UOC
Universitat Oberta de Catalunya
www.uoc.edu

Projecte Fi de Carrera

Signatura Digital

CARLOS VILA MATEOS

Firmado digitalmente por CARLOS VILA MATEOS
Nombre de reconocimiento (DN): c=ES, ou=Vegeu https://www.catcert.net/verIDCat (c)03, sn=VILA MATEOS, givenName=CARLOS, serialNumber=46358509P, cn=CARLOS VILA MATEOS
Fecha: 2010.12.30 15:33:03 +01'00'

Carlos Vila Mateos

Enginyeria en Informàtica

Josep Maria Camps Riba

30/12/2010

➔ Control de versions

Versió	Data	Objecte
1.0	03/09/10	1. Fase 0: Definició del Pla de Treball: + Descripció del PFC + Objectius generals i específics + Metodologies, entorns tecnològics i eines + Planificació amb fites i temporalització
1.1	17/09/2010	3. Fase I: Anàlisi Funcional
1.2	29/09/2010	4. Fase II: Anàlisi Orgànic: Dissenyar la capa d'integració entre una aplicació J2EE i els serveis de PKI.
1.3	25/10/2010	Revisió AF i AO.
1.4	11/11/2010	Tancament Fase I: Anàlisi Funcional i Fase II: Anàlisi Orgànic
2.0	12/11/2010	Inici documentació Fase III: Construcció
2.1	13/11/2010	Afegit AO: 4.4 Mòdul Frontal de Signatura
2.2	14/11/2010	Fase III: Construcció ProveidorSignatura
2.3	18/11/2010	Modificació dels Vo per un d'únic DigitalSignatureContentVo. Modificacions al AO
2.4	01/12/2010	Fase IV: Verificació de la construcció i lliurament. Creació de Test Cases ProveidorSignatura i empaquetament.
2.5	20/12/2010	Fase IV: Verificació de la construcció i lliurament.

Versió	Data	Objecte
2.6	27/12/2010	Tancament Fase III: Construcció i Fase IV: Verificació de la construcció i lliurament.
2.7	29/12/2010	Repàs general de la Memòria: reestructuració capítols, objectius.
3.0	30/12/2010	Lliurament Projecte i Memòria.

➔ Interlocutors

	Nom	email
Coordinador Àrea J2EE	Santi Caballé Llobet	scaballe@uoc.edu
Consultor/Tutor	Josep Maria Camps Riba	jcampsri@uoc.edu
Alumne	Carlos Vila Mateos	cvilama@uoc.edu

2 Dedicatòria i agraïments

A tots els que han patit les hores de tancament i dedicació al projecte, principalment la meva dona Cristina Morell i els meus pares pel suport que sempre m'han donat durant tots els estudis.

A proposit de l'actitud de les persones que m'han donat suport m'agradaria citar a Arturo Graf, *"La constancia és la virtud por la que todas las cosas dan su fruto."*

3 Resum

Aquest projecte es centra en com funciona la Signatura Digital, conèixer els seus conceptes fonamentals, aprofundir en el seu disseny i les variacions de funcionalitats que té.

L'enfoc del projecte i de la memòria es posicionarà des del punt de vista d'un Enginyer de Software que ha d'iniciar un projecte que doni serveis de Signatura Digital a un projecte més ampli d'Administració Electrònica (AE). Des d'aquest punt de vista anirem coneixent els conceptes bàsics de la Signatura Digital, les particularitats d'aquesta i quines funcionalitats i serveis ens poden proporcionar a una plataforma de tramitació electrònica dintre de l'àmbit de l'AE.

En aquest punt aprofundirem en l'ús de les primitives de Signatura Digital, els requeriments de Signatura Digital que necessita una plataforma d'AE, com cobrir aquests requeriments amb les eines de mercat existents, quins proveïdors de serveis d'aquest tipus hi ha per cobrir els requeriments del projecte en aquest àmbit, realitzar la selecció d'un proveïdor de serveis de Signatura i com podem dissenyar el nostre sistema per integrar aquests serveis en la nostra plataforma.

En el nostre cas realitzarem un mòdul que pugui subministrar autenticació, identificació i signatura digital a una plataforma d'AE. Aquestes plataformes han de ser capaces de modelar i treballar amb qualsevol tràmit o procés de l'administració de forma que es puguin executar de forma electrònica o on-line i que cap ciutadà o empresa hagi de presenciar-se a l'administració, donat que té el dret a realitzar-ho on-line d'igual manera que l'intercanvi d'informació i documentació entre administracions LAECAP (Llei d'Administració Electrònica).

Aquest projecte es centra doncs en aquest mòdul expert en la **Signatura Digital.**

4 Índex

2 Dedicatòria i agraïments 4

3 Resum 5

4 Índex 6

5 Cos de la memòria 8

5.1 Capítol 1, Introducció 8

5.1.1 Justificació del PFC i context en el qual es desenvolupa: punt de partida i aportació del PFC 10

5.1.2 Objectius del PFC 10

5.1.3 Enfocament i mètode seguit. 11

5.1.3.1 Metodologies, entorns tecnològics i eines. 13

5.1.4 Planificació del projecte. 14

5.1.5 Productes obtinguts. 16

5.1.6 Descripció dels altres capítols de la memòria. 19

5.2 Capítol 2: Fase I - Anàlisi Funcional 20

5.2.1 Conèixer les capacitats dels serveis PKI i necessitats del projecte. 20

5.2.1.1 Premisses Bàsiques de la Plataforma de Signatura: 20

5.2.1.2 Dispositiu de creació de signatura electrònica: 21

5.2.1.3 Certificats Tipus (X509) 21

5.2.1.4 Contenidors de Certificats i Signatures 26

5.2.1.5 Topologia de les signatures 27

5.2.1.6 Segell de temps 29

5.2.1.7 Signatures Avançades 30

5.2.1.8 Digital Signature Services protocol DSS 30

5.2.1.9 Plataforma PSIS de CatCert 31

5.2.2 Casos d'us 33

5.2.2.1 Us dels serveis de PSIS: 34

5.2.2.2 Frontal de Signatura: 35

5.2.2.3 Diagrama seqüència dels Casos d'us 37

5.2.2.4 Procés validació PSIS-CATCert 38

5.2.3 Identificar les necessitats funcionals del PortaSignatures per utilitzar els serveis de PKI. 39

5.3 Capítol 3: Fase II - Anàlisi Orgànic: Dissenyar la capa d'integració entre una aplicació J2EE i els serveis de PKI. 41

5.3.1 Aspectes i premisses tècniques de la Plataforma de Signatura 41

5.3.2 Definició de l'arquitectura general de la Plataforma de Signatura. Enterprise Appl. (J2EE). 45

5.3.3 El mòdul de Proveïdor de Signatura 46

5.3.3.1 Composició modular del Proveïdor de Signatura. 48

5.3.3.2 Composició Missatges DSS: 50

a Missatges DSS de validació (VerifyRequest): 52

b Missatges DSS de signatura (SignRequest): 53

c Missatges DSS de resposta (Response): 54

d Exemple: missatge DSS de validació de certificat x509 i normalització d'atributs contra PSIS. 56

5.3.3.3 Constants configuració: 58

5.3.3.4 Els VOs. 64

5.3.3.5 Mòduls EJB 66
5.3.4 Mòdul frontal de signatura 69
5.3.4.1 Parametrització i comportament JWS. 70
a Paràmetres principals (obligatoris) 71
b Paràmetres de xarxa 73
c Paràmetres d'entrada (document / llibreria / àlies-CN) 73
d Paràmetres de sortida 75
e Paràmetres signatura XML 76
f Paràmetres signatura CMS / PDF 76
5.4 Capítol 4: Fase III - Construcció 78
5.4.1 ProveidorSignaturaCore. 78
5.4.1.1 Estructura del projecte ProveidorSignaturaCore: 80
5.4.1.2 Llibreries necessàries: 81
5.4.1.3 Codi font, classes i mètodes: 82
5.4.2 ProveidorSignaturaCoreTest. 90
5.4.2.1 Estructura del projecte ProveidorSignaturaCoreTest: 90
5.4.2.2 Llibreries necessàries: 91
5.4.2.3 Codi font, classes i mètodes: 91
5.5 Capítol 5: Fase IV - Verificació de la construcció i lliurament 92
5.5.1 Programari, llibreries i instal·lació 92
5.5.1.1 Productes obtinguts 92
5.5.1.2 IDE i eines de desenvolupament 96
5.5.2 Verificació de la construcció 97
5.5.3 Test de Validació de certificats digitals X509 98
5.5.4 Test de Validació de Signatures CMS 107
5.5.5 Test de Validació de Signatures en PDF 116
5.5.6 Test de Signatura CMS i Segell de Temps 122
5.6 Capítol 6: Conclusions i extensions de futur 128
6 Glossari: 129
7 Bibliografia i referències 132

5 Cos de la memòria

5.1 Capítol 1, Introducció

El món a l'actualitat està immers en nous canvis tecnològics que avancen molt ràpidament; la forma de comunicar-se de la societat, de treballar, de relacionar-se, de pensar, d'educar-se, de l'oci, tot això està canviant a formes més àgils, més immediates, més complertes, amb més mitjans i variants gràcies a les noves tecnologies.

Les noves tecnologies o TIC estan impactant en la societat de manera que canvien les formes tradicionals del funcionament de les organitzacions públiques i privades, l'economia on apareixen els nous mercats i negocis (pure-players o clicks-and-bricks), l'impacte en la política, comerç, marketing, impacte cultural i educatiu, a aquesta època la podem anomenar l'Era Digital o l'Era de la Informació.

En l'Era Digital la societat es comunica i es mou amb uns nous paradigmes de comportament i uns efectes que encara estan en estudi. S'ha trencat amb l'època anterior, l'Era Industrial, hi ha però encara resistències als nous canvis però amb adaptacions progressives. En molts casos a la societat, educació, família, treball, polítics, es mantenen encara els costums i els canals tradicionals que van evolucionant cap a la societat digital, s'ha de dir però, que les bases fonamentals del comportament humà no han variat gaire amb l'evolució humana durant els segles, però sí els canals i les tecnologies que estan al nostre abast i que permeten potenciar i accelerar allò que la humanitat pot realitzar amb un món global i amb informació immediata.

En aquest punt és on les noves tecnologies incideixen tant en la societat i per tant van construint una base de comunicacions i una xarxa d'eines capaços de donar suport a aquestes noves necessitats creades i formes de comportar-se i comunicar-se.

Per això aquells processos més tradicionals que abans requerien molt de temps per realitzar-los i, que es perdia temps en el transport o en els passos físics del procés, ara mitjançant les noves tecnologies ens permeten reduir els temps als mínims possibles, sense desplaçaments físics, i permeten que ens centrem en poder realitzar l'objectiu directe de la nostra fita. Podem aprofitar millor el temps en altres coses de la nova Era Digital que ens demana més comunicació, més informació instantània i més capacitat d'estar connectats en tot el món globalitzat, sense fronteres per qualsevol persona d'aquest món.

En aquest sentit per exemple podem veure que les noves aplicacions corporatives et permeten treballar a distancia, les webs 2.0 et permeten navegar amb un món interactiu integrat amb un munt de serveis, les noves formes de relacions personals i laborals, les xarxes socials comunicar-te al instant amb una xarxa de contactes de tot el món privades i públiques amb informació abundant de tothom a l'instant. Les TIC estan impulsant aquest canvi i s'han de posar més eines que segueixin permetent aquesta evolució.

Evoluciona la Societat Digital i les TIC és l'eina de suport? o be són les TIC que forcen aquesta evolució de la societat?

Deixant a banda moltes qüestions filosòfiques que poden sortir d'aquests raonaments, ens centrarem amb una de les eines tecnològiques fonamentals necessàries per a que aquesta Societat Digital pugi ser-ho, la Signatura Digital.

En l'entorn que hem d'escrit de la societat s'havia de cobrir un àmbit essencial per a que la gent i les empreses poguessin realitzar els processos habituals bàsics de la seva rutina diària: la signatura d'un contracte laboral (empresa-treballador) o qualsevol altre tipus de contractes que es realitzen, la creació d'avals, els processos de facturació, els metges i els seus processos, les receptes, els accessos als bancs de forma segura, les interaccions entre AAPP (interoperabilitat), les interaccions entre AAPP i ciutadans, etc. Qualsevol procés que requerís la identificació de la persona física o jurídica, qualsevol procés que requerís documentació física (papers), tots aquests processos que es poden englobar perquè requereixen aquesta característica d'identificació i documentació fins ara es tenien que realitzar de forma presencial i amb el trasllat de documentació física cap a tots llocs. S'ha hagut de buscar una eina que permetés portar aquests processos a l'Era Digital, la digitalització i la signatura digital.

La Signatura Digital i les seves variants ens ha permès convertir aquest processos en digitals i reconeguts amb la mateixa validesa legal que els processos tradicionals.

Actualment ja no cal presenciar-se a cap lloc per realitzar aquests processos, ja no cal traslladar la documentació física, han sorgit i seguiran sorgint aquests processos en l'àmbit digital com: la recepta digital, la factura digital, la identificació digital (DNIe), els càrrecs digitals, els notaris digitals, els contractes digitals, evidències digitals, etc.

Quin gran pas és aquest per l'evolució de les tècniques d'interacció humana i de les empreses, quin estalvi de paper (i per tant del medi ambient), quin estalvi econòmic i de millora dels processos empresarials i dels mercats, quina acceleració dels processos tradicionals i de la productivitat. Quan arribem a una madureça de l'Era Digital aquest processos estaran totalment integrats en la societat, seran molt àgils, eficients i automatitzats de manera que una persona no haurà de perdre els temps morts de la gestió d'aquestes activitats i les seves dependencies i podrà actuar des de qualsevol moment i lloc del món.

La Signatura Digital ha estat des del seu naixement un tema complex i amb una lenta implantació a la societat, a costat temps que la Informàtica li dones una bona solució. Altres famílies de software com les BBDD o els SO han estat més presents en la consciència general de la societat. Encara i això avui en dia podem començar a veure l'inici de la inmersió social en l'Era Digital i l'ús de la Signatura Digital associada als processos habituals en la societat.

Per a donar solució a aquestes necessitats de la Signatura Digital es va crear la infrastructura de clau pública o PKI, que veurem en detall en els següents capítols.

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

En aquest projecte podem veure com funciona la Signatura, conèixer els seus conceptes fonamentals, aprofundir en el seu disseny i les variacions de funcionalitats que té.

Aquest projecte s'envolta en l'entorn de les AAPP i com la signatura digital pot ajudar a convertir el procésos de les AAPP en un servei pel ciutadà digital. Convertint els processos tradicionals de les AAPP en digital, és a dir l'Administració Electrònica.

En aquest àmbit els legisladors s'han posat mans a l'obra per donar un impuls a la conversió i modernització de les AAPP cap a l'Era Digital i optimització de processos. Per això s'han creat les lleis d'Administració Electrònica.

En el nostre cas realitzarem un mòdul PKI que pugui subministrar autenticació, identificació i signatura digital a una plataforma d'AE. Aquestes plataformes han de ser capaces de modelar i treballar amb qualsevol tràmit o procés de l'administració de forma que es puguin executar de forma electrònica o on-line i que cap ciutadà o empresa hagi de presenciar-se a l'administració donat que té el dret a realitzar-ho on-line al igual que l'intercanvi d'informació i documentació entre administracions LAECAP (Llei d'Administració Electrònica).

Aquest mòdul coneixerà la complexitat de la Signatura Digital i tots els seus elements o serveis de PKI (Public Key Infrastructure), serà l'expert del negoci dels serveis de PKI i servirà aquests serveis al mòdul PortaSignatures o a d'altres de forma transparent.

Aquest projecte es centra doncs en aquest mòdul expert en la **Signatura Digital.**

Opcionalment, com a línia de futur es crearà una aplicació web per recollir tots els documents de la plataforma d'AE pendents de signar digitalment per una persona integrat amb un frontal de signatura electrònica.

5.1.2 Objectius del PFC

En l'àmbit d'un projecte global d'AE (Administració Electrònica), aquest projecte té com objectiu principal **conèixer les possibilitats de la signatura digital i les primitives PKI**, **els proveïdors** de serveis d'aquest tipus per cobrir els requeriments del projecte en aquest àmbit sense haver de desenvolupar la complexitat de la signatura digital com a tal que no és part del nostre negoci. D'aquesta forma la signatura digital serà un procés transparent pels usuaris de la nostra plataforma i nosaltres ho integrarem i adaptarem per a que doni resposta als requeriments del nostre programari.

L'altre objectiu principal és el de realitzar una **integració amb les funcionalitats de signatura digital i proveir dels serveis de Signatura** de forma que per la resta de programadors de la plataforma i usuaris sigui transparent i no hagin de ficar-se ni conèixer la complexitat de la signatura digital, donant així una capa de serveis de més alt nivell, que per sota es connectarà amb diferents, proveïdors, llibreries i serveis experts en la signatura digital.

En el proper apartat veurem els **objectius agrupats per fases, objectius de primer nivell i objectius de segon nivell** segons l'estratègia de gestió de projecte per assolir els objectius descrits.

5.1.3 Enfocament i mètode seguit.

Els objectius del projecte de Signatura Digital és conèixer les funcionalitats i capacitats dels actuals sistemes que implementen serveis de Signatura Digital, les eines i mecanismes d'integració d'aquestes i analitzar i dissenyar un projecte que utilitzi i integri aquestes infrastructures en un cas real, un cas d'us d'un projecte real que requereix d'aquestes funcionalitats.

Per assolir aquests objectius s'ha preparat un desglossament en Work Breakdown Structure (WBS) a tres nivells per facilitar l'organització del projecte i el seguiment en blocs d'aquest. Aquesta estructura esta basada en l'extensió dels objectius de primer nivell en tasques realitzables per aconseguir aquests objectius.

Per això els objectius s'han desglossat en tres nivells:

Nivell de Fase: Identifiquem 4 grans fases en el projecte que segueixen el sistema més clàssic de cicle de desenvolupament d'un projecte informàtic: Anàlisi Funcional (AF), Anàlisi Orgànic (AO), Construcció (CO), Proves i validació (PV)

Nivell macro-activitat o objectius de primer nivell: Els objectius de primer nivell són els objectius principals a assolir, essent els objectius claus a complir.

Nivell tasques o objectius de segon nivell: Els objectius clau, es descomponen en objectius de segon nivell o tasques amb la resolució de les quals s'han de veure complits els objectius de primer nivell (claus).

Fase I: Anàlisi Funcional

1. Conèixer les capacitats dels serveis PKI.
   - Serveis PKI, tipus de signatures CMS vs XML.
   - Certificats de clau pública X509
   - Topologia de les signatures.
   - Segell de temps.
   - Signatures Avançades.
   - Plataforma PSIS de CatCert.
   - Digital Signature Services protocol DSS.
2. Identificar les necessitats funcionals del PortaSignatures per utilitzar els serveis de PKI.
   - Bústia d'ordres de signatura: Disposarà d'un frontal on l'usuari podrà consultar els documents pendents de signatura.
   - Històric de l'usuari: Opció de consulta de l'històric dels documents signats.

- Realitzar signatura de documents PDF (CMS).
- Rebuig de l'ordre de signatura.
- Signatura per lots: signar fins a 10 documents o ordres de signatura d'un sol cop, tan si els ha de signar un sol usuari o, be, per múltiples usuaris, independentment de l'ordre de les signatures. També es podrà realitzar aquesta signatura, en el cas de la que hi hagi més d'una persona, de forma niuada. No es podrà saltar l'ordre de signatura establert pel document a signar.
- Suport a la signatura múltiple en paral·lel. El document es podrà signar per diverses persones, sense la necessitat de que s'hagi de seguir un ordre específic.

Fase II: Anàlisi Orgànic

3. Dissenyar la capa d'integració entre una aplicació J2EE i els serveis de PKI.
   - Definició de l'arquitectura general de la Plataforma de Signatura i PortaSignatures. Enterprise Appl. (J2EE).
   - Disseny orgànic d'integració PortaSignatures amb serveis de PKI, mitjançant la plataforma PSIS i a través del protocol DSS amb les extensions especifiques de CatCert. Realitzar aquesta integració a nivell de lògica de negoci (com a serveis) i a nivell de presentació com a frontal web de signatura.

Fase III: Construcció

4. Construir la capa d'integració entre una aplicació J2EE i els serveis de PKI (PSIS)
   - Validació de certificats X509 amb implementació DSS: Webservices.
   - Construcció de capa d'integració de serveis web de PSIS.
5. Construir la integració entre una aplicació web i un frontal de signatura.
   - Modificació de l'applet de signatura de CatCert per ajustar-ho als requeriments del client i de l'aplicació de PortaSignatures.

Fase IV: Verificació de la construcció i lliurament.

6. Verificar la construcció de les primitives i integració PKI mitjançant els casos d'us.
   - Implementar casos de prova (test cases) que permetin la validació de la construcció.

Per assolir aquesta estructura d'objectius i tasques s'anirà treballant i estenent (delimitant o ampliant l'abast) en els casos necessaris durant el transcurs del projecte. Principalment s'haurà de detectar aquests canvis en la Fase inicial d'AF, però també es podran donar en la d'AO i inclos en la fase de construcció però en menys mesura o de menys calat.

5.1.3.1 Metodologies, entorns tecnològics i eines.

Metodologies:

- El Llenguatge Unificat de Modelat (UML).
- Anàlisi i Disseny OO amb UML.
- Desenvolupament Avançat de Components de Negoci amb Tecnologia EJB.
- Desenvolupament d'Aplicacions J2EE. Desenvolupament d'arquitectures per a aplicacions empresarials.
- Anàlisi i disseny d'arquitectures: filosofies SOA, Patrons: MVC, Facade, Singleton, EJB.

Entorns tecnològics:

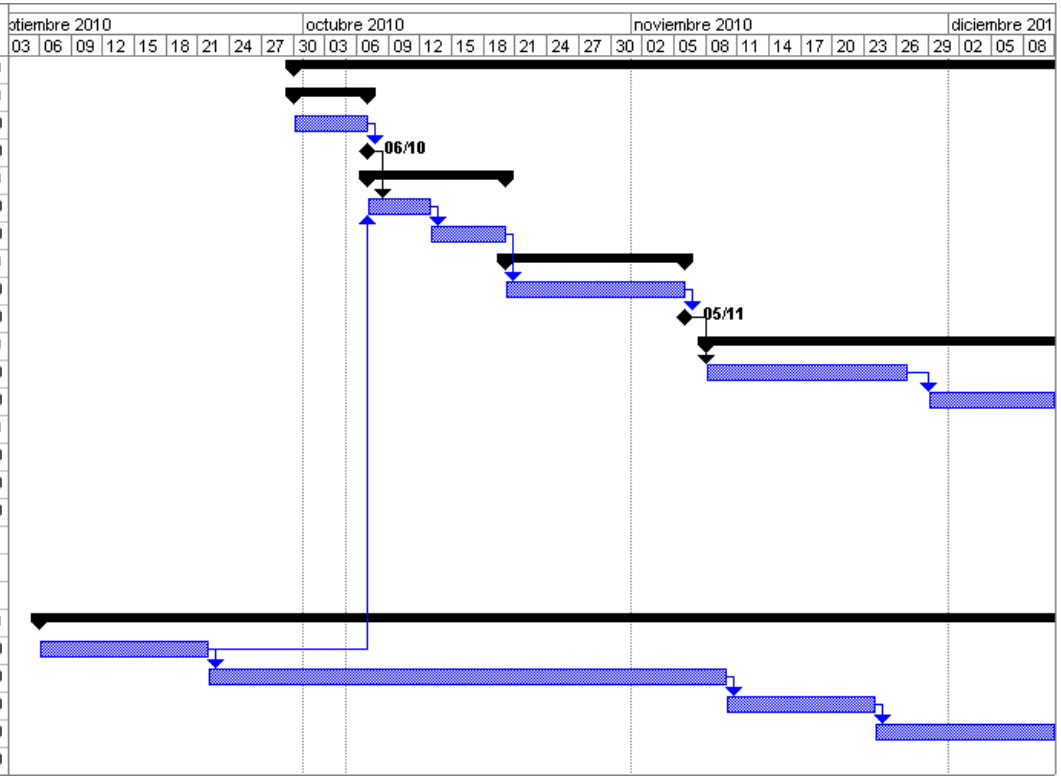
- J2SE, J2EE (java server), Arquitectura J2EE.
- Desenvolupament de components web: servlets, JavaServer Pages
- Desenvolupament de components distribuïts: EJBs
- Disseny i implementació arquitectures MVC, EJBs, Struts
- APIS de J2EE: JMS, JAXP, JSP, JDBC, EJB, JTA.
- Signatura Digital: XMLDsig, CMS, PDF-CMS, DSS, PSIS, X509

Eines:

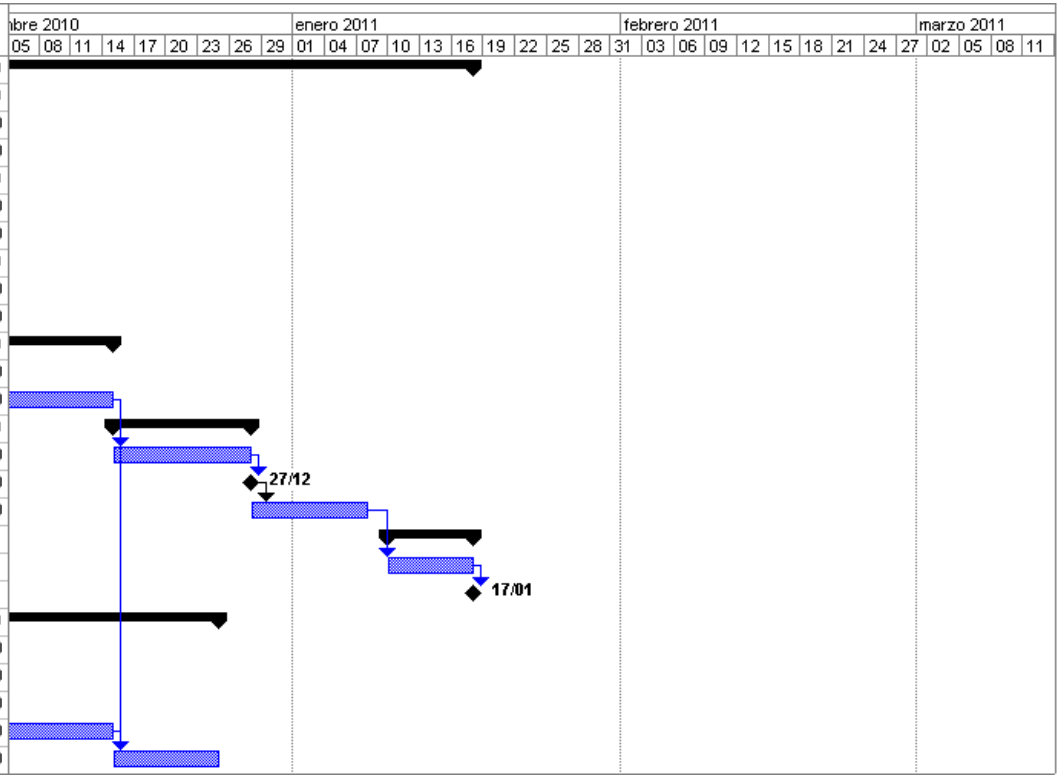
- Microsoft Office Project 2003
- Microsoft Office Visio 2003.
- Eclipse Helios
- Ant

