial a nivell d'anàlisi efectuada al final de

la fase anterior, i posteriorment es podran definir els diferents aspectes del disseny del producte (la seva arquitectura, els casos d'us i diagrames *UML* associats, l'estructura de la base de dades i les interfícies d'interacció amb l'usuari).

- Implementació parcial:

[del 6 de Novembre al 4 de Desembre de 2008]
Aquesta fase inclou totes les tasques d'implementació del producte final, i també aquelles proves i estudis que encara no s'hagin dut a terme sobre les tecnologies que cal aplicar. Coincidirà completament amb el període destinat a la PAC4. Serà en aquest punt on caldrà implementar el gruix de codi que conformarà el producte final, així com realitzar les primeres proves i documentació d'usuari. Cal dir que en aquesta fase s'espera assolir una concepció parcial del producte, on s'hagin assolit unes mínimes funcionalitats:

  - *Client*: Com a mínim serà capaç de treballar amb un certificat predefinit (facilitat directament per codi) d'algun tipus estàndard (generat amb *OpenSSL*). No s'espera incloure encara la funcionalitat d'accés al magatzem de certificats i accedir al certificat del *DNI-e*, i el *proveïdor d'identitat* haurà d'utilitzar un llistat de certificats de confiança per poder validar el certificat del client (i/o la CA simulada que l'emeti) en aquesta implementació parcial.
  - *Proveïdor de serveis*: Es disposarà d'un únic recurs, però s'implementarà gairebé tot el protocol complert de verificació de signatures i validació de certificats del *proveïdor d'identitat*, així com una interfície mínima per l'usuari (amb la part *JSP*, *Servlet* i *HTML* més bàsica). Es procurarà implementar també l'obtenció del segell de temps.
  - *Proveïdor d'identitat*: s'implementarà el servei web capaç d'interactuar amb el client mitjançant *SOAP*, i capaç també de validar un segell de temps. Les signatures del client es validaran simulant-ho sobre un llistat propi de certificats de confiança, per tal d'incloure més endavant el tractament de certificats i signatures de *DNI-e*. La interfície de manteniment de recursos i gestió dels administradors es deixarà per una entrega posterior.

Cal destacar que, donada la interacció que les diferents parts tenen entre sí, és complicat definir una delimitació sobre quina part s'implementarà abans que una altra, i el més probable és que la major part del temps es treballi sobre totes tres parts de forma més o menys simultània durant la tasca anomenada "*generació de codi*".

- Implementació final i redacció de la memòria:

[del 5 de Desembre de 2008 a 12 de Gener de 2009]
En aquesta fase s'inclouen tots els aspectes relacionats amb la finalització de la implementació del producte (i fase de proves final del mateix), així com la redacció dels apartats de la memòria relatius a les instruccions per la instal·lació i posada en marxa, i aquells aspectes que no s'hagin redactat en les fases anteriors. Coincidirà completament amb el període destinat a l'entrega final de la memòria i del producte amb la funcionalitat completa.

- Tancament:

[del 13 de Gener al 19 de Gener de 2009]
Finalment aquesta fase constituirà la redacció de la presentació del TFC per efectuar l'entrega al tribunal d'avaluació del mateix. Coincidirà completament amb el període destinat a l'entrega de la presentació.

Cal tenir en compte, tal com s'ha comentat anteriorment, que de forma paral·lela s'efectuaran tantes proves, investigacions i estudis d'exemples sobre les diferents tecnologies a aplicar com sigui possible, per tal d'arribar a la fase de implementació parcial amb el màxim d'aspectes treballats i agilitzar al màxim la implementació del producte final.

5.1.5.3. Fites

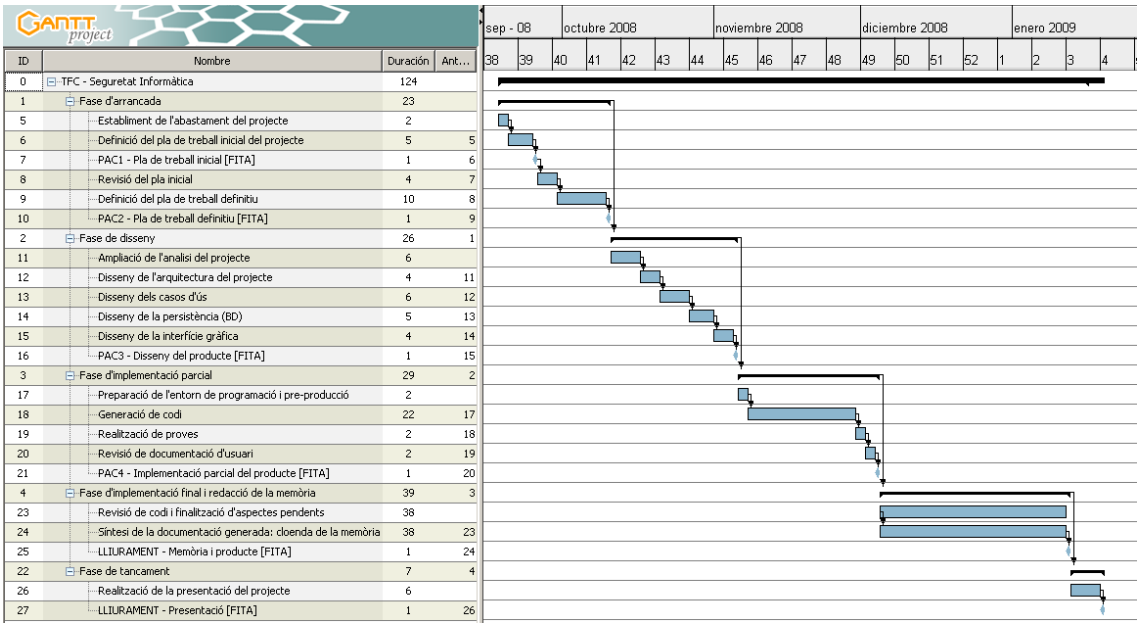
Com s'ha pogut observar de la planificació anterior, el transcurs de les accions que esdevindran necessàries per la correcta elaboració del projecte, en els temps estimats, s'han amollat a un esquema marcat per la prioritat d'assolir unes fites en unes dates concretes. Aquestes fites corresponen a les entregues de les diferents PACs de l'assignatura del TFC, juntament amb l'entrega final i l'entrega de la presentació:

- PAC1 – Pla de treball inicial: dia 25/09/2008
- PAC2 – Pla de treball definitiu: dia 10/10/2008
- PAC3 – Disseny del producte: dia 5/11/2008
- PAC4 – Implementació parcial del producte: dia 4/11/2008
- LLIURAMENT – Memòria i producte: dia 12/01/2009
- LLIURAMENT – Presentació: dia 19/01/2009

El caràcter inamovible d'aquestes fites és el que regiria una hipotètica mobilitat o dilatació en el temps de la resta de les tasques.

5.1.5.4. Cronograma

Finalment, el diagrama de Gantt resultant de la planificació de tasques, i que les desplega en la línia de temps, seria el següent:



ID	Nombre	Duración	Ant...
0	TFC - Seguretat Informàtica	124	
1	Fase d'arrancada	23	
5	Establiment de l'abastament del projecte	2	
6	Definició del pla de treball inicial del projecte	5	5
7	PAC1 - Pla de treball inicial [FITA]	1	6
8	Revisió del pla inicial	4	7
9	Definició del pla de treball definitiu	10	8
10	PAC2 - Pla de treball definitiu [FITA]	1	9
2	Fase de disseny	26	1
11	Ampliació de l'analisi del projecte	6	
12	Disseny de l'arquitectura del projecte	4	11
13	Disseny dels casos d'ús	6	12
14	Disseny de la persistència (BD)	5	13
15	Disseny de la interfície gràfica	4	14
16	PAC3 - Disseny del producte [FITA]	1	15
3	Fase d'implementació parcial	29	2
17	Preparació de l'entorn de programació i pre-producció	2	
18	Generació de codi	22	17
19	Realització de proves	2	18
20	Revisió de documentació d'usuari	2	19
21	PAC4 - Implementació parcial del producte [FITA]	1	20
4	Fase d'implementació final i redacció de la memòria	39	3
23	Revisió de codi i finalització d'aspectes pendents	38	
24	Síntesi de la documentació generada: cloenda de la memòria	38	23
25	LLIURAMENT - Memòria i producte [FITA]	1	24
22	Fase de tancament	7	4
26	Realització de la presentació del projecte	6	
27	LLIURAMENT - Presentació [FITA]	1	26

5.1.6. Productes obtinguts

Els productes que s'espera obtenir en aquest Treball de Final de Carrera son els següents:

- El pla de projecte (inclòs a la present memòria) que recull la planificació i estimació de les tasques necessàries per dur a terme els objectius previstos.
- El producte en sí, que serà el software desenvolupat més la documentació tècnica associada (manual d'usuari i de posada en funcionament, que també estaran inclosos a la present memòria).
- La presentació, que resumirà de forma concisa el treball realitzat i els resultats obtinguts.
- La present memòria, que és el document que sintetitza tot el treball realitzat al llarg del projecte, incloent la documentació resultant del seguiment de les diferents fases del projecte (pla de treball, anàlisi del projecte, disseny i construcció del sistema).

5.1.7. Descripció dels altres capítols de la memòria

En els següents capítols es definiran les parts crítiques pel disseny i implementació del sistema. En primer lloc, en el capítol *Fonaments i estat de l'art* es repassaran aquells conceptes que cal entendre i que s'han utilitzat en la concepció i implementació del projecte (com ara els relacionats a la tecnologia que s'ha utilitzat, els tipus estructures o la persistència).

Tot seguit, en el capítol *Visió general de l'arquitectura del projecte* es repassarà, a mode molt general, el comportament de cadascun dels components que conformen el sistema (*client*, *proveïdor de serveis* i *proveïdor d'identitat*), així com la descripció de la plataforma de treball i desenvolupament que s'ha utilitzat.

El següent capítol es dedicarà al *disseny del producte*, entrant en detall en el *disseny de l'arquitectura del projecte* (mitjançant diagrames *UML*) i en el *disseny de la persistència* (on es definiran les taules de la *base de dades*), del *format de dades XML*, i de la *interfície gràfica* (on es descriuran les diferents pantalles d'interacció amb l'usuari de les que constarà l'aplicació).

A continuació es dedicarà un capítol al *funcionament, instal·lació i posada en marxa del producte*, on es descriuran els passos necessaris per tal d'instal·lar el sistema i fer-lo funcionar, mitjançant exemples i captures de pantalla.

Seguidament es comentaran les possibles *variacions del producte final respecte el disseny inicial previst*, així com la presentació de les *conclusions i futures vies de treball*, on també es posaran de manifest les possibles millores que caldria dur a terme en cas de tractar-se d'un projecte comercial.

Finalment es presentaran els *glossaris* i *bibliografies/webgrafies* utilitzades, així com els annexos que s'hagi considerat oportú adjuntar (com ara la documentació tècnica del lector *USB* de *DNI-e* que s'ha utilitzat per efectuar les proves en la implementació d'aquest projecte.

5.2. Fonaments i estat de l'art

5.2.1. Infraestructura de clau pública (*PKI*)

A més de la tecnologia necessària per implementar l'escenari proposat com a objectiu del projecte, cal definir exhaustivament en primer lloc la tecnologia criptogràfica sobre la que es sustenta la comunicació que s'establirà entre els diferents elements d'aquest escenari: la infraestructura de clau pública (coneguda comunament amb les sigles de l'anglès *PKI*).

Aquesta infraestructura ens proveeix dels elements de seguretat necessaris per tal de garantir les identitats dels emissors dels missatges que s'intercanviaran en el procés, així com també els mecanismes per signar aquests missatges.

La base de la *infraestructura de clau pública* es troba en la utilització de dues claus, una de privada (que només ha de conèixer el propietari) i una de pública (que ha d'estar a l'abast de tothom). Aquestes dues claus tenen la particularitat de que un missatge xifrat amb una de les dues claus es pot desxifrar únicament amb l'altra clau associada. Per tant, un missatge xifrat amb la clau privada de l'emissor només pot ésser desxifrat amb la seva clau pública. Aquest fet, a més, afegeix la característica de no repudi en aquesta infraestructura, ja que si el missatge ha estat desxifrat amb èxit, l'emissor no pot negar la seva autoria.

Per tal de poder fer un ús eficient d'aquest mecanisme de xifrat, cal introduir el terme *signatura digital*, que consisteix en l'ús d'aquesta operació de xifrat per tal de vincular la identitat del signatari amb el document a signar, mitjançant un resum del document (normalment obtingut gràcies a una funció *hash*), i que a més en garanteix la integritat i el no repudi. Així doncs, en xifrar el resum d'un document amb la clau privada de l'emissor d'un missatge, el receptor té la possibilitat de recuperar aquest resum amb la clau pública de l'emissor, i comprovar, per tant, que el resum es correspon amb el document, garantint-ne així la integritat en cas d'haver estat modificat després d'haver estat signat.

Aquest mecanisme, però, presenta el problema de que la clau pública ha d'estar disponible per a tothom. Cal, doncs, introduir un nou terme: el de *certificat digital*. Un *certificat digital* és una estructura de dades que conté la informació necessària i referent al propietari d'un conjunt de claus criptogràfiques, a més de la clau pública i la signatura de tot plegat per part d'una *autoritat de certificació de confiança*. Per tant, es necessita establir uns mecanismes de confiança que garanteixin (de forma arbitraria però coneguda) si una *autoritat de certificació* és de confiança. Normalment existeix un conjunts d'*autoritats de certificació de confiança* conegudes i a disposició de qualsevol que necessiti verificar l'autenticitat d'un certificat.

5.2.2. XML

Aquest projecte, doncs, fa ús de la *infraestructura de clau pública* a l'hora de relacionar els diferents missatges, que esdevindran en la comunicació entre el component *client* que executarà l'usuari i el *servidor d'identitat* o el *servidor de servei,* amb el seu emissor.

Però aquests missatges necessitaran certa estructura que els encapsuli, i una de les millors que es pot utilitzar, segons el context en que s'elaborarà aquest projecte, és mitjançant estructures *XML*.

El *XML* (de l'anglès *Extensible Markup Language*) és un llenguatge de marques creat pel *World Wide Web Consorcium* (*W3C*) i que ha esdevingut un estàndard en el intercanvi d'informació estructurada. A més, gràcies a les especificacions contemplades per l'estàndard *XMLDSig* es

disposa de la infraestructura necessària per tal de signar aquests missatges de 3 formes diferents:

- *Enveloped*: on la signatura queda continguda en el cos del missatge.
- *Enveloping*: on la signatura esdevé l'embolcall del missatge (i l'acaba incloent).
- *Detached*: on el missatge i la signatura acaben conformant estructures independents.

5.2.3. Tecnologies web – clients

Un dels objectius d'aquest projecte és crear una estructura, dins de l'escenari plantejat, per tal que els usuaris puguin interactuar amb els elements necessaris per accedir als recursos del *proveïdor de serveis*.

Esdevé, doncs, imprescindible utilitzar alguna de les plataformes disponibles per tal d'establir el canal en el que s'interactuarà i, tal com s'havia recomanat en l'especificació inicial de requisits, s'ha optat per utilitzar les pàgines *HTML* i els *applets Java* com a tecnologia utilitzada.

El *HTML* (de l'anglès *Hyper Text Markup Language*) és un llenguatge de marques que va sorgir cap al 1990 fruït de la unificació de tecnologies anteriors sobre enllaços i estructures. L'estàndard actual, definit pel *W3C*, encara és la base de la comunicació i accés al *WWW* (*World Wide Web*), i admet diverses millores com ara la utilització de sentències senzilles en un llenguatge anomenat *javascript* que permet ampliar la funcionalitat de les interaccions d'una plana web amb els usuaris.

Els *applet Java* també ens permetran ampliar molt més l'abastament en la interacció amb l'usuari, gràcies a aquesta tecnologia basada en *Java* (que es definirà en el punt sobre la visió general de l'arquitectura del producte), fins al punt de poder obtenir un certificat del magatzem de certificats de l'usuari i operar amb ell, efectuant signatures digitals sobre dades, així com interactuar amb serveis web allotjats en sistemes diferents als que ens proveiran aquests *applets*.

Cal considerar també la tecnologia *SSL* (de l'anglès *Secure Socket Layer*) que afegeix components criptogràfics al protocol estàndard d'accés a documents *HTML.* Aquest protocol estàndard és el *HTTP* (de l'anglès *Hypertext Transfer Protocol*), però gràcies a la característica criptogràfica de xifrat de les connexions que li atorga *SSL* es pot parlar del protocol *HTTPS* (protocol *HTTP* segur).

5.2.4. Tecnologies web – servidors

Donat el caràcter dinàmic dels elements que composen aquest projecte, es necessitarà efectuar diverses operacions des de la banda dels servidors, tant del que executarà el rol de *proveïdor d'identitat* com de *proveïdor de serveis*. Es necessitarà, per tant, utilitzar una tecnologia *Java* que sigui capaç de processar les diferents sol·licituds de l'usuari (en vers al *proveïdor de serveis*) i els diferents intercanvis d'informació (entre el component *client* i el servei web del *proveïdor d'identitat*).

S'utilitzarà, doncs, la tecnologia *JSP* (de l'anglès *Java Server Pages*) i *servlets Java*, desenvolupades per *Sun Microsystems*, i que ens permetran generar de forma dinàmica tant els documents *HTML* que s'utilitzaran per interactuar amb l'usuari (tant en el *proveïdor de serveis* com en la part d'administració del *proveïdor d'identitat*), com totes aquelles operacions relatives a la infraestructura de clau pública que es duran a terme.