5.1.4 Planificació del projecte.

Id	Nombre de tarea	Duración	Comienzo
1	**Signatura Digital**	**78 días**	**jue 30/09/10**
2	**Fase 0: Pla de projecte**	**5 días**	**jue 30/09/10**
3	Adquisició i preparació del pla de projecte	5 días	jue 30/09/10
4	PAC1: Lliurament Pla de Projecte	0 días	mié 06/10/10
5	**Fase I: Anàlisis Funcional**	**9 días**	**jue 07/10/10**
6	Identificar les necesitats funcionals del PortaSignatures per utilitzar els servei	4 días	jue 07/10/10
7	Conèixer les capacitats dels serveis PKI	5 días	mié 13/10/10
8	**Fase II: Anàlisis Orgànic**	**13 días**	**mié 20/10/10**
9	Dissenyar la capa d'integració entre una aplicació J2EE i els serveis de PKI	13 días	mié 20/10/10
10	PAC2: Lliurament AF i AO	0 días	vie 05/11/10
11	**Fase III: Construcció**	**27 días**	**lun 08/11/10**
12	Construir la capa d'integració entre una aplicació J2EE i els serveis de PKI (PS	15 días	lun 08/11/10
13	Construir la integració entre una aplicació web i un frontal de signatura	12 días	lun 29/11/10
14	**Fase IV: Verificació de la construcció i lliurament**	**9 días**	**mié 15/12/10**
15	Verificar la construcció de les primitives i integració PKI mitjançant els casos d	9 días	mié 15/12/10
16	PAC3: Lliurament Programari	0 días	lun 27/12/10
17	Folgança	9 días	mar 28/12/10
18	**Fase V: Lliurament final**	**6 días**	**lun 10/01/11**
19	Preparació de la presentació	6 días	lun 10/01/11
20	Lliurament final	0 días	lun 17/01/11
21	**Porta Signatures**	**80 días**	**lun 06/09/10**
22	Fase AF i AO	12 días	lun 06/09/10
23	Desenvolupament	35 días	mié 22/09/10
24	Proves	10 días	mié 10/11/10
25	Folgança	15 días	mié 24/11/10
26	Ajustos d'integració Signatura Digital	8 días	mié 15/12/10



Id	Nombre de tarea	Duración	Comienzo
1	**Signatura Digital**	**78 días**	**jue 30/09/10**
2	**Fase 0: Pla de projecte**	**5 días**	**jue 30/09/10**
3	Adquisició i preparació del pla de projecte	5 días	jue 30/09/10
4	PAC1: Lliurament Pla de Projecte	0 días	mié 06/10/10
5	**Fase I: Anàlisis Funcional**	**9 días**	**jue 07/10/10**
6	Identificar les necesitats funcionals del PortaSignatures per utilitzar els servei	4 días	jue 07/10/10
7	Conèixer les capacitats dels serveis PKI	5 días	mié 13/10/10
8	**Fase II: Anàlisis Orgànic**	**13 días**	**mié 20/10/10**
9	Dissenyar la capa d'integració entre una aplicació J2EE i els serveis de PKI	13 días	mié 20/10/10
10	PAC2: Lliurament AF i AO	0 días	vie 05/11/10
11	**Fase III: Construcció**	**27 días**	**lun 08/11/10**
12	Construir la capa d'integració entre una aplicació J2EE i els serveis de PKI (PS	15 días	lun 08/11/10
13	Construir la integració entre una aplicació web i un frontal de signatura	12 días	lun 29/11/10
14	**Fase IV: Verificació de la construcció i lliurament**	**9 días**	**mié 15/12/10**
15	Verificar la construcció de les primitives i integració PKI mitjançant els casos c	9 días	mié 15/12/10
16	PAC3: Lliurament Programari	0 días	lun 27/12/10
17	Folgança	9 días	mar 28/12/10
18	**Fase V: Lliurament final**	**6 días**	**lun 10/01/11**
19	Preparació de la presentació	6 días	lun 10/01/11
20	Lliurament final	0 días	lun 17/01/11
21	**Porta Signatures**	**80 días**	**lun 06/09/10**
22	Fase AF i AO	12 días	lun 06/09/10
23	Desenvolupament	35 días	mié 22/09/10
24	Proves	10 días	mié 10/11/10
25	Folgança	15 días	mié 24/11/10
26	Ajustos d'integració Signatura Digital	8 días	mié 15/12/10



La planificació del projecte de Signatura Digital segueix les Fases i l'estructura WBS definida anteriorment.

5.1.5 Productes obtinguts.

1) ProveïdorSignaturaCore.jar: És una llibreria que ens permetrà a partir d'una API simplificada utilitzar serveis de Signatura Digital. Les Constants de Configuració i parametritzacions bàsiques estan predefinides de forma que les funcionalitats bàsiques tenen un fàcil accés i us. Aquesta llibreria també permet funcionalitats més complexes de Signatura afegint les parametritzacions avançades. Aquesta llibreria s'anirà ampliant amb noves funcionalitats.

A) Paquet complet amb llibreries incloses: Per utilitzar aquesta llibreria s'ha de desplegar al servidor d'aplicacions el ProveïdorSignaturaCore.jar que conté totes les llibreries necessàries.

Per fer això seguirem els següents pasos:

- Per obtenir aquesta utilitat hem de llançar l'script ant que està dintre del paquet ProveidorSignaturaCore\proveidor_build.xml
- Aquest ens empaquetarà el projecte i el deixarà preparat per executar a \ProveidorSignaturaCore\inst\ ProveidorSignaturaCore.jar

B) Paquet complet amb llibreries excloses: Si les llibreries incloses poden donar problemes de versions o de classloaders amb d'altres llibreries ja incloses en un projecte global o servidor d'aplicacions, s'hauria de desplegar la llibreria ProveïdorSignaturaCore.jar amb les llibreries necessàries no incloses per a que es puguin incloure al classpath només les necessàries que no col·lisionessin en les del projecte en curs.

Per fer això seguirem els següents pasos:

- Llançar l'script ant que està dintre del paquet ProveidorSignaturaCore\proveidor_libs_build.xml
- Aquest ens empaquetarà el projecte amb totes les llibreries necessaries pel funcionanment incloses en el paquet i amb els classpath necesaris configurats i el deixarà preparat per executar a \ProveidorSignaturaCore\inst\ ProveidorSignaturaCore.jar

Les llibreries necessàries són:

- JDK v1.5 o superior
- psis-beans.jar
- XMLBeans 2.1.0
- XFire 1.1.2
- Commóns-httpclient-3.0
- Commóns-loggin-1.0.4
- Wsdl4j-1.5.1
- Wss4j-1.5.0
- Jdom-1.0

2) ProveidorSignaturaCoreTest: Paquet que ens permet provar les classes de signatura a través de la classe que exten TestCase SignaturaTest. Aquesta classe es centrarà en la lògica de negoci pròpia per realitzar les proves unitàries. Tindrem diferents mètodes per a provar cadascun de les primitives PKI implementades.

Hem organitzat aquest paquet ProveidorSignaturaCoreTest com a un paquet independent en comptes d'afegir la classe a ProveidorSignaturaCore perquè així podem realitzar la prova definitiva des de fora del paquet core i crear el ProveidorSignaturaCore.jar, provar d'incloure'l en un altre paquet i veure que es poden realitzar totes les crides de forma correcte i amb les llibreries necessàries.

Les llibreries necessàries d'aquest paquet són:

- JDK v1.5 o superior:
- XMLBeans 2.1.0
- Commóns-httpclient-3.0
- commóns-codec-1.3
- Commóns-loggin-1.0.4
- JUnit 3

3) ProveïdorSignatura.ear: El component de ProveidorSignatura estarà format per subcomponents o paquets que formaran una gran aplicació (enterprise o EAR) i que implementaran la lògica de negoci necessària per respondre a les necessitats de l'actor (appl. externa) segons els seus casos d'us. S'encarregarà de resoldre els casos d'us de l'actor appl. externa: validar certificat, normalització d'atributs, validar signatura, segell de temps, consultar documents signats i validar-los en un moment donat de temps. Aquest component donarà una interfície per accedir a les seves funcionalitats IProveidor. A aquesta interfície només podran accedir les aplicacions amb els permisos necessaris. Com hem vist en l'apartat anterior lliuràvem un .jar, una llibreria d'utilitats que hom podrà utilitzar al afegir al seu projecte. Si es vol incloure les mateixes funcionalitats amb un component distribuït o EJB en una plataforma escalable donem la possibilitat de fer-ho convertint la llibreria en aquesta tecnologia.

Els paquets que formaran l'aplicació (EAR):

- **ProveidorSignaturaCore**: Aquest paquet és la llibreria que conté tota la lògica de negoci implementada, és qui realitza les crides a les funcionalitats de PSIS a través de la implementació DSS mitjançant XML i comunicació a través de webservices i SOAP. Aquest component conté les classes de configuració, els objectes Vo a utilitzar i els mètodes principals així com les llibreries i classes d'ajuda (utilities).

  Aquest component es pot utilitzar com a llibreria independent .jar. Com per exemple afegir la llibreria en un altre projecte i utilitzar les seves funcions públiques.

  Per exemple, en el cas de la plataforma de tramitació amb que està relacionat aquest projecte de signatura, aquest component com a llibreria ProveidorSignaturaCore.jar s'utilitzarà en el mòdul de SSO. Servirà per realitzar amb l'Open SSO una autentificació amb certificats digitals de l'usuari en la plataforma de tramitació. De manera que l'usuari

es connecta amb certificats digitals s'agafa les seves credencials i s'envien a través d'aquesta llibreria a validar el certificat i normalitzar l'atribut del NIF, la classe "Validacio" tornarà si és vàlid el certificat i quin és el NIF normalitzat del l'usuari en qüestió, d'aquesta forma se li donarà accés a l'usuari amb les credencials que aquest hagi de tenir en la plataforma de contractació. Aquest usuari quan arribi al Porta Signatures, de forma automàtica, només se li permetrà signar amb el mateix certificat de l'usuari autenticat en la plataforma (via les credencials recollides en el SSO).

- **ProveidorSignaturaEJB**: Aquest component s'implementarà amb un EJBs de sessió ProveidorSignaturaEJB seguint per tant el patró facade (session-facade: aconseguim desacoblar la capa del model de la capa de negoci cadascú fa la seva feina/responsabilitat i reduïm notablement el trànsit de xarxa). Aquest component permetrà el desplegament del component entrerprise ProveidorSignatura en un servidor d'aplicacions de forma que les seves funcionalitats públiques estiguin disponibles per realitzar peticions des de qualsevol punt de la plataforma de tramitació de forma distribuïda.
- **ProveidorSignaturaEJBClient**: Interfície del component per a lliurar als programadors o programari que vulgui utilitzar el nostre component de ProveidorSignatura.
- **ProveidorSignaturaEAR**: Aquest paquet ens permetrà empaquetar tots els paquets en un de sol com a únic recurs distribuïble i usable.

Amb aquesta Memoria s'adjunten els següents fitxers:

LLIBRERIA PRINCIPAL:

- ProveidorSignaturaCore.zip -> Codi font de la lliberia principal del projecte.
- ProveidorSignaturaCore.jar -> Executable de la lliberia principal del projecte.

PAQUET DE TEST:

- ProveidorSignaturaCoreTest.zip -> Codi font del paquet de test principal del projecte. Inclou una referència a la llibreria ProveidorSignaturaCore_libs.jar. -> Executable de la lliberia principal del projecte amb totes les llibreries necessàries incloses.

Carregar únicament aquest paquet en el Eclipse Helios i executar els JUnits per validar tota la construcció.

PAQUETS SECUNDARIS, LLIBRERIA EN EJB:
Els següents paquets són la infrastructura necessària per a poder cridar la llibreria anterior des de una interfície EJB RMI.

ProveidorSignaturaEAR.zip que conté:

- ProveidorSignaturaEAR -> Codi font del paquet ear.
- ProveidorSignaturaEJB -> Codi font del paquet ejb.
- ProveidorSignaturaEJBClient -> Codi font del paquet ejb de la part client.
- ProveidorSignaturaEAR.ear -> Aplicació executable de desplegament en un servidor d'aplicacions distribuïda amb interfícies EJB RMI.

5.1.6 Descripció dels altres capítols de la memòria.

Els altres capítols de la memòria s'organitzen segons la definició que ja hem realitzat en l'apartat de 5.1.3 "Enfocament i mètode seguit" en resum:

Capítol 2: Fase I - Anàlisi Funcional

Capítol 3: Fase II - Anàlisi Orgànic

Capítol 4: Fase III - Construcció

Capítol 5: Fase IV - Verificació de la construcció i lliurament

Capítol 6: Conclusions i extensions de futur

5.2 Capítol 2: Fase I - Anàlisi Funcional

5.2.1 Conèixer les capacitats dels serveis PKI i necessitats del projecte.

5.2.1.1 Premisses Bàsiques de la Plataforma de Signatura:

Per realitzar aquest projecte s'ha triat la Signatura Electrònica Avançada, que es basa en l'ús d'algorismes de xifrat asimètrics (es diuen així perquè són sistemes amb dues claus diferents amb unes propietats molt concretes). La clau privada de signatura s'anomena "dada de signatura electrònica" i la clau pública de signatura s'anomena "dada de verificació de signatura electrònica". Aquesta última ve garantida o certificada per totes les entitats certificadores del certificat. Els que signen ho fan amb la clau privada, mentre els tercers que reben documents signats els verifiquen amb la clau pública.

El sistema haurà de tenir un dispositiu de verificació de signatura electrònica, el dispositiu estarà implementat en el component de Signatura Digital i ha de garantir que qui ha signat és qui diu ser. El procés de verificació haurà d'obtenir la validesa a partir de la clau publica de la signatura, i per estar segurs que el certificat és vàlid, haurà de verificar la ruta de certificació fins una arrel fiable i validar totes les entitats certificadores de la ruta (CA). A més, el component de Signatura Digital comprovarà on-line les llistes de revocació vigents (CRL) contra els serveis de CATCert.

En una plataforma tecnològica com aquesta s'haurà d'utilitzar elements per garantir:

- Integritat documental electrònica: Garanteix que el document no ha estat modificat des de la seva creació i signatura (algorismes de resum i algorismes de signatura electrònica i segell de temps certificat per una tercera entitat).
- Autenticació: Garanteix que la persona que ha signat el document és qui diu ser (algorismes de signatura electrònica asimètrics).
- Si s'afegeix un segell de temps d'una tercera part de confiança com CATCert (TSA), es garanteix a més que aquell document signat per una persona concreta ho va fer a una data donada i no ha estat modificat des d'aleshores.

El sistema de Signatura Digital garantirà que qui ha signat és qui diu ser i que el document no ha estat modificat des que aquesta persona ho va signar (autenticitat, integritat i veracitat). La garantia de que qui ha signat és qui ho ha de fer o té potestats per fer-ho en un moment donat és responsabilitat del component de PortaSignatures.

La Plataforma de Signatura es desenvoluparà segons les normatives i pautes marcades per CATCert per tal de complir amb la legislació vigent en l'entorn d'aplicacions de Signatura Digital. Per tal d'acomplir aquestes normatives es realitzaran estudis, procediments i seguiments del desenvolupament d'aquesta plataforma adequat a les necessitats dels processos i tràmits del client. Aquesta forma de treballar conjunta garanteix un producte de qualitat i que compleix les normatives de les aplicacions segures de creació de signatura electrònica per a complir la llei vigent.

La Plataforma de Signatura garantirà la integritat de les dades sensibles i es comunicarà amb interfícies segures i utilitzant sempre un canal segur.

5.2.1.2 Dispositiu de creació de signatura electrònica:

El dispositiu de creació de signatura electrònica és aquell on es guarda la clau privada o dada de signatura. Es poden trobar tres maneres principals d'emmagatzemar aquestes claus de forma segura:

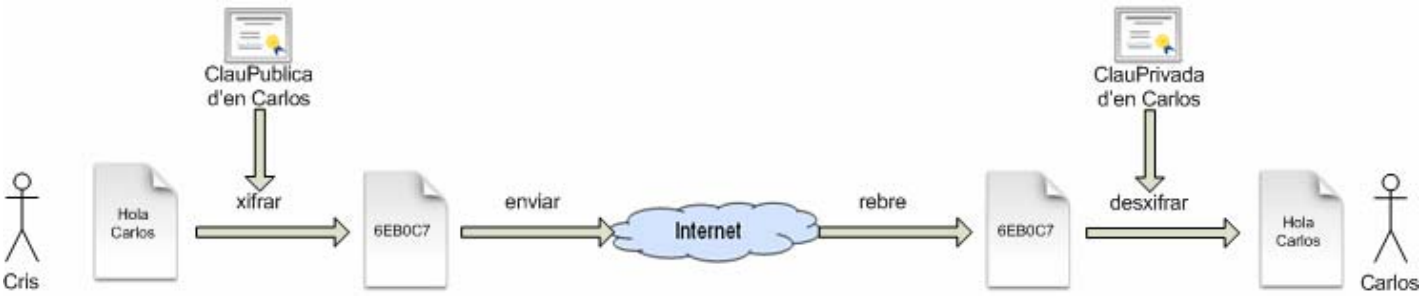
- Per software (fitxer PFX, PKCS#12): es guarda directament al sistema del client el certificat. Aquest sistema és molt sensible a intromissions ja que un virus, programa maligne o un accés no desitjat al sistema del client es pot obtenir el certificat fàcilment.
- Per hardware, token (usbkey o targeta criptogràfica):
    - Utilitzant el middleware (llibreria PKCS#11).
    - És molt segur però és un sistema car de distribució.
    - No és compatible amb tots els SO's i requereix de drivers propis per a cadascun.
    - Es pot perdre. En el cas de pèrdua s'hauria de demanar un altre i es tindria més cost de distribució. A més si es porten més certificats, es perden tots. En aquest cas, una altre solució podria ser una Autoritat de Signatura Centralitzada (ACS), que té un repositori de tots els teus certificats identitaris i d'atribut, i accedeix a aquest amb qualsevol sistema d'autentificació (pot ser amb un dels certificats).
    - Utilitzant el magatzem de certificats del compte personal d'usuari de Windows. Windows guarda referències a la targeta criptogràfica com si tingués els certificats privats.

En el projecte acceptarem tots aquests tipus de dispositius que poden garantir que la signatura és del propietari d'aquesta i no ha estat suplantat. Aquests dispositius han de garantir que, cap altre persona pot obtenir l'activació de la signatura. El dispositiu de creació de signatura electrònica és la "targeta electrònica". Aquest dispositiu, permet l'activació de la signatura (mitjançant PIN o altre codi). També es permetrà utilitzar certificats emmagatzemats als keystores dels navegadors. El procés de signatura, haurà de mostrar els certificats de tots els possibles keystores ja sigui als navegadors com a la targeta electrònica.

5.2.1.3 Certificats Tipus (X509)

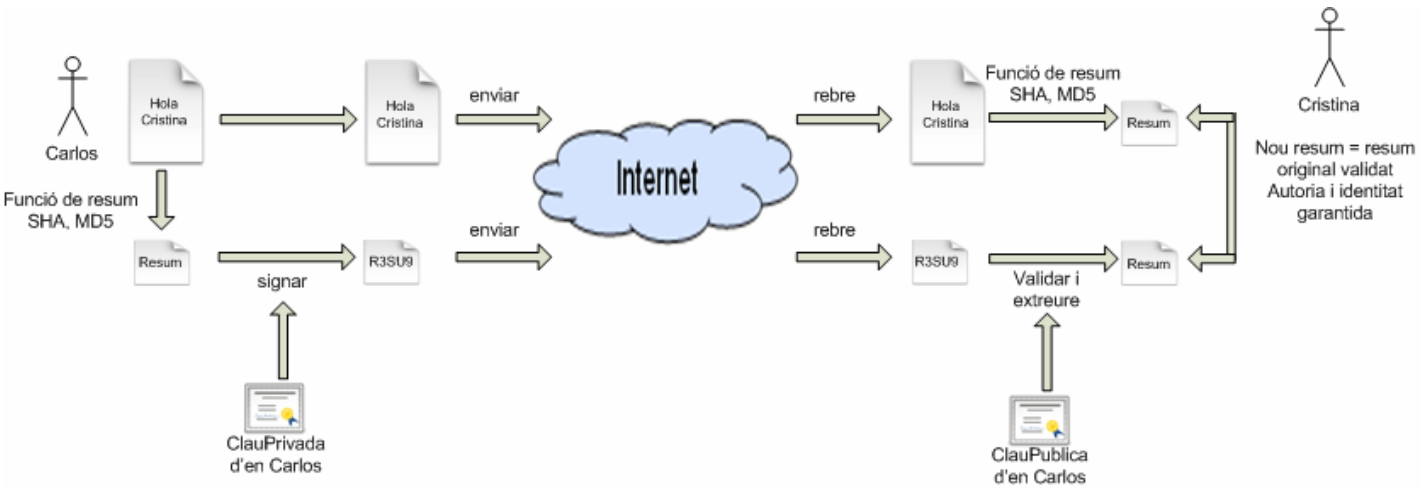
Els certificats digitals basats en els algoritmes de clau pública o asimètrica s'han demostrat que són pel moment els més efectius en aconseguir la protecció de les dades i les comunicacions. Sobre tot en aquest món de les comunicacions per Internet aquests sistema ens permet garantir l'autenticitat i no repudi de la identitat "digital". No entrarem a debatre aquests punts perquè no son objecte del projecte però si explicarem en que consisteix aquests sistema i que ens proporciona pels objectius del projecte.

Garantia de secret de les dades i comunicacions: El sistema de clau pública ens permeten xifrar amb la clau pública del receptor i desxifrar amb la clau privada també del receptor. D'aquesta manera es pot garantir el secret davant de tercers interposats a la comunicació.



- Pas 0: Carlos té un certificat digital de clau asimètrica
- Pas 1: Carlos ha de lliurar la clau pública als interessats (els que volen enviar quelcom encriptat a en Carlos), en aquest cas Cristina.
- Pas 2: Na Cristina utilitza la clau pública per xifrar el missatge que vol enviar al Carlos.
- Pas 3: Na Cristina envia el missatge xifrat al Carlos i aquest utilitza la clau privada per desxifrar el missatge que li ha enviat la Cristina i ningú més a pogut veure el missatge en clar.

Garantia d'Autenticació i no repudi: El sistema de clau pública ens permet xifrar amb la clau privada de l'emissor i desxifrar amb la clau pública de l'emissor. Amb això podem assegurar que algú és qui diu ser i tampoc pot repudiar-ho. A més que garantim que allò que s'ha enviat no ha estat modificat per ningú altre des de la seva creació i signatura, **Integritat documental electrònica**.



- Pas 0: Carlos té un certificat digital de clau asimètrica i vol autenticar un document per a que ningú pugui alterar-ho o indicar que és l'autor. Només Carlos serà l'autor i no ho podrà repudiar.
- Pas 1: Carlos realitzarà un resum (digest) del seu document.
- Pas 2: Carlos signa digitalment el resum amb la seva clau privada.
- Pas 3: Carlos envia el document original i la signatura del resum d'aquest als receptors.
- Pas 4: Els receptors reben el document original i la signatura del resum del document original.
- Pas 5: El receptor torna a fer el resum del document original.
- Pas 6: El receptor valida que la signatura del resum enviat és correcta, és d'en Carlos i no s'ha alterat i extreu el resum original.
- Pas 7: Es compara el resum realitzat pel receptor del document original amb el resum enviat pel Carlos i validat amb la signatura. Si ambdos son iguals podem garantir que aquell document és el que crea en Carlos i que ningú no l'alterat des d'aleshores. La Cristina pot estar tranquila perquè el contingut del document està doncs verificat.

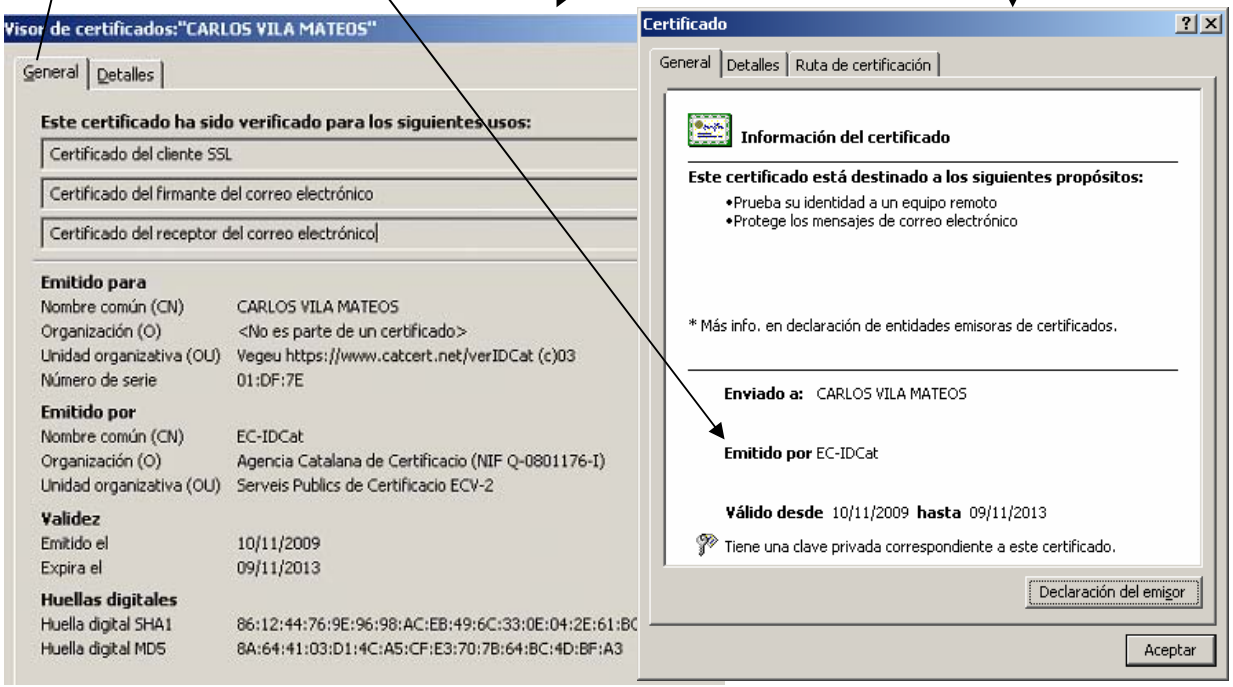
Nota: Si algú hages alterat o remplaçat la identitat del Carlos, la seva signatura digital no la podriem validar contra una CA de confiança i per tant també detectariem la identitat falsa.

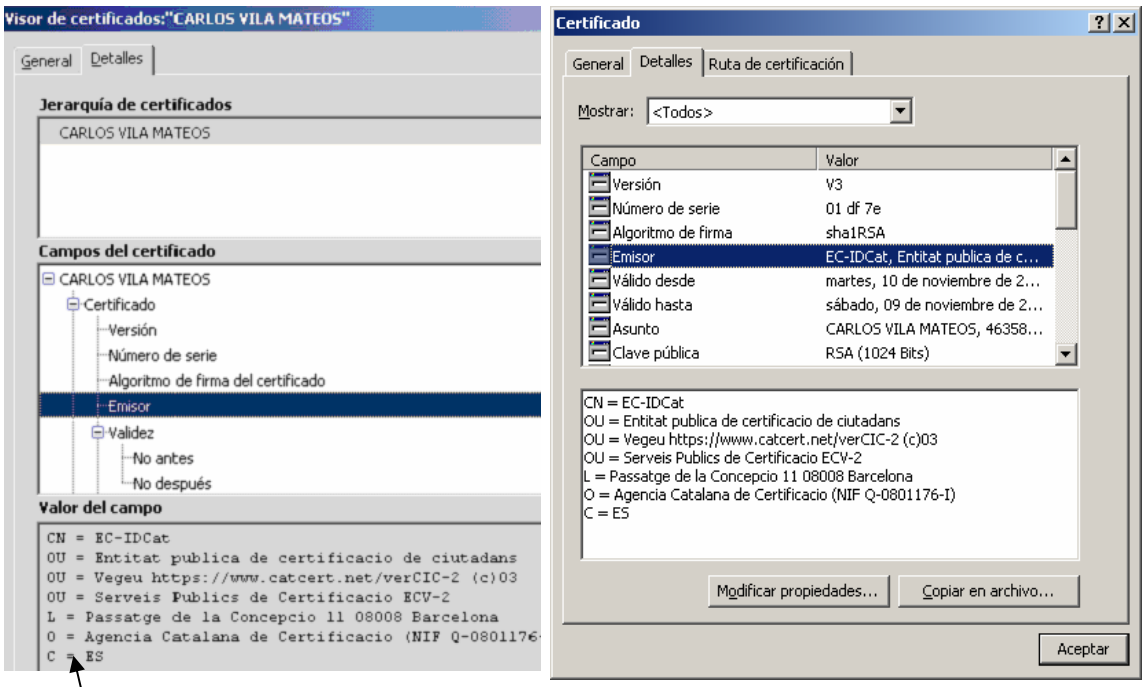
Amb aquests sistemes de claus asimètriques es va crear la infrastructura de clau pública i el certificat X509v3 i es va solucionar el problema de disposar d'un sistema identificatiu electrònic d'una persona o entitat.

Amb el sistema de clau pública o asimètrica una persona disposa de dos claus, una pública, que pot tenir tothom, i una altra privada, que només pot conèixer el propietari (es fonamental guardar-la ben protegida en el sistema de dispositiu de creació de signatura electrònica custodiant be la clau d'accés). Amb aquest sistema podem enviar un missatge confidencial a un altra persona amb el xifrat. Per estar segurs de qui ho envia és qui diu ser i no un suplantador podem signar el contingut amb la nostra clau privada de forma que el receptor (que té la clau pública) podrà amb aquesta assegurar que nosaltres hem enviat aquell contingut i que no s'ha modificat per això serveixen els certificats digitals X509.

Els certificats X509, és per tant un document electrònic que conté les dades identificaves d'una persona o entitat (servidor, programari, empresa, etc.) i la clau pública de la mateixa. Aquest certificats estan emesos i acreditats per les Autoritats Certificadores (CA) que es fan responsables de l'autenticitat de les dades que hi figuren com és el cas de CATCert.

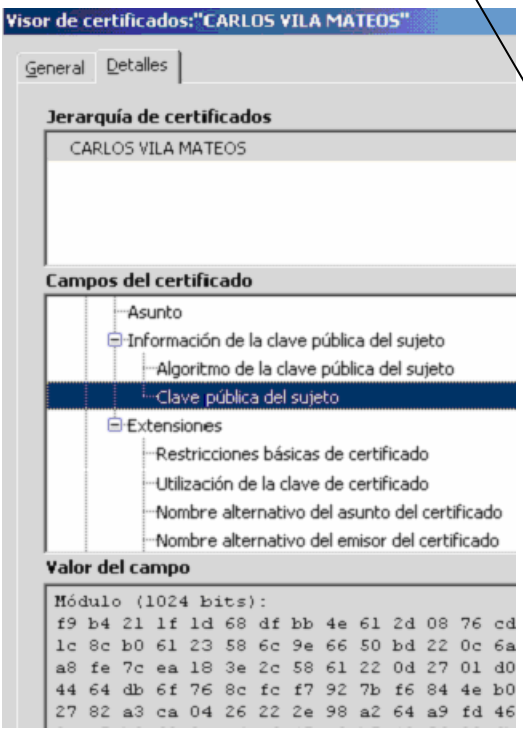
1) Els certificats X509v3 tenen una informació comuna y unes extensions que no son obligatòries.
2) Podem veure en exemple de certificat emes per CatCert per l'Entitat de certificació vinculada ciutadans (EC-IDCat):
veurem dos exemples segons mostra el Firefox primera columna i segons mostra l'IExplorer segona columna.

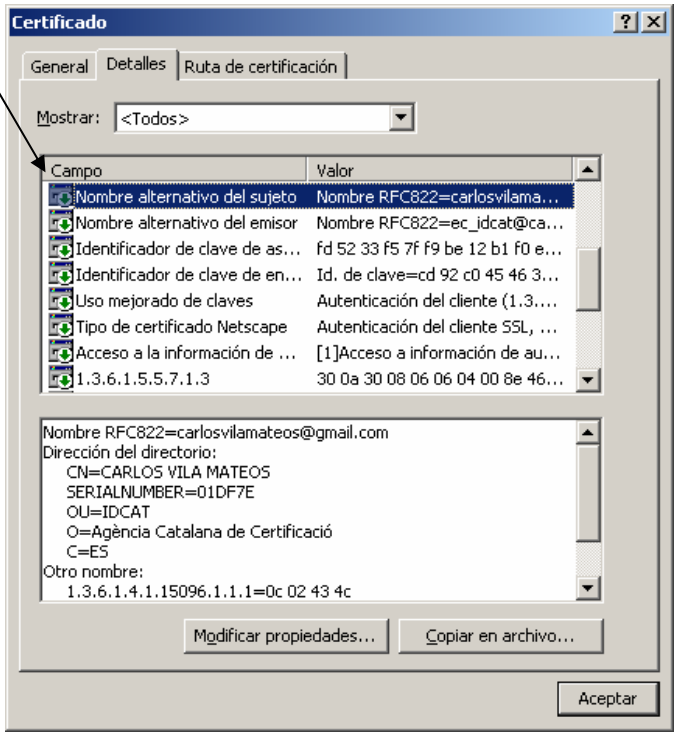




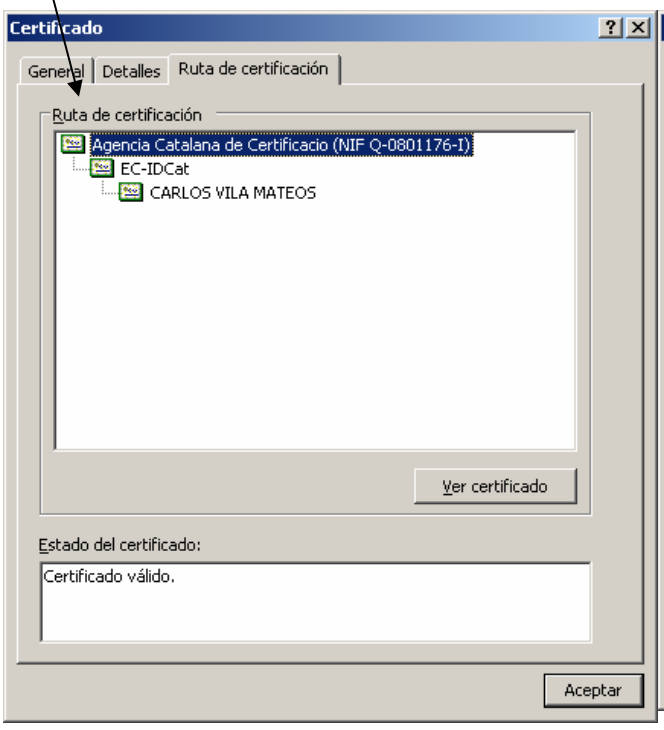
En l'IExplorer podem veure un visor més agradable a la vista i intuitiu on marca amb fletxes de color verd les extensions del certificat.
En el Firefox es pot veure quina es l'organització de l'estructura del certificat, perque esta tabulat i agrupat segons aquesta té un format que recorda al d'un LDAP: Certificado\Emisor\CN="";OU="";

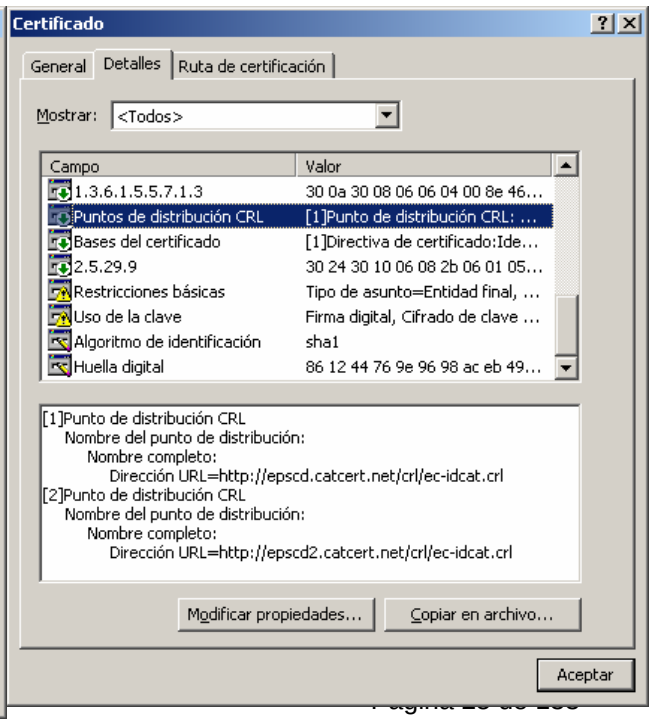
En el IExplorer podem veure un visor més agradable a la vista i intuitiu on marca amb fletxes de color verd les extensions del certificat.
En el Firefox es pot veure quina es l'organització de l'estructura del certificat, perque esta tabulat i agrupat segons aquesta té un format que recorda al d'un LDAP: Certificado\Emisor\CN="";OU="";





Per tal d'assegurar l'autenticitat d'aquest document, cal que algú es faci càrrec d'aquesta garantia, creant d'aquesta manera el que es coneix com a cadena de certificació, concepte que ens acompanyarà durant tot el tema de la PKI. A la següent imatge podem veure el que es diu la "Ruta de Certificación", aquesta cadena identifica quines entitats autoritzades i reconegudes han emés el teu certificat/identitat, és a dir, totes aquelles entitats que tenen potestats per certificar que un és qui diu ser (certificar la teva identitat, el teu certificat d'identitat X509). Aquesta cadena de certificació ha de ser validada també tota ella, és a dir, comprovar que són qui diuen ser.





Aquests certificats tenen, a més a més, una validesa temporal (una data d'emissió i una data de caducitat) que permet acotar el seu període de validesa en el temps.

Però que passa si ens roben el certificat o li roben a l'entitat que l'ha emès? Per a solucionar aquest problema, les entitats emissores de certificats (a partir d'ara CA's del seu acrònim en anglès) disposen d'una informació sobre l'estat dels certificats que emeten per a poder saber si la cadena que estem creant és vàlida o no. Això aporta complexitat a la validació de la signatura electrònica, ja que hem de validar que tant el resum encriptat com el certificat, i tota la seva cadena, siguin vàlids. Tanmateix, la informació de revocació també està signada, amb el què es torna a repetir en cascada el procés de validació...i s'ha de tenir en compte que la signatura ha d'ésser vàlida en el moment en el qual es va crear. La informació de revocació s'emet en unes llistes que cada cert temps es publiquen i on figuren els certificats no vàlids. Aquests llistats es coneixen bé com a CRL, de Certificate Revocation List, o bé com a OCSP, d'Online Certificate Status Protocol, un servei de consulta de l'estat on-line que permet consultar l'estat del certificat en el moment actual.

A més, hem de poder establir una manera fiable de determinar el temps de creació de la signatura, donat que potser quan es va crear el certificat era vàlid però ara ja ha caducat i, tot i això, la signatura hauria de ser determinada com a vàlida en l'actualitat.

Aquest problema s'ha resolt amb la figura de l'entitat emissora de segells de temps (TSA de Time Stamp Authority), que emet missatges on consta el temps de creació i que signa el contingut de la signatura per a poder garantir que aquesta existia en el moment que el segell diu.

Per una altra banda, es planteja el problema de la interoperabilitat de la informació signada. En aquest àmbit tenim dues famílies de signatures que, tot i que ser molt semblants, no són homogènies i difereixen, a banda de certs aspectes sintàctics, sobretot en les seves regles d'encoding.

Així tenim les signatures CMS (de l'acrònim Criptographic Message Syntax - RFC 3852) on els missatges estan codificats fent servir regles ASN.1; i el XMLDsig, on les signatures estan codificades en format XML.

5.2.1.4 Contenidors de Certificats i Signatures

En l'entorn de la signatura digital, principalment, existeixen dues gran famílies de signatures. Les **CMS** (són un contenidor de signatures codificat en ASN.1), regles de codificació binàries de documents, que permet múltiples signatures dintre del mateix missatge i que transporta el certificat amb el qual se signa i la informació de revocació del mateix.

Per la seva part, les signatures **XML**, són una mica diferents. Permeten signar múltiples continguts, però la signatura és única. Tanmateix, possibiliten aplicar transformacions sobre els documents abans de signar o validar (com ara processos de c14n molt útils per als documents xml) o bé signar i validar documents localitzats remotament (incloent la seva URI de referència).

Aquests dos estàndards de signatura són molt diferents, però tenen una funció comuna: defineixen un format de missatge per a les signatures i unes regles de procés que tant, qui signa com qui valida, han de seguir per a crear o validar signatures digitals.

El CMS és l'evolució de l'antic PKCS#7 i ve especificat segons el RFC 3852. Aquest format és molt madur, reconegut i molt extens. És el mateix format que utilitza el pdf. CMS permet manejar signatures múltiples, atributs signats i no signats (segell de temps) de manera molt eficient i directa. Segons **l'entorn del projecte** i les necessitats detectades, on s'utilitzaran, pdfs com a documents certificats, és més adient l'ús del **contenidor CMS**. Aquest contenidor té una estructura que permet inserir un segell de temps juntament amb la signatura de l'usuari (tal i com es pot veure en un pdf el segell de temps queda inserit com a una propietat del pdf).

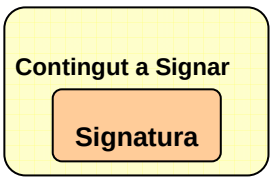
CATCert proporciona els mateixos serveis per tots dos sistemes, sent més òptim i adequat implementar el CMS pel nostre cas de gestió de documents pdf signats, però quedarà preparat i reflectit tant en disseny com en construcció d'interfícies tots dos mètodes de signatura, amb la intenció de seguir un estàndard que permeti comunicar-nos amb d'altres aplicacions externes que realitzin processos de signatura (documents signats, etc.). Per a realitzar aquesta integració i permetre aquesta interoperabilitat utilitzarem el Protocol DSS (Digital Signature System). Bàsicament aquest procotol treballa amb signatures de tipus CMS i XMLDsig i per tant coneixent com "parla" aquest protocol, el podrem utilitzar i acceptar signatures en qualsevol format i qualsevol contingut recollit en les especificacions d'aquest protocol. A més, DSS proporciona mecanismes d'extensió per ampliar funcionalitats. En el projecte en centrarem principalment en la signatura CMS, PSIS suporta ambdós tipus de signatures, i per això s'utilitzarà com a proveïdor d'aquest servei. D'aquesta manera, la plataforma de signatura estarà preparada en infrastructura, per afegir les primitives de Signatura (validació, signar, etc.) en ambdós casos.

5.2.1.5 Topologia de les signatures

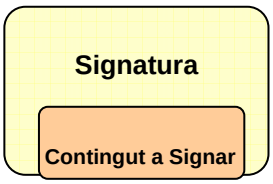
A banda del tipus de signatura i les seves codificacions les signatures poden agrupar-se de diferents formes o topologies. Això vol dir com es contenen les signatures i el seu contingut, és a dir, si la signatura conté el contingut signat o es mantenen per separat, segons aquestes tipologies podem distingir en els següents casos:

Signatures annexades (attached): Són signatures que estan unides al contingut a signar.

- **Signatura embolicada (attached enveloped):** Es dona en el cas de les signatures de tipus XMLDsig on la signatura s'inclou dintre el contingut a signar. En el XML l'element signature està contingut com un node més del document.



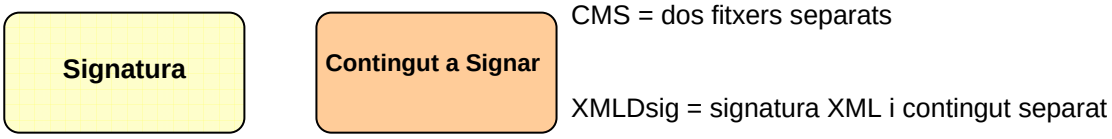
- **Signatura envolupant (attached enveloping):** Aquesta signatura inclou el document que s'ha signat realitzant un embolcall.



Per signatures CMS = attached

Per signatures XMLDsig = enveloping

Signatures separades (detached): En aquest cas es realitza la signatura del documento o dades i s'emmagatzemen per separat el contingut a signar i la signatura per un altre banda.



Signatures PDF: Aquest tipus de signatura no és un tipus "oficial" estàndard, sinó que és una signatura CMS dettached però incrustada en un arxiu PDF. Això vol dir que realitzem la signatura de tipus CMS dettached, és a dir, signem el fitxer PDF (amb resum) i obtenim la signatura a banda dettached (en fitxer) i després a través de les estructures que proporciona PDF afegim la signatura incrustada dintre del propi PDF. Això ho farem mitjançant unes llibreries que ens permetran tractar aquesta estructura i que veurem en l'anàlisi orgànic (toolkit Bouncy Castle i llibreria lowagie Itext).

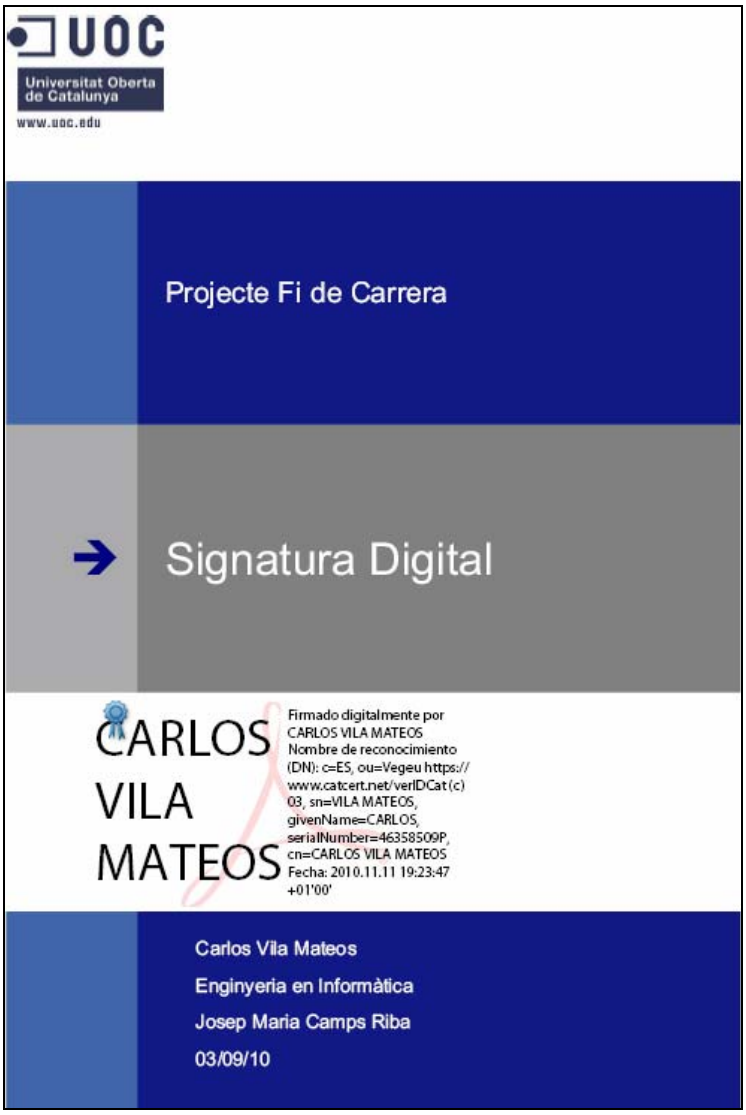


Figura: Exemple de document pdf signat amb signatura visible.

5.2.1.6 Segell de temps

És molt important determinar l'instant temporal de creació de la signatura o, si més no, deixar constància que un contingut existia en un instant temporal. Per a cobrir aquesta necessitat, es va crear la figura dels segells de temps (o Timestamps), on una autoritat (TSA) signa un missatge afegint un atribut protegit per la pròpia signatura on es reflecteix l'instant temporal de la creació del mateix.

Amb aquests tipus de signatures podem garantir l'instant de creació d'una signatura, l'existència d'un document i d'altres processos que necessitin de la determinació d'un instant en el temps, delegant la resolució d'aquest instant en una tercera autoritat de confiança (TSA).

Aquests missatges signats com no són més que signatures digitals amb un instant temporal segueixen també la sintaxi i regles de creació i validació determinades per a signatures CMS i XML.

En les fases d'anàlisi orgànic, de construcció i verificació podrem coneixer més a detall aquesta funcionalitat i comprovar el seu funcionament.

5.2.1.7 Signatures Avançades

Les ADES (Advanced Electronic Signatures) defineixen ampliacions sobre les ja mencionades signatures XML (Xades) com en format CMS (Cades).

Les ADES és una signatura que permet contenir el màxim número possible d'atributs i propietats necessàries totals per a validar una signatura i no dependre de cap sistema extern o conectivitat. És a dir, si la signatura digital li afegim tota la cadena de certificats de les CAs que la validen amb tots els certificats d'aquestes, afegim les llistes de revocació o CRLs, afegim un segell de temps, etc. El procés per a validar una signatura serà molt més complet, aquest procés contempla molts altres processos, com ara construir la cadena de certificació, validar-la, determinar l'instant de creació de la signatura o bé determinar amb quin certificat es va realitzar la signatura amb garantia total de la no alteració de cap dels passos citats.

Si la signatura pogués ser autocontinguda, és a dir, que portés tota la informació necessària per a poder-se validar en sí mateixa (alguna d'aquesta informació protegida per la pròpia signatura per a reforçar encara més la seva seguretat) reduint d'aquesta manera els graus de llibertat durant el procés de validació de la mateixa.

5.2.1.8 Digital Signature Services protocol DSS

DSS és l'abreviatura de Digital Signature Services protocol i proporciona un protocol basat en missatgeria XML per a poder oferir i demanar serveis de signatura digital. Aquest protocol ha estat definit i estandarditzat per OASIS (Organization for the Advancement of Structured Information Standards), grup internacional sense ànim de lucre que proposa estàndards per al desenvolupament, convergència i adaptació de l'e-business.

El problema amb la missatgeria emprada abans de DSS és que cada integrador de solucions PKI creava el seu propi protocol per tal de poder proporcionar el servei del qual normalment era propietari, tancat i no interoperable amb altres implementacions. Això provocava que cada integració amb un proveïdor diferent necessités d'un desenvolupament nou dels connectors així com una mancança de regles de procés d'alt nivell i estructures per a la prestació de servei de signatura digital amb el que cada proveïdor feia la seva solució a la seva manera i donant més o menys funcionalitats segons el seu criteri. Un altre avantatge de fer servir DSS és que té un nucli que permet la definició d'estructures addicionals a les definides donat que la seva estructura és ampliable i oberta. A més, permet definir perfils, que no són més que ampliacions del mateix protocol, per tal de donar solució a una casuística no contemplada al nucli del protocol. Exemples d'aquests perfils són el xifrat o arxivat, on es defineixen estructures i regles de processament per a poder donar serveis no contemplats al nucli del protocol. Un altre avantatge de la manera com es va concebre aquest protocol és que defineix les seves regles de procés i estructures per als diferents tipus de signatures disponibles, i no només per a un tipus concret. A més, també permet diferents combinacions i tipologies de documents tant per a signar com per a validar.