Per tal de poder implementar també els serveis web necessaris en el *proveïdor d'identitat* s'utilitzarà aquesta mateixa tecnologia complementada per les funcionalitats de les llibreries *Axis* i *JAX-RPC* que permetran la crida a aquests serveis mitjançant missatges *SOAP*.

5.2.5. SOAP

Cal fer una menció específica a aquest protocol anomenat *SOAP* (de l'anglès *Simple Object Access Protocol*) creat per *Microsoft*, *IBM* i d'altres i actualment sota els estàndards definits pel *W3C*. A diferència d'altres protocols d'interacció entre processos distribuïts, com ara *DCOM* o *CORBA*, que implementen aquesta interacció en mode binari, el protocol *SOAP* es sustenta sobre codi *XML* (i les últimes implementacions del mateix suporten la inclusió de documents adjunts binaris).

Com ja s'ha comentat anteriorment, les llibreries *Axis* i *JAX-RPC* ens proveiran de les interfícies necessàries per efectuar la comunicació mitjançant aquest protocol, en la seva versió 1.1, sense prestar massa atenció a la com s'han format aquests missatges (gràcies també als assistents que disposarem en l'entorn de desenvolupament i que ens facilitaran enormement la feina).

5.2.6. DNI-e i OCSP

Actualment, el document nacional d'identitat ha sofert importants actualitzacions en el seu format tradicional per tal d'implementar els mecanismes criptogràfics necessaris, mitjançant un circuït integrat (xip) en el mateix document, que permetin la identificació electrònica indubtable de qualsevol ciutadà. A més, aquests mecanismes permeten que el ciutadà signi digitalment qualsevol document.

Per tal de poder efectuar operacions amb aquest nou *DNI-e*, els usuaris poden adquirir un lector de *DNI-e* que els permetrà connectar la seva identitat digital a l'ordinador, per tal de poder efectuar un nou ventall d'operacions que, fins ara, no s'havien pogut realitzar a través d'Internet degut als problemes d'identificació segura que ha patit sempre aquest mitjà:

- Compres signades.
- Tràmits complerts amb l'administració pública, a qualsevol hora.
- Transaccions segures amb entitats bancàries.
- Accés a zones físiques.
- Utilització segura d'un entorn informàtic.
- Qualsevol forma de teleconferència amb la seguretat de que els interlocutors son qui diuen ser.

Aquest projecte utilitzarà els avantatges que suposen aquest nou sistema d'identificació, per tal de treballar i assolir els conceptes necessaris que demostrin l'eficiència dels mecanismes d'autenticació segura proposats en el model del *DNI-e*.

En aquest sistema, es treballarà principalment amb la característica de *signatura digital* mitjançant el certificat facilitat per un *DNI-e*. A l'hora de validar aquesta signatura serà necessari disposar del certificat arrel de l'autoritat de certificació que ha emès dels certificats del *DNI-e* (disponible a http://www.dnielectronico.es), així com utilitzar l'accés a una autoritat de validació, mitjançant el protocol *OCSP*, que informi de l'estat del certificat.

El protocol *OCSP* (de l'anglès *Online Certificate Status Protocol*) és un mètode que permet, de forma més eficient que les llistes de revocació (o *CRL*, de l'anglès *Certificate Revocation List*), determinar l'estat de revocació d'un certificat. Aquest protocol es descriu en el *RFC 2560* i és absolutament requerit per treballar en el context de certificats emesos per la identificació mitjançant el *DNI-e* (donat que obliga sempre a que el client efectuï una petició d'estat a l'autoritat de validació online, permetent una àgil resposta davant certificats perduts o robats, degudament denunciats).

En aquest projecte s'utilitzaran les llibreries de l'API *Java Certification Path* de *J2SE* per tal de treballar amb *OCSP*.

5.2.7. Segells de temps

Es tracta d'un mecanisme, disponible *online*, que ens permet demostrar que unes dades han existit en una certa forma, sense alteracions, i en un instant determinat del temps. El protocol en el que es basa aquest mecanisme està descrit en el *RFC 3161*, essent necessària una *autoritat de segellat de temps* per dur a terme les tasques de generació i validació d'aquests *segells de temps*.

En aquest projecte no es contactarà directament amb l'autoritat de segellat, sinó que s'utilitzaran els serveis públics *DR* i *DSV* (per generar el segell i verificar-lo, respectivament) de *TrustedX*, una plataforma pública que s'ofereix des de http://www.safelayer.com.

Més concretament, aquest mecanisme permetrà actualitzar una signatura digital existent amb un *segell de temps*, i efectuar-ne la verificació més endavant, per tal de garantir que les dades han estat generades en el moment que es diu que s'han generat.

5.2.8. Persistència

Per tal de poder emmagatzemar certes dades de forma permanent en el sistema, serà necessari disposar d'algun dispositiu que ens proveeixi del que s'anomena *persistència*. Aquest dispositiu normalment acostuma a ser una *base de dades*, que es configura per tenir-la a disposició en un *servidor de bases de dades*.

En el mercat existeixen infinitat de solucions igualment vàlides per proveir persistència en un sistema com aquest, totes elles basades en l'estàndard *SQL*. Avaluant les necessitats i la funcionalitat necessàries, s'ha estimat oportú utilitzar un servidor *MySQL 5.0*, que cobrirà perfectament les necessitats requerides en l'àmbit de l'aplicació, a més de disposar d'una versió *opensource* (*MySQL Comunity Server*) que ens permetrà operar sense restriccions de licències.

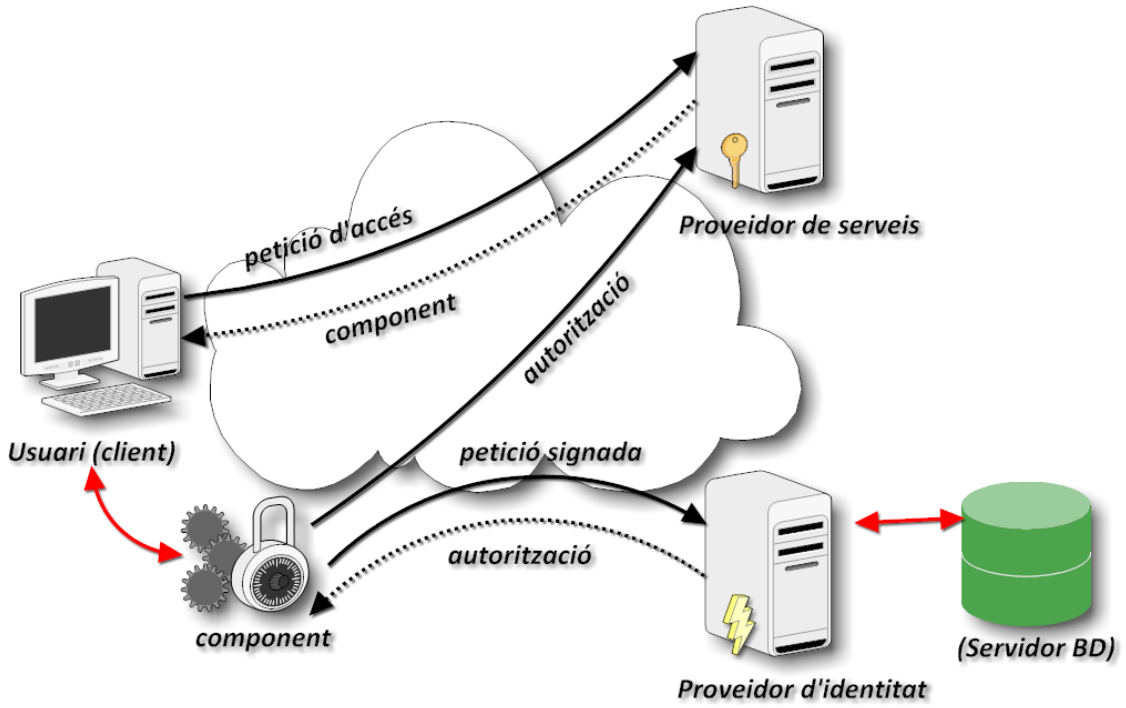
Aquest estàndard *SQL* (de l'anglès *Structured Query Language*), és un llenguatge que permet l'accés a bases de dades relacionals, i especificar diferents tipus d'operacions sobre les mateixes, utilitzant com a base l'àlgebra relacional.

Tot plegat esdevindrà una plataforma suficient per allotjar les dades que necessitarà tenir al seu abast el *proveïdor d'identitat* d'aquest sistema a l'hora de determinar els permisos d'accés a un recurs.

5.3. Visió general de l'arquitectura del producte

5.3.1. Estructura general de l'escenari

A grans trets, i tal com s'ha anat introduint al llarg d'aquest document, l'escenari es pot representar segons els elements de la següent figura:



En aquesta representació no es pretén representar exhaustivament cada element (ni la seva relació) que conforma l'escenari i, per tant, s'ha utilitzat un punt de vista molt generalista i esquemàtic per tal de poder plasmar, de forma visual, la forma en que interactuen els diferents elements.

5.3.2. Plataforma de treball i desplegament

Tal com s'ha comentat en punts anteriors, la plataforma sobre la que es treballarà en la implementació i desplegament d'aquest projecte serà aquella que suporti *Java*, tecnologia desenvolupada els anys 90 per la companyia *Sun Microsystems* (actualment ja rep la denominació de *Java2*). Concretament s'utilitzarà la plataforma de desenvolupament JDK 6 (amb el *JavaEE SDK*) per tal de no tenir problemes amb les llibreries necessàries per generar *serveis web* o *servlets*, i que poden no estar incloses en altres modalitats més senzilles de la plataforma de desenvolupament de *Java*.

Per tal de simplificar les tasques de programació, s'ha optat per utilitzar la interfície d'usuari per desenvolupament anomenada *Eclipse*, en la seva variant preparada per treballar sobre *JEE* (anomenada *Ganymede*) i amb la versió 3.4.1.

A l'hora d'efectuar el desplegament dels components s'utilitzarà un servidor *Apache Tomcat v.6* (última versió estable a data de la realització del projecte), que ens permetrà executar de forma més o menys senzilla les tecnologies implementades en el producte. Es tracta d'un servidor robust i que permet la creació d'accessos mitjançant *SSL*, configurant-lo de forma

adequada, a més de suportar sense problemes tota la funcionalitat *Java* requerida per assolir amb èxit els objectius del aplicatiu.

Pel que fa al servidor de persistència, s'utilitzarà un servidor *MySQL 5.1* que disposa els mecanismes suficients d'accés a dades d'una forma eficient, sobre el que es durà a terme el disseny de la base de dades necessària per la persistència en algunes de les parts del producte.

L'accés del client a les aplicacions web i als components necessaris per a la interacció amb els *proveïdors d'accés i de serveis* es prepararà per tal que sigui compatible amb els productes més comuns del mercat (*Internet Explorer* i *Firefox*), i en generar es procurarà seguir un disseny amb uns patrons estàndard que garanteixin la funcionalitat amb qualsevol altre producte del mercat que accepti aquests estàndards.

5.3.3. Client

La implementació del *client* es durà a terme mitjançant la codificació d'un *applet Java* que inclogui les funcionalitats necessàries per tal que l'usuari pugui seleccionar quin certificat vol utilitzar per signar la petició d'accés al recurs del *proveïdor de serveis*, i que sigui capaç de comunicar-se, mitjançant la crida a serveis web (*SOAP*), amb el servei web del *proveïdor d'identitat*.

La tecnologia d'*applets Java* permetrà carregar en el sistema de l'usuari un component que realitzarà la interacció amb els elements necessaris per aconseguir una correcta validació de la identitat del mateix. Cal tenir en compte, però, que aquesta tecnologia ha estat concebuda sobre uns esquemes de seguretat que caldrà configurar per tal de poder obtenir privilegis d'accés extraordinaris. Per defecte, un *applet Java* s'executa sobre un entorn completament aïllat de les parts sensibles del nostre sistema (el que en anglès s'anomena *Sandbox*, com a similitud d'aquell lloc del parc amb sorra on sovint els nens poden jugar sense fer-se mal) i, per tant, caldrà sol·licitar una escalada de privilegis a l'usuari, mitjançant la signatura de l'*applet* (que provocarà que l'usuari hagi de declarar la seva confiança en el signant) i poder així accedir sense problemes a operacions relacionades amb l'obtenció d'un certificat del magatzem de certificats del sistema de l'usuari, o amb l'accés a un servidor diferent del que ha proveït aquest component (operacions que no es poden efectuar dins del *Sandbox*), fet indispensable per poder implementar la interacció mitjançant missatges SOAP amb el *proveïdor d'identitat*.

Cal tenir en compte que aquest *applet Java* rebrà com a paràmetre un missatge *XML* signat pel *proveïdor de servei* (del que caldrà definir la estructura en la fase de disseny) on figuraran les dades necessàries per conformar el missatge de petició d'autorització que s'enviarà al *proveïdor d'identitat*. Tanmateix l'*applet* també serà capaç de signar aquest missatge i enviar-lo (tal com s'ha comentat, mitjançant el protocol *SOAP* d'interacció amb serveis web) gràcies a les llibreries *Axis* i *JAX-RPC*.

La resta d'interaccions amb el client, més senzilles, seran les pròpies de qualsevol aplicació web simple que dona accés a certs recursos (evidentment, un cop la identificació s'ha realitzat amb èxit i el *proveïdor d'identitat* ha autoritzat l'accés al client), i el que el client veurà es correspondrà amb una estructura *HTML* (que es generarà per l'aplicació web del *proveïdor de serveis*).

5.3.4. Proveïdor de serveis

Les tecnologies utilitzades en el proveïdor de serveis seran les que permetran implementar una aplicació web capaç de facilitar el component *applet Java* al client i processar la petició d'accés segons si s'autoritza o no mitjançant el *proveïdor d'identitat*. S'utilitzarà, doncs, la

tecnologia *JSP* i *servlets Java* que permetran, de forma senzilla, facilitar una pàgina que carregui l'*applet* quan es sol·liciti un nou accés a un recurs per part d'un usuari, i gestionar els certificats i signatures rebuts quan l'*applet* faciliti la resposta del *proveïdor d'identitat* referent a l'autorització (per tal d'implementar aquestes signatures, al llar del projecte, s'utilitzarà l'estàndard *XMLDSig* i les llibreries disponibles al *JDK* de *Sun*). Aquesta tecnologia es pot desplegar sense cap problema en el tipus de servidor d'aplicacions web proposat (*Tomcat v.6*) i també es podria desplegar en altres servidors que la suportessin (com ara *Glassfish* de *Sun Microsystems*).

Tal com s'ha comentat pel component del *client*, el *proveïdor de serveis* haurà de ser capaç de generar un missatge *XML* i lliurar-lo al *client* per a que aquest pugui elaborar la sol·licitud d'accés al recurs. Per tant, caldrà definir com es durà a terme la persistència necessària per controlar la numeració d'identificadors de peticions (o decidir si no cal persistència en el cas que ens ocupa). De la mateixa forma, també haurà de ser capaç d'interactuar amb els serveis web *SOAP* de no repudi i verificació de la plataforma *TrustedX* per tal de poder generar una signatura de segell de temps.

5.3.5. Proveïdor d'identitat

D'una banda, la tecnologia que ens faciliten les llibreries *Axis* i *JAX-RPC* ens permetrà desplegar el servei *SOAP* necessari per rebre les peticions d'accés del component del *client* cap a algun recurs del *proveïdor de servei*. Gràcies a aquesta tecnologia, el servei web podrà rebre un missatge *XML* signat amb les dades necessàries per conformar una petició d'accés (tal com s'ha comentat, caldrà definir l'estructura exacte d'aquests missatge més endavant) i haurà de ser capaç de verificar les signatures i validar els certificats implicats en aquest missatge. També implementarà la tecnologia necessària per interactuar amb la plataforma *TrustedX* per tal de validar el segell de temps emès pel *proveïdor de serveis,* i inclòs en aquest missatge *XML* rebut.

També implementarà, en la part relativa al servei web, les consultes oportunes a la base de dades, mitjançant la llibreria *JDBC* facilitada pel *SDK* de *Sun*. D'aquesta forma es tindrà accés a la informació relativa als permisos dels usuaris als recursos i les franges de temps definides, per tal de generar un missatge *XML* de resposta, signat, amb la resolució sobre l'autorització d'accés.

D'altra banda, i utilitzant la possibilitat de configurar l'accés *SSL* en el servidor *Apache Tomcat*, així com la tecnologia *JSP* i *Servlets* comentada anteriorment, la implementació del *proveïdor d'identitat* es completarà mitjançant una interfície senzilla d'administració i manteniment dels recursos, dels usuaris i de les franges de temps permeses per cada recurs i usuari. En aquest cas, també s'utilitzarà la tecnologia facilitada per les llibreries *JDBC* en els aspectes relatius a la persistència, ja que esdevé suficient per cobrir la funcionalitat necessària (inserció, modificació i eliminació de registres mitjançant sentències *SQL*).

5.4. Disseny del producte

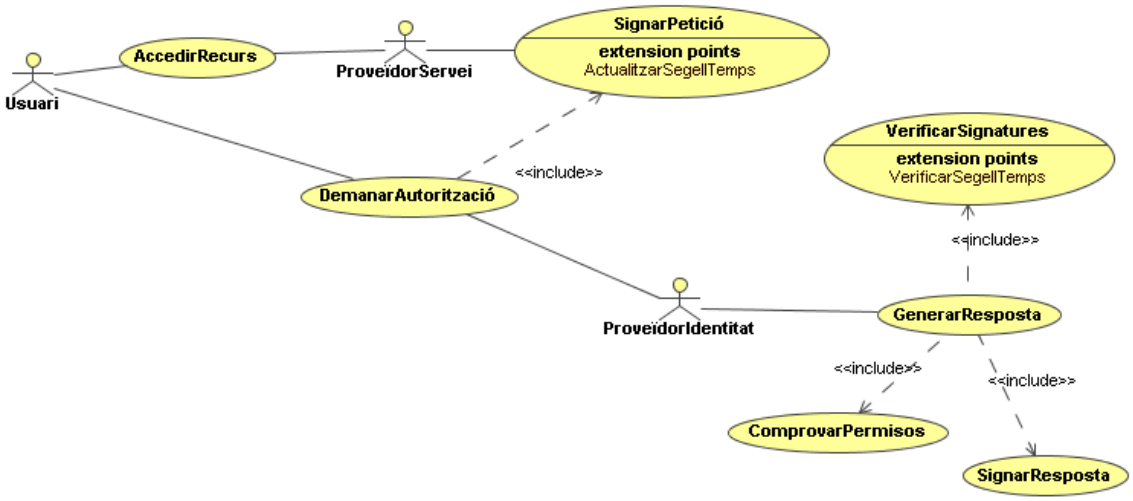
Els diagrames UML que es presenten a continuació han estat elaborats mitjançant el software *MagicDraw UML Personal Edition* en la seva versió 12. En alguns casos ha estat necessari afegir o eliminar elements dels suggerits per aquesta aplicació en alguns tipus de diagrames.

5.4.1. Disseny de l'arquitectura del projecte

Els diagrames UML que es presenten a continuació constituiran la base per l'arquitectura d'aquest projecte, tot i que, rigorosament parlant, aquesta arquitectura també inclouria el disseny de la persistència, de les dades i de la interfície d'interacció amb l'usuari. S'ha volgut, però, diferenciar el disseny del comportament de l'aplicació (que s'intentarà descriure el més estrictament possible en els següents diagrames) del disseny dels components més "físics" (com les dades o la interfície).

5.4.1.1. Diagrama de casos d'ús

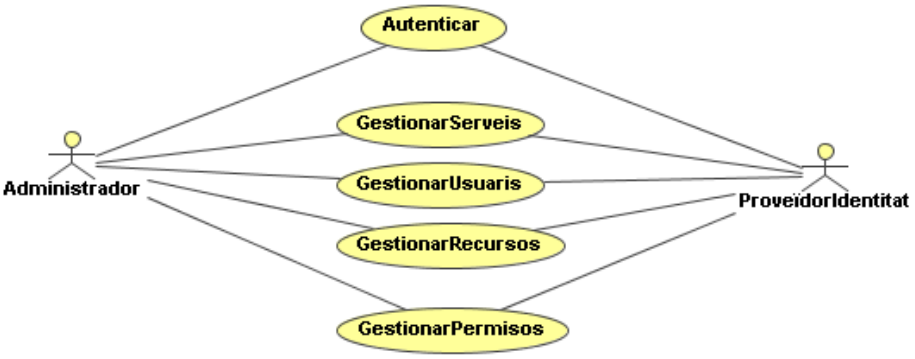
- *General*



En els casos d'ús del cas general del funcionament del projecte podem apreciar ja la majoria d'actors i de situacions que prendran part activa durant l'execució del mateix. D'una banda, l'actor *Usuari* serà el desencadenador principal de l'acció, accedint a un recurs del *proveïdor de serveis*. Així doncs, existiria un actor *ProveïdorServei* que seria qui ens facilitaria el *ComponentClient*, i qui ens facilitaria el recurs en última instància i si l'autorització ha estat satisfactòria. El paper que juga aquest actor s'ha procurat definir també en altres diagrames amb més detall. Però donat que les funcionalitats del *ComponentClient* son executades directament pels usuaris, no s'ha considerat oportú fer-lo aparèixer com a actor en aquest diagrama, designant a l'actor *Usuari* com el veritable executor del cas d'ús *DemanarAutorització* (dins d'aquest cas d'ús és on residiria l'àmbit d'acció de l'actor *ComponentClient*). En aquest punt sí que cal tenir en compte l'actor *ProveïdorIdentitat*, que serà qui generarà la resposta comprovant els permisos a la base de dades i verificant les signatures, i finalment decidirà si s'atorga o no accés al recurs, sense oblidar que aquesta resposta es retornarà signada. Com a últim punt a destacar d'aquest diagrama, cal comentar que els casos d'ús *SignarPetició* i *VerificarSignatures* són comuns per les signatures tant pel *ProveïdorServei* (es veurà més endavant) com pel cas d'ús *DemanarAutorització*, amb la salvetat (representada com un punt d'excepció) del fet que la signatura del p*roveïdor de serveis* caldrà

actualitzar-la amb un segell de temps. En el cas d'ús *SignarResposta* es farà servir la mateixa classe per signar que a *SignarPetició*, però donat que, conceptualment, implica una certa diferència, s'ha considerat oportú denotar aquesta diferenciació de casos d'ús.

- *Administració de serveis*

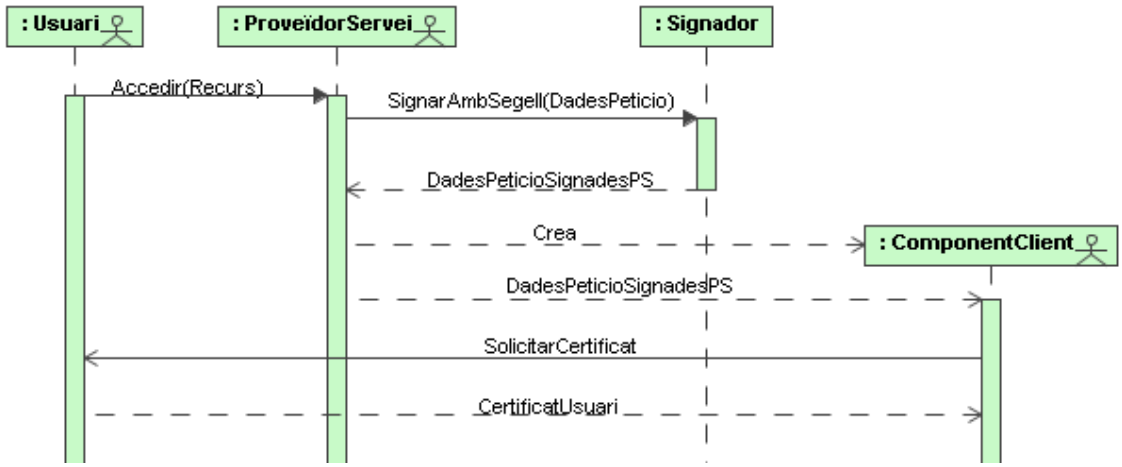


També s'ha de tenir en compte que en el p*roveïdor d'identitat* caldrà dur a terme les tasques de manteniment oportunes per tal de tenir les dades correctes sobre usuaris, serveis, recursos i permisos a la base de dades. Així doncs, la gestió d'aquests elements (que no seran altra cosa que altes, baixes o modificacions) la realitzarà l'actor *Administrador*, prèviament autentificat en el *ProveïdorIdentitat*. Cal tenir en compte que aquest actor *Administrador* no hereta en cap cas de l'actor *Usuari* del diagrama anterior, ja que si bé poden coincidir en algun cas, es tracta de sistemes diferents i de mitjans d'autenticació diferents, i en general, per tant, d'usuaris completament diferents.

5.4.1.2. Diagrama de seqüència

Per tal de fer més entenedors aquests diagrama, s'ha optat per subdividir-lo en els diagrames corresponents als casos d'ús més significatius, per tal de que facilitar-ne la lectura.

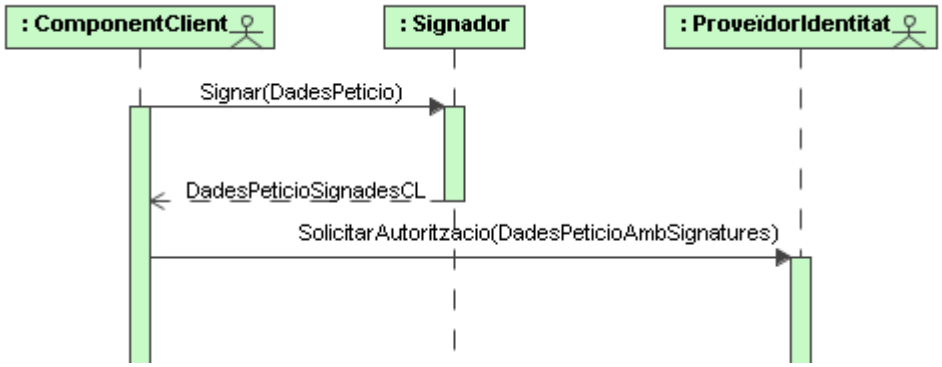
- *Accedir a recurs*



L'actor *Usuari* és qui inicialment efectua l'accés a un recurs del *proveïdor de servei* qui, després de signar les dades de la petició (mitjançant una classe dissenyada per a tal

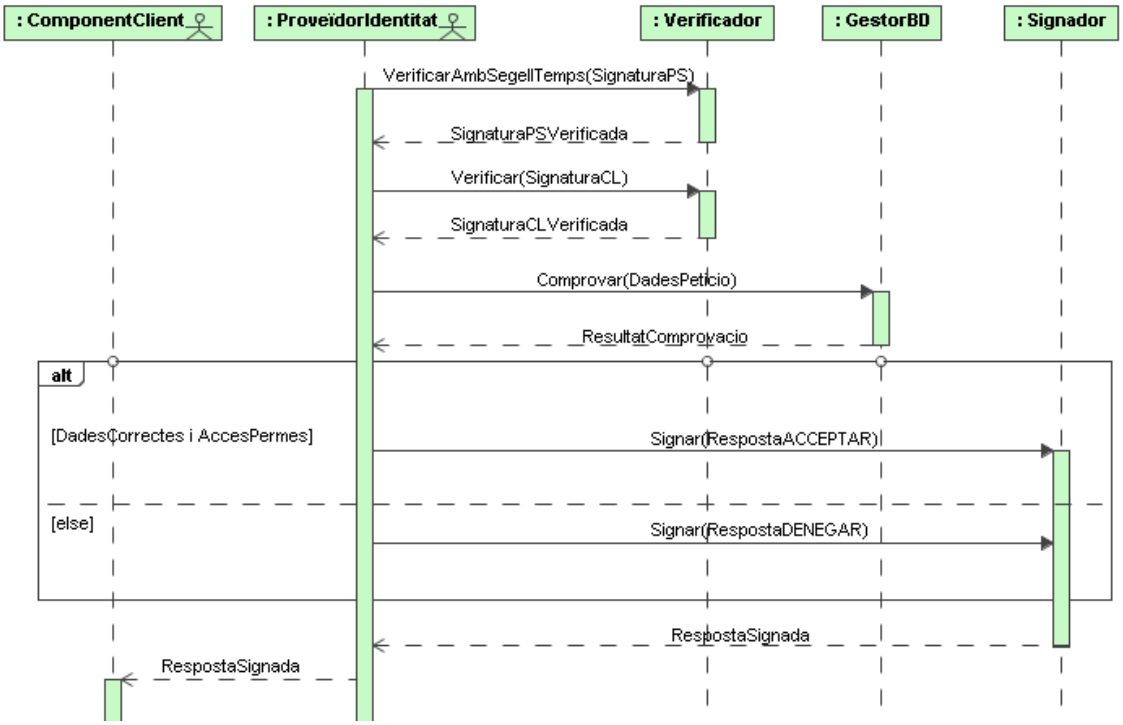
efecte) crearà el *ComponentClient* (i li facilitarà les dades signades de la petició). En aquest punt sí que s'ha considerat el *ComponentClient* com a un actor, per tal de poder descriure el paper que hi juga a l'hora de demanar el certificat a l'usuari, i es podrà considerar veritablement com l'actor que demanarà l'autorització al *proveïdor d'identitat* en els següents passos.

- *Demanar autorització*



Un cop que el *ComponentClient* té les dades del certificat de l'usuari i les dades de la petició d'accés al recurs (signades pel *proveïdor de serveis*), pot efectuar la petició de l'autorització a l'accés al recurs, signant en primer lloc les dades de la petició, i deixant en mans del *proveïdor d'identitat* les operacions oportunes per decidir l'accés al recurs.

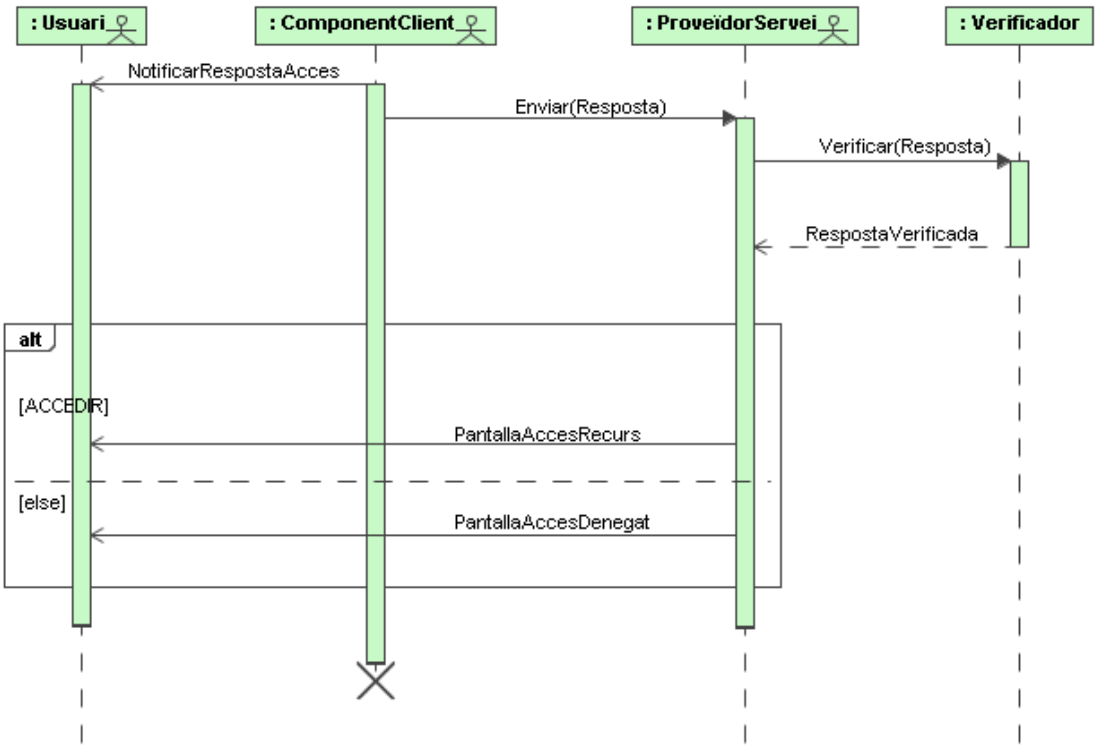
- *Generar resposta*



Per generar la resposta primer de tot caldrà que el *proveïdor d'identitat* efectuï les verificacions de les signatures (validant els certificats, el segell de temps en el cas de la

signatura del *proveïdor de serveis* i l'estat de revocació, mitjançant *OCSP*, en el cas de la signatura del *client*), així com també li caldrà comprovar que les dades (tant l'usuari que efectua la petició, com el servei o el recurs, i els permisos que existeixin, en funció de la data en que s'hagi efectuat la petició) es troben a la base de dades, i finalment decidir si *ACCEPTAR* o *DENEGAR* l'accés. Un cop decidit, es signaran les dades de la resposta i s'enviaran (dades i signatura) cap al *ComponentClient*.

- *Presentar l'accés*

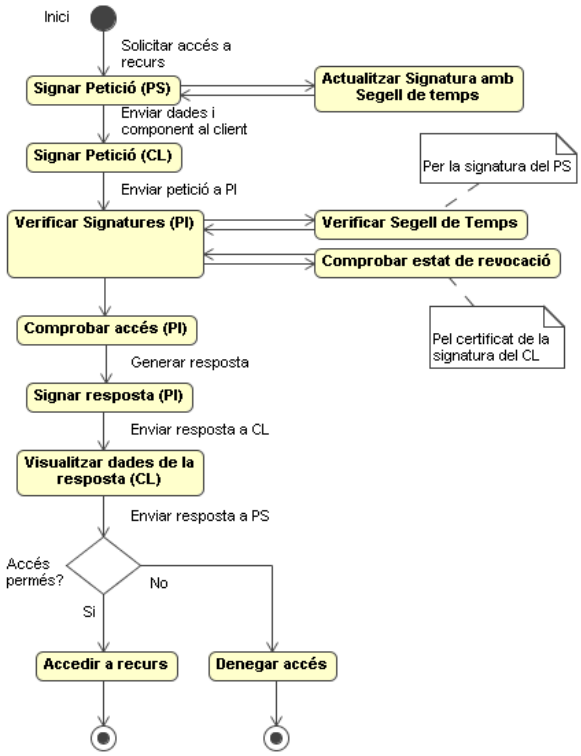


Finalment el *ComponentClient* notificarà a l'usuari les dades de la resposta que li han estat enviades pel *proveïdor d'identitat*, i les enviarà al *proveïdor de servei* qui, en funció de si la resposta ha estat favorable, i de si la signatura ha estat verificada (validant el seu certificat), presentarà al usuari una pantalla per accedir al recurs, o una pantalla informant de l'accés denegat al mateix.