Una de les raons perquè hem escollit PSIS com a proveïdor de primitives i serveis PKI és l'ús d'aquest protocol DSS. Com hem comentat aquest protocol proporciona un nucli principal de primitives de signatura i un canal estàndard de comunicació amb una estructura que permet l'extensió amb noves funcionalitats, PSIS implementa el DSS i l'estén per afegir noves funcionalitats interessants pel nostre projecte com són els serveis/perfils de validació i normalització de certificats digitals, el servei especific de tractament de PDF o el servei de segell de temps.

Aquests perfils estessos (validació i normalització de certificats digitals, el servei especific de tractament de PDF o el servei de segell de temps) possibiliten que, sense alterar cap estructura bàsica, es puguin construir una sèrie de funcionalitats adaptades a necessitats més concretes, però sense descuidar el joc mínim de funcionalitats que tot sistema que proporciona serveis de PKI a través de DSS ha d'oferir.

PSIS suporta el Core de DSS i, a més a més, estén diversos dels seus perfils per aportar noves funcionalitats. En l'anàlisi orgànic veurem amb més detall el protocol DSS la seva organització i com usarem les extensions de CATCert.

5.2.1.9 Plataforma PSIS de CatCert

La Plataforma de PSIS (Plataforma de Serveis d'Identificació i Signatura) de CATCert proporciona uns serveis per augmentar la seguretat certificant per tercers diversos fets d'autenticitat i seguretat. PSIS proporciona un gran nombre de les operacions més demandades en el camp de la PKI relacionades, sobretot, amb la gestió de signatures electròniques. Proporciona aquests serveis integrats en un únic proveïdor. D'aquesta manera, el client no ha d'anar buscant cada servei a un lloc diferent o s'ha d'implementar les llibreries criptogràfiques o específiques.

Les funcionalitats de PSIS s'articulen al voltant de grans unitats funcionals que guarden certa similitud amb els diferents tipus d'autoritats definides al món de la signatura digital.

PSIS s'agrupa en tres grans autoritats:

- **AVS (Autoritat de Validació Semàntica):** El PFC Aquest projecte es centrarà en les funcionalitats d'aquesta autoritat i entrarem en més profunditat als següents apartats.

- **ASC (Autoritat de Signatura Centralitzada):** Permet la delegació del procés de signatura a CATCert. CATCert té la custodia de les teves claus privades, s'autentica contra el seu servidor (TLS) i ells s'encarreguen del procés de signatura.

- **AAI (Autoritat d'Acreditació i Federació de la Identitat):** L'ús d'aquesta autoritat permet que un sistema entri en un anell de federació on els sistemes comparteixen certa informació sobre els seus usuaris. Això permet que un sistema, que no té registrat a un usuari, demani a un altre, amb qui té una relació de confiança i on es troba registrat aquest usuari, certes dades que li permetin realitzar determinades tasques com ara autenticació o autorització del mateix.

El PFC utilitzarà l'Autoritat de Validació Semàntica ja que aquesta proporciona els serveis que es necessiten implementar en aquesta fase del projecte.

Aquesta autoritat proporciona serveis de validacions per a diferents tipus de missatges criptogràfics. Permet validar signatures digitals (CMS i XMLDsig), certificats de clau pública X509v3, segells de temps CMS i Xades i efectua operacions de validació i compleció de signatures digitals avançades (ADES). L'ús de signatures digitals avançades que permet que el validador (en aquest cas l'AVS), pugui afegir tots els atributs derivats de la validació que consideri necessaris dintre de la pròpia signatura. Així, la signatura pot incorporar la cadena de certificats amb la qual s'ha fet la validació, la informació de revocació emprada en el procés de validació o fins i tot, afegir segells de temps sobre certes parts de la signatura per tal de poder allargar la seva vigència en el temps. Aquestes signatures són més "independents" i permeten major grau d'autovalidació.

Funcionalitats que proporciona el AVS que utilitzarem al projecte:

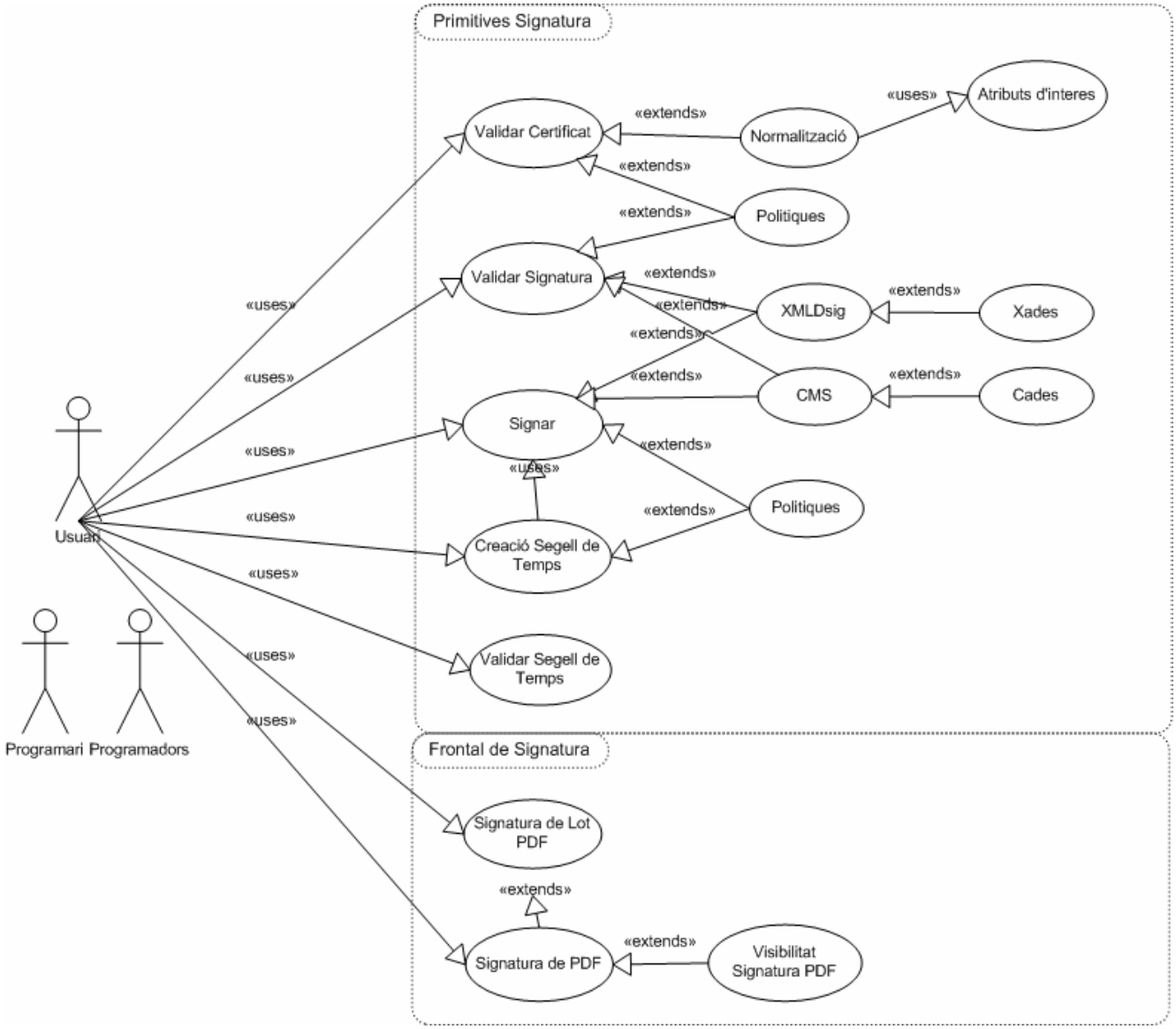
- Validació de signatures digitals (CMS i Cades CMS).
- Signatures Digitals Avançades (ADES):
  - Validar signatura.
  - Validar atributs: timestamps, validar cadena de certificació.
  - Completar atributs ADES.
- Validar segell de temps (CMS).
- Creació de segells de temps.
- Validació de certificats X509
- Extracció d'informació de certificats i signatures (normalització atributs).
- Validar Signatures CMS sobre documents pdf.

Funcionalitats que proporciona el AVS que no utilitzarem al projecte:

- Arxivat: validar una signatura i emmagatzemar-la.
- Xifrat: permet xifrar documents per un o múltiples destinataris.
- Polítiques de Signatura (Semàntica): Validar contra polítiques, depenen de si és un tipus de document o altre aplicar polítiques de signatura, per exemple, posar més restriccions per a signar un contracte, o una clau més llarga, o obligar de posar un segell de temps.

5.2.2 Casos d'us

De les funcionalitats que proporciona l'AVS incorporarem directament a la plataforma les que es necessiten pel projecte en aquesta primera fase, les hem agrupat doncs per casos d'us que detallem a continuació.



5.2.2.1 Us dels serveis de PSIS:

L'ús d'aquestes primitives romandrà principalment en utilitzar les primitives proporcionades pels serveis de PSIS i proporcionar una capa d'un nivell superior d'abstracció cap a dintre de la plataforma de tramitació on els serveis de signatura siguin més transparents. D'aquesta forma qualsevol programador o programari de la plataforma de tramitació o de les seves aplicacions satèl·lits podrà utilitzar aquests mètodes simplificats. D'aquesta manera l'objectiu d'aquest mòdul serà deixar els mètodes preparats i provats (test) per a que qualsevol programador de la plataforma de tramitació en pugui fer us. Els serveis a utilitzar de PSIS seran:

- Validació de signatures digitals (CMS i Cades).
- Validar segell de temps (CMS).
- Creació de segells de temps.
- Validació de certificats X509
- Extracció d'informació de certificats i signatures (normalització atributs).
- Validar Signatures CMS sobre documents pdf.
- Signatures Digitals Avançades (ADES):
    - Validar signatura.
    - Validar atributs: timestamps, validar cadena de certificació.
    - Completar atributs ADES.

5.2.2.2 Frontal de Signatura:

El frontal de signatura ens permetrà en les extensions de futur, en un segon projecte, cobrir els requeriments bàsics de la plataforma en quant a signatura de documents, autenticitat i integritat del signant. Per això utilitzarem un component dissenyat per CATCert que haurem de conèixer el seu us en profunditat parametritzar-lo per aprofitar aquelles funcionalitats que cobreixen els requeriments de la plataforma i modificar algunes d'aquestes funcionalitats en els casos adients.

Aquest component de frontal web disposa de les primitives i serveis PKI abans comentats ja implementats però l'utilitzarem per integra-ho amb el PortaSignatures amb les següents funcionalitats principals necessàries:

- Signatura de documents PDF visible, a través del frontal de signatura (applet).

  Utilitzarem la signatura CMS dettached però incrustada en un arxiu PDF. Això vol dir que realitzem la signatura de tipus CMS dettached, és a dir, signem el fitxer PDF (amb resum) i obtenim la signatura a banda dettached (en fitxer) i després a través de les estructures que proporciona PDF afegim la signatura incrustada dintre del propi PDF.

  Això ho farem mitjançant unes llibreries que ens permetran tractar aquesta estructura i que veurem en l'anàlisi orgànic (toolkit Bouncy Castle i llibreria lowagie Itext).

  El projecte té com objectiu conèixer les possibilitats de la signatura digital i les primitives PKI, els proveïdors de serveis d'aquest tipus per cobrir els requeriments del projecte en aquest àmbit sense haver de desenvolupar la complexitat de la signatura digital com a tal que no és part del nostre negoci. D'aquesta forma la signatura digital serà un procés transparent pels usuaris de la nostra plataforma i nosaltres ho integrarem i adaptarem per a que doni resposta als requeriments del nostre programari.

  En el projecte treballarem amb aquesta signatura, CMS en PDF, com la signatura principal per l'ús dels documents electrònics en un expedient i la seva tramitació. Aquesta decisió es dona perquè el document pdf amb signatura incrustada es un document fàcilment transportable, autocontingut i amb interfície agradable per l'usuari i que permet n-signatures. Amb això volem dir que podem transportar fàcilment el document amb la signatura amb un sol fitxer reconegut i estandarditzat (pdf), aquest document es pot versionar dintre del gestor documental, expedient i plataforma de tramitació electrònica. Aquest document és totalment vàlid si s'envia fora de la plataforma, de forma independent i seguirà sent vàlid, íntegre, i verificable per qualsevol que ho rebi. Es port validar la signatura en aquest mateix contenidor de forma agradable i visual (Acrobat Reader). Es poden afegir camps a la signatura visibles per l'usuari amb una interfície agradable i visual.

  Per això tindrem que integrar l'applet de CATCert amb la nostra aplicació i adaptar alguna part del codi inclòs de la llibreria de Bouncy Castle i lowagie Itext per modificar alguna funcionalitat.

  - Modificar els paràmetres visibles de la signatura en PDF: Com hem comentat en l'apartat anterior de signatura de PDFs, haurem de modificar les llibreries proporcionades per CATCert per a afegir nous camps visibles al PDF a la signatura CMS continguda en aquest, com la rubrica digitalitzada emmagatzemada als sistemes corporatius de l'usuari que està signant en la plataforma de tramitació electrònica, permet visualitzar la rúbrica, comentaris, data (segell de temps), localització, etc. Per a fer això caldrà modificar les

llibreries de itext i accedir en temps de signatura als repositoris corporatius a recuperar en el perfil de l'usuari la rubrica digitalitzada.

Una signatura en un document pdf pot contenir diversos camps visibles i opcionals que els posarem com atributs signats:

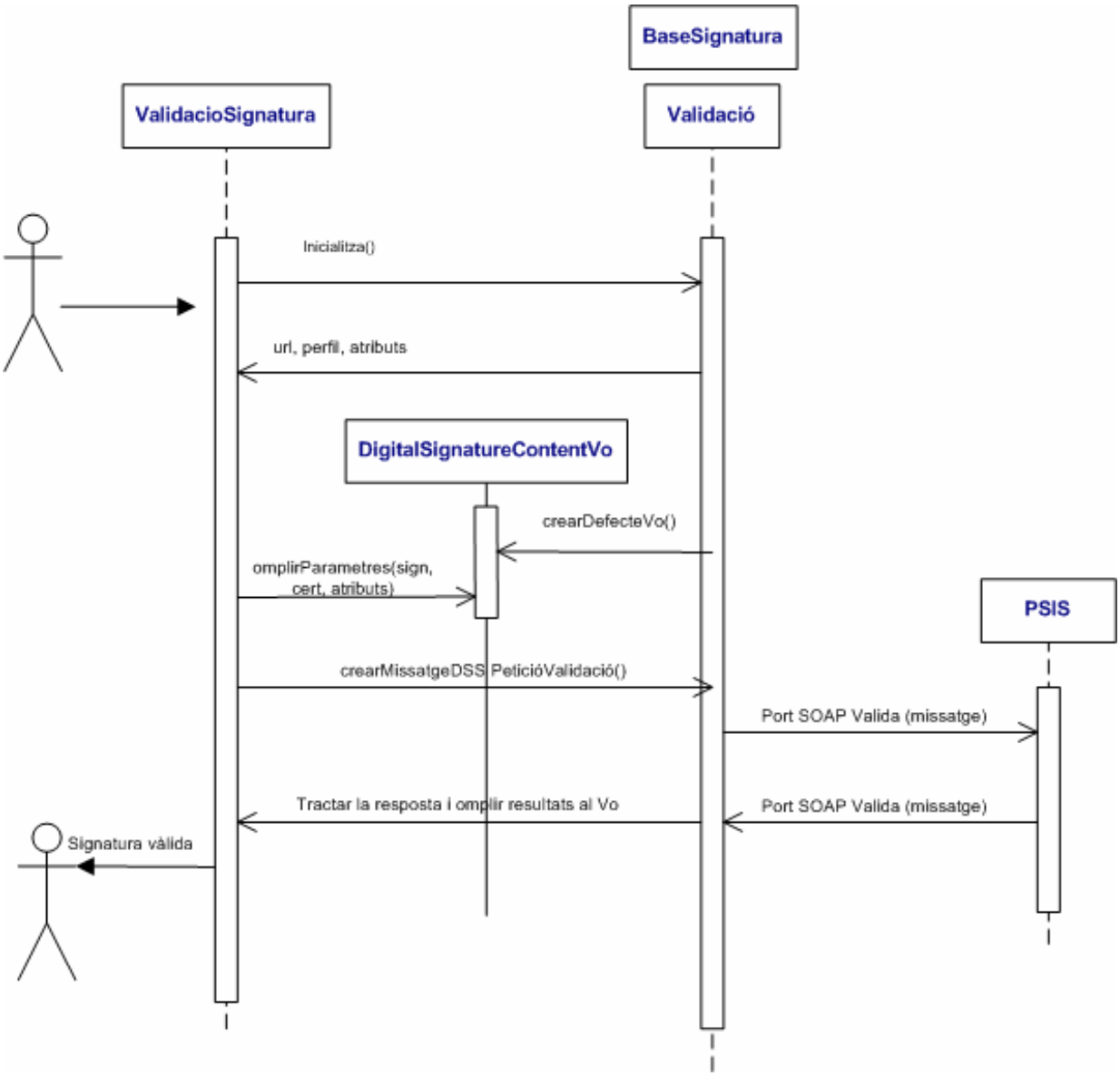
- Una imatge, logotip o només text; el nom del signant.
- La informació pròpia del certificat (CN=, DN, càrrec, etc.)
- La raó de signatura o comentaris.
- La localització i informació de contacte.
- La data de signatura local o certificada que és més segura, podem recollir-la d'un proveïdor de segell de temps, TSA, en el nostra cas CATCert.
- Podem marcar la signatura per a qui requereixi revisar el document i les seves alertes abans de signar-ho.

- Signatura de Lots, a través del frontal de signatura (applet):

S'estudiarà el cas d'ús de realitzar Signatures per Lots. Cal destacar que el fet de realitzar signatures per lots sense obrir, visualitzar i signar cada document un a un, és un procediment en que recaurà la responsabilitat en l'usuari. L'opció que s'utilitzarà en cas de signatura de lots de documents serà realitzant la selecció de n documents del PortaSignatures i enviar els n documents al JWS de signatura, de manera que aquest tindrà que demanar el PIN i confirmació un sol cop. Segons les normatives recollides a la "Guia dels dispositius i les aplicacions segures de creació de signatura electrònica" de CATCert apartat 1.3, es desprèn que l'aplicació ha de donar l'opció de poder visualitzar el document abans de signar, però la responsabilitat de visualitzar-lo abans de signar recau en l'usuari. Per tant, des del punt de vista de l'aplicació, només cal assegurar que l'usuari té l'opció de visualitzar-lo abans de signar si així ho desitja, però serà la seva responsabilitat si donant-li aquesta opció l'aplicació, l'usuari no el visualitza i el signa directament. Això dependrà de la confiança que tingui l'usuari en els documents que ha de signar, si s'han generat automàticament, o si els ha preparat alguna persona de confiança, etc... en molts casos els signarà directament perquè té constància de la font que provenen i en la qual hi confia. En altres potser preferirà veure-ho abans.

Hi ha altres mecanismes però, per donar aquesta funcionalitat: PSIS suporta la signatura centralitzada i en aquesta es pot realitzar un contracte amb CATCert per signar un tipus de documents per lots de forma centralitzada (per exemple: per a signar les factures d'un determinat client).

5.2.2.3 Diagrama seqüència dels Casos d'us



Aquest diagrama de seqüència és un exemple del prototipus de seqüència que és seguirà a l'hora de construir les relacions entre les classes i mètodes principals. En aquest cas es tracta de la validació d'una signatura, per la validació d'un certificat o creació de signatures CMS o un segell de temps seguirem la mateixa dinàmica però amb les classes especifiques pròpies de cada servei. L'estructura de classes s'explicarà amb detall a la fase de construcció.

5.2.2.4 Procés validació PSIS-CATCert

CATCert proporciona uns serveis de certificació digital amb la plataforma de PSIS. Ja s'ha descrit les diverses funcionalitats i quines s'utilitzaran, la principal, però és la validació del certificat o signatura que inclou la normalització d'atributs com el NIF. Aquesta funcionalitat permet que les administracions puguin reconèixer els certificats emesos per entitats de certificació terceres sense que els calgui conèixer els detalls del diferents perfils de certificats i sistemes de consulta d'informació de l'estat de revocació dels certificats.

Els passos d'aquest procés són els següents:

1. Comprovarà si el certificat pertany a alguna de les entitats de certificació classificades per CATCert. D'aquesta forma es delega en CATCert la feina necessària d'auditoria de les entitats de certificació i dels certificats que aquestes emeten.

2. Comprovarà si el certificat ha estat revocat. D'aquesta forma es delega en CATCert la feina necessària de contínua actualització de la informació de l'estat de revocació dels certificats emesos per totes les entitats de certificació classificades.

3. Farà l'homogenització dels continguts dels certificats, extraient les dades del titular o del signatari. Així no cal que coneguem els detalls dels perfils de cada certificat classificat.

4. Retornarà una resposta, signada pel servei de validació, indicant la validesa de la signatura o certificat. Es crearan els missatges de petició de serveis i interpretar el missatge de resposta segons el protocol DSS i els serveis de PSIS.

5.2.3 Identificar les necessitats funcionals del PortaSignatures per utilitzar els serveis de PKI.

Com hem comentat aquest projecte estarà composat per dos components: el **frontal web de signatura** i el mòdul de serveis de PKI (**ProveidorSignatura**), a més disposarem d'una aplicació web, el **PortaSignatures**, per realitzar la integració d'aquests serveis amb un cas real d'us i a través d'aquest la validació dels serveis de signatura.

El mòdul responsable de conèixer el negoci de serveis PKI i donar resposta a les funcionalitats de Signatura[1] que es requereixen dintre de la plataforma tecnològica i amb les seves particularitats concretes serà el mòdul de ProveidorSignatura.

Aquest mòdul podrà anar adquirint noves funcionalitats i donant nous serveis dins de la plataforma, ja sigui per què sorgeixen noves necessitats, tecnologies o suports específics. Aquest mòdul s'ha dissenyat per no dependre exclusivament d'un sol proveïdor de serveis de signatura. Es tindran unes interfícies d'entrada i sortida "estàndards".

Aquest component serà el que implementi el protocol DSS, un protocol estàndard que ens permetrà sol·licitar diversos serveis de Signatura a diversos proveïdors. Com a proveïdor inicial farem servir PSIS de CATCert, com a proveïdor que implementa el protocol DSS i, què, ens proporciona molts serveis estàndard i estesos que, en l'actualitat, no estan implementant altres proveïdors. Però, en el cas que per tots o per algun servei en concret es vulgui canviar de proveïdor, es podrà fer amb un impacte reduït, sempre i quan aquest proveïdor implementi l'estàndard DSS.

El PortaSignatures és l'aplicació que gestiona les ordres de signatura. És un programari amb frontal web on els usuaris els hi arriben els documents que tenen pendents de signatura de diferents expedients i tràmits ordenats cronològicament.

Els usuaris que participen en la tramitació dels expedients disposen d'una eina que els permet accedir de forma ràpida i fàcil als documents que tenen pendents de signar, amb l'objectiu de signar-los, rebutjar-los, validar-los i consultar-los de forma individual i col·lectiva.

De manera resumida, el PortaSignatures recull tota una sèrie de documents que requereixen la signatura d'algun que forma part de la corporació. En aquest sentit, s'han generat en algun moment del procediment una sèrie de documents electrònics i, per tant, ja no existeixen documents físics. En aquest punt, les persones implicades són informades del fet que han de signar una sèrie de documents, que tenen pendents de signatura. Se'ls hi facilita l'accés als mateixos i, a través d'un certificat digital que garanteix l'autenticitat de la persona requerida, es signa el/s document/s pendents.

El conjunt del procés i la implantació d'una eina com la que es proposa té les següents avantatges i objectius pel negoci:

[1] Es parlarà de Signatura, d'ara endavant, de forma genèrica com totes les funcionalitats relacionades a la signatura digital: serveis de validacions per a diferents tipus de missatges criptogràfics, validar signatures digitals (CMS i XMLDsig), certificats de clau pública X509v3, segells de temps CMS, Xades i efectuar operacions de validació i compleció de signatures digitals avançades (ADES).

- S'elimina el procés físic de signatura del document que es realitzava fins aquest moment de manera manuscrita, gestionant de manera eficient i clara tant la seva signatura com el seu rebuig, amb tots els efectes que aquestes accions suposen.
- Permet estandarditzar el seu ús per a qualsevol tipus de document que requereixi una signatura electrònica, complint amb tots els aspectes de legalitat i seguretat requerits.
- Es constitueix com una solució de gestió integral, segura i en consonància amb els requeriments de la normativa electrònica actual d'aplicació.

A més, des de la seva mateixa implementació permet un estalvi de temps i costos en dos aspectes ben diferenciats:

- Gestió: reducció dels temps de tramitació, unificació de criteris de signatura, informació accessible i explotable i, en última instància, una estandardització de la metodologia de treball per a la corporació.
- Producció: estalvi de paper i de material associat a la signatura manuscrita.

El PortaSignatures, és una simple aplicació web que gestiona documents o ordres pendents de signatura, en si no té cap complexitat addicional que la d'utilitzar un mòdul web i una lògica de negoci per a gestionar-la.

Aquest disposarà de les funcionalitats web per poder gestionar els documents a signar per cada usuari en concret. Els Casos d'us principals del Porta Signatures són:

- Bústia d'ordres de signatura: Disposarà d'un frontal on l'usuari podrà consultar els documents pendents de signatura.
- Històric de l'usuari: Opció de consulta de l'històric dels documents signats.
- Realitzar signatura de documents PDF (CMS).
- Rebuig de l'ordre de signatura.
- Signatura per lots: signar fins a 10 documents o ordres de signatura d'un sol cop, tan si els ha de signar un sol usuari o, be, per múltiples usuaris, independentment de l'ordre de les signatures. També es podrà realitzar aquesta signatura, en el cas de la que hi hagi més d'una persona, de forma niuada. No es podrà saltar l'ordre de signatura establert pel document a signar.
- Suport a la signatura múltiple en paral·lel. El document es podrà signar per diverses persones, sense la necessitat de que s'hagi de seguir un ordre específic.
- Es podrà signar un document per múltiples usuaris de forma niuada. Els usuaris hauran de seguir un ordre per tal de signar un document. No es podrà signar si l'usuari anterior no ha realitzat la signatura.
- Amb supervisió de l'usuari, es podrà realitzar una signatura de múltiples documents de forma automàtica, sense l'assistència de l'usuari i sempre demanant l'aprovació d'aquestes signatures.
- Validació de credencials de l'usuari a través de certificats X509.