- *Administració de serveis*

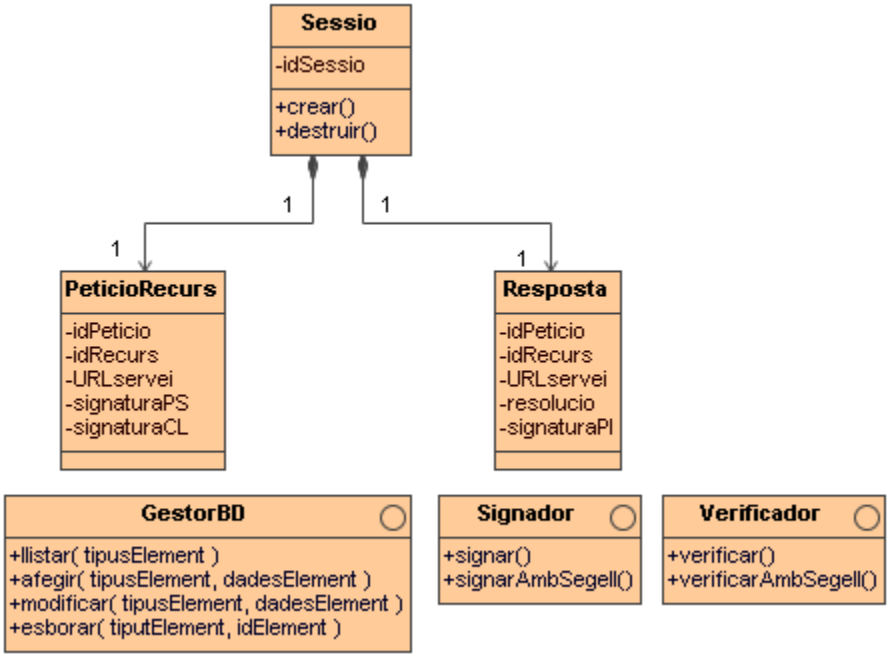
En el cas de les tasques de manteniment que l'administrador pot dur a terme sobre els recursos, serveis, usuaris i permisos que el *proveïdor d'identitat* ha de tenir en compte (emmagatzemats gràcies a la persistència), no s'ha considerat prou significativa la informació que aportaria un diagrama de seqüències, donat que més endavant es pararà especial atenció al disseny de la persistència, molt més rellevant pel que es refereix a aquest aspecte del projecte. Les operacions a realitzar sobre els elements persistents (usuaris, recursos, serveis i permisos) seguirien un patró similar i força estàndard (llistar elements, alta d'elements, modificació i eliminació, entre d'altres).

5.4.1.3. Diagrama d'estats



En aquest diagrama es representen, esquemàticament, els diferents estats i transicions que adoptarà el sistema. Donat que el sistema s'executa, de forma distribuïda, en tres elements, s'ha procurat marcar els estats amb un identificador de l'element on té lloc (*PS* pel *proveïdor de serveis*, *PI* pel *proveïdor d'identitat* i *CL* pel *component client*).

5.4.1.4. Diagrama de classes



En aquest diagrama es posen de manifest les classes principals que serviran com a contenidors de les dades significatives que prenen part durant el transcurs de l'aplicació. Cal tenir en

compte que el projecte es suportarà de classes de tipus *interfície* per dur a terme les operacions necessàries per gestionar la base de dades (representada de forma molt esquemàtica en aquest diagrama) o per signar i verificar les dades que s'intercanviaran els diferents elements del sistema, essent trivial la relació d'aquestes interfícies amb les dades en sí mateixes. Així doncs, pel que respecta a les dades, s'ha diferenciat entre les necessàries a l'hora de completar els passos de l'aplicació fins a la decisió de l'autorització d'accés. A partir d'aquest punt, caldrà accedir a un altre tipus de dades, les dades persistents, que quedaran descrites més oportunament en la secció de *disseny de la persistència*.

Afegint concreció en la descripció de les dades que es defineixen en aquest diagrama, cal dir que representen les dades que es construiran des de que l'usuari genera la petició d'un recurs del *proveïdor de serveis*, fins que el *component del client* envia aquestes dades signades (tant pel *proveïdor de serveis* com per ell mateix) al *proveïdor d'identitat*, així com també les dades que el *proveïdor d'identitat* retornarà al *component del client* com a resposta autoritzant o no la petició. En aquestes estructures, a més dels identificadors necessaris per tal que el *proveïdor d'identitat* pugui trobar en els elements persistents els permisos i horaris establerts per l'accés a un recurs, també s'inclouen implícitament els identificadors tant de l'usuari com del *proveïdor de serveis* al que es vol accedir, gràcies a les signatures.

Finalment, cal posar de manifest que les verificacions de signatures per part del *Verificador* realitzaran una comprovació de l'estat de revocació del certificat quan es tracti de verificar la signatura generada pel *ComponentClient*.

5.4.2. Disseny de la persistència

En aquest projecte es contempla la necessitat de mantenir persistència de les dades que necessita el *proveïdor d'identitat* per tal de decidir si pot atorgar l'accés a un usuari. Així doncs, caldrà emmagatzemar les dades referents a l'usuari, als serveis als que dona accés el *proveïdor d'identitat*, als recursos que hi ha disponibles en aquests serveis i als permisos que té un usuari en vers aquests serveis (i en cas de tenir permís, la franja horària en la que es té aquest permís).

5.4.2.1. Taules de la base de dades

Les entitats principals de la base de dades són *Clients, Serveis* i *Recursos*, que ens serviran per identificar els elements que prenen part en el procés de decisió de l'autorització d'accés a una petició.

- *Clients*
  - **Id_client (varchar)**: Clau principal de la taula *clients*. Haurà de coincidir amb el camp *subject* del certificat del client.
  - **Nom (varchar)**: Nom del client.

- *Serveis*
  - **Id_servei (varchar)**: Clau principal de la taula *serveis*. Haurà de coincidir amb la URL del servei.
  - **Cert_servei**: Camp *subject* del certificat del *proveïdor de serveis.*
  - **Descripció (varchar)**: Descripció del servei.

- *Recursos*
  - **Id_recurs (varchar)**: Clau principal de la taula *recursos*. Haurà de ser el mateix identificador que faciliti el *proveïdor de serveis* quan es sol·liciti accés al recurs.
  - **Descripció (varchar)**: Descripció del recurs.

D'altra banda, també hi ha les taules corresponents a les relacions entre les entitats anteriors, que ens descriuran les relacions entre elles. Així doncs disposarem de la taula *RecursosServeis* que ens definirà els recursos que conté un servei determinat, i de la taula *Permisos* que ens definirà si un usuari té o no permís per accedir a un recurs determinat dins d'un servei, i en quin horari té permès aquest permís. Cal tenir en compte que la no existència d'un registre en aquesta taula que relacionés un client determinat amb un recurs i servei implicaria que aquest client NO té permís per accedir a aquest recurs i servei.

- *RecursosServeis*
  - **Id_servei (varchar)**: Clau forana que fa referència a un registre de la taula *Serveis*.
  - **Id_recurs (varchar)**: Clau forana que fa referència a un registre de la taula *Recursos*.

- *Permisos*
  - **Id_client (varchar)**: Clau forana que fa referència a un registre de la taula *Clients*.
  - **Id_servei (varchar)**: Clau forana que fa referència a un registre de la relació *RecursosServeis* (i, indirectament, a un servei).
  - **Id_recurs (varchar)**: Clau forana que fa referència a un registre de la relació *RecursosServeis* (i, indirectament, a un recurs).

Aquestes relacions es configuraran segons el model *cascada* per tal de definir el comportament davant la modificació d'alguna clau de les taules d'entitats, o l'esborrament d'algun registre amb relacions existents d'aquestes mateixes taules. Aquest model propagarà els canvis a totes les taules de relacions, modificant o esborrant les relacions afectades.

Tal i com s'ha concebut aquest model, es pot afirmar que, com a mínim, es compleix fins la *3ª forma normal* (a la pràctica és suficient per garantir un disseny correcte i coherent de les dades per la poca magnitud de la persistència en aquest projecte) de la *Teoria de la normalització de les bases de dades*, atès que:

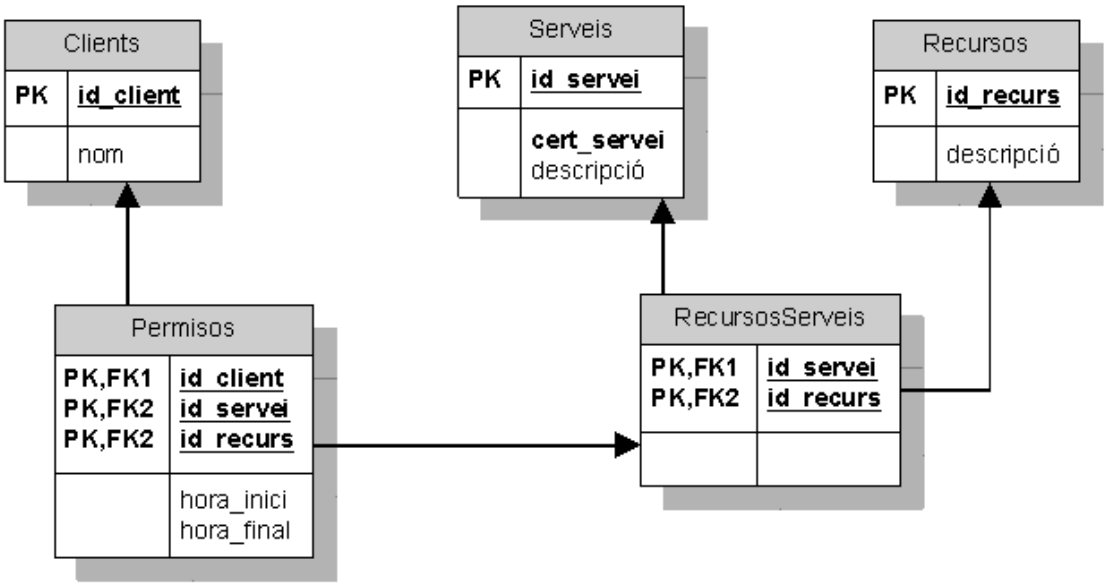
- Cap atribut de cap relació és en sí mateix una relació (*1ª forma normal*).
- Tots els atributs no clau depenen funcionalment, de forma completa, de la seva clau primària (*2ª forma normal*).
- Cap atribut no clau depèn funcionalment de cap altre conjunt d'atributs no clau (*3ª forma normal*).

En el nostre cas concret, s'ha considerat oportú que la relació *RecursosServeis* existeixi independentment de si hi ha cap client amb un permís associat, així com la possibilitat de que un recurs pugui existir en més d'un servei.

Tanmateix, el camp *cert_servei* de la taula *Serveis* no s'ha considerat que aportés informació a la clau d'aquesta taula (ja que un servei serà únic per la seva *URL*), tot i que sí que s'utilitzarà com a suport a efectes de comprovació de la identitat del *proveïdor de servei* que ofereix aquell servei i en signa les dades de la petició.

Finalment, posar en rellevància que no s'ha considerat el fet de dissenyar una taula d'administradors que tinguin accés al manteniment d'aquestes dades a través de la interfície del *proveïdor d'identitat*. Donat el caràcter del sistema i la poca rellevància que suposaria disposar d'aquest control d'administradors, s'ha preferit codificar l'accés dels administradors mitjançant un fitxer de recursos en el codi (es comentarà en l'apartat de funcionament).

5.4.2.2. Model relacional de la base de dades



Formalment, la descripció de les taules és la següent :

Clients(id_client, nom)
Serveis(id_servei, cert_servei, descripció)
Recursos(id_recurs, descripció)
RecursosServeis(id_servei, id_recurs)
on {id_servei} fa referència a *Serveis*.
on {id_recurs} fa referència a *Recursos*.
Permisos(id_client, id_servei, id_recurs, hora_inici, hora_final)
on {id_client} fa referència a *Clients*.
on {id_servei} fa referència a *RecursosServeis*.
on {id_recurs} fa referència a *RecursosServeis*.

5.4.3. Disseny del format de dades XML

Les dades que s'intercanviaran els elements del sistema que conforma l'aplicació estaran, com ja s'ha comentat, en format XML. En aquest format s'encapsularan les dades necessàries segons les següents etapes d'intercanvi de dades:

5.4.3.1. Proveïdor de serveis → Client

El format per l'intercanvi de dades que s'utilitzarà en aquesta etapa serà:

```
<PeticioRecurs>
        <IdPeticio>Identificador_de_peticio</IdPeticio>
        <IdRecurs>Identificador_de_recurs</IdRecurs>
        <URLServei>URL_del_servei</URLServei>
</PeticioRecurs>
```

També es facilitarà la signatura *detached* d'aquestes dades, per part del *proveïdor de serveis* (cal recordar que aquesta signatura ja inclourà un segell de temps).

Aquestes dades es facilitaran al *component client* (*applet Java*) en forma de paràmetres (un paràmetre per les dades i un per la signatura) en la pàgina que l'embeurà, i que serà la que rebrà el client després de la seva petició de recurs.

5.4.3.2. Client → Proveïdor d'identitat

El format per l'intercanvi de dades que s'utilitzarà en aquesta etapa serà el mateix que en l'etapa anterior, amb el fet que ara també s'enviarà la signatura *detached* de les dades per part del client, amb el certificat de l'usuari.

El *component client* enviarà aquestes dades mitjançant un missatge *SOAP* amb un paràmetre per les dades i un paràmetre per cadascuna de les signatures.

5.4.3.3. Proveïdor d'identitat → Client → Proveïdor de serveis

En aquesta etapa s'afegirà la dada *resolució* que indicarà si s'ha acceptat o s'ha denegat l'accés al recurs per part del *proveïdor d'identitat*. Aquesta dada únicament contemplarà els valors "ACCEPTAT" i "DENEGAT" segons pertoqui.

També cal tenir en compte que el *proveïdor d'identitat* signarà aquestes dades amb una signatura de tipus *enveloping* (per tal d'obtenir una única resposta que contingui tant les dades com la signatura).

Per tant, el format per l'intercanvi de dades que s'utilitzarà en aquesta etapa serà:

```
<PeticioRecurs>
        <signature>
        (camps de la signatura)
        </signature>
        <IdPeticio>Identificador_de_peticio</IdPeticio>
        <IdRecurs>Identificador_de_recurs</IdRecurs>
        <URLServei>URL_del_servei</URLServei>
        <Resolucio>[ACCEPTAT/DENEGAT]</Resolucio>
</PeticioRecurs>
```

En tots els casos cal tenir present que les signatures contindran els certificats dels signants per tal de poder-les verificar sense la necessitat de disposar d'un repositori de certificats compartit.

També cal remarcar, en aquest punt, que aquesta estructura de dades no contempla, en cap cas, que el *proveïdor de serveis* tingui accés a cap atribut o dada que pugui identificar a l'usuari, tal com ha estat definit en els requisits del sistema.

5.4.4. Disseny de la interfície gràfica

Hi ha dos interfícies d'interacció amb els usuaris clarament diferenciades en aquest sistema. D'una banda, cal dissenyar la interfície que es trobarà l'usuari en accedir al recurs del *proveïdor de serveis*. Aquesta interfície serà inicialment senzilla (en forma de pàgina web amb enllaços als recursos que qualsevol usuari pot sol·licitar, i seguidament, en quant es carregui el *component client*, esdevindrà una mica més complexa per tal de permetre la selecció del certificat d'usuari de la llista de certificats de la màquina, així com veure les dades de la resposta obtinguda pel *proveïdor d'identitat*.
La segona interfície a considerar és la que utilitzarà un usuari administrador sobre el *proveïdor d'identitat*. En aquest cas es tractarà d'una interfície web senzilla que permetrà, després de

que l'usuari s'identifiqui (i accedeixi mitjançant una connexió segura *SSL*), efectuar les altes, modificacions i baixes oportunes sobre els registres de les entitats que s'han descrit en l'apartat de persistència, així com establir les relacions entre elles (incloent-hi els permisos d'accés).

5.4.4.1. Accés a un recurs

Caldrà tenir en compte el fet que el botó d'enviar resposta al *proveïdor de serveis* haurà d'estar desactivat fins que no s'hagi sol·licitat l'accés (primer botó) i s'hagi rebut la resposta (que apareixerà en el quadre destinat a tal efecte). Els quadres de selecció en realitat seran informatius (no es podran operar), i simplement serviran per veure a primer cop d'ull si la resposta del *proveïdor d'identitats* ha estat favorable o no (es podria haver pensat qualsevol altre sistema per representar-ho).

En cas de que no es disposi de cap certificat *DNI-e* disponible, el primer botó també haurà d'aparèixer desactivat, i en el primer quadre apareixerà l'únic element "*(No hi ha cap certificat DNI-e disponible)*".

5.4.4.2. Administració del proveïdor d'identitat

En primer lloc, l'usuari s'haurà d'identificar mitjançant una pantalla similar a aquesta:

En cas que l'accés no sigui correcte, l'usuari rebrà alguna mena de missatge d'accés denegat, i en cas de que l'accés sigui correcte, accedirà a la pantalla que li permetrà administrar els diferents elements de la base de dades. Una pantalla similar a:

Inicialment l'usuari apareixerà en el manteniment de *Clients*, i podrà canviar al manteniment de *Serveis* o de *Recursos* seleccionant-ho en el menú del que disposarà en la part superior de la pantalla. En aquest manteniment, l'usuari veurà la llista de clients que estan donats d'alta en el sistema, i disposarà dels botons *Afegir*, *Modificar* i *Esborrar*, a més d'un botó que li permetrà assignar els recursos i serveis que estiguin donats d'alta.

En cas de prémer el botó *Afegir* o *Modificar*, la interfície canviarà a quelcom semblant a:

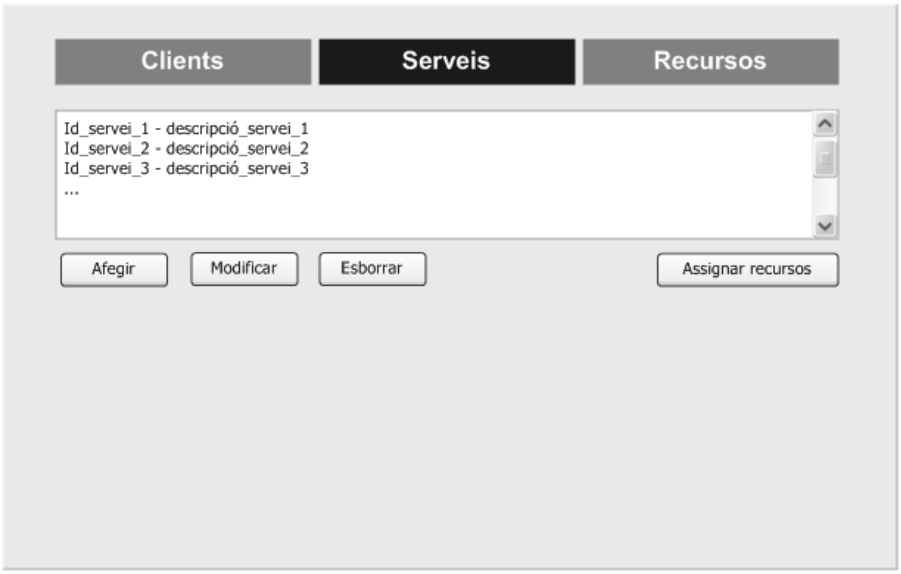
La diferència estarà en que quan l'administrador hagi seleccionat *Afegir*, els camps li seran mostrats en blanc, i quan hagi seleccionat *Modificar*, en els camps hi hauran les dades del client que s'hagi seleccionat en la primera llista. Evidentment, els botons *Modificar* i *Esborrar* només estaran actius en cas que existeixin clients i se n'hagi seleccionat un. Quan l'administrador seleccioni el botó *Guardar* es tornarà a la interfície anterior, amb la llista de clients actualitzada.

Si el que es prem és el botó *Assignar recursos-serveis*, la interfície serà similar a:

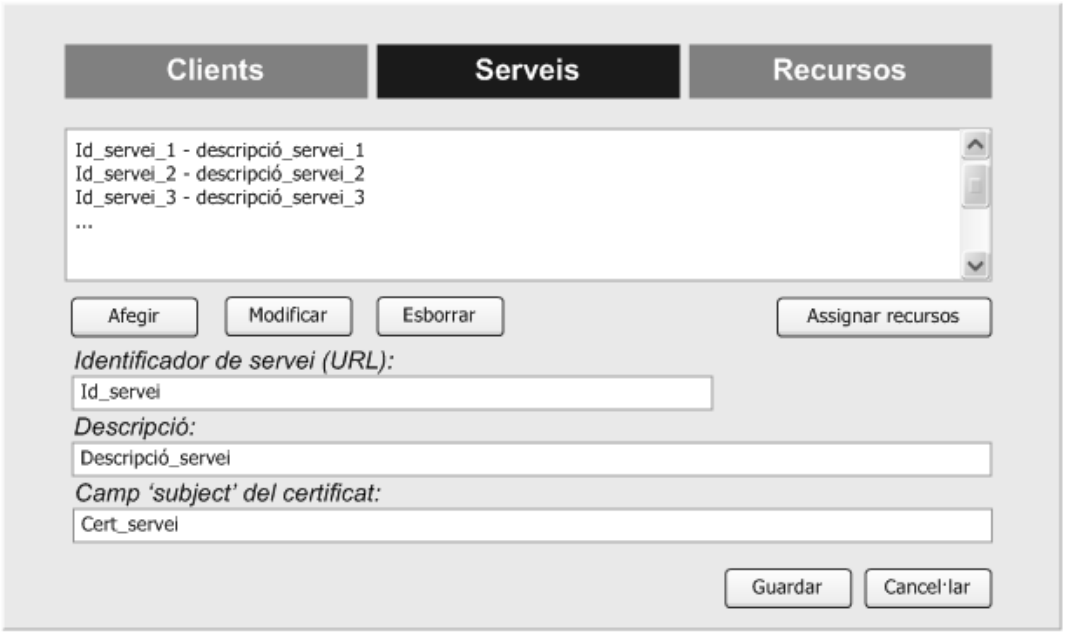
En aquest cas, en primer lloc se'ns afegirà una llista amb els *recursos-serveis* que ja tenim assignats per aquest client. I en segon lloc se'ns afegirà una llista per a seleccionar un *recurs-servei* que no tinguem assignat, podent guardar la relació (amb una hora d'inici i de final). En cas de deixar els camps d'hora en blanc, el permís serà per tot el dia. El sistema disposarà de comprovacions per tal que les hores entrades siguin coherents, mostrant els missatges informatius o d'error pertinents.

Un cop vista la interfície pel manteniment de clients, es pot treure la conclusió de que el sistema només permet assignar un *recurs-servei* a un client quan aquests ja han estat creats (tant el client com la relació entre el recurs i el servei).

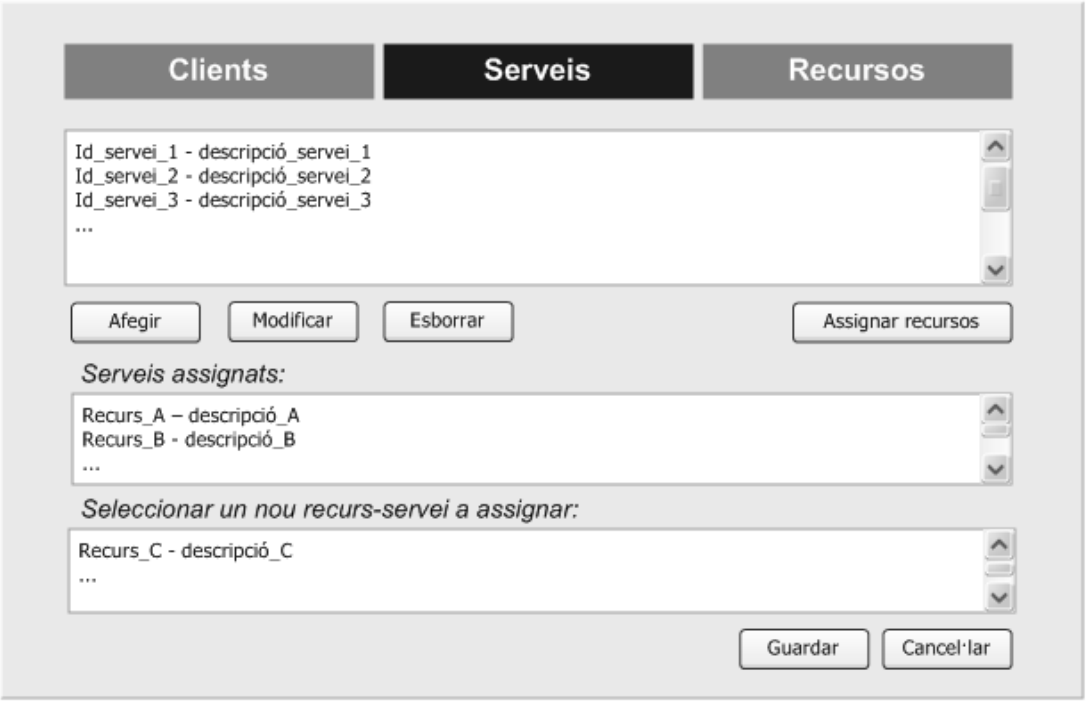
Si l'usuari selecciona *Serveis* en el menú superior, la pantalla que se li presentarà per efectuar el manteniment de serveis serà similar a:



De la mateixa forma que en el cas de clients, l'administrador podrà *Afegir*, *Modificar* i *Esborrar* els serveis del sistema, a més de poder *Assignar recursos* a aquests serveis (recursos que existeixin). En el cas d'*Afegir* i *Modificar*, es presentarà una interfície com:



I en el cas de voler *Assignar recursos* a un servei, la interfície serà similar a:



El seu funcionament serà força anàleg al de l'assignació de *recursos-serveis* en el manteniment de clients, permetent-nos crear les relacions *recursos-serveis* que es presenten en la interfície de manteniment de clients. Cal tenir en compte, però, que el camp *cert_servei* haurà de contenir el camp *subject* del certificat que utilitzarà el *proveïdor de serveis* per signar les peticions.

Finalment, si l'usuari selecciona *Recursos* en el menú superior, se li presentarà una interfície com:

I en el cas d'*Afegir* o *Modificar* un recurs, la interfície s'ampliarà a:

En qualsevol de les pantalles, els botons *Esborrar* demanaran confirmació de que es vol esborrar el registre seleccionat, i posteriorment refrescaran els llistats pertinents.

5.5. Funcionament, instal·lació i posada en marxa del producte

5.5.1. Instal·lació de l'entorn necessari de funcionament

Per tal de poder executar el producte, primer de tot cal disposar de l'entorn de funcionament necessari. La descripció de la instal·lació d'aquest entorn s'ha centrat en sistemes operatius *Windows*, però qualsevol administrador amb coneixements de *Linux* hauria de ser capaç també d'instal·lar i desplegar el producte sobre aquest sistema operatiu.

5.5.1.1. Servidor de bases de dades. MySQL Comunity Server

El primer del que cal disposar és d'un servidor de bases de dades MySQL, disponible a http://dev.mysql.com/downloads/ . D'aquest servidor existeixen diferents versions, però per les funcionalitats del nostre producte s'ha optat per la versió *5.0.67* del *Comunity Server*, sota llicència *OpenSource*. Cal dir, però, que en el moment de redactar aquesta memòria la versió vigent d'aquest producte és la *5.1.30*, però aquest fet hauria d'ésser irrellevant.

Donat que no és la intenció d'aquest escrit aprofundir en els aspectes sobre les instal·lacions d'aquest servidor de bases de dades, simplement cal afegir que el servidor es pot instal·lar en moltes plataformes diferents, i que es disposa de suficient informació com per efectuar una instal·lació estàndard en qualsevol d'elles (suficient per les especificacions del nostre producte).

Finalment, cal tenir en compte que aquest servidor només és necessari instal·lar-lo en el sistema que efectuarà les funcions de *proveïdor d'identitat.* Tot i que a efectes de desenvolupament i proves és el mateix sistema que acaba executant tots els rols, a efectes pràctics el producte té un caràcter distribuït molt important.

5.5.1.2. Servidor d'aplicacions web. Apache Tomcat

El següent del que cal disposar és d'un servidor d'aplicacions web. S'ha optat per utilitzar el servidor *Apache Tomcat 6.0* disponible a http://tomcat.apache.org/ . Una vegada més, aquest servidor gratuït també es pot trobar disponible per a instal·lar en diferents plataformes, si bé les indicacions sobre com fer-ho en cadascuna d'elles es poden trobar en la mateixa web.

Aquest servidor s'ha d'instal·lar tant en el sistema corresponent al *proveïdor de serveis* com al corresponent al *proveïdor d'identitat*. Un cop més cal denotar que a efectes de proves i desenvolupament el sistema serà el mateix, però és necessari remarcar aquest punt en els casos pràctics d'aplicació, on les parts del producte estaran efectivament distribuïdes.

Cal parar atenció en 2 aspectes de configuració importants, un cop s'ha instal·lat el servidor:

- D'una banda, cal afegir al seu directori */lib* el connector *MySQL* necessari per a que les aplicacions web del nostre producte puguin enllaçar correctament amb la persistència. Aquest connector, en forma de fitxer *.jar* que caldrà copiar a aquest directori, es pot trobar a http://dev.mysql.com/downloads/connector/j/5.1.html. Concretament, cal utilitzar el fitxer *mysql-connector-java-5.1.7-bin.jar* contingut dins de qualsevol de les distribucions disponibles. Aquesta configuració serà estrictament necessària en la instal·lació que haurà d'efectuar el rol de *proveïdor d'identitat*.

- D'altra banda, caldrà configurar també el servidor per a l'ús de connexions *SSL*. Per tal d'efectuar aquesta operació, caldrà copiar el magatzem de claus *Server.p12*, disponible en el directori */pki* de la distribució del producte, al directori arrel de la instal·lació de

Tomcat 6.0, i afegir ó modificar la següent entrada del fitxer *server.xml* del directori */conf* del servidor:

```
<Connector port="8443" protocol="HTTP/1.1" maxThreads="150"
keystoreFile="Server.p12" keystorePass="mefisto" keystoreType="PKCS12"
SSLEnabled="true"    scheme="https"    secure="true"    clientAuth="false"
sslProtocol="TLS" />
```

Aquesta configuració caldrà efectuar-la tant en la instal·lació del sistema del *proveïdor d'identitat* com en la del *proveïdor de serveis*, si bé el magatzem de claus utilitzat pot ésser el mateix (de fet, en la distribució es disposa del mateix fitxer tant pel *proveïdor de serveis* com pel *proveïdor d'identitat*).

5.5.1.3. Navegador del client

Tot i que no es pretén descriure en aquest document com cal proveir al sistema del client d'un navegador funcional, ni tampoc existeix cap consideració important a tenir en compte sobre aquest, sí que és necessari destacar alguns aspectes.

El producte ha estat testejat tant en el navegador *Firefox* com en el *Internet Explorer* (ambdós sobre *Windows XP* i sobre *Windows Vista 64bit*), donat que aquests dos productes cobreixen la major part de navegadors que es poden trobar en qualsevol sistema operatiu d'escriptori. Però els resultats més satisfactoris han esdevingut amb *Firefox* sobre *Windows XP*, sorgint diversos problemes en les altres combinacions (o bé problemes del *lector de targetes* i *Windows Vista x64*, o bé el fet que sobre *Internet Explorer* no funciona la crida amb la que finalment es descarrega el recurs mentre es refresca la plana web del *proveïdor de serveis*). Cal tenir en compte, però, que caldrà disposar d'una *màquina virtual Java* actualitzada (mínim *JRE 6*) per poder executar l'*applet Java* per signar les peticions amb certificats *DNIe* i comunicar-se amb el *proveïdor* d'identitat sense problemes.

També cal tenir en compte que el client necessitarà disposar d'un dispositiu lector de *DNI-e* degudament instal·lat i configurat (mitjançant el software disponible a http://www.dnielectronico.es) per tal de que l'*applet Java* pugui accedir als certificats i claus privades del *DNI-e* i efectuar les signatures de les peticions de recursos sense problemes (cal recordar que el producte està dissenyat per funcionar amb aquesta mena de certificats, obligatòriament). En els annexos d'aquest document es poden trobar les especificacions del dispositiu lector de *DNI-e* de la marca *Chipnet* que s'ha utilitzat en el desenvolupament i proves del producte.

5.5.1.4. Màquina virtual Java

Tot i que s'ha comentat en el cas del navegador del client, cal fer èmfasi en el fet que tots els sistemes sobre els que es desplegarà el producte han de disposar, com a mínim, de l'entorn d'execució *Java JRE 6* (que posarà a disposició la *màquina virtual Java* necessària en el client). En el cas dels sistemes sobre els que es desplegaran els servidors (*proveïdor de serveis* i *proveïdor d'identitat*) cal tenir en comte que caldrà definir correctament la variable d'entorn *JAVA_HOME* cap al directori on estigui instal·lat aquest entorn d'execució. Eventualment, algunes distribucions del servidor *Tomcat 6.0* poden requerir un entorn de desenvolupament *Java* (*JDK 6*), enlloc de simplement l'entorn d'execució, definint aleshores la variable *JAVA_HOME* en relació a aquest entorn.

5.5.2. Desplegament del producte

Per tal de distribuir el producte es facilitaran dos fitxers comprimits .ZIP que contindran tot el necessari per desplegar el producte sobre la plataforma proposada:

- *ProveidorServeis.ZIP*:

  Dins d'aquest fitxer trobarem el següent:

  - *ProveidorServeis.war*: aquest fitxer és el que conté la part del producte corresponent al *proveïdor de serveis* pròpiament dit. Un cop es desplega en el servidor d'aplicacions (normalment hi ha prou amb copiar aquest fitxer en el directori */webapps* de *Tomcat*, però també es pot efectuar mitjançant el gestor web del propi servidor) es crearà la següent estructura dins del directori */webapps/ProveidorServeis* de *Tomcat* (es comentaran els fitxers i directoris més rellevants):

    - En l'arrel d'aquest directori trobarem el fitxer *index.html* (que serà al primer que s'accedirà del *proveïdor de serveis*), així com el fitxer *testapplet.html* (per efectes de proves) i els fitxers *.jar* corresponents tant a l'*applet Java* que carregarà el *proveïdor de serveis*, com a les llibreries que necessita aquest applet per funcionar correctament (necessàries per efectuar les crides *SOAP* al servei web del *proveïdor d'identitat*). També es pot trobar el fitxer *SignApplet* que conté el magatzem de claus utilitzades per signar l'*applet* (per més informació, consulteu l'annex referent als documents *PKI* utilitzats).