Un cop adquirits els coneixements de Signatura Digital, realitzat el disseny i la construcció del mòdul principal de ProveidorSignatura, aquests nous casos d'us ens ajudaran en un segon projecte a realitzar la verificació de la construcció de l'applet de CATCert convertit a JWS i integrat amb la plataforma.

5.3 Capítol 3: Fase II - Anàlisi Orgànic: Dissenyar la capa d'integració entre una aplicació J2EE i els serveis de PKI.

En la fase de definició Orgànica ens preocuparem dels aspectes més tècnics del projecte, quines eines, llibreries, metodologies tècniques utilitzarem per cobrir les premisses tècniques per després, suportat amb aquestes bases, definir el disseny orgànic de la solució i entrar en el detall en cada mòdul definit en aquest.

5.3.1 Aspectes i premisses tècniques de la Plataforma de Signatura

La Plataforma de Signatura amb el seu mòdul principal ProveidorSignatura estan composats per una capa de presentació i lògica de negoci pròpies, que permeten complir les funcionalitats dels components descrits en aquest document.

Estan desenvolupats, fonamentalment, en **programari lliure (codi obert)** i en **arquitectures** obertes, **complexes, robustes**, **escalables** i de fàcil **manteniment,** com és la plataforma *J2EE*.

Per tant, tots els mòduls estan analitzats, dissenyats, implementats i distribuïts amb aquesta filosofia: obert, escalable, distribuïble, mantenible, interoperable.

Serveis de signatura, acreditació i certificació digital

Aquesta solució està basada en la plataforma de serveis de signatura (PKI) de CATCert, PSIS (Plataforma de Serveis d'Identificació i Signatura), adoptant estàndards universals seguint les pautes marcades per CATCert i juntament amb l'ús, integració i certificació de les seves eines al nostre projecte.

La Plataforma de Signatura està desenvolupada segons les normatives i pautes marcades pels organismes oficials de serveis de signatura i identificació digital per tal de complir amb la legislació vigent en l'entorn d'aplicacions de Signatura Digital

- La Plataforma de Signatura garanteix la signatura digital reconeguda i la gestió segura de documents, la integritat de les dades sensibles i es comunicarà amb interfícies segures i utilitzant sempre un canal segur.
- Integritat documental electrònica: Garanteix que el document no ha estat modificat des de la seva creació i signatura (algorismes de resum i algorismes de signatura electrònica i segell de temps certificat per una tercera entitat).
- Autenticació: Garanteix la identitat electrònica del ciutadà i del treballador públic. Garanteix que la persona que ha signat el document és qui diu ser (algorismes de signatura electrònica asimètrics).
- Si s'afegeix un segell de temps d'una tercera part de confiança com CATCert (TSA), es garanteix a més que aquell document signat per una persona concreta ho va fer a una data donada i no ha estat modificat des d'aleshores.
- PSIS és una plataforma basada també en Java, J2EE i en tecnologies Open Source, com ara Spring i Hibernate, que s'aprofita dels avantatges derivats de l'ús de les mateixes, com ara oferir el seu servei en un entorn fiable, escalable i transaccional, aconseguint d'aquesta

manera convertir l'ús de la PKI en una *appliance* més que es dóna als clients com a un servei de caixa negra.

- Aquesta plataforma presenta les funcionalitats abans esmentades com un servei remot que s'invoca mitjançant un WebService SOAP estàndard. D'aquesta manera, l'aplicació client, el component de Signatura Digital, es podrà integrar sempre que compleixi els requeriments mínims d'integració amb el WebService. Això ens permet un espectre de tecnologies compatibles molt més gran del que normalment permeten aquest tipus d'entorns PKI.

- La Plataforma de signatura s'ha dissenyat amb la mateixa filosofia per poder escalar-la i cobrir tots els serveis de signatura, és compatible amb qualsevol tipus de certificat electrònic i qualsevol tipus de signatura: CMS, XMLDSig, ADES, XADES. I qualsevol servei de signatura: validar certificats, normalització d'atributs (obtenir el NIF o altres atributs per a qualsevol tipus de certificat), validar signatura, segell de temps (CATCert TSA), consultar documents signats i validar-los en un moment donat de temps, signatura de formularis, signatura detached.

- El component Proveïdor de Signatura ens oferirà les funcionalitats de Signatura a la resta de la Plataforma Tecnològica o altres aplicacions externes mitjançant serveis web (arquitectura SOA): comunicació per XML (DSS), validació de certificats, normalització d'atributs (obtenir el NIF o altres atributs per a qualsevol tipus de certificat), validar signatura, segell de temps (CATCert TSA), consultar documents signats i validar-los en un moment donat de temps, signatura de formularis i signatura detached.

Dispositiu de verificació de signatura electrònica

- El procés de verificació haurà d'obtenir la validesa a partir de la clau pública de la signatura, i per estar segurs que el certificat és vàlid, haurà de verificar la ruta de certificació fins una arrel fiable i validar totes les entitats certificadores de la ruta (CA). La verificació de la signatura la farem a partir de dues opcions:

- Aquest projecte es fonamenta en la validació de Signatures Digitals. Així doncs, el component de ProveidorSignatura, comprovarà on-line les llistes de revocació vigents (CRL) contra els serveis de CATCert. S'hauria de tenir en compte que, si els serveis de CATCert (verificació de signatura, segell de temps, etc) no estiguessin disponibles, s'hauria d'actuar bé permetent o bé denegant el procés de signatura digital, donant, això si, el missatge adient a l'usuari, deixant el procés de signatura i del tràmit en el seu estat anterior amb tot l'entorn inicialitzat i disponible per noves operacions. Per aquest supòsit, PSIS garanteix un SLA del 99,7%. Un altra opció seria implementar una altra validació contra un proveïdor diferent que implantés el protocol estàndard de Signatura DSS (Digital Signature Service), d'aquesta manera totes les funcionalitats que ens serveix PSIS també ens les serviria aquest altre proveïdor (inclosa la normalització) però cal tenir en compte que, ara per ara, no hi ha altre proveïdor que segueixi aquest estàndard i que implementi totes les funcionalitats que requerim.

- En el propi pc de l'usuari podrà tenir instal·lats diferents validadors de signatura que segueixin les normatives del procediment adient per validar-les: validar el certificat (sistema asimètric clau pública / clau privada), la cadena de certificació, el segell de temps, les CRLs. Com que en la plataforma de tramitació s'utilitzaran com a documents signats reconeguts els documents pdf amb signatura visible (CMS dettached però incrustada en un

arxiu PDF) podem utilitzar com a eina corporativa el propi visor de documents pdf (Acrobat Reader) com a dispositiu de verificació de signatura electrònica. Aquest utilitza l'estàndard OCSP (Online Certificate Status Protocol) per verificar les CRLs.

Arquitectura i tecnologies emprades

A continuació, es detalla tota la graella de programari que configurarà l'arquitectura de la plataforma:

Projecte en curs:

- **J2EE**: Serveis; JAAS, SOA, SOAP, JAXP, JMX, JTA, JMS, RMI, EJB, JNDI, JAF, JDBC, etc.
- **Programari i normatives de CATCert i l'AOC**: Ús i integració del programari i normatives dels serveis d'identificació i signatura digital: funcionalitats de la plataforma PSIS, protocol DSS sobre XML i extensions, CMS, XMDSig, XADES, CADES, Certificats X509 (compatibilitat amb múltiples emissors: DNI digital, idCat, etc.), TSAs, OCSP, Acrobat Reader, CRL's, atributs de certificats, Bouncy Castle, JCA/JCE, PKI, comunicació i autenticació amb certificats servidor i client: HTTPS i TLS.
- **XMLBeans 2.1**: Llibreria per facilitar el tractament d'arxius XML definits amb XSD.
- **XFire 1.2 distributed**: Framework per la connexió i configuració de serveis web
- **JDK v1.6** o superior.
- **Enterprise JavaBeans (EJB):** API que forma part de l'estàndard J2EE de Sun i que proporciona un model de components distribuït estàndard per al costat servidor d'una aplicació.
- **CSS i recursos de propietats**: La interfície web amb l'usuari es podrà personalitzar per tal d'adaptar-se a l'aparença de la resta de les aplicacions corporatives. Es suportada pels diferents navegadors i versions que compleixin l'estàndard del Consorci W3C (les diverses versions d'Internet Explorer i Mozilla Firefox).
- **Eclipse Helios (3.6.1)**: Eclipse Java EE IDE for Web Developers. IDE obert de programació.

Segon projecte:

- **Struts**: framework de programació de frontals web (MVC).
- **Hibernate**: framework d'accés a dades (motor de persistència): per a l'accés a la base de dades amb objectes POJO i permet treure la responsabilitat de l'accés a dades d'on ell és expert. Permet treballar de forma no acoblada amb diferents proveïdors de BBDD: Oracle, DB2, Microsoft SQL Server, Sysbase, MySQL, PostgreSQL, TimesTen, HypersonicSQL, SAP DB...
- **Jboss i OPENSSO**: L'aplicació ha de permetre la validació dels usuaris a través d'un registre d'usuaris LDAP, com poden ser Active Directory o OpenLDAP.
- **Oracle**: La plataforma actual treballa amb aquest programari de BBDD.

La filosofia J2EE

És una plataforma de **serveis** (APIS) **estanditzats** que permet desenvolupar, mantenir i escalar grans aplicacions o aplicacions *entreprise*. Aquesta plataforma utilitza serveis estàndards que fan la feina transversal i **d'infrastructura**, evitant la complexitat d'aquests serveis i permetent que ens centrem en la nostra lògica de negoci, la tramitació electrònica, amb totes les garanties de seguretat, comunicació global (interoperatibilitat), transaccionalitat, identificació, de forma transparent. Els serveis estàndards són:

- Seguretat **JAAS**: Java Authentication and Authorization Service JAAS
- Arquitectura orientada a serveis (**SOA**): És una arquitectura basada en **SOAP** i XML que permet la interoperabilitat entre aplicacions i xarxes de diferents topologies. Permet que ens centrem en la nostra lògica de negoci i que la responsabilitat de la comunicació i interoperabilitat sigui transparent i la complexitat d'aquesta sigui responsabilitat d'aquest Servei J2EE (API).
- Servei per el processament de **XML**: Java API for XML Processing (JAXP)
- Servei de mónitorització del servidor d'aplicacions, sistema d'indicadors d'explotació: **JMX** (Java Management Extension).
- Servei de llançador en el temps: Timer.
- Servei de control transaccional **JTA**: Java Transaction Service (JTS).
- Servei de missatgeria síncrona i asíncrona: Java Message Service (**JMS**).
- Serveis de missatgeria de correu electrònic: JavaMail, JavaBeans Activation Framework (JAF).
- Serveis d'invocació remota (components **EJB**): **RMI** sobre l'Internet Inter-Object Protcol (IIOP).
- Servei de directori i localització de recursos: Java Naming and Directory Interface (**JNDI**), Network Information Service Plus (NIS+) i el Lightweight Directory Access Protocol (**LDAP**).
- Servei d'accés a dades (servei de persistència): Java DataBase Connectivity (**JDBC**).
- Serveis de presentació: eines de treball en la presentació: Patró **MVC**, JSPs, servlets, taglibs, etc.

Aquesta plataforma es pot fer servir per a grans aplicacions de negoci i ens donarà tots els serveis esmentats, permeten seguir tot el cicle de vida de creació del programari, incloent la qualitat i el manteniment.

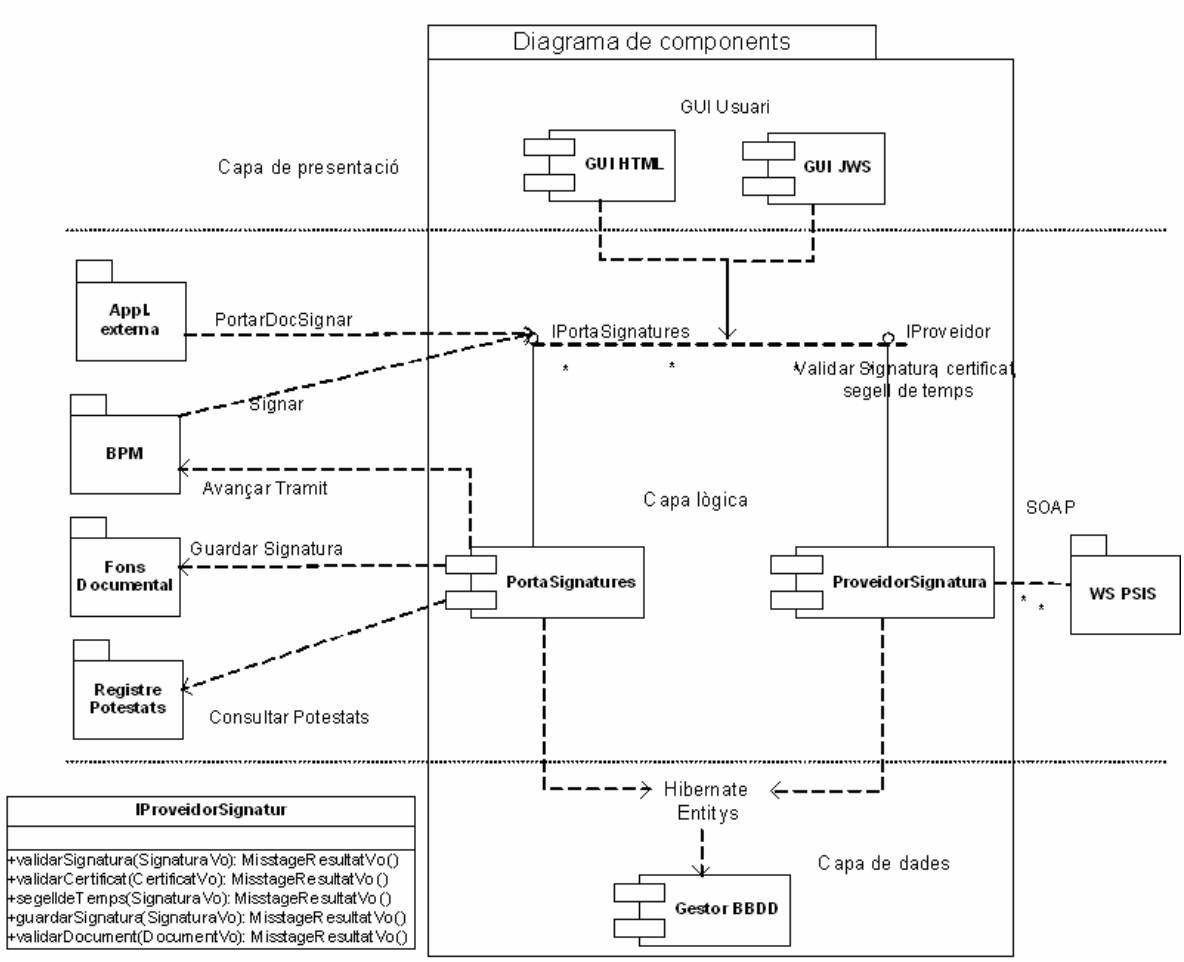
Donat que l'especificació *J2EE* no obliga a les següents funcionalitats: **escalabilitat**, **la tolerància a fallides o el balanceig de càrrega**, s'escollirà un servidor d'aplicacions que les implementi, Es planteja com a servidor d'aplicacions *Jboss*, que és també un contenidor d'aplicacions web i contenidor de grans aplicacions de negoci (*J2EE*-EJB), amb els serveis mencionats i la **infrastructura** que ho suporta.

Aquest producte permet el treball en xarxa sense limitacions d'usuaris i amb un rendiment equivalent, ja que és escalable i redundant (clúster) sense que això suposi un augment del cost de llicències de tot el programari indicat anteriorment.

5.3.2 Definició de l'arquitectura general de la Plataforma de Signatura. Enterprise Appl. (J2EE).

En la definició de l'arquitectura tindrem en compte els objectes de negoci que han d'assumir cada mòdul definit de forma que estiguin cohesionats però no s'acoblin entre els diferents mòduls independents. Aquesta arquitectura, la separació i distribució de components en responsabilitats especialitzades permet la reusabilitat i manteniment d'aquests.

Com hem comentat anteriorment aquest projecte estarà composat per dos components principals: el **mòdul de serveis de PKI** (ProveidorSignatura) i el **frontal web de signatura (JWS),** a més, en el futur, disposarem d'una aplicació web, el **PortaSignatures**, per realitzar la integració d'aquests serveis amb un cas real d'us i a través d'aquest la validació dels serveis de signatura. En aquest apartat entrarem en detall de quins són aquests mòduls i com es configuren.



5.3.3 El mòdul de Proveïdor de Signatura

El mòdul responsable de conèixer el negoci de serveis PKI i donar resposta a les funcionalitats de Signatura que es requereixen dintre de la plataforma tecnològica i amb les seves particularitats concretes serà el mòdul de ProveidorSignatura. Aquest, per tant, serà el mòdul expert en la signatura digital i proporcionarà una capa d'abstracció de forma que aïllarà la complexitat de la signatura digital que romandrà en PSIS. Aquest mòdul doncs serà expert perquè sabrà parlar amb PSIS i implementar el protocol de comunicació DSS de forma que proporcionarà una capa o API transparent que hom podrà utilitzar en la plataforma de tramitació o aplicacions satèl·lits per invocar serveis de signatura de forma senzilla i sense que haver d'entrar a les entranyes. L'objectiu és que els programadors i altres aplicacions podran invocar a mètodes d'alt nivell per utilitzar aquestes funcionalitats.

Aquest mòdul podrà anar adquirint noves funcionalitats i donant nous serveis dins de la plataforma ja sigui que sorgeixen noves necessitats, tecnologies o suports específics (BD, etc.). El ProveidorSignatura serà el que implementi el protocol DSS, un protocol estàndard que ens permetrà sol·licitar diversos serveis de Signatura a diversos proveïdors. Com a proveïdor inicial utilitzarem PSIS de CATCert perquè és un proveïdor que implementa el protocol DSS i proporciona molts serveis estàndard i estesos que no ho fan d'altres. Però en el cas que per tots o per algun servei en concret es vulgui canviar de proveïdor es podrà fer amb un impacte mínim sempre i quan aquest proveïdor implementi l'estàndard DSS.

L'arquitectura de l'aplicació es basarà en components EJBs de negoci, per al component ProveidorSignatura. En la part client s'implementaran la invocació als serveis de PSIS i la implementació de crides mitjançant el protocol estàndard DSS. Es realitzaran signatures de tipus CMS, Segell de Temps CMS i validacions d'aquestes signatures i de certificats X509v3.

PSIS és una plataforma basada també en Java, J2EE i en tecnologies Open Source, com ara Spring i Hibernate, que s'aprofita dels avantatges derivats de l'ús de les mateixes, com ara oferir el seu servei en un entorn fiable, escalable i transaccional, aconseguint d'aquesta manera convertir l'ús de la PKI en una *appliance* més que es dóna als clients com a un servei de caixa negra.

Aquesta plataforma presenta les funcionalitats abans esmentades com un servei remot que s'invoca mitjançant un WebService SOAP estàndard. D'aquesta manera, l'aplicació client, el component de ProveidorSignatura, es podrà integrar sempre que compleixi els requeriments mínims d'integració amb el WebService. Això ens permet un espectre de tecnologies compatibles molt més gran del que normalment permeten aquest tipus d'entorns PKI.

Aquest component utilitzarà el protocol DSS per comunicar-se amb el proveïdor de signatura PSIS. DSS és un protocol definit per OASIS que pretén estandarditzar la missatgeria per tal d'invocar serveis de signatura digital. Defineix un protocol de missatges en XML sobre SOAP estàndard que permet, mitjançant la creació d'uns missatges tipus, la validació o la creació de signatures i documents signats. A més defineix una estructura oberta anomenada OptionalInputs/OptionalOutputs per tal de poder definir paràmetres i sortides addicionals al procés de validació o creació, permetent una gran capacitat modular i unes grans possibilitats d'ampliació del protocol sense haver d'alterar la seva estructura bàsica.

De fet, el protocol defineix un nucli on es llisten i defineixen els processos i estructures bàsiques per a dur a terme la majoria de les funcionalitats més comunes derivades de la signatura digital. També permet, mitjançant el seu perfilat, la definició de nous comportaments i funcionalitats.

Aquests perfils possibiliten que, sense alterar cap estructura bàsica, es puguin construir una sèrie de funcionalitats adaptades a necessitats més concretes, però sense descuidar el joc mínim de funcionalitats que tot sistema que proporciona serveis de PKI a través de DSS ha d'oferir.

PSIS suporta el Core de DSS i, a més a més, diversos dels seus perfils.

- Defineix també el perfil XSS, que amplia el Core amb funcionalitats que permeten validar certificats X509 o extreure'n certa informació de les signatures o els certificats a validar entre d'altres. Aquest perfil serà clau al nostre projecte i l'utilitzarem per normalitzar diferents tipus de certificats i poder obtindré la mateixa informació de cada tipus de certificat i altre informació estesa, aquesta és una de les funcionalitats clau en el projecte per tal de poder identificar a l'usuari i la seva autenticitat (autentificació).

- Un altre extensió de CATCert que utilitzarem és un perfil per a PDF, que incorpora certes funcionalitats que, amb les mateixes estructures emprades per PSIS, permeten validar documents PDF signats i incorporen OptionalInputs/Outputs específics per a tractar amb aspectes derivats de la natura especial dels PDF's, com ara per exemple obtenir les raons de les signatures entre d'altres. Aquest perfil tam´be serà clau en el projecte, com ara per exemple obtenir les raons de les signatures entre d'altres.

- També, el perfil de segellat de temps que aporta restriccions addicionals a l'hora de crear i validar segells de temps tant en el seu format Xades com CMS. Aquest perfil l'utilitzarem per obtindré segell de temps.

- Defineix perfil de Xades que aporta regles de procés i estructures per a poder treballar amb signatures avançades Xades i Cades, permetent la seva actualització i compleció segons les formes definides. Aquest perfil l'utilitzarem per obtindré signatures avançades en el projecte.

- El perfil d'arxivat que defineix les regles i les estructures de missatgeria per a permetre l'arxivat de signatures definint les operacions típiques del cicle de vida i aspectes relatius a la configuració de la política d'arxivat corresponent. Aquest perfil no l'utilitzarem en l'àmbit d'aquest projecte.

Nosaltres ens centrarem en l'ús del protocol DSS (amb propietats directes), el perfil XSS i el de validació i normalització de certificats digitals, l'específic de tractament PDF i el de segell de temps.

El protocol DSS també permet la integració entre XMLDSig i CMS de forma transparent i interoperable.

La definició de les regles de procés i les estructures emprades tant al Core com als diferents perfils estan disponibles en la documentació de referència a la bibliografia (OASIS). Els perfils XSS i especifics de PSIS estan disponibles en la documentació de referència a la bibliografia (CATCert).

PSIS pot treballar fent servir un protocol de l'Autoritat de Validació de CATCert que permet validar signatures PKCS#7 (CMS) i certificats, a més d'extreure'n la seva informació associada de manera similar a la mostrada per XSS. Aquest protocol està basat en un WebService que funciona sobre SOAP i no és més que un altre punt d'entrada a PSIS, però que comparteix tota la lògica de validació amb el servei equivalent de DSS. De manera similar, PSIS també por parlar protocols no basats en SOAP, com ara per exemple el definit pel RFC 3161, que permet l'estampació de segells de temps CMS. Així podem fer que qualsevol sistema previ que contactes amb una TSA pugui demanar el mateix servei a PSIS sense cap tipus d'alteració en la seva lògica.

Per a la Plataforma Tecnològica que ens ocupa hem decidit no implementar el protocol de CATCert perquè encara que és un protocol històricament molt utilitzat, molt simple i fàcil d'integrar no és estàndard i permet un ventall de possibilitats molt limitades a més de no tenir projecció de futur (només implementa una petita part de les funcionalitats de PSIS). Per contra, el DSS és un protocol estàndard molt potent i dóna moltes possibilitats i queda obert a noves funcionalitats. L'altre protocol, el RFC 3161, és un protocol que treballa en binari i s'utilitzava molt fins ara perquè no hi havien altres possibilitats però és un protocol desfasat.

5.3.3.1 Composició modular del Proveïdor de Signatura.

Aquest component implementarà la integració amb PSIS i donarà per tant accés als seus serveis. Per això utilitzarà un servei basat en SOAP, com ara tots els basats en DSS.

La integració amb el servei web basat en DSS requereix la compilació d'un WSDL (Web Service Definition Language), aquest és el descriptor dels serveis que proporciona PSIS. A partir d'aquest descriptor obtindrem els *stubs* que ens donaran la infrastructura client necessària per a poder realitzar les crides als serveis PSIS de forma transparent (per a nosaltres com una crida a una API qualsevol, com si fos una crida a una llibreria local però en realitat s'encapsula amb SOAP i es realitza una crida remota), la URL del descriptor dels serveis de PSIS és http://psis.catcert.net/wsdl/dss.wsdl.

Per tal de poder compilar necessitem un client de WebServices que suporti invocacions amb model Document/Literal, donat que la invocació es fa mitjançant l'enviament de missatges DSS a un port SOAP concret i no fent una invocació a un mètode remot. Dels clients de WebServices que suportin aquesta característica podem fer servir o bé XFire o Axis 2.0, per a implementacions basades en Java, nosaltres escollirem XFire.