    - En el directori */WEB-INF* podem trobar el directori */cnf* que conté el fitxer *pkiconfig.properties* on s'ha definit el magatzem de claus, i el password necessari per utilitzar-lo, per tal que el proveïdor de serveis pugui signar les dades de la petició. Cal tenir en compte que la ruta definida per al magatzem de claus és relativa al directori arrel del desplegament del *proveïdor de serveis*. També conté la informació relativa al directori *trustedCACerts* per tal de validar la signatura de la resposta del *proveïdor d'identitat*.

    - També en el directori */WEB-INF* trobem el directori */pki* que conté el magatzem de claus esmentat anteriorment. Dins d'aquest directori es troba el subdirectori *trustedCACerts* amb el certificat arrel necessari per validar la signatura de la resposta del *proveïdor d'identitat*.

    - Finalment, dins del directori */WEB-INF* trobem el fitxer *web.xml* amb la informació necessària pel funcionament de l'aplicació web (aquest fitxer ha estat creat automàticament per l'entorn de desenvolupament) així com els directoris */classes* i */lib* amb els *bytecodes* (executables *Java*) del *servlet Java* corresponent al proveïdor de serveis i a la llibreria pròpia de funcions criptogràfiques i *XML* que s'ha utilitzat en la confecció del producte.

  - *Directori /pki_ssl*: aquest directori conté el magatzem de claus proposat per configurar les capacitats *SSL* del servidor d'aplicacions on es desplegarà el *proveïdor de serveis*.

- *ProveidorIdentitat.ZIP*:

  Dins d'aquest fitxer trobarem el següent:

- *ProveidorIdentitat.war*: aquest fitxer és el que conté la part del producte corresponent al *proveïdor d'identitat* pròpiament dit. Un cop es desplega en el servidor d'aplicacions (de la mateixa forma que s'ha comentat amb el *proveïdor de serveis*) normalment hi ha prou amb copiar aquest fitxer en el directori */webapps* de *Tomcat*, però també es pot efectuar mitjançant el gestor web del propi servidor) es crearà la següent estructura dins del directori */webapps/ProveidorIdentitat* de *Tomcat* (es comentaran els fitxers i directoris més rellevants):

  - En l'arrel d'aquest directori trobarem el fitxer *index.html* (que serà al primer que s'accedirà del *proveïdor d'identitat*), i que ens donarà accés a la interfície per administrar els clients, serveis i permisos definits en la base de dades (i necessaris per a que el *proveïdor d'identitat* reconegui les parts implicades en la petició d'un recurs, i poder resoldre aquesta petició), implementada mitjançant el fitxer *Administrar.jsp*, també ubicat en aquest directori, juntament amb els recursos (gràfics i definició de pàgines d'estil) del que fa ús per qüestions de disseny.

  - En el directori */WEB-INF* podem trobar el directori */cnf* que conté els fitxers *dbconfig.properties* i *adminconfig.properties*. Cal posar de rellevància que aquests fitxers contenen informació relativa a les configuracions de l'aplicació web encarregada de la gestió de les dades persistents, i no pas del servei web del *proveïdor d'identitat*. En el primer fitxer trobarem les dades relatives a la connexió amb la base de dades (cadena de connexió que inclou usuari i *password*), i en la segona trobarem les dades d'usuari i *password* necessàries per entrar com a administrador a aquesta aplicació web encarregada de la gestió de dades. Cal remarcar el fet que el *password* no s'emmagatzema directament, sinó que s'emmagatzema el *hash MD5* que s'utilitzarà per comparar amb el que l'usuari entri en la finestra de *login*.

  - També en el directori */WEB-INF* trobem el directori */classes* que conté la classe corresponent al servei web del *proveïdor d'identitat*, així com els fitxers *dbconfig.properties* i *pkiconfig.properties* que utilitza aquest servei web. El fet que aquests fitxers de configuració es trobin en aquesta ubicació és pel fet que el context dels serveis web en *Java* no permet accedir a rutes relatives per sota del directori */classes* (el directori on es troba la classe del servei web), a no ser que s'indiqui una ruta absoluta. És per això també que la configuració *pki*, tal com s'indica en l'interior del fitxer de configuració, necessita fer referència a un directori mitjançant una ruta absoluta, i no relativa com era el cas del *proveïdor de serveis*.

  - Finalment, en el directori */WEB-INF* també es troba el directori */lib* amb les llibreries necessàries (les pròpies i les que permeten rebre les crides *SOAP* des de l'*applet Java*), així com els directoris i fitxers necessaris per a que tant el servei web com l'aplicació de gestió de dades funcionin correctament en el servidor d'aplicacions.

- Directori */pki_ssl*: aquest directori, de la mateixa forma que en el *proveïdor de serveis*, conté el magatzem de claus proposat per configurar les capacitats *SSL* del servidor d'aplicacions on es desplegarà el *proveïdor d'identitat*.

- Directori */pki*: aquest directori contindrà tots els fitxers necessaris per a que el servei web del *proveïdor d'identitat* pugui signar la resposta i validar les peticions rebudes (també conté el directori */trustedCACerts* amb els certificats arrel necessaris per validar les signatures del *client* i del *proveïdor de serveis*) i caldrà ubicar-lo en una localització de la qual s'obtingui la ruta absoluta i s'afegeixi en el fitxer *pkiconfig.properties* de dins del directori */classes* (el del servei web) que s'ha comentat anteriorment. Per defecte està definit com a *C:/pki*, però es pot definir qualsevol altre localització sempre i quan es mantinguin els permisos de lectura (fet a tenir en compte especialment si es vol desplegar sobre sistemes *Linux*).

- Directori */db_struct*: en aquest directori est podrà trobar el fitxer *tfcseguretat.sql* amb les instruccions *SQL* per tal de crear les taules que conformen la estructura de la base de dades de l'aplicació del *proveïdor d'identitat*. Es recomana crear-ho en la base de dades *tfcseguretat* mantenint les configuracions (relatives a nom i *password* d'usuari) que es troben predefinides en els fitxers de configuracions de bases de dades comentats anteriorment, o bé propagar els possibles canvis en ubicació o nom d'usuaris a aquests fitxers de configuració. S'ha advertir que aquest usuari que cal definir en el servidor és necessari que disposi de permisos, com a mínim, per efectuar *selects, inserts, updates* i *deletes* sobre la base de dades. Aquestes instruccions SQL també afegeixen alguns camps que han de permetre funcionar inicialment, si bé caldrà afegir, mitjançant l'aplicació de gestió de dades del *proveïdor d'identitat*, les dades relatives a nous usuaris i permisos associats (consultar la secció de funcionament). Com a últim comentari al respecte, denotar que en el desenvolupament del projecte s'ha fet ús de l'eina gratuïta *Navicat 8 Lite for MySQL*, que ha facilitat molt les tasques de creació de taules i usuaris de la base de dades (tractant-se d'un *front-end* molt senzill d'utilitzar).

Addicionalment, també es proporcionarà un fitxer *CodiFont.ZIP* que contindrà un directori per cada projecte, efectuat mitjançant l'*IDE NetBeans 6.5*, dels fitxers que contenent el codi font de totes les aplicacions del producte.

Per tal de facilitar la distribució del producte, enfocat a l'entrega final del *TFC*, és possible que s'hagi unificat tot en un únic *.ZIP* que contingui les carpetes *CodiFont*, *ProveidorServeis* i *ProveidorIdentitat* homòlogues en contingut als fitxers que s'acaben de comentar.

5.5.3. Funcionament

A continuació es descriurà el funcionament del producte, un cop efectuat el seu correcte desplegament tal com s'ha indicat en els punts anteriors.

5.5.3.1. Accés a un recurs del proveïdor de serveis

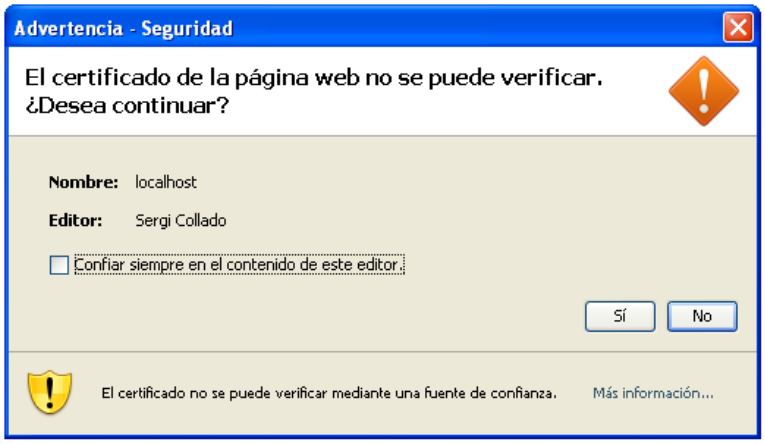
Un client que vol accedir a un recurs del *proveïdor de serveis* ha d'accedir, mitjançant el navegador web, a la seva *URL*. Si suposem que el client i el *proveïdor de serveis* fossin a la mateixa màquina, la *URL* del *proveïdor de serveis* seria:

https://localhost:8443/ProveidorServeis
Cal tenir present que també es pot utilitzar:
http://localhost:8080/ProveidorServeis
però si el desplegament ha estat correcte, es veurà com realment el servidor ens adreçarà a la primera *URL* indicada, de forma automàtica.

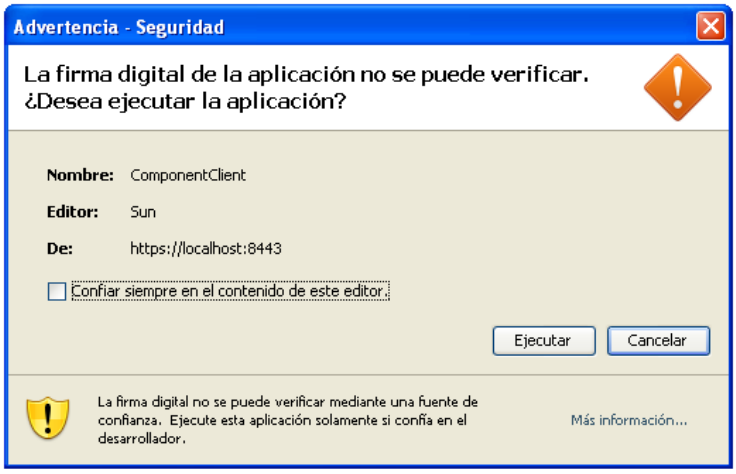
L'accés a aquesta adreça provocarà una violació de seguretat (el navegador ens avisarà) donat que el certificat utilitzat per configurar la connexió *SSL* del protocol *HTTPS* no és de confiança. En aquest punt s'aconsella afegir una excepció de seguretat i configurar el navegador per tal que consideri el certificat com si fos de confiança. D'aquesta forma, el servidor finalment ens redirigiria a la pantalla de selecció de recurs, dels recursos disponibles d'aquest servei:

Un cop seleccionat un recurs (fent *clic* sobre l'enllaç corresponent), el *proveïdor de serveis* carregarà l'*applet Java* en la mateixa finestra del navegador. En carregar aquest *applet*, donat que es tracta d'un *applet* signat (per tal de conferir-li els privilegis necessaris per accedir al magatzem de claus del sistema on s'executa (i per tant als certificats i claus del *DNI-e*) el primer que apareixeran seran les peticions d'autorització per executar-lo:

Una primera petició d'autorització relativa encara al certificat de la connexió:



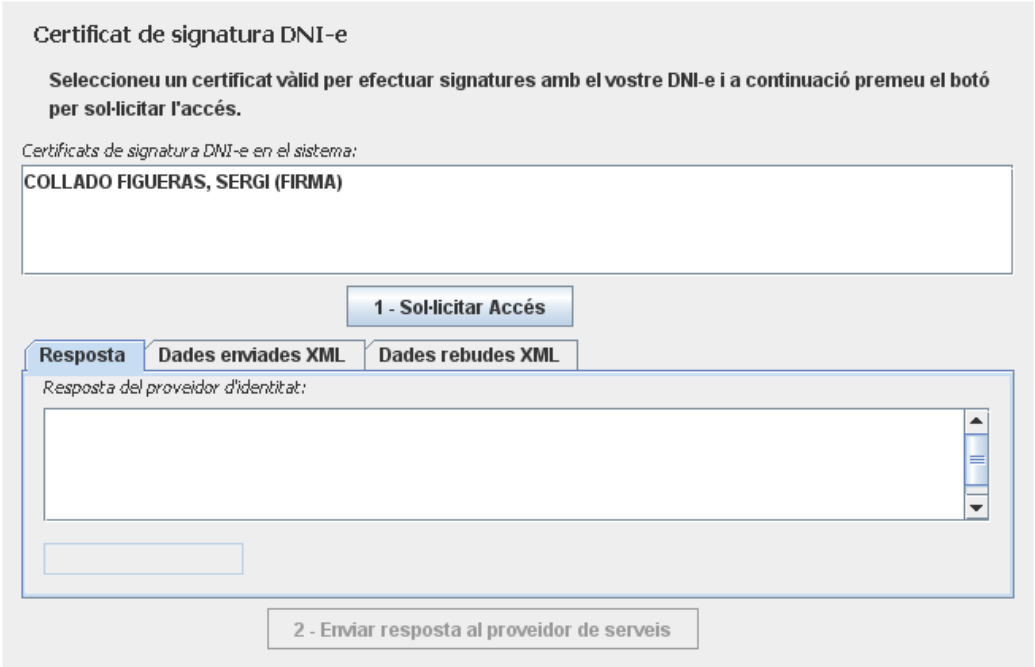
I una segona relativa a la signatura amb la que s'ha signat l'*applet java* (cal recordar que certes funcionalitats de l'*applet* requereixen més permisos que s'obtenen signant-lo):



Caldrà, doncs, prémer el botó *Ejecutar* per tal que l'*applet* es carregui correctament.

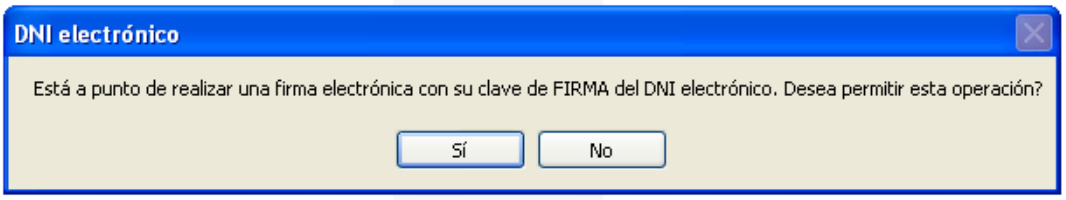
En carregar-se accedirà al magatzem de claus per tal de seleccionar aquells certificats *DNI-e* habilitats per signar documents. Per tant, ens apareixerà un parell de vegades la finestra per a que introduïm el *PIN* del nostre *DNI-e* (aquesta funcionalitat ve atorgada pels controladors de instal·lats del dispositiu lector de *DNI-e,* tal com s'explica en l'annex corresponent).

Un cop s'hagi carregat l'*applet*, es disposarà del següent formulari:

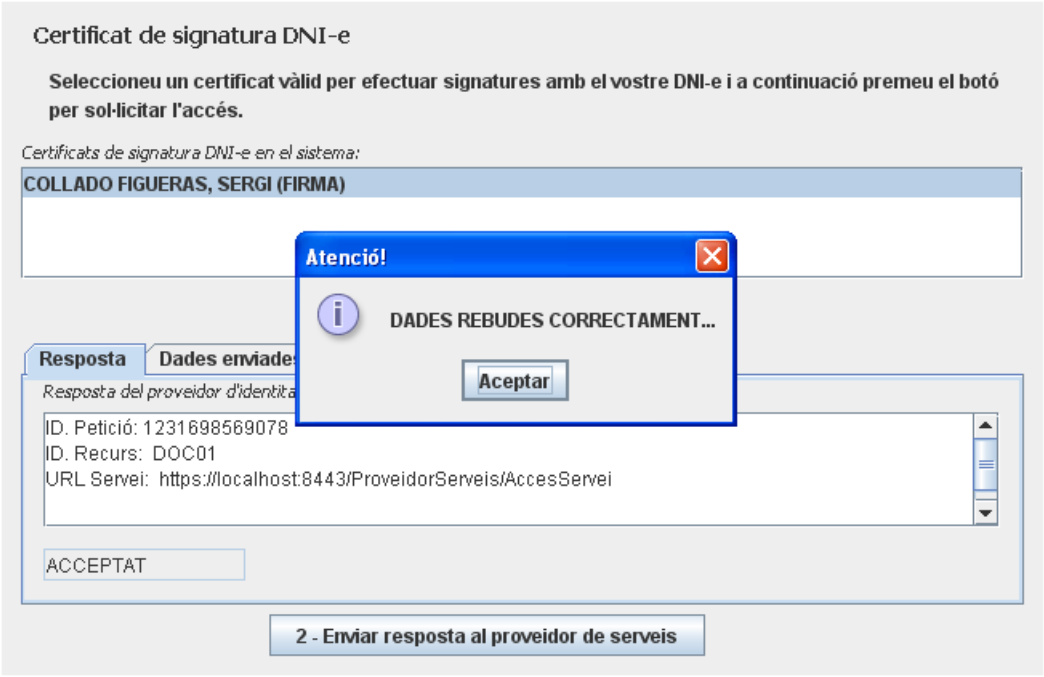


En aquest formulari, tal com s'ha definit en la documentació del disseny de la interfície, es disposa d'una llista de certificats *DNI-e* vàlids per signar i que estiguin presents en aquest moment en l'equip. En cas de no existir-ne cap, el botó de *Sol·licitud d'accés* romandrà desactivat. Com a lleugera diferència respecte al disseny de la interfície proposat, finalment s'ha n afegit dues pestanyes per tal de fer més exhaustiva la informació disponible sobre les dades que s'intercanviaran amb el *proveïdor d'identitat* (en la pestanya *dades enviades XML* es disposarà dels documents literals *XML* enviats, en format text, i en la pestanya *dades rebudes XML* es disposarà de la resposta literal per part del servei web del *proveïdor d'identitat*).

En seleccionar el certificat *DNI-e* que es vol utilitzar i prémer el botó *Sol·licitar Accés* apareixerà l'advertència de que s'està a punt de fer servir una signatura electrònica amb la clau de signatura del *DNI-e*.

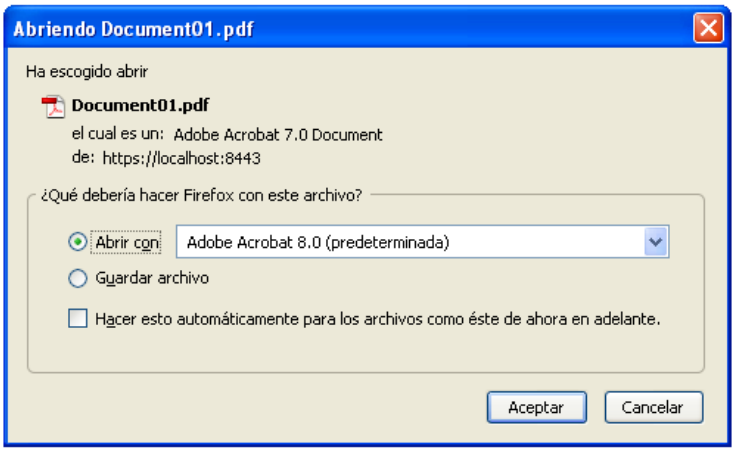


Un cop acceptat aquest missatge, l'*applet* enviarà les dades signades al *proveïdor d'identitat* i esperarà la seva resposta. Finalment, quan es rebi la resposta, es mostrarà la informació sobre la petició i l'estat de resolució al que ha arribat el *proveïdor d'identitat*:



En aquest punt, el botó *enviar resposta* quedarà habilitat i l'usuari el podrà prémer per tal de tornar a sol·licitar l'accés al recurs del *proveïdor de serveis*, aquest cop amb la resolució signada del *proveïdor d'identitat*.

En cas que la resolució estigui acceptada, el *proveïdor de serveis* facilitarà l'accés al recurs sol·licitat:



5.5.3.2. Administració de les dades del proveïdor d'identitat

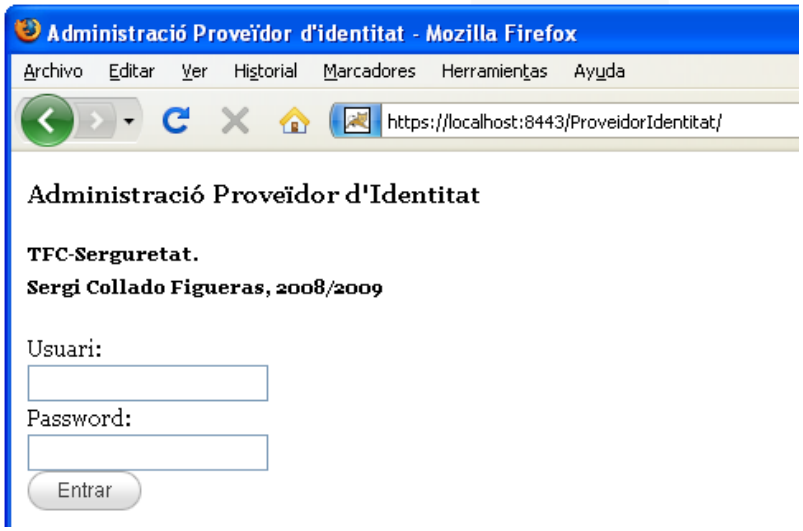
El funcionament de l'aplicació d'administració de les dades del *proveïdor d'identitat* està àmpliament descrit en l'apartat *Administració del proveïdor d'identitat* del punt dedicat a descriure el disseny de la interfície de l'aplicació.

Amb tot i això, cal dir que les dades per poder accedir a aquesta aplicació, tal com s'ha comentat amb anterioritat, s'emmagatzemen en un fitxer de propietats en els recursos de l'aplicació, pel que no està prevista cap interfície per administrar els usuaris que tindran accés.

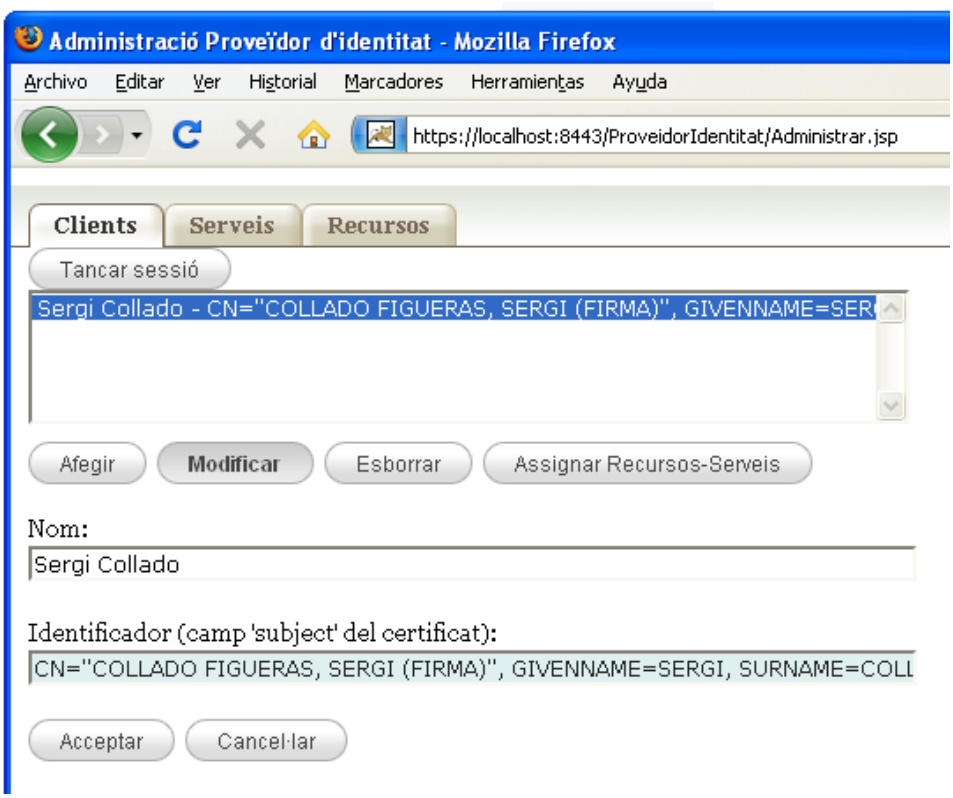
Actualment, per poder entrar a aquesta aplicació d'administració caldrà utilitzar les següents dades:

> ***Connectar a:*** https://localhost:8443/ProveidorIdentitat (de la mateixa forma que en el *proveïdor de serveis*, la connexió via *HTTP* i el port 8080 ens haurà de redirigir cap a aquesta *URL*)
> ***Usuari:*** admin
> ***Password:*** tfcseguretat

L'aspecte del formulari de presentació és el següent:



I l'aspecte de l'aplicació, vista en el navegador web, és el següent:



Donat que les dades i la funcionalitat ja han estat extensament comentades en apartats anteriors, no s'incidirà en el funcionament d'aquesta part del producte, per tal de no reincidir en explicacions anteriors. Cal tenir en compte, únicament, que les modificacions en vers al concepte descrit anteriorment només han implicat l'aparició d'algun botó per eliminar alguna associació, però el funcionament general roman intacte.

5.6. Variacions del producte final respecte el disseny inicial previst

Finalment no existeixen grans variacions en el producte final respecte el que s'havia previst inicialment en el disseny. Realment, les variacions que han existit han estat prou petites com per haver estat modificades durant el transcurs de la mateixa fase de disseny del producte, i han estat fruït de recomanacions o observacions del consultor que han fet reflexionar sobre la pertinència de la variació en qüestió.

Com a canvi més destacat es pot considerar l'afegiment del camp *cert_servei* a la taula *serveis* per tal d'emmagatzemar també el camp *subject* del certificat utilitzat per *proveïdor de serveis* per signar les dades de la petició. Aquest camp no es va contemplar inicialment i segurament ha estat el que més modificacions, tant de disseny com de codificació, ha acabat comportant, però com es pot apreciar, i tal com s'ha comentat amb anterioritat, estat un canvi menor.

Per sort, la concepció de l'escenari sobre el que s'ha desenvolupat aquest *TFC* ha estat ben entesa des d'un començament, fet que ha esdevingut en un enfocament correcte de l'anàlisi i el disseny des d'un bon principi.

Però un fet que potser sí que val la pena remarcar ha estat el fet de que, tot i que en aquest document sempre s'ha parlat de fer servir l'eina *Eclipse* com a eina de desenvolupament en *Java*, en un cert punt de l'etapa de codificació es va optar per migrar tot el desenvolupament del producte cap a l'eina *NetBeans*, que ha facilitat enormement algunes tasques (com, per exemple, el disseny de la interfície de l'*applet java*) gràcies a alguns dels *wizards* dels que disposa. Aquest canvi d'interfície va provocar que es modifiqués la selecció del framework que s'havia seleccionat inicialment per la creació del servei web. En les primeres versions del producte es va començar a utilitzar *Axis* i *JAX-RPC*, tal com s'ha comentat als primers punts de la present memòria, però finalment es va optar per configurar-ho utilitzant *JAX-WS* de forma que el *client* (l'*applet java*) ja no necessitava un munt de llibreries que el feien augmentar molt de pes. Amb aquesta modificació es van aprofitar les facilitats de *NetBeans* per configurar serveis web, així com es va augmentar també l'eficiència, al disposar ara d'un *applet* veritablement lleuger i ràpid de descarregar.

5.7. Conclusions i futures vies de treball

Com a conclusió, un cop realitzat aquest *TFC*, es pot assegurar que la implementació de l'escenari proposat és completament factible amb les eines utilitzades. La plataforma de programació sobre *Java*, i el servidor d'aplicacions que el suporta (*Apache Tomcat*) permeten dur a terme totes les operacions necessàries: des de la utilització de tots els mecanismes d'una *infraestructura criptogràfica de clau pública*, fins la comunicació amb processos distribuïts, passant per l'accés als certificats allotjats al xip del *DNI-e*, o l'accés a la persistència implementada sobre un servidor de bases de dades (*MySQL*).

Amb tot i amb això, cal tenir en compte diverses consideracions respecte al transcurs d'aquest projecte:

- En primer lloc, el temps i esforços dedicats han estat exageradament superiors a les expectatives. Aquest fet, lluny de semblar negatiu, no està enfrontat amb el fet que la realització del projecte ha esdevingut, també, exageradament més apassionant del que s'havia previst. Però cal tenir-ho en compte: en termes de realisme, enfrontar-se a un projecte d'aquestes pretensions, que requereix nombrosos fronts d'investigació en diverses tecnologies, amb uns coneixements inicials molt bàsics de Java, ha deixat la sensació d'haver superat un dels reptes informàtics més titànics als que mai s'havien fet front per part de l'autor, ni tant sols en l'àmbit professional.

- En segon lloc, cal posar de manifest que l'enorme documentació de que es disposa sobre la utilització de *Java* en els diferents aspectes de la implementació del projecte, de vegades ha esdevingut una arma de doble tall. Si bé és cert que molts dels problemes que han sorgit en el transcurs de la implementació han estat ràpidament solucionats gràcies tant a la documentació oficial de *Sun* sobre *Java* com a la no oficial extreta de diverses pàgines, fòrums i llibres de temàtica específica, sovint l'autor s'ha trobat navegant sense rumb en un mar d'informació, sense saber massa cap a on calia dirigir la mirada, aclaparat per tan immensa quantitat d'informació. Val a dir que certs problemes han estat prou específics com per dificultar, a més, la troballa d'alguna solució funcional.

- Finalment, s'ha pogut arribar a la conclusió de que, si bé els sistemes d'identificació mitjançant una targeta electrònica (com el *DNI-e*) esdevenen un nou paradigma en quant a la fiabilitat de les comunicacions que podem establir des de un ordinador domèstic per efectuar operacions quotidianes amb plena seguretat, val a dir que els dispositius electrònics utilitzats (al menys el que ha utilitzat l'autor, de la casa *Chipnet*, distribuït per uns grans magatzems de renom, i recomanat com a estàndard per la pròpia web www.dnie.es) encara no disposen del grau d'efectivitat necessari com per operar al 100% sense problemes. La lectura de les dades electròniques esdevé impossible sobre una màquina funcionant amb *Windows Vista x64*, i provoca molts problemes (lectures errònies de les claus, no identificació dels certificats continguts a la targeta) sobre un equip funcionant amb *Windows XP*. A més, cal remarcar també que el funcionament normal de qualsevol d'aquests equips es veu alterat si es deixa el lector *USB* connectat, amb el *DNIe* inserit, donat que molts processos *normals* del sistema operatiu provoquen l'aparició del diàleg en que se'ns demana la clau per accedir als certificats de la targeta.

Un cop esmentades aquestes consideracions, cal posar també de manifest el fet que l'aspecte funcional i d'utilitat real del projecte ha quedat relegat a un segon o tercer pla (doncs no era l'objectiu d'aquest *TFC*), però en realitat en una hipotètica funcionalitat comercial autèntica

(com per exemple accedir a dades veritablement confidencials o crítiques dels clients) és on es veuria justificat l'escenari proposat.

Al respecte, cal destacar un petit detall detectat en les últimes fases de desenvolupament de l'entrega final d'aquest *TFC* referent al fet que el temps en que s'ha efectuat la signatura de la petició del recurs no es comprova abans de facilitar l'accés a aquest. Es podria pensar que una petició és vàlida si el moment en que s'ha efectuat està contemplat pels permisos establerts en la base de dades, independentment del moment en que s'acabi accedint a aquest recurs (es pot tenir l'*applet* carregat amb la resolució acceptada durant hores, abans d'accedir finalment al recurs). Però és possible que algun escenari més realista requereixi una comprovació més restrictiva en aquest punt, i calgui tenir en compte també el moment en que s'acaba accedint al recurs.

Per tant, una possible via de treball futur podria esdevenir de l'aprofitament de les tecnologies implementades en aquest projecte per atorgar un caràcter completament robust de seguretat i gestió d'usuaris a qualsevol projecte *J2EE* de gran envergadura.

En l'àmbit personal també es pot afirmar amb rotunditat que s'han assolit els objectius previstos. La conclusió dels estudis que l'autor va començar l'any 1995 en la *Facultat d'Informàtica de Barcelona* (*UPC*), que va continuar, intermitentment, en la *Escola Universitària d'Informàtica de Sabadell* (*UAB*) fins al 2003, i que va reprendre l'any 2007 en la *UOC*, no podia deixar millor satisfacció que la realització amb èxit d'aquest *TFC*, en un àrea desconeguda però que ha esdevingut apassionant.

6. Glossari

- **Administrador:** Usuari amb privilegis especials per tenir accés a recursos o zones de gestió i configuració d'una aplicació o sistema concret.

- **Apache Tomcat:** Servidor d'aplicacions i serveis web en Java, desenvolupat per la companyia *ApacheSoftware Foundation* sota llicència *opensource*.

- **Applet java:** Aplicació *Java* que s'executa sobre el navegador de l'usuari, i que ve proveïda pel servidor de la pàgina web que la carrega. En certs casos, aquestes aplicacions estan signades de forma que aconsegueixen sol·licitar una escalada de privilegis per tal d'accedir a més recursos de la màquina que les executa.

- **Axis:** Llibreries *Java* que permeten, entre d'altres coses, implementar una infraestructura de comunicació amb serveis web mitjançant *SOAP*.

- **Bytecode:** Fitxer binari *Java*. Seria l'equivalent al tradicional *codi màquina*, però per una *màquina virtual Java*.

- **CA emissora:** Les sigles CA (de l'anglès *Certificate Authority*) simbolitzen el que es coneix com a *Autoritat de Certificació*, i en aquest cas, el terme fa referència a qualsevol *CA* que emet un certificat.

- **Certificat digital:** Document digital mitjançant el qual una CA emissora (normalment de confiança) garanteix la vinculació que hi ha entre la identitat d'un subjecte i la seva *clau pública*.

- **Chipnet:** Companyia que fabrica i distribueix el model de lector USB de *DNI electrònic* utilitzat en el desenvolupament i proves d'aquest *TFC*.

- **Criptografia:** Estudi de les tècniques matemàtiques per protegir la informació contra accessos per part d'entitats no autoritzades.

- **Criptografia de clau pública:** Infraestructura criptogràfica que defineix unes polítiques i procediments de seguretat que permeten executar amb garanties operacions criptogràfiques com el signat de documents electrònics o el xifrat de comunicacions.