Per tant un cop implementat aquesta part client de la invocació a PSIS podem utilitzar tots els serveis comentats anteriorment enviant els missatges (format XML) DSS adequats en cada cas per a sol·licitar el servei requerit amb la resposta que ens interessi, és a dir, amb els missatges DSS parametritzem les crides i respostes dels serveis de PSIS. Existeixen unes guies per a la integració amb PSIS que contemplen pas a pas el procés de creació dels Stubs del client a partir del WSDL i aporten diversos exemples de com invocar un gran nombre de les funcionalitats de PSIS (veure Guia bàsica pels integradors de PSIS.pdf en l'apartat de referències).

Per a poder invocar l'Autoritat de Segellat de Temps, es pot fer servir qualsevol client que generi peticions en format RFC 3161. Aquí podem trobar diferents implementacions com ara les brindades per OpenSSL, Bouncy Castle i d'altres eines propietàries. Nosaltres utilitzarem el servei de PSIS per a demanar segell de temps de tipus RFC 3161 i ho manegarem posteriorment amb Bouncy Castle.

Com hem comentat a part d'utilitzar el protocol SOAP per la comunicació l'altre important estàndard que principalment utilitzarem són els missatges DSS. Aquests tenen un format definit a les especificacions d'OASIS (veure especificació del protocol al apartar de referències), nosaltres implementarem aquests missatges seguint aquests estàndards i amb les extensions segons les necessitats actuals de la Plataforma Tecnològica. Per a implementar el protocol DSS i la formació de missatges XML compatibles utilitzarem les llibreries d'OASIS combinades amb XmlBeans. Com es pot veure en aquest document el disseny i implementació es deixa preparat i obert per possibles ampliacions (apareixen nous perfils DSS, que poden incorporar-se, etc..), de moment les funcionalitats, atributs i extensions contemplades estan definides a aquest document (veure referències "Guia bàsica pels integradors de PSIS.pdf") per a conèixer amb més profunditat el format DSS (missatges, profiles, etc.) veure les referències donades.

El component de ProveidorSignatura estarà format per subcomponents o paquets que formaran una gran aplicació (enterprise o EAR) i que implementaran la lògica de negoci necessària per respondre a les necessitats de l'actor (appl. externa) segons els seus casos d'us. S'encarregarà de resoldre els casos d'us de l'actor appl. externa: validar certificat, normalització d'atributs, validar signatura, segell de temps, consultar documents signats i validar-los en un moment donat de temps. Aquest component donarà una interfície per accedir a les seves funcionalitats **IProveidor**. A aquesta interfície només podran accedir les aplicacions amb els permisos necessaris.

Els paquets que formaran l'aplicació (EAR):

- **ProveidorSignaturaCore**: Aquest paquet és la llibreria que conté tota la lògica de negoci implementada, és qui realitza les crides a les funcionalitats de PSIS a través de la implementació DSS mitjançant XML i comunicació a través de webservices i SOAP. Aquest component conté les classes de configuració, els objectes Vo a utilitzar i els mètodes principals així com les llibreries i classes d'ajuda (utilities).

  Aquest component es pot utilitzar com a llibreria independent .jar. Com per exemple afegir la llibreria en un altre projecte i utilitzar les seves funcions públiques.

  Per exemple, en el cas de la plataforma de tramitació amb que està relacionat aquest projecte de signatura, aquest component com a llibreria ProveidorSignaturaCore.jar s'utilitzarà en el mòdul de SSO. Servirà per realitzar amb l'Open SSO una autentificació amb certificats digitals de l'usuari en la plataforma de tramitació. De manera que l'usuari es connecta amb certificats digitals s'agafa les seves credencials i s'envien a través d'aquesta llibreria a validar el certificat i normalitzar l'atribut del NIF, la classe "Validacio" tornarà si és vàlid el certificat i quin és el NIF normalitzat del l'usuari en qüestió, d'aquesta forma se li donarà accés a l'usuari amb les credencials que aquest hagi de tenir en la plataforma de contractació. Aquest usuari quan arribi al Porta Signatures, de forma

automàtica, només se li permetrà signar amb el mateix certificat de l'usuari autenticat en la plataforma (via les credencials recollides en el SSO).

- **ProveidorSignaturaEJB**: Aquest component s'implementarà amb un EJBs de sessió ProveidorSignaturaEJB seguint per tant el patró facade (session-facade: aconseguim desacoblar la capa del model de la capa de negoci cadascú fa la seva feina/responsabilitat i reduïm notablement el trànsit de xarxa). Aquest component permetrà el desplegament del component entrerprise ProveidorSignatura en un servidor d'aplicacions de forma que les seves funcionalitats públiques estiguin disponibles per realitzar peticions des de qualsevol punt de la plataforma de tramitació de forma distribuïda.
- **ProveidorSignaturaEJBClient**: Interfície del component per a lliurar als programadors o programari que vulgui utilitzar el nostre component de ProveidorSignatura.
- **ProveidorSignaturaCoreTest**: Aquest paquet ens permetrà provar totes les funcionalitats i casos d'us del component de ProveidorSignatura. **Aquest punt forma part de la Fase IV: Verificació de la construcció i lliurament.**
- **ProveidorSignaturaEAR**: Aquest paquet ens permetrà empaquetar tots els paquets en un de sol com a únic recurs distribuïble i usable.

5.3.3.2 Composició Missatges DSS:

En aquest apartat veurem les estructures principals del protocol DSS que utilitzarem en el projecte. El protocol DSS (Digital Signature Services), del consorci d'estandardització OASIS (Organization for the Advancement of Structured Information Standards), és un protocol per a la prestació de serveis de signatura digital obert i extensible (mitjançant l'ús de perfils)

Aquestes estructures del DSS ens permetran sol·licitar la primitiva PKI que ens interessi a través de la creació de missatge XML i enviant la petició al serveis de PSIS mitjançant protocol SOAP (Simple Object Access Protocol).

SOAP és un protocol estàndard sobre el qual es fonamenta la tecnologia de serveis web (Web Services). SOAP es basa en documents de text pla codificats en format XML. L'avantatge principal de codificar en XML és que els missatges són llegibles per éssers humans; però, per contra, aquests documents resultants són, en general, de tamany gran.

SOAP està dissenyat per a funcionar sobre qualsevol protocol d'Internet, tot i que l'ús més habitual és sobre HTTP. El fet d'utilitzar HTTP minimitza l'impacte de dispositius com Firewalls i similars, i fa accessible SOAP a pràcticament qualsevol tipologia de comunicació client-servidor.

En els namespaces (els espais de noms dels XML) els prefixos que permet l'estàndard DSS són:

- **xd**: http://www.w3.org/2000/09/XMLDsig# -> permet utilitzar les estructures XMLDsig del (W3C)
- **dss** : urn:OASIS:names:tc:dss:1.0:core:schema -> permet utilitzar les estructures del CoreDSS -> Asynchronous Processing Abstract Profile of the OASIS Digital Signature.
- **xss** : urn:OASIS:names:tc:dss:1.0:profiles:XSS -> permet utilitzar les estructures de les extensions del protocol DSS, PSIS ha creat l'extensió del DSS, el perfil XSS. Ampliació de

DSS que permet, entre d'altres validar certificats X509 de clau púbica, extreure informació dels mateixos i fer servir polítiques de signatura.

- **pdf**: urn:OASIS:names:tc:dss:1.0:profiles:DSS_PDF-> perfil que permet utilitzar les estructures de les extensions del protocol DSS, PSIS implementa el perfil DSS_PDF per la signatura de .pdf
- **timestamp (CMS)**: urn:oasis:names:tc:dss:1.0:profiles:timestamping (urn:ietf:rfc:3161)-> perfil que permet utilitzar les estructures de les extensions del protocol DSS, PSIS implementa el perfil timestamping per la realització de segells de temps.

Els perfils de DSS son extensions del protocol DSS, el core d'aquest ens permet les funcionalitats globals de Signatura Digital, però els perfils ens aporten noves funcionalitats o ampliacions de las del nucli (core). CATCert ha desenvolupat els perfils estesos XSS, PDF, i timestamp. La documentació detallada amb les descripcions de les estructures i elements que formen part de la missatgeria DSS i les extensions de CATCert amb les particularitats dels perfils de PSIS la podem trobar a l'apartat de referències i bibliografia d'aquesta memòria.

DSS permet també l'ús de documents digerits (l'usuari aplica una funció de digestió a la banda client) que permet que els documents no hagin de viatjar a la plataforma, evitant així el cost de pujada dels mateixos i protegint la seva privacitat ja que no abandonen mai el client i només hi viatja un petit resum criptogràfic.

En el cas dels documents XML es poden aportar també un tercer tipus de documents anomenats transformats. Aquests han passat per una sèrie de transformacions (com de c14n,XPath o qualsevol de les definides a XMLDsig) per tal que les dades validades hagin estat ja pre-processades per l'entitat client.

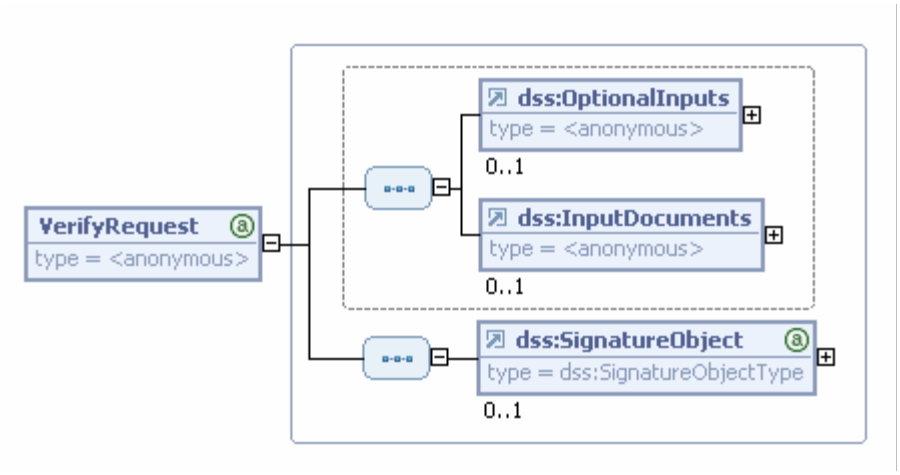
Finalment hi trobem l'estructura OptionalInputs. Aquesta estructura és una estructura oberta i sense una cardinalitat definida on podem encabir els diferents paràmetres i opcions que poden modificar el procés de validació. El Core de DSS defineix els OptionalInputs més comuns aplicats en el procés de validació així com les seves regles de procés associades. Els diferents perfils defineixen també diferents inputs per tal de poder donar resposta a noves funcionalitats sense haver de modificar l'estructura base, perquè en definitiva el que es vol validar sempre és el mateix (signatures i elements signats) i només volem customitzar el procés concret de validació.

Finalment, un altre punt important és l'atribut de l'element VerifyRequest anomenat profile. Aquest atribut contindrà una URL que identificarà al servidor segons quin perfil s'ha de processar la petició (la mateixa petició serà processada seguint diferents esquemes de procés depenent del valor d'aquest atribut). Si aquest element no és aportat per l'usuari, el servidor assumirà el perfil que ell consideri per defecte.

En aquest apartat aprofundirem en l'estructura de les peticions XML definides per DSS. Utilitzarem dos objectes principals arrel (root), VerifyRequest per a validacions i SignRequest per a signatures. Amb aquestes estructures podrem crear els XML de petició de servei a PSIS i rebre la seva resposta de servei.

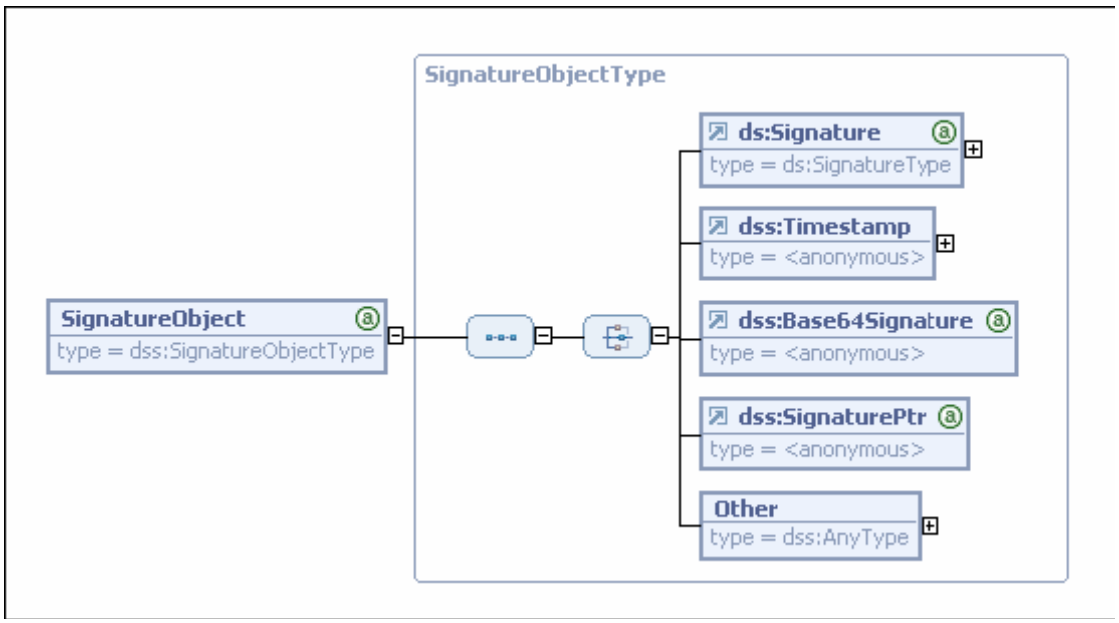
a Missatges DSS de validació (VerifyRequest):

En aquest apartat ens centrarem en l'estructura de les **peticions de validació** definides per DSS. Aquestes validacions es creen fent servir un tipus de missatge anomenat **VerifyRequest**. Estructura de la VerifyRequest DSS:



Aquesta petició està formada per tres grans tags on aniran els paràmetres de la validació així com els elements de confiança a validar. A l'element **VerifyRequest.SignatureObject** inclourem l'element de confiança a validar. Aquest element pot ser del tipus signatura XML (on podem encabir signatures XMLDsig o Xades) del tipus CMS (on podem trobar signatures CMS o Cades), segells de temps en format CMS o Xades i certificats X509 de clau pública a l'element Other.

En el nostre cas utilitzarem els següents tipus del **SignatureObject**:



Validar Certificats: Per validar els certificats digitals hem d'utilitzar la següent estructura VerifyRequest/SignatureObject/**Other**/X509Certificate. Utilitzem l'estructura *Other* perquè els certificats no són de l'estàndard DSS, aquest només contempla signatures i validacions de signatures, el tag Other del protocol DSS és el que ens permet futures ampliacions.

Validar Signatures: Per validar les signatures digitals hem d'utilitzar la següent estructura VerifyRequest/SignatureObject/**Base64Signature**. Utilitzarem aquest que és el format per a validar signatures CMS, en cas que volguéssim validar signatures XMLDsig utilitzaríem ds:Signature, si en un futur es vol ampliar tant els nostres components com el protocol DSS hi està preparat.

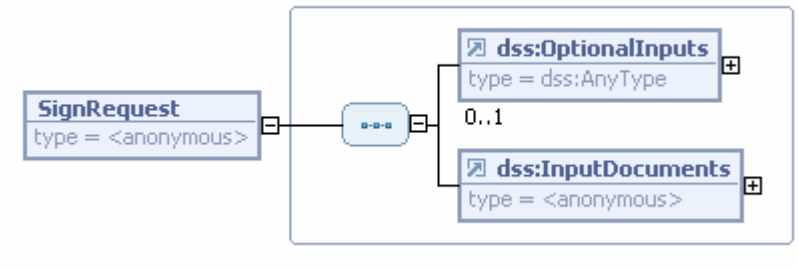
Validar Segell de Temps: Per validar els segells de temps hem d'utilitzar la següent estructura VerifyRequest/SignatureObject/**Timestamp**/RFC3161TimeStampToken (validarem un segell tipus RFC3161).

Validar Documents: Al camp **VerifyRequest**.**InputDocuments** inclourem el document o documents a validar. Aquests poden incloure les signatures a validar o poden estar apuntats per aquestes. Els documents poden ésser fitxers XML que poden estar passats com a un element fill de l'element InputDocuments, codificats en Base64, o bé inclosos com a text escapant els caràcters especials de XML. Com en el nostre cas podem també aportar documents binaris codificats en Base64 (Base64Data).

Nosaltres però, ens centrarem en l'ús principalment d'un nou perfil fet específicament per al cas de **validació de documents pdf** (*Profile="urn:oasis:names:tc:dss:1.0:profiles:DSS_PDF"*). Donat que és un format àmpliament utilitzat s'ha creat un perfil concret per aquest cas, ens fa la validació de la signatura tipus attached d'un document pdf i permet extreure informació de signatura. En aquest perfil haurem d'afegir simplement el document pdf signat en Base64 (VerifyRequest.InputDocuments.**Document.Base64Data**) i indicant el perfil i els atributs extres que volem obtenir (això ho podem configurar a través de l'element OptionalInputs) podrem validar la signatura (podrem veure l'ús dels atributs i operacions opcionals en l'apartat següent de "Constants Configuració"). Els elements optionalInputs i optionalOutputs seran els que ens permetran parametritzar les crides als diferents serveis fent que aquests es comportin d'una manera o un altra, demanant una informació o altre.

b Missatges DSS de signatura (SignRequest):

En aquest apartat ens centrarem en l'estructura de les peticions de signatura definides per DSS. Aquestes validacions es creen fent servir un tipus de missatge anomenat **SignRequest**, aquest funciona de manera anàloga al de verificació (VerifyRequest). Estructura de la SignRequest DSS:



La funcionalitat de signatura ens permet realitzar signatures o segell de temps (donat que aquest és una signatura). Fa servir input documents per enviar el document o documents a signar o els seus resums (hash).

La resta del procés es configura mitjançant OptionalInputs i les respostes mitjançant els seus corresponents outputs.

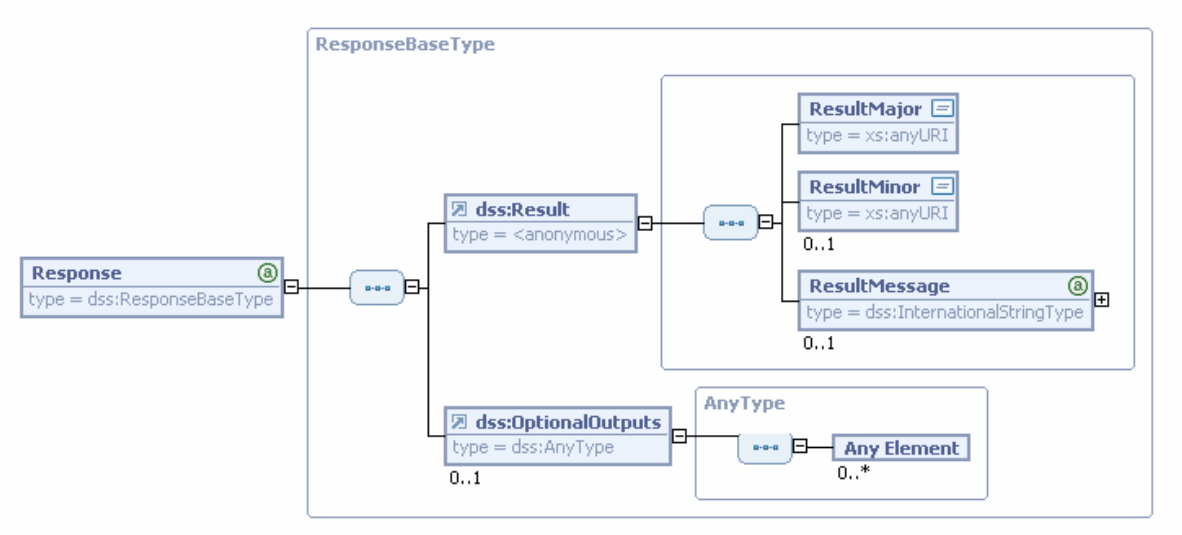
Per indicar quina TSA vols que et realitzi el segell de temps s'ha de passar el certificat de la TSA per l'optional Inputs (optionalInputs.addNewKeySelector().addNewKeyInfo().addNewX509Data()).

Un segell de temps el podem definir com *SignatureType* en els atributs de configuració de la petició del servei que podem veure en l'apartat següent "Constants de Configuració".

- Timestamp (CMS): urn:ietf:rfc:3161
- Timestamp (XMLDsig): oasis:names:tc:dss:1.0:core:schema:XMLTimeStampToken
- Timestamp (XAdES): oasis:names:tc:dss:1.0:core:schema:XAdESTimeStampToken

c Missatges DSS de resposta (Response):

Davant un petició com les mostrades, el servidor ens retornarà un missatge de resposta que segueix el següent esquema definit.



L'estructura de la resposta s'articula en dos gran blocs:

Result: El primer és l'element Result on es retorna el resultat de la petició. Aquest està format per tres camps.

- **ResultMajor:** El primer o Major dóna informació general sobre el procés. Aquí ens pot dir si el resultat del processat ha estat correcte o, si pel contrari, hi ha hagut problemes i no s'ha pogut donar servei a la petició ja sigui per causes derivades d'un mal ús per part del client o per un mal funcionament del servei.
- **ResultMinor:** El segon o Minor ens dona amb més detall el resultat del procés. En cas d'una fallada ens diu la causa i en cas d'una petició que ha estat processada correctament ens diu el resultat de la mateixa.

- **ResultMessage:** Finalment el missatge ens dóna un detall textual sobre el resultat de la invocació dels clients de manera que un usuari humà pugui entendre amb més detall el resultat de la invocació.

DSS defineix al seu Core un conjunt de Majors i Minors que donen resposta a les diferents peticions definides a aquest document. Els diferents perfils defineixen nous codis per tal de poder donar resposta a les noves funcionalitats que s'hi defineixen i hi conserven els definits al Core per a les funcionalitats comuns o n'hi amplien la seva semàntica per al context concret en el qual s'hi apliquen.

Ara definirem els principals Majors i Minors definits per DSS. La llista complerta així com la seva definició normativa i regles de procés estan disponibles als diferents documents de definició dels mateixos.

urn:oasis:names:tc:dss:1.0:resultmajor: + [valor]

Valor	Tipus	Descripció
Success	OK	Protocol complert correctament
RequesterError	Error	El missatge que conté la petició del client és incorrecte, ja sigui sintàcticament o semànticament
ResponderError	Error	La petició no s'ha pogut completar per un error al servidor

El catàleg d'atributs i de codificacions d'entrada o de resposta com els de ResultMinor es poden veure al document de referència "Guia bàsica pels integradors de PSIS.pdf.

OptionalOutputs: Aquesta estructura ens permet retornar tots els elements addicionals demanats en els OptionalImputs, com per exemple en el cas de la normalització d'atributs si sol·licitem el NIF (<xss:AttributeDesignator Name="urn:catcert:psis:certificateAttributes:KeyOwnerNIF"/>) del propietari del certificat digital ens retornarà el seu valor.

d Exemple: missatge DSS de validació de certificat x509 i normalització d'atributs contra PSIS.

Per tal d'anar coneixent aquestes tecnologies comentades i com les podem utilitzar per realitzar les funcionalitats descrites, ara veurem un exemple de com crear un missatge XML complint el protocol DSS per enviar una petició al servei de validació de certificats X509 de PSIS. Aquesta és una de les funcionalitats claus del projecte donat el problema que hi ha per validar certificats diferents de forma homogènia, certificats emesos per diferents entitats certificadores.

Tots aquests certificats segueixen l'estàndard X509 v3 però cada entitat certificadora guarda els atributs personals en una estructura diferent estil LDAP (CN=, DN, càrrec, etc.) per això aquest servei de PSIS ens permet validar l'autenticació de la persona y la validesa del certificat i les seves claus asimètriques i al mateix temps recuperar de forma normalitzada les diverses informacions de cada tipus de certificat com el NIF.