- **CRL:** Llistes de Certificats Revocats (de l'anglès *Certificate Revocation List*). És un mecanisme per detectar aquells certificats que ja no són vàlids, mitjançant els seus números de sèrie. Al ser un sistema *off-line* requereix actualitzar constantment aquestes llistes amb les últimes versions disponibles per tal de maximitzar l'efectivitat.

- **DNI electrònic:** Nou Document Nacional d'Identitat que disposa d'un mecanisme electrònic (un xip) gràcies al qual un ciutadà té al seu abast els certificats d'identificació i signatura digital necessaris per l'autenticació i utilització d'alguns sistemes segurs.

- **Eclipse:** Entorn de desenvolupament pensat principalment per facilitar les tasques de producció de programari en *Java*.

- **Hash:** Una funció *hash* és aquella capaç de calcular un resum a partir de les dades d'un document.

- **HTML:** Llenguatge de marques d'hyper-text (de l'anglès *Hyper Text Markup Language*) utilitzat de forma predominant en la construcció de pàgines web.

- **HTTP:** Protocol de transmissió d'hyper-textos (de l'anglès *Hyper Text Transfer Protocol*) ideat per definir la sintaxis i la semàntica necessàries per la transmissió de documents *HTML*.

- **HTTPS:** Protocol *HTTP* implementat sobre un canal xifrat *SSL*.

- **Infraestructura de clau pública:** Veure *Criptografia de clau pública*.

- **J2EE:** Plataforma de desenvolupament *Java* (de l'anglès *Java Enterprise Edition*) enfocada a una arquitectura distribuïda, basant-se en components modulars executant-se en un servidor d'aplicacions.

- **Java:** Llenguatge de programació orientat a objectes desenvolupat per *Sun Microsystems*. Gràcies a que està enfocat a ésser executat en una *màquina virtual java*, és un dels llenguatges de programació més portables i versàtils.

- **JAX-RPC:** Llibreries *Java* que permeten, entre d'altres coses, implementar una infraestructura de comunicació amb serveis web mitjançant *SOAP*.

- **JAX-WS:** Llibreries *Java* que permeten, entre d'altres coses, implementar una infraestructura de comunicació amb serveis web mitjançant *SOAP*.

- **JDBC:** Llibreria Java que permet connectar amb dades persistents (com per exemple un servidor de bases de dades *MySQL)*.

- **JSP:** Tecnologia *Java* (de l'anglès *Java Server Pages*) que permet generar contingut web dinàmicament, en format *HTML*, *XML* o d'altre tipus.

- **Keytool:** Eina que proveeix el *JDK* (de l'anglès *Java Development Kit*, que fa referència a les eines de desenvolupament *Java*) i que permet generar certificats de forma senzilla per ésser utilitzats en temps de desenvolupament (per exemple, per signar un *applet java*).

- **Lector de DNI electrònic:** Dispositiu electrònic, normalment connectat a un ordinador mitjançant un port *USB*, que permet interactuar amb els certificats de signatura i identificació ubicats al xip del *DNI-e*.

- **Màquina Virtual Java:** Plataforma que permet executar els *bytecodes java* en un sistema concret. Cada sistema diferent ha de disposar de la seva *màquina virtual java* per tal de poder executar aquests *bytecodes*.

- **MySQL:** Servidor de bases de dades que disposa d'una distribució gratuïta i prou robusta com per utilitzar en projectes de poca i mitjana envergadura.

- **Navegador web:** Software utilitzat per accedir a les pàgines web allotjades o generades per un servidor. Els més comuns són el *Microsoft Internet Explorer* i el *Mozilla Firefox*.

- **NetBeans:** Entorn de desenvolupament pensat principalment per facilitar les tasques de producció de programari en *Java*.

- **OCSP:** Protocol *online* de comprovació d'estat de certificats digitals (de l'anglès *Online Certificate Status Protocol*). Guanya en eficiència la comprovació mitjançant *CRL* donat que aquest protocol permet accedir sempre a la informació més actualitzada.

- **OpenSSL:** Paquet d'eines criptogràfiques que permet operar amb diversos protocols de criptografia de clau pública.

- **PKI:** De l'anglès *Public Key Infrastructure*, veure *Criptografia de clau pública*.

- **Proveïdor de servei:** Aplicació web capaç de donar accés a uns determinats recursos, segons l'acceptació, per part d'un *proveïdor d'identitat* de les condicions i actors d'una petició formulada per un *client*.

- **Proveïdor d'identitat:** Servei web capaç de decidir si una petició formulada per part d'un *client* a un recurs d'un *proveïdor d'identitat* ha de ser acceptada o no, en funció dels elements criptogràfics adjunts a les dades de la petició, i als permisos establerts en la persistència del sistema.

- **Safelayer:** Proveïdor de segells de temps (entre d'altres serveis) que posa a disposició un servei web de proves, utilitzable per obtenir aquests segells de temps.

- **Segell de temps:** Característica temporal en una signatura. Mitjançant l'actualització d'una signatura amb un segell de temps, es disposa de les dades temporals en que s'ha efectuat aquesta actualització, de forma irrefutable.

- **Servlets java:** Tecnologia *Java* que permet generar contingut web dinàmicament, en format *HTML*, *XML* o d'altre tipus, gràcies a la creació d'objectes que amplien les funcionalitats d'un servidor d'aplicacions.

- **Signatura digital:** Mecanisme criptogràfic que associa la identitat d'un subjecte a un document o missatge.

- **SOAP:** Protocol (de l'anglès *Simple Object Access Protocol*) que defineix com es poden comunicar dos objectes mitjançant el intercanvi de dades *XML*.

- **SQL:** Llenguatge d'accés a bases de dades relacionals (de l'anglès *Standard Query Language*).

- **TrustedX:** Denominació del paquet de llibreries amb les que es pot operar amb els serveis web de *SafeLayer.*

- **W3C:** Consorci internacional de producció d'estàndards web (de l'anglès *World Wide Web Consortium*).

- **X.509:** Conjunt de recomanacions i estàndards pels algoritmes, certificats i dades en general de la *infraestructura de clau pública*.

- **XML:** Metallenguatge extensible d'etiquetes (de l'anglès *eXtensible Markup Language*) desenvolupat pel *W3C* i que ha esdevingut un estàndard en el intercanvi d'informació estructurada entre diferents plataformes.

7. Bibliografia i Webgrafia

Hook, D. (2005). *Beginning Cryptography with Java*. Wrox Press.

McLaughlin, B. (2001). *Java & XML, 2nd Edition*. O'Reilly.

Bloch, J. (2008). *Effective Java, Second Edition*. Addison-Wesley. (Sun Microsystems).

Gosling, J.; Joy, B.; Steele, G.; Bracha, G. (2005). *The Java Language Specification, Third Edition*. Addison Wesley Professional.

Esteban, Á. (2000). *Tecnologías de Servidor con Java: Servlets, Jababeans, JSP*. Grupo Eidos.

Perry, B. W. (2004). *Java Servlet & JSP Cookbook*. O'Reilly.

Hall, M. (2000). *Core Servlets and JavaServer Pages*. Prentice Hall.

Parsian, M. (2006). *JDBC Metadata, MySQL and Oracle Recipes: A problem-solution approach*. Apress.

Knudsen, J.; Niemeyer, P. (2005). *Learning Java, 3rd Edition*. O'Reilly.

Cole, B.; Eckstein, R.; Elliott, J.; Loy, M.; Wood, D. (2002). *Java Swing, 2nd Edition*. O'Reilly.

Adamson, c.; Marinacci, J. (2005). *Swing Hacks*. O'Reilly.

Tidwell, D.; Snell, J.; Kulchenko, P. (2001). *Programming Web Services with SOAP*. O'Reilly.

Englander, R. (2002). *Java and SOAP*. O'Reilly.

Domingo, J.; Herrera, J.; Rifà, H. (2006). *Criptografia*. UOC.

Herrera, J.; García, J.; Perramón, X. (2008). *Seguretat en xarxes de computadors*. UOC.

Campderrich, B. (2004). *Enginyeria del Programari*. UOC.

Sistac, J.; Camps, R.; Costal, D.; Martín, C.; Rodríguez, E. (2005). *Bases de dades I.* UOC.

Collado, S. (2008). *Pràctica 2 Criptografia*. UOC. [*Memòria i codi font*]

Calavera, D. (2007). *Java 6 y tu DNI electronico*. [en línia] Think in code. http://thinkincode.net/2007/2/14/java-6-y-tu-dni-electronico [data de consulta: Oct. 2008]

Diversos. (des de 1997). *(Diversos articles sobre Java)*. [en línia] http://www.rgagnon.com/howto.html [data de consulta: Oct. 2008]

TheServerSide; Palos, J. A. (2008). *Introducción a los servicios web en Java*. [en línia] http://www.programacion.com/java/tutorial/servic_web/ [data de consulta: Oct. 2008]

Apache; Palos, J. A. (2008). *El API Apache SOAP v2.2.* [en línia]. http://www.programacion.net/java/tutorial/apachesoap/3/ [data de consulta: Nov. 2008]

Chileno, J. P. (2007). *Creación de un website seguro mediante SSL*. [en línia]. http://lacasadeljota.blogspot.com/2007/10/creacin-de-un-webservice-seguro.html [data de consulta: Nov. 2008].

García, C. (2006). *Guia de Apache Axis*. [en línia] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=axis [data de consulta: Nov. 2008]

Diversos. (des de 2002). *(Diversos articles)*. [en línia] http://es.wikipedia.org [data de consulta: des de Sep. 2008 fins Gen. 2009]

Sun Microsystems. (2008). *Documentació de les API*. [en línia] http://www.sun.com/documentation/ [data de consulta: Oct. 2008]

Apache Software Foundation. (2008). *Documentació Apache Tomcat 6.0*.[en línia] http://tomcat.apache.org/tomcat-6.0-doc/index.html [data de consulta: Oct. 2008]

Apache Software Foundation. (2008) *Documentació Apache Axis Web Services*. [en línia] http://ws.apache.org/axis/java/index.html [data de consulta: Oct. 2008]

Direcció General de la Policia i la Guardia Civil. (2008). *Guia de Referència del DNI-e (i documents associats disponibles a la mateixa web)*. [en línia] http://www.dnielectronico.es/Guia_Basica/index.html [data de consulta: Oct. 2008]

8. Annexos

8.1. Material PKI utilitzat

El material PKI utilitzat per proporcionar els mecanismes de seguretat necessaris en el producte final ha estat:

- ***Server.p12:*** Certificat i claus generades amb l'eina *OpenSSL* (i autoritzats per uns certificats i claus CA generats amb la mateixa eina) i emmagatzemats en format PKCS#12. S'utilitza per configurar les connexions *SSL* dels servidors d'aplicacions.
  Els passos que s'han seguit per crear aquest fitxer han estat:

  - Cal crear la següent estructura de directoris en el lloc on es vulguin crear els certificats:

    ./demoCA [directori]
    ./demoCA/newcerts [directori]
    ./demoCA/index.txt [fitxer buit]
    ./demoCA/serial [fitxer amb el valor 01 a dins]

    (també caldria deixar comentada l'entrada *x509_extensions* del *config.cnf* de configuració de *OpenSSL*).

  - Per crear el certificat CA propi:

    ```
    openssl req -new -x509 -keyout ./demoCA/private/CAkey.pem
    -out ./demoCA/private/CAcert.pem -days 360
    ```

  - Per crear les peticions d'un nou certificat:

    ```
    openssl req -new -keyout newkey.pem -out newreq.pem -days
    360
    ```

  - Per signar les peticions amb la nostra *CA*:

    ```
    openssl ca -policy policy_anything -out certserver.pem -in
    newreq.pem
    ```

  - Per signar les peticions amb la nostra *CA*:

    ```
    openssl ca -policy policy_anything -out certserver.pem -in
    newreq.pem
    ```

  - Finalment, per aconseguir el fitxer ***Server.p12*** una possible forma és crear el fitxer *convertServer.pem* i copiar a dins el contingut de *CAcert.pem*, *certserver.pem* i *newkeys.pem* (el certificat de la *CA emissora*, el certificat pròpiament i les claus), i executar:

    ```
    openssl pkcs12 -export -in convertServer.pem -out
    Server.p12 -name tomcat
    ```

- ***SergiSafelayer.p12***: Certificat i claus demanats al lloc http://demo.safelayer.com/, i que s'utilitzen per signar tant les dades de la petició per part del *proveïdor de serveis* com la resolució per part del *proveïdor d'identitat*. Té la particularitat d'acceptar les

actualitzacions del segell de temps del mateix proveïdor *SafeLayer*. També es troben en format PKCS#12.

- ***KS_ProvIdentitat.p12*:** Certificat i claus generades amb l'eina *OpenSSL* (seguint els mateixos passos que els descrits pel fitxer *Server.p12*, utilitzant també els certificats de la nostra *CA* per signar). S'utilitza per a que el *proveïdor d'identitat* signi la resolució de la petició, abans de retornar-la al *client*.

- ***/trustedCACerts*:** Directori on hi ha dipositats els certificats de les *autoritats certificadores* de confiança. En el cas del *proveïdor d'identitat* hi trobem el certificat arrel dels certificats *DNI-e*, així com els certificats de les *CA* subordinades (n'hi ha tres). Un certificat *DNI-e* es validarà contra aquests últims certificats de les *CA* subordinades (una d'elles serà el que l'haurà generat). També trobem el certificat arrel de la *CA* de *SafeLayer*, que es pot utilitzar per validar els certificats esmentats en el punt anterior. Però cal tenir en compte que en una de les últimes modificacions del producte s'ha passat a utilitzar el servei web *DSV* de verificació de *SafeLayer*, fet que deixa de fer necessari verificar el certificat, o l'estat de revocació, de forma local i, per tant, ja no caldria ni aquest certificat de la *CA* de SafeLayer ni tampoc el fitxer *.CRL* de la llista de revocació. En el cas del *proveïdor de serveis* aquest directori únicament conté el certificat de la nostra *CA* en format *DEM binari* (exportat dels certificats de confiança de l'explorador) per tal de poder verificar la signatura del *proveïdor d'identitat* en la resolució de la petició. No s'ha contemplat l'ús de *CRL* o algun altre mecanisme de validació de l'estat d'un certificat donat que es pressuposa una estreta confiança entre servidors. Però en cas que un hipotètic escenari real disposés de molts *proveïdors d'identitat* diferents, caldria implementar aquest tipus de mecanisme.

- ***kstore_trustedx.jks*:** Aquest magatzem de claus és el que s'utilitza per definir els certificats de confiança necessaris per crear la connexió *HTTPS* cap als serveis web de *SafeLayer*. En última instància (des del 23 de Desembre) aquest fitxer no seria necessari, donat que *SafeLayer* ja disposa de certificats emesos per *CA* de confiança *estandard* (concretament, signats per *Thawte Consulting*, de forma que tant qualsevol explorador com el *JRE* trobaran la connexió de confiança).

- ***Connexió SSL al nostre servei web*:** En el cas de la connexió a través d'*HTTPS* al nostre servei web per part de l'*applet java*, s'ha comprovat que una vegada s'ha definit el certificat arrel de la nostra *CA* com a certificat de confiança, l'*applet* no necessita definir cap *truststore* (magatzem de certificats de confiança) per tal d'efectuar la comunicació correctament).

- ***mystore***: Certificat i claus per signar l'*applet java*. Donat que no es disposa d'un certificat *real* per signar l'*applet*, s'ha optat per utilitzar una alternativa que ha facilitat enormement la implementació i les proves amb l'*applet*: signar-lo amb unes claus generades automàticament per l'eina *keytool* de *java*. Així doncs, s'han afegit les següents comandes en el *build.xml* del projecte corresponent a l'*applet java*:

```
<target name="Keytool" description="keytool-generate">
    <exec executable="C:\Archivos de programa\Java\jdk1.6.0_11\bin\keytool.exe">
            <arg value="-genkey" />
            <arg value="-alias" />
            <arg value="signFiles" />
            <arg value="-keystore" />
            <arg value="mystore" />
            <arg value="-keypass" />
            <arg value="mefisto" />
            <arg value="-noprompt" />
```

```
            <arg value="-dname" />
            <arg value="cn=Sun" />
            <arg value="-storepass" />
            <arg value="mefisto" />
        </exec>
    </target>

    <target name="Jarsigner" description="jarsigner-generate">
   <exec executable="C:\Archivos de programa\Java\jdk1.6.0_11\bin\jarsigner.exe">
            <arg value="-keystore" />
            <arg value="mystore" />
            <arg value="-storepass" />
            <arg value="mefisto" />
            <arg value="-keypass" />
            <arg value="mefisto" />
            <arg value="dist/ComponentClient.jar" />
            <arg value="signFiles" />
        </exec>
    </target>
```

Gràcies al primer, s'aconsegueixen crear les claus per signar, i gràcies al segon es signa l'*applet* cada cop que es genera el *.JAR* que el conté.

8.2. Dispositiu lector de DNI-e

Tant pel desenvolupament com per les proves efectuades en aquest producte s'ha utilitzat un lector de *DNI-e,* homologat i amb interfície *USB,* amb les següents especificacions:

Lector per signatura electrònica i aplicacions de Seguretat Informàtica

Marca: Chipnet (més informació a www.chipnet.es, o bé a www.dnielectronico.es)

Característiques i estàndards:

- Norma ISO 7816 1, 2 i 3
- Targetes amb protocol T=0 i T=1
- Microsoft WHQL – PC/SC i CCID
- CE – FCC – RoHS
- EMV 2000 Nivell 1
- USB 1.1/2.0 (full speed) mín. 9600 bps
- Alimentació per port USB
- PKCS#11
- CSP de Microsoft