Podem veure com es forma un missatge dss de VerifyRequest.
Podem veure que els namespaces utilitzen els prefixes de dss i xss, d'aquesta forma podem utilitzar les estructures esteses de CATCert per a sol·licitar informació addicional (tag dss:ReturnProcessingDetails) com la normalització de certs atributs del certificat, com per exemple el NIF.

```
<dss:VerifyRequest Profile="urn:oasis:names:tc:dss:1.0:profiles:XSS" xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema"
xmlns:xd="http://www.w3.org/2000/09/xmldsig#" xmlns:xss="urn:oasis:names:tc:dss:1.0:profiles:XSS">
<dss:OptionalInputs>
<dss:ReturnProcessingDetails/>
        <xss:ReturnX509CertificateInfo>
        <xss:AttributeDesignator Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:Version" />
        <xss:AttributeDesignator Name="urn:catcert:psis:certificateAttributes:KeyOwnerNIF" />
        <xss:AttributeDesignator Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SerialNumber" />
        <xss:AttributeDesignator Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:Signature" />
        <xss:AttributeDesignator Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SignatureAlgorithm" />
        <xss:AttributeDesignator Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:IssuerDistinguishedName" />
        <xss:AttributeDesignator Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SubjectDistinguishedName" />
        <xss:AttributeDesignator
        Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SubjectDistinguishedName:commonName" />
        <xss:AttributeDesignator
        Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SubjectDistinguishedName:givenName" />
        <xss:AttributeDesignator
        Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SubjectDistinguishedName:surname" />
        <xss:AttributeDesignator
        Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SubjectDistinguishedName:organizationName"/>
        <xss:AttributeDesignator
        Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SubjectDistinguishedName:serialNumber"/>
        <xss:AttributeDesignator Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:NotAfter" />
        <xss:AttributeDesignator Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:NotBefore" />
        <xss:AttributeDesignator Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SubjectPublicKeyAlgorithm" />
        <xss:AttributeDesignator Name="urn:o
        </xss:ReturnX509CertificateInfo>
</dss:OptionalInputs>
<dss:SignatureObject>
<dss:Other>
        <xd:X509Data>
        <xd:X509Certificate>MIIHxTCCBq2gAwIBAgIQOiZLlU8OHRFCG0+XhWb/RjANBgkqhkiG9w0BAQUFADCCASYxCzAJBgNVB
AYTAkVTMTswOQYDVQQKEzJBZ2VuY2lhIENhdGFsYW5hIGRlIENlcnRpZmljYWNpbyAoTklGIFEtMDgwMTE3Ni1JKTE0MDIGA1UEBx
MrUGFzc2F0Z2UgZGUgbGEgQ29uY2VwY2lvIDExIDA4MDA4IEJhcmNlbG9uYTEuMCwGA1UECxMlU2VydmVpcyBQdWJsaWNzIGRlIE
NlcnRpZmljYWNpbyBFQ1YtMjE2MDQGA1UECxMtVmVnZXUgaHR0cHM6Ly93d3cuY2F0Y2VydC5uZXQvdmVyQ0lDLTIgIChjKTAzMSw
wKgYDVQQLEyNBZG1pbmlzdHJhY2lvbnMgTG9jYWxzIGRlIENhdGFsdW55YTEOMAwGA1UEAxMFRUMtQUwwHhcNMDUwMjIyMTUy
ODIzWhcNMDkwMjIyMTUyNzQyWjCCASoxCzAJBgNVBAYTAkVTMSQwIgYDVQQKFBtELkcuIEluZm9ybeF0aWNhIFRyaWJ1dGFyaWE
xNzA1BgNVBAsTLlZlZ2V1IGh0dHBzOi8vd3d3LmNhdGNlcnQubmV0L3ZlckNQSVNSLTEgYygwMykxM ==</xd:X509Certificate>
        </xd:X509Data>
</dss:Other>
</dss:SignatureObject>
</dss:VerifyRequest>
```

A banda d'afegir el node els inputs opcionals (dss:OptionalInputs) indicant que utilitzem l'extensió de CATCert per la normalització d'atributs, també hem de passar el contingut del certificat en si codificat en Base64 això seguint els passos indicats en els apartats anteriors sobre la formació de missatges DSS o inclourem en el node (dss:other\xd:X509Data\Xd:X509Certificate)

1) Podem veure el resultat de la petició de validació a PSIS en el missatge DSS de resposta VerifyResponseDocument.
2) En el detall del missatge de resposta podem veure que el certificat és vàlid això s'emmagatzemarà al DigitalSignatureContentVo HashMap key ResultMinor

INFO: <dss:VerifyResponse Profile="urn:oasis:names:tc:dss:1.0:profiles:XSS" xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<dss:Result>
<dss:ResultMajor>urn:oasis:names:tc:dss:1.0:resultmajor:Success</dss:ResultMajor>
<dss:ResultMinor>urn:oasis:names:tc:dss:1.0:profiles:XSS:resultminor:valid:certificate:Definitive</dss:ResultMinor>
</dss:Result>
<dss:OptionalOutputs>
<dss:ProcessingDetails>
<dss:ValidDetail Type="urn:oasis:names:tc:dss:1.0:detail:ValidityInterval">
<dss:Message xml:lang="en">The signing key is inside its static validity interval.</dss:Message>
</dss:ValidDetail>
<dss:ValidDetail Type="urn:oasis:names:tc:dss:1.0:detail:IssuerTrust">
<dss:Message xml:lang="en">The issuer of the given key is trusted.</dss:Message>
</dss:ValidDetail>
<dss:ValidDetail Type="urn:oasis:names:tc:dss:1.0:detail:RevocationStatus">
<dss:Message xml:lang="en">The signing key is not revoked.</dss:Message>
</dss:ValidDetail>
</dss:ProcessingDetails>
<urn:X509CertificateInfo xmlns:urn="urn:oasis:names:tc:dss:1.0:profiles:XSS">
<urn:Attribute Name="urn:catcert:psis:certificateAttributes:KeyOwnerNIF">
<urn1:AttributeValue xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">46358509P</urn1:AttributeValue>
</urn:Attribute>
<urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:Version">
<urn1:AttributeValue xsi:type="xs:integer" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">3</urn1:AttributeValue>
</urn:Attribute>
<urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SerialNumber">
<urn1:AttributeValue xsi:type="xs:integer" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">122750</urn1:AttributeValue>
</urn:Attribute>
<urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:Signature">
<urn1:AttributeValue xsi:type="xs:base64Binary" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">driSCWODTvIDsqfyI8PQzYsLyL+IoKORcOfTmfDNw47kY4JFyj4jnwLTz/POrinaaWppmLlU41eGj1PZMUqwSe4bha9dFOzbjpKAuOQxFKWvd
w8IapgkCCW/87thXsadVXBo1RtkkC9B7S0TbuHDk/srK2n9p31p18TPYyveOQLJfOMKstqtfWvj5a0gx3EDjp+diWWbh5PYnMMXtGzhtKfEswivJA3Sb8C9
6Qq4erOC5is+N/aLJJJ2IVDlCZhgH1/4NPzkppWSN3eCspYWuHLExbnD6cZr+2WnxuDiz6JAf+X5ZLtIzGj+lG/DcekTMdnWQOLq1Fve5+qd9Rrnag==</ur
n1:AttributeValue>
</urn:Attribute>
<urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SignatureAlgorithm">
<urn1:AttributeValue xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">1.2.840.113549.1.1.5</urn1:AttributeValue>
</urn:Attribute>
<urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:IssuerDistinguishedName">
<urn1:AttributeValue xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">CN=EC-IDCat,OU=Entitat publica de certificacio de ciutadans,OU=Vegeu
https://www.catcert.net/verCIC-2 (c)03,OU=Serveis Publics de Certificacio ECV-2,L=Passatge de la Concepcio 11 08008 Barcelona,O=Agencia Catalana
de Certificacio (NIF Q-0801176-I),C=ES</urn1:AttributeValue>
</urn:Attribute>
<urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SubjectDistinguishedName:givenName">
<urn1:AttributeValue xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">CARLOS</urn1:AttributeValue>
</urn:Attribute>
<urn:Attribute Name="Urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SubjectDistinguishedName:surname"/>
<urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SubjectPublicKeyAlgorithm">
<urn1:AttributeValue xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">RSA</urn1:AttributeValue>
</urn:Attribute>
<urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SubjectPublicKey">
<urn1:AttributeValue xsi:type="xs:base64Binary" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQD5tCEfHWjfu05hLQh2zSOvHIywYSNYbJ5mUL0iDGp36aj+fOoYPixYYSINJwHQfvRE
ZNtvdoz895J79oROsOIVJ4KjygQmIi6YomSp/UYaUizHuGgKr+ZHwLdCViNrWhTBWwx7rCdYI0zgJxVC2uXqHpCrCM06UvIxHqdjvTErVwIDAQAB</urn1:
AttributeValue>
</urn:Attribute>
<urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SubjectDistinguishedName:organizationName"/>
</urn:X509CertificateInfo>
</dss:OptionalOutputs>
</dss:VerifyResponse>

En aquest missatge de resposta podem veure l'estructura XSS X509CertificateInfo on ens retornarà tots els atributs del certificat que hem demanat normalitzats i aquests els inclourem en el hash d'atributs de resposta DigitalSignatureContentVo.atributs.get(ApplicationConfig.NIF) per a que hom que utilitzi la nostra api pugui obtenir fàcilment aquesta informació.

5.3.3.3 Constants configuració:

Aquestes constants i operacions vindran recollides a una classe de constants que juntament amb els value objects i les interfícies dels serveis de Signatura estaran incloses al paquet de client (EJBClient) del component ProveidorSignatura. Qui vulgui utilitzar aquests serveis tindrà que importar exclusivament aquest paquet client i podrà gaudir de les interfícies i classes clients per poder invocar als mètodes de signatura de l'EJB remot (ProveidorSignaturaBean) amb la lògica de negoci de forma transparent per qui el crida.

La clau de la classe que contingui les constants de configuració és que el programador vegi simplificat l'entorn de funcionalitats de signatura poden fàcilment invocar els serveis acotats amb paràmetres predefinits i més simples.

Atributs de Certificat

NOM ATRIBUT (VO)	PER DEFECTE	NOM ESTÀNDARD DSS O PROFILE XSS O CATCERT	DESCRIPCIÓ
NIF	Si	urn:catcert:psis:certificateAttributes:KeyOwnerNIF	Nif del propietari del certificat.
versio	Si	urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:Version	Versió del certificat
UIDCertificat	Si	urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SerialNumber urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SubjectDistinguishedName:serialNumber	Número de serie del certificat.
signaturaCA	Si	urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:Signature	Signatura de la CA emissora
algoritmeSignaturaCertificat	Si	urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SignatureAlgorithm	Algoritme de la signatura
nomEmissor	Si	urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:IssuerDistinguishedName	Nom de l'emissor CA del certificat.
nomDePilaPersona	No	Urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SubjectDistinguishedName	Nom de pila de la persona.

		:commónName	
nomPersona	Si	urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SubjectDistinguishedName :givenName	Nom de la persona.
cognomPersona	Si	Urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SubjectDistinguishedName :surname	Cognom de la persona.
algoritmeClauPublica	Si	urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SubjectPublicKeyAlgorithm	Algoritme de generació de la clau pública.
clauPublica	Si	urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SubjectPublicKey	Clau Publica.
organitzacio	Si	urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SubjectDistinguishedName :organizationName	Nom de l'organització de la que forma part el subjecte del certificat
dataIniciCertificat	No	urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:NotBefore	Data d'inici de validesa del certificat
dataFiCertificat	No	urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:NotAfter	Data fi de validesa del certificat

Atributs de Signatura

NOM ATRIBUT (VO)	PER DEFECTE	NOM ESTANDARD DSS O PROFILE XXS O CATCERT	DESCRIPCIÓ
funcioResumAlgoritme	Si	urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:DigestAlgorithm	Algoritme utilitzat per generar el valor de "Digest"
algoritmeEncriptacioResum	Si	urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:DigestEncryptionAlgorithm	Algoritme d'encriptació aplicat al digest per generar la signatura.
algoritmeSignatura	Si	urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:SignatureAlgorithm	Algoritme utilitzat per generar la signatura.
signatura	Si	urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:SignatureValue	Valor de signatura.

Atributs Pdf

NOM ATRIBUT (VO)	PER DEFECTE	NOM ESTANDARD DSS O PROFILE XXS O CATCERT	DESCRIPCIÓ
horaSignaturaPdf	No	ReturnSigningTime	Retornar l'hora de signatura.
signant	No	ReturnSignerIdentity	La identitat del signant.
raoSignatura	No	ReturnSignatureReason	La raó de la signatura.

Atributs de Segell

NOM ATRIBUT (VO)	PER DEFECTE	NOM ESTANDARD DSS O PROFILE XXS O CATCERT	DESCRIPCIÓ
tipusSignatura	No	signatureTypeCMS	Indica el tipus de signatura que volem, per exemple un segell de temps CMS (urn:ietf:rfc:3161). O bé un segell de tipus Xades (oasis:names:tc:dss:1.0:core:schema:XAdESTimeStampToken).

Atributs de Protocol

NOM ATRIBUT (VO)	PER DEFECTE	NOM ESTANDARD DSS O PROFILE XXS O CATCERT	DESCRIPCIÓ
perfilDSS	Si	El de defecte de DSS o altres perfils (urn:oasis:names:tc:dss:1.0:profiles:XSS)	Perfils d'implementació de funcions

Operacions: Aquestes seran suboperacions possibles dintre de les principals com validar certificat, validar signatura o signar. Hi hauran operacions que s'han considerat per defecte i obligatòries per l'abast de la plataforma i d'altres opcionals. Aquestes operacions són operacions estàndards definides al Core de DSS o als diferents perfils que han sorgit i que PSIS implementa (XSS, Timestamp...), d'aquests perfils i del Core. PSIS implementa la gran majoria, definirem els més importants així com la seva semàntica. Els que estan en verd són els que s'utilitzaran de moment a la Plataforma de Signatura. Hi ha moltes més operacions possibles però que surten de l'abast del projecte i en qualsevol cas es poden afegir posteriorment (veure document de DSS i profiles). Passem a mostrar una taula dels més interessants i que quedaran recollits al ApplicationConfig de l'aplicació. Si estan prefixats amb dss indiquem que aquest Input està definit al document Core, xss vol dir que està definit al perfil XSS.

Nom Operació	Per Defecte	Nom estàndard DSS o profile XSS o Xades	Descripció
politicaSignatura	No	xss:SignaturePolicy	Realitza la signatura segons unes polítiques en concret definides. Insta al servidor a fer la validació fent servir els criteris contemplats a una política de signatura. Aquest input és el que dóna el caràcter semàntic a l'Autoritat de Validació, ja que el resultat de la mateixa validació fent servir una política o una altra pot diferir donat que els criteris que hi intervenen no són els merament sintàctics. Si es defineixen polítiques segons el tipus de document aquella validació es farà sobre la política concreta amb les restriccions concretes.
validacioAData	No	dss:VerificationTime	Indica l'instant temporal en el qual es durà a terme la validació. És especialment útil quan validem certificats o bé no tenim indicat a la signatura el temps en el qual es va crear ja que permet determinar un instant temporal passat en el qual volem efectuar la validació.
detallsProces	Si	dss:ReturnProcessingDetails	Demana al servidor que doni un detall addicional sobre el procés de validació. Retorna detall sobre els diferents passos duts a terme pel servidor així com el resultat dels mateixos de manera que el client pot determinar amb més detall la causa del resultat retornat.
horaSignatura	Si	dss:ReturnSigningTime	Demana al servidor que retorni el temps en el qual es va dur a terme la signatura. En cas que sigui al·legat com a un atribut signat dintre de la signatura o bé certificat per l'existència d'un segell de temps sobre la signatura hi esmentarà la natura del mateix i quin és el seu valor.
mostrarAtributsSignatura	Si	xss:ReturnSignatureInfo	Demana al servidor que retorni certa informació sobre la signatura. Aquesta pot anar des del valor del resum criptogràfic emprat, l'algorisme amb el (Veure taula atributs de signatura) qual es va signar o el tipus i tamany de les claus. Els diferents valor definits així com la seva definició es troben definits al doc del perfil de XSS.

mostrarAtributsCertificat	Si	xss:ReturnX509CertificateInfo	Aquest input és similar a l'anterior però demana informació sobre el certificat a validar (en cas que validem un certificat) o sobre el certificat amb el que es va generar la signatura (en cas de validar signatures). La definició dels diferents atributs que es poden demanar estan contemplats a la definició del perfil XSS. (Veure taula atributs de certificat)
informacioAddicional	No	dss:AdditionalKeyInfo	Proporciona CA's intermèdies, així com informació de revocació addicional que pot ésser útil en la validació.
certificatQualificat	No	xss:RequireQualifiedCertificate	Requereix que els certificats d'entitat final involucrats a la validació siguin qualificats per a ser vàlids.
politica	No	dss:ServicePolicy	Requereix una política de servei concreta. Si el servidor que la rep no la coneix o no pot servir d'acord amb aquesta ha de rebutjar la petició indicant que no la suporta.
opcionsValidacioCertificat	No	xss:X509CertificateValidationOptions	Aporta modificacions sobre el procés de validació dels certificats de la signatura (o del certificat a validar en cas de validacions de certificats). Això permet configurar arrels de confiança específiques per al procés i per aquestes donar restriccions sobre les característiques del camí de certificació (llargada, restriccions sobre polítiques de certificació permeses o noms d'entitats permesos) així com la majoria de les restriccions contemplades el RFC de Certificate Path Building.
signarResposta	No	xss:ReturnSignedResponse	Demana al servidor que signi la seva resposta per tal de protegir la integritat de la mateixa i poder donar al client garantia d'autenticitat. Un cop rebuda, el client haurà de validar la resposta signada fent servir els seus propis mitjans i la signatura que protegeix aquesta resposta és de format Xades.
transformarDocument	No	dss:ReturnTransformedDocument	Demana al servidor que retorni els documents a validar després d'haver-hi aplicat totes les transformacions necessàries. Això només es pot demanar en cas que estiguem validant-hi signatures XML i permet obtenir el conjunt de dades a validar justament abans d'aplicar-hi 'algorisme de resum criptogràfic.
completarADES	No	dss:ReturnUpdatedSignature	Completa signatures avançades ADES (Cades, Xades). Retorna la signatura actualitzada a una forma en concret. Això només aplica en signatures avançades (ADES) i permet definir la forma concreta a la que s'actualitzarà la signatura afegint els atributs que siguin necessaris en cada cas.
schemas	No	dss:Schemas	Aporta una sèrie d'esquemes per a la validació sintàctica dels documents a validar. Només aplica en cas que els documents a validar siguin XML i el servidor els validarà contra l'esquema indicat abans de començar a processar la petició. Rebutjarà la petició en cas que la validació no sigui satisfactòria

5.3.3.4 Els VOs.

El següent apartat descriu els objectes de transport de dades que tenen una estructura comuna més algun atribut específic per a resoldre la seva funcionalitat dintre de la Plataforma de Signatura, és a dir, que conté la informació bàsica necessària per realitzar totes les funcionalitats relacionades amb la Signatura Digital (validació certificats, creació/ validació de segell de temps, creació/validació de signatures, etc.). Tots els value objects tindran el mateix contingut bàsic però tindran algun atribut específic per realitzar la seva funcionalitat en concret. D'aquesta forma l'intercanvi entre la Plataforma Tecnològica sobretot el component de PortaSignatures amb el ProveidorSignatura sempre serà amb objectes VO específics X509CertificateVo, Base64DataVo, Base64DataVo, perquè poden aportar informació addicional a tractar al mòdul de ProveidorSignatura.

- **X509CertificateVo:**
  - X509Certificate: El contingut del Certificat de l'usuari en base64.
- **Base64SignatureVo:**
  - Base64Signature: El contingut de la signatura de tipus CMS.
- **Base64DataVo:**
  - Base64Data: El contingut del document pdf amb signatura CMS.
- **InlineXMLVo:**
  - InlineXML: El contingut d'una signatura XML detached.
- **DocumentHashVo:**
  - DocumentHash: El resum del document a signar (o posar segell de temps).
  - X509Certificate: El contingut del Certificat de l'usuari en base64 o bé en el cas d'una sol·licitud de segell de temps contindrà el contingut del Certificat de la TSA que volem que ens emeti el segell (el segell tornarà signat per la seva clau privada).
- **Camps comuns a tots els Vos:**
  - **Array [] attributsOpcionals**: un array amb els atributs opcionals a normalitzar i obtenir que es guardaran com a constants a la classes del component de constants d'aplicació ApplicationConfig. Els atributs obligatoris els passarà automàticament el component. (veure taula Atributs de l'apartat de Constants de Configuració). Poden anar apareixent nous atributs al estàndard DSS o els seus profiles que ens puguin interessar o noves necessitats que es detectin, només caldrà afegir-ho a la nostra classe de constants ApplicationConfig.
  - **Array [] operacionsAddicionals**: un array amb les suboperacions addicionals dintre de l'operació de validació del certificat. Aquestes operacions addicionals i

opcionals quedaran registrades també a la classe d'ApplicationConfig (Veure taula operacions addicionals).

- **Array [] atributsConfig**: un array amb altres atributs de configuració com ara el perfil DSS.
- **Hashmap atributs:** on la key serà el nom dels atributs (veure taula Atributs) i el valor el contingut de l'atribut retornat. Per exemple X509CertificateVo.atributs.get(ApplicationConfig.NIF) retornarà el valor del NIF obtingut a la validació. Els atributs obligatoris sempre vindran complimentats o amb valor buit i els atributs opcionals s'afegiran posteriorment.
- **boolean procesOk**: un camp de resultat booleà que indicarà si el procés és correcte (true – s'ha validat positivament la signatura) o ha fallat (false – no s'ha validat la signatura). Pot ser correcte si valida el certificat, si valida la signatura, si retorna segell de temps o pot se fals si dona qualsevol excepció (no hi ha connexió, etc.) o no valida el certificat, no valida la signatura, etc.
- **HashMap msg**: Construirem un missatge de resposta al hash "msg" de Base64DataVo. El msg contindrà un missatge descriptiu del procés.

Finalment en la Fase III "Construcció" s'ha variat la decisió de disseny, hem decidit utilitzar un únic vo per a tots els casos. Donat que varies de les funcions de signatura necessiten diferents atributs com X509Certificate, Data en Base64, XML en string o un document també en base64, hem definit un Vo que contingui aquests atributs i que omplirem o no en cas que es necessitin per cada servei de signatura. Aquest VO contindrà tots els camps enumerats anteriorment i en la Fase III "Construcció" s'indicarà la definició exacta i ús. Aquest VO serà doncs DigitalSignatureContentVo i la resta quedaran deprecats i no s'utilitzaran.

5.3.3.5 Mòduls EJB

L'objectiu del mòdul ProveidorSignatura és invocar el serveis de signatura de PSIS i encapsular-los amb funcions i una estructura parametritzable per que qualsevol programador de la plataforma no hagi de conèixer els mètodes concrets de PSIS i el seu detall.

Els EJBs de l'aplicació s'encarregaran d'implementar la lògica de negoci responsable de solucionar els diversos casos d'us que hem de resoldre. I ja havent definit el protocol DSS i la seva missatgeria les següents funcions tractaran la informació sol·licitada pel programador o programari per a composar els missatges DSS adients i comunicar-se amb PSIS.

En cada cas, a partir de la informació que s'omple en els Vo i operacions opcionals que es vulguin fer, ProveidorSignatura, s'encarregarà de cridar el servei corresponent de PSIS i de crear el missatge DSS apropiat segons les opcions escollides i les pautes indicades anteriorment (veure Composició de missatges DSS).

ProveidorBean: implementació del SessionEJB que implementa els casos d'ús per tal de proporcionar les funcionalitats de Signatura a tota la plataforma.

- **validarCertificat(DigitalSignatureContentVo):** aquesta funció ens permetrà validar un certificat X509, certificat d'usuari amb les claus públiques. Li passarem un X509CertificateVo que contindrà la informació necessària per a validar un certificat més altres atributs de control (veure apartat de Vos els possibles atributs de control).

  1. Agafarà la informació del certificat i afegint la lògica necessària i les operacions obligatòries (per defecte) i atributs i composarà la crida especifica (attributsOpcionals, operacions Addicionals) i sabent el servei que volem utilitzar de PSIS crearem el missatge DSS apropiat segons les pautes indicades anteriorment (veure Composició de missatges DSS).

  2. Ens tornarà els atributs normalitzats i tota la informació complementaria que haguem demanat. En un map X509CertificateVo.*atributs* amb key (veure taula Atributs de l'apartat Constants Configuració) i valor de l'atribut retornat. Per exemple **X509CertificateVo.atributs.get(ApplicationConfig.NIF)** retornarà el valor del NIF obtingut a la validació.

  3. Construirem un missatge de resposta al hash "msg" de X509CertificateVo. El msg contindrà un missatge descriptiu del procés.

  4. X509CertificateVo també contindrà un camp de resultat booleà que indicarà si el procés és correcte (true – s'ha validat positivament la signatura) o ha fallat (false – no s'ha validat la signatura).

- **validarSignatura(Base64SignatureVo, Data) Base64SignatureVo**: aquesta funció ens permetrà validar una signatura tipus CMS (Base64Signature). Li passarem una Base64SignatureVo que contindrà la informació necessària per a validar una signatura més

altres atributs de control (veure apartat de VOs), per exemple li passarem el moment en el qual volem validar la signatura si ho deixem a null (**Base64SignatureVo.DataValidacio**) la signatura es validarà al moment actual.

1. Rebrà com a paràmetre la informació de la signatura i afegim les operacions i atributs per defecte (attributsOpcionals, operacionsAdiccionals). Si la signatura es vol validar en el temps (en un moment concret) haurem afegit l'operació opcional validacioAData (veure apartat operacions) Base64SignatureVo.operacionsAddicionals[i]=validacioAData). L'usuari o aplicació usuària (PortaSignatures o d'altres) només cal que cridin la funcionalitat que els interessa i el component ProveidorSignatura serà el responsable d'omplir els atributs o operacions necessàries i fer el mapeig que calgui a l'hora de cridar les diferents funcionalitats. En aquest cas l'usuari ens envia una data a omplir doncs nosaltres afegirem l'opció validacioAData i passarem el paràmetre de data. Si l'usuari crida a la funció validarSignatura(Base64SignatureVo) sense data, nosaltres posarem la data a null i cridarem a validarSignatura(Base64SignatureVo, Data null). Si la data és null es farà la validació en la data actual sinó la validació es farà a la data que s'indiqui.

2. Ens tornarà el resultat de la validació i tota la informació complementaria que haguem demanat. En un map *atributs* amb key (veure VOs) i valor de l'atribut retornat. Per exemple **Base64SignatureVo.atributs.get(ApplicationConfig.signatura)** retornarà la signatura realitzada (encriptació i base64).

3. Construirem un missatge de resposta al hash "msg" de Base64SignatureVo. El msg contindrà un missatge descriptiu del procés.

4. Base64SignatureVo també contindrà un camp de resultat booleà que indicarà si el procés és correcte (true – s'ha validat positivament la signatura) o ha fallat (false – no s'ha validat la signatura).

- **validarPdf(Base64DataVo, Data) Base64DataVo**: aquesta funció ens permetrà validar una signatura tipus CMS attached; un document pdf amb la signatura adjunta que els documents pdf implementen com una estructura CMS. Com que els documents pdfs implementen aquesta estructura i és un format molt potent de gran ús i amb molta demanda, PSIS ha implementat una funcionalitat específica per validar signatures en documents pdf, donant la possibilitat, junt amb la validació, d'extreure la informació d'aquesta estructura (Retornar l'hora de signatura, la identitat del signant, la raó de la signatura, etc. veure taula de Constants Configuració). Per aquesta funcionalitat per tant haurem d'annexar tot el document a la petició (**InputDocument - Base64Data**). Li passarem una Base64DataVo que contindrà la informació necessària per a validar una signatura en un document pdf més altres atributs de control (veure apartat de VOs).

  1. Si la signatura es vol validar en el temps (en un moment concret) haurem afegit l'operació opcional validacioAData (Base64Data.operacionsAddicionals[i]= validacioAData). En aquest cas l'usuari ens envia una data a omplir, doncs nosaltres afegirem l'opció validacioAData i passarem el paràmetre de data. Si l'usuari crida a la funció validarPdf(Base64DataVo) sense data, nosaltres posarem la data a null i cridarem a validarPdf(Base64DataVo, Data). Si la data és a null es farà la validació en la data actual sinó la validació es farà en la data que s'indiqui.

  2. Ens tornarà el resultat de la validació i tota la informació complementària que haguem demanat. En un map atributs amb key i valor de l'atribut retornat. Per exemple

Base64DataVo.atributs.get(ApplicationConfig.raoSignatura) retornarà la raó de la signatura.

3. Construirem un missatge de resposta al hash "msg" de Base64SignatureVo. El msg contindrà un missatge descriptiu del procés.

4. Base64SignatureVo també contindrà un camp de resultat booleà que indicarà si el procés és correcte (true – s'ha validat positivament la signatura) o ha fallat (false – no s'ha validat la signatura).

- **crearSegell (DocumentHashVo) DocumentHashVo**: aquesta funció ens permetrà crear un segell de temps signatura tipus CMS (rfc:3161). Li passarem un DocumentHashVo que contindrà la informació necessària per a crear un segell de temps.

1. Utilitzarà un objecte i root del missatge XML del tipus SignRequest i crearem el missatge DSS apropiat.

2. Requerirà omplir els atributs:

   - **DocumentHashVo.X509Certificate:** Ficarem el Certificat de la TSA que volem que ens emeti el segell, per això haurem de tenir aquest certificat.

   - **DocumentHashVo.DocumentHash**: El resum del document a signar (o posar segell de temps).

   - Posarem l'operació addicional **DocumentHashVo.operacionsAddicionals**[]=informacióAddicional que ens permet habilitar l'opció per demanar el segell de temps afegint el certificat de la TSA com a informació addicional (AdditionalKeyInfo.addNewX509Data on el valor serà el **DocumentHashVo.X509Certificate)**.

3. Ens tornarà els segell de temps i tota la informació complementaria que haguem demanat. En un map atributs amb key i valor de l'atribut retornat. Per exemple **DocumentHashVo.atributs.get(ApplicationConfig.Segell)** retornarà el valor del segell de temps.

4. Construirem un missatge de resposta al hash "msg" de Base64SignatureVo. El msg contindrà un missatge descriptiu del procés.

5. Base64SignatureVo també contindrà un camp de resultat booleà que indicarà si el procés és correcte (true – s'ha validat positivament la signatura) o ha fallat (false – no s'ha validat la signatura).

5.3.4 Mòdul frontal de signatura

Finalment l'abast del projecte en curs es centra en el mòdul de Proveïdor Signatura, des de l'AF, AO, en la seva construcció i verificació, el mòdul de frontal de signatura està en procés de desenvolupament i es plantejarà com una línia de futur. Aquest projecte ens ha servit per assentar les bases de la Signatura Digital i poder disposar de primitives de signatura, amb aquestes bases podem iniciar aquesta segona fase del projecte per integrar el Frontal de Signatura amb una aplicació web com el PortaSignatures.

Un cop ja coneixem be totes les primitives de PKI, la formació de missatges DSS i els serveis de PSIS. Podem utilitzar un mòdul de frontal de signatura que permeti a l'usuari final o aplicació web signar documents de forma àgil i usable.

Per això en aquest apartat dissenyarem com l'applet existent de CATCert s'ha d'integrar amb l'aplicació web PortaSignatures, quines opcions té i com el podem aprofitar per les necessitats de signatura de la nostra plataforma de tramitació.

El PortaSignatures, és una simple aplicació web que gestiona documents o ordres pendents de signatura, en si no té cap complexitat addicional que la d'utilitzar un mòdul web i una lògica de negoci per a gestionar-la.

Però aquest mòdul web, el PortaSignatures, necessitarà d'un mòdul que conegui la complexitat de la Signatura Digital i tots els seus elements o serveis de PKI (Public Key Infrastructure). Aquest mòdul serà l'expert del negoci dels serveis de PKI i servirà aquests serveis al mòdul PortaSignatures amb l'applet com a frontal de signatura.

L'applet de CATCert és una eina web que ens permetrà realitzar les mateixes funcionalitats PKI que mostràvem al mòdul de ProveidorSignatura tot i que en comptes d'utilitzar aquest mòdul a través de l'applet, aquest applet de CATCert ja té implementades les crides directament als serveis de PSIS, per això aquesta lògica de negoci la deixarem ja tal i com està implementada.

Ens centrarem doncs en integrar l'applet amb el PortaSignatures i modificar alguns aspectes puntuals com vàrem comentar a l'anàlisi funcional els casos d'us: **Signatura de documents PDF visible, Modificar els paràmetres visibles de la signatura en PDF, Signatura de Lots**.

Aquest component de frontal applet de CATCert el convertirem en un JWS (Java Web Start) ja que JWS permet funcionar sobre qualsevol tecnologia i navegador, perquè no depèn d'aquests com l'applet. A més és mes robust que l'applet i més resistent a fallides i penjades. També ens interessa per la forma de distribució de versions automàtica que aporta el JWS i que només s'instal·la un cop per versió al ordinador del client.

Es treballarà en la descarrega automàtica de la JVM en la versió especifica requerida, la signatura del JWS amb un signant de confiança (el certificat de servidor que ja estarà acceptat pel client) i amb els permisos d'escriptura necessaris.

Dins de les tecnologies que permeten signar un document utilitzarem una eina que proporciona diverses APIs amb les funcions més comunes. Aquest toolkit permet treballar amb estructures PKI (Public Key Infrastructure), Bouncy Castle.

El Bouncy Castle és un proveïdor criptogràfic que permet tractar els contenidors CMS i que implementa les especificacions JCA/JCE de manera molt més robusta i fiable que no pas les pròpies llibreries de SUN Microsystems. Les llibreries de Bouncy Castle estan àmpliament reconegudes i compleixen el RFC. La pròpia API de PSIS incorpora les llibreries de Bouncy Castle.

Aquest proveïdor disposa bàsicament dels següents serveis:

- JCA/JCE
- Missatgeria CMS
- Missatgeria TSP
- Engine validació certificats
- Objectes d'extensió d'atributs (CRL's, atributs de certificats, etc.).

La signatura que utilitzarem inicialment a la plataforma serà a documents amb format pdf i de forma attached. El format pdf té una estructura molt madura i permet la incorporació de signatures (comentaris, rubrica) simples o múltiples, segells de temps i la seva validació (CMS - PKCS#7). La plataforma bàsicament tractarà documents d'aquesta tipologia.

Hi ha una API per manegar els pdfs i afegir signatures: llibreria lowagie per treballar amb java i pdfs i que a més integra la llibreria Bouncy Castle; la classe PdfStamper que permet modificar els pdf, PdfSignatureAppearance que permet afegir la signatura, el segell de temps i diversos atributs: imatge, observacions, etc. El component de CATCert utilitza aquestes llibreries més una d'Apache Security per a la signatura de XML.

5.3.4.1 Parametrització i comportament JWS.

En aquest apartat es donaran les pautes de com utilitzar l'eina, els paràmetres que permeten configurar-la per a diferents entorns i objectius.

A mode de resum, per a signar es poden utilitzar certificats en software, targetes criptogràfiques accessibles a través del middleware (llibreria PKCS#11) i certificats emmagatzemats al magatzem personal de Windows o de Mac OS X. El tipus de signatures que es poden generar poden ser XMLdsig (també en la seva forma avançada, XAdES), CMS i CMS incrustades en un document PDF (PDF signat).

Nosaltres ens centrarem en l'entorn actual dels requeriments del nostre sistema: Windows i unificació de keystores PKCS#11 i keystores de navegadors (Firefox i Internet Explorer) i la signatura de documents tipus pdf (CMS) o bé signatura detached (CMS). La resta de parametrizacions possibles també queden reflectides en aquest document encara que no se'ls hi donarà ús en aquesta fase del projecte.

L'aplicació frontal de Signatura Digital està basada en la tecnologia Java, concretament és Java Web Start, vindrà signada en tots els seus components i treballarà amb els recursos necessaris i les comunicacions necessàries amb el servidor (JBoss) de forma segura (HTTPS). Aquesta aplicació té un mòdul que requereix compilació amb JDK 1.6 i la resta de mòduls amb JDK 1.5, per

les diverses compilacions, signatura dels diversos mòduls i l'ensamblaje es proveeix d'un script ant que ho manega.

El fet de que la implementació sigui en java obre el ventall de possibilitats de sistemes operatius i navegadors que poden suportar el seu ús.

El JWS consta de 5 paquets (jar) que es descarreguen amb el control de versions implícit que proporciona el JWS amb l'arquitectura JNLP (Java Network Launching Protocol). Amb aquesta tecnologia podem definir quins recursos necessitem, quines versions de cadascun i quins paràmetres necessiten: fitxers del projecte: JnlpSteria\files\app\signatura.jnlp i JnlpSteria\version.xml. D'aquesta forma automàticament es descarregarà o actualitzarà aquells recursos que siguin necessaris segons les indicacions d'aquests fitxers, també se seleccionarà que el client requereixi la versió de JRE 1.6 que en cas de manca s'instal·larà automàticament.

- JWSSteriax.x.jar – Conté un paquet amb la lògica del JWS i les diferents implementacions de signatura i magatzems de certificats. I un segon paquet de llibreries d'Apache Commóns per a connexions http, necessàries per a poder generar segells de temps (RFC3161 o XML). Suporta comunicació bidireccional, utilitzant sockets segurs amb el servidor d'aplicacions. Des de l'HTML (PortaSignatures) s'envien les cookies del procés en que es troba l'usuari i el JWS es comunica posteriorment amb el servidor d'aplicacions (PortaSignatures) recuperant la sessió de l'usuari i finalitzant el procés.

- CATCertCMSlibx.x.jar – Es tracta d'un paquet reduït de la llibreria de Bouncy-Castle per a la generació de signatures CMS.

- CATCertXMLlibx.x.jar – Conté el paquet d'Apache XML. Un paquet reduït amb petites modificacions de les llibreries d'Apache Security per a la generació de signatures XML.

- CATCertPDFlibx.x.jar – Conté un paquet reduït de la llibreria iText per al tractament de documents PDF. Permetrà incrustar signatures CMS en els documents, i que aquestes siguin vàlides (i visibles) si el document s'obre amb l'eina d'Adobe, l'Acrobat Reader.

Es disposa de multitud de paràmetres que permeten configurar l'eina segons les necessitats. Per tal de facilitar-ne l'ús, es llistaran agrupats segons funcionalitat, donant detalls del que cada un implica i de com s'utilitza. Com hem comentat nosaltres només en configurarem algunes segons les necessitats actuals del projecte, la configuració d'aquests paràmetres es pot modificar al properties psconfig.properties, d'altres ja vindran fixades.

a Paràmetres principals (obligatoris)

Els paràmetres principals s'encarreguen de definir el comportament bàsic del JWS és a dir, de quin magatzem es recuperaran les claus que s'utilitzaran en el procés de signatura i quina implementació de signatura s'utilitzarà.

- keystore_type – Indicarà el tipus de magatzem de certificats que s'utilitzarà en el procés de signatura. Nosaltres treballarem amb el mode 1, que unifica els certificats del navegador i de la targeta criptogràfica (PKCS#11) però aquestes propietats com hem dit es poden configurar al fitxer psconfig.properties.

  Cal utilitzar un codi numèric dels següents:

  1 – Magatzem de certificats del compte personal d'usuari de Windows. Per a que aquesta opció funcioni, el JWS crearà una carpeta CATCert dins de la carpeta personal de l'usuari i hi guardarà la llibreria dll nativa (sunmscapi.dll) que necessita per a poder-hi accedir.

  2 – Certificat en software (fitxer PFX, PKCS#12). Cal indicar el camí absolut al fitxer que conté el certificat.

  3 – Targeta criptogràfica, utilitzant el middleware (llibreria PKCS#11). Cal indicar el camí absolut a la llibreria.

  4 – Magatzem de certificats de Mozilla. Opció encara no disponible.

  5 – Magatzem de certificats personals de Java. Opció encara no disponible.

  6 – Magatzem de certificats personals de Mac OS X, .Mac.

- signature_mode – Permet seleccionar el format de representació de la signatura electrònica que es generarà. Nosaltres treballarem amb els modes 2 i 4 però aquestes propietats com hem dit es poden configurar al fitxer psconfig.properties.

  Cal utilitzar un codi numèric dels següents:

  1 – CMS attached (signatura CMS conté el document original).

  2 – CMS detached (signatura CMS no conté el document original).

  3 – CMS detached, utilitzant hash pre-calculat. Opció encara no disponible.

  4 – PDF (CMS detached incrustada en un document PDF).

  5 – XMLdsig enveloped (document XML embolcalla la signatura).

  6 – XMLdsig enveloping (signatura XML embolcalla el document original).

  7 – XMLdsig detached (signatura XML no conté el document original)

  8 – XMLdsig detached, utilitzant hash pre-calculat.

9, 10, 11 i 12 – XAdES-BES enveloped, enveloping, detached i detached amb hash pre-calculat. Protegeix el certificat del signant.

13, 14, 15 i 16 – XAdES-T enveloped, enveloping, detached i detached amb hash pre-calculat. Afegeix un segell de temps a la forma BES.

13, 14, 15 i 16 – XAdES-C enveloped, enveloping, detached i detached amb hash pre-calculat. Afegeix informació de revocació a la forma T.

b Paràmetres de xarxa

En cas d'haver de fer connexions a serveis externs, com és el cas d'utilitzar segells de temps, caldrà tenir en compte si l'usuari es troba darrere d'un proxy. La configuració per defecte detecta l'ús del proxy configurat en el navegador. Si la configuració per defecte no funciona o es desitja utilitzar un proxy diferent del que hi ha configurat, caldrà utilitzar el paràmetre següent:

- proxy_settings – Indica les dades del proxy que haurà d'utilitzar el JWS en el cas d'haver de fer accessos a serveis remots (servei de segellat de temps de PSIS). Cal indicar el nom del proxy i el port separats per un espai (proxy_settings ="serverName serverPort"). Aquestes propietats com hem dit es poden configurar al fitxer psconfig.properties.

c Paràmetres d'entrada (document / llibreria / àlies-CN)

Aquests paràmetres facilitaran al JWS què serà i com ha de tractar el document a signar.

- document_to_sign – Forma abstracta del document a signar en funció del que s'indiqui en el paràmetre doc_type. Es pot obviar sempre i quan s'actualitzi amb el mètode públic abans de fer la crida al mètode de generació de signatura.

  Nosaltres treballarem passant el l'UDI des del PortaSignatures (identificador únic del document a signar al GestorDocumental), però aquestes propietats com hem dit es poden

  configurar al fitxer psconfig.properties.

- doc_type – Codi numèric que indicarà a l'eina com recuperar el document que cal signar. Cal utilitzar-ne un dels següents. Si no s'indica res, el valor per defecte és 2 (document únic). Nosaltres treballarem amb aquest valor però aquestes propietats com hem dit es poden configurar el psconfig.properties.

  1 – Directori local. Permet signar tots els documents que hi hagi en una carpeta accessible des del client on s'executa el JWS. Amb aquesta opció, cal indicar al paràmetre document_to_sign el camí absolut al directori.

2 – Document únic. Cal indicar el camí absolut al document que es vol signar en el paràmetre document_to_sign. Utilitzant el mètode públic per actualitzar paràmetres es pot crear un diàleg amb l'usuari per seleccionar el document a signar.

3 – Hash pre-calculat. Amb aquesta opció, caldrà que el paràmetre document_to_sign contingui el valor del hash del document codificat en Base64.

4 – Contingut del document codificat en Base64. Com ja indica, permet que en el paràmetre document_to_sign si pugui carregar el document sencer. És una opció útil si no es pot pre-calcular el hash i no es disposa d'accés local al document.

5 – Llista de camins absoluts als fitxers locals a signar. Permetrà indicar una llista de documents locals a signar (no tenen perquè estar en la mateixa carpeta). La llista haurà d'estar separada amb punts i comes (;).

Els següents paràmetres permeten indicar, en cas d'utilitzar la targeta o certificat en software on es troba el fitxer que cal utilitzar. Nosaltres indiquem la llibreria pkcs11 de CATCert pkcs11_file=C:/WINDOWS/system32/aetpkss1.dll) però aquestes propietats com hem dit es poden configurar el psconfig.properties.

- pkcs11_file – Indica el camí absolut on cal anar per a trobar la llibreria PKCS#11que ha d'utilitzar el JWS per poder treballar amb la targeta criptogràfica.

- pkcs12_file – Indica el camí absolut al fitxer que conté el certificat en software.

Els paràmetres que hi ha a continuació faciliten la selecció del certificat que cal utilitzar, si per exemple, l'aplicació ja el coneix (cal utilitzar l'alies del certificat o el CommónName del subjectDistinguishedName). En el nostre cas si tenim sessió de SSO per certificat X509 en el mòdul de login es guardarà el CN per transmetreu a la resta d'aplicacions i en especial a l'aplicació de JWS de Signatura Digital, aquest paràmetre es pot modificar directament per la seva propietat al psconfig.properties.

- selected_alias – Indica l'àlies del certificat que cal utilitzar en la signatura. Es comprova que existeixi en el dispositiu / magatzem seleccionat.

- selected_CN – Indica el CommónName dins del SubjectDistinguishedName del certificat que cal utilitzar en la signatura.

En cas de que es desitgi filtrar des de l'aplicació quins certificats mostrar com a disponibles, en funció de l'autoritat de certificació que ha emès el certificat, es disposa del següent paràmetre:

- allowed_CAs – Permet filtrar els certificats a mostrar en el diàleg mitjançant el CommónName de l'IssuerDistinguishedName que apareix en el certificat. Es poden indicar múltiples entrades separades per ";". No té en compte la distinció majúscules/minúscules. Exemple: "EC-SAFP;EC-idCAT".

d Paràmetres de sortida

Amb aquests paràmetres es controla com obtenir el resultat del procés de signatura. Hi ha 3 mètodes i es poden utilitzar a la vegada (no cal limitar-se a un forçosament). Per defecte estan tots desactivats, cosa que obliga a incloure com a paràmetre, com a mínim, un d'ells.

Sortida amb event Javascript

- js_event – Per activar la sortida utilitzant un event Javascript cal incloure aquest paràmetre amb valor true. El valor per defecte és false. La sortida es captura creant una funció Javascript amb el nom onSignOK(signature). Aquesta funcio l'hem interceptat per cas que la signatura sigui correcta i l'usuari l'accepti. Aleshores ens comuniquem amb el servidor d'aplicacions (PortaSig) mitjançant streams HTTPS i recuperem la sessió de l'usuari per a finalitzar el procés de signatura en el PortaSignatures.

Sortida a document local

- local_file – Per crear el document que conté la signatura en un fitxer local cal incloure aquest paràmetre amb valor true. El valor per defecte és false.

- output_filename – Indica el camí absolut del fitxer en el que es desarà la signatura sempre que el paràmetre local_file estigui activat.Si local_file està activat però no s'especifica el nom del fitxer de sortida, aquest prendrà el nom del document original i li afegirà la cadena "_signat.ext", on ext indicarà el tipus de document. Hi ha dues excepcions al comportament per defecte: quan es signen tots els documents d'una carpeta i es genera una signatura n_enveloping i quan el document a signar viatja al propi HTML (veure apartat 3.4). En el primer cas es crea un document de sortida amb el nom document_multisignatura.xml a la carpeta on hi ha la resta de documents. En el segon, es crea un document amb el nom "signatura.ext" a la carpeta CATCert que s'ha creat prèviament a la carpeta personal de l'usuari.

Format de sortida

- output_mode – Amb aquest paràmetre es pot decidir el format de representació de la signatura. En principi tan sols té sentit en el cas de signatures CMS en que es pot triar entre codificació binaria o Base64. Per defecte pren el valor de Base64. Per a signatures XML o PDF el format de sortida no es pot modificar. El paràmetre es configura amb un valor numèric:

  1 – Codificació binaria.

2 – Codificació Base64, opció per defecte en signatures CMS.

3 – XML, opció per defecte en signatures XML.

4 – PDF, opció per defecte signant documents PDF.

e Paràmetres signatura XML

Paràmetres específics per a les signatures XML (XMLdsig i XAdES en les formes suportades).

- n_enveloping – En el cas d'utilitzar qualsevol de les formes de signatura XML enveloping i havent de signar múltiples documents (utilitzant l'opció de signar tots els documents d'una carpeta), és possible generar una única signatura que inclogui referències a tots els documents de la carpeta.

- signature_policy – Permet incloure la política contra la que s'haurà de validar la signatura generada. El valor del paràmetre haurà de ser l'OID de la política de signatura (implica l'ús del paràmetre signature_policy_hash). Paràmetre disponible tan sols per a signatures XAdES (convertim la forma de la signatura a EPES).

- signature_policy_hash – El valor d'aquest paràmetre conté el hash codificat en Base64 del document XML que descriu la política de signatura contra la que es validarà la signatura generada (ve a ser una mena de control de versió).

f Paràmetres signatura CMS / PDF

Paràmetres opcionals i específics per a signatures CMS i incrustades (CMS detached) en documents PDF:

- TimeStamp_CMS_signature – Com el seu nom indica, permet afegir un segell de temps a les signatures CMS i per extensió a les que s'incrusten en els PDF. Per activar-ho cal posar el valor del paràmetre a true. Per defecte el valor és false. El servei de segellat de temps (TSA) utilitzat és el que s'ofereix a PSiS (http://psis.catcert.net/psis/catcert/tsa).

- pdf_signature_field – Si el document PDF a signar disposa de camps de signatura per buits, és possible indicar el nom del camp que es desitja emplenar. D'aquesta manera s'evita que en el diàleg que apareix a l'hora de signar un document PDF s'hagi d'escollir aquesta opció.

- pdf_visible_signature – Permet indicar al component que la signatura que es crearà al document PDF sigui invisible (valor a false). Per defecte el valor és true (visible). Si hi ha camps de signatura, aquest paràmetre no té valor i s'intentarà emplenar un dels camps buits.

- pdf_signature_rectangle – Quan no hi ha camps de signatura i la signatura ha de ser visible, hi ha l'opció de seleccionar on es crearà: coordenades de la pàgina i número de pàgina. El valor d'aquest paràmetre per defecte és 100 100 200 200 1. Les coordenades s'indiquen de forma numèrica i separades per espais: llx lly urx ury page_nr.

- pdf_certificate_doc – Posant aquest paràmetre a true, la signatura que s'incrustarà en el PDF serà del tipus MDP (document certificat). Per defecte pren el valor false.

- pdf_reason – Permet indicar el motiu de la signatura (camp propi d'Adobe). L'ús d'aquest paràmetre deshabilita aquesta opció en el diàleg de signatura de documents PDF.

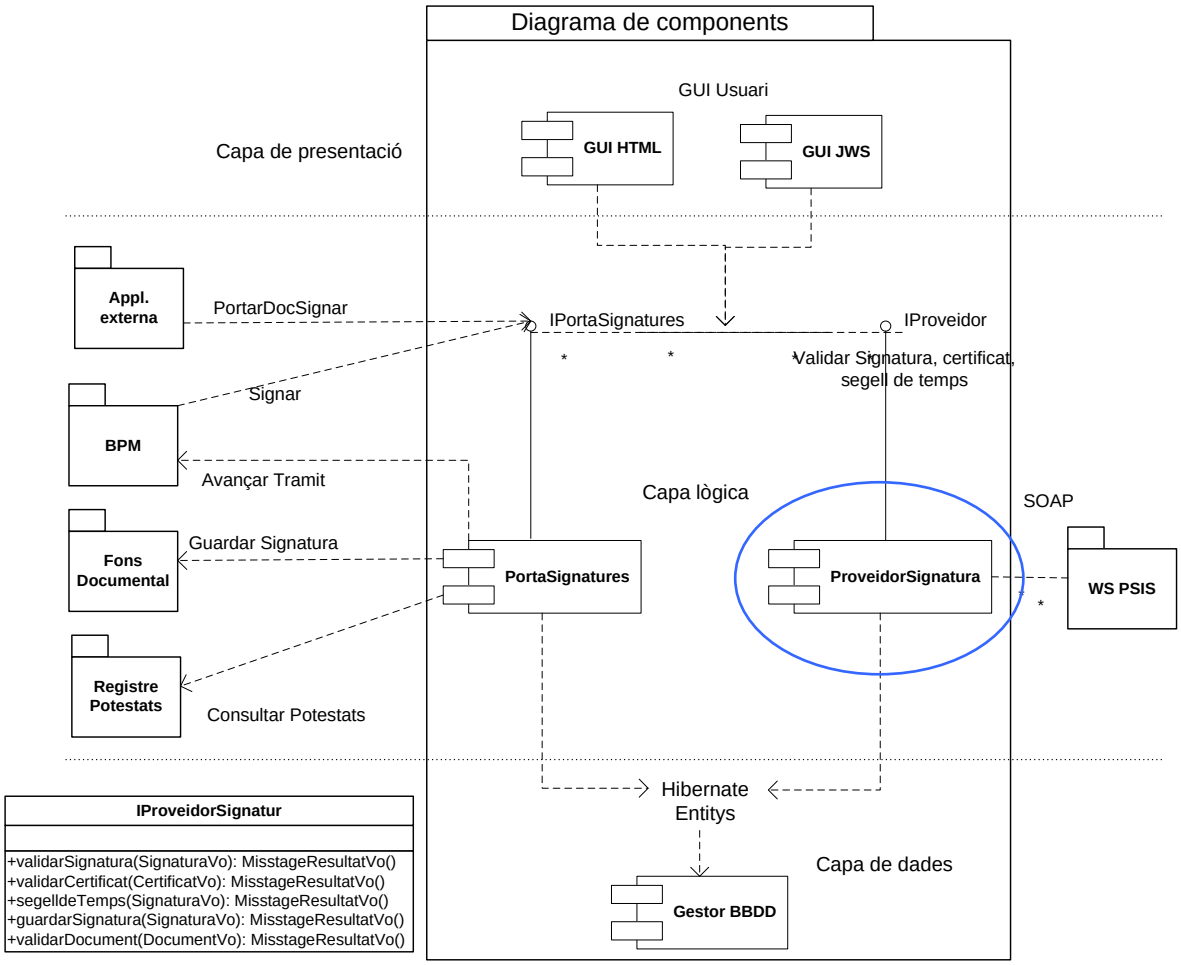
- pdf_location – Permet indicar una localització (camp propi d'Adobe). Com en el cas anterior, si s'utilitza el paràmetre, desapareix del diàleg de signatura de documents PDF.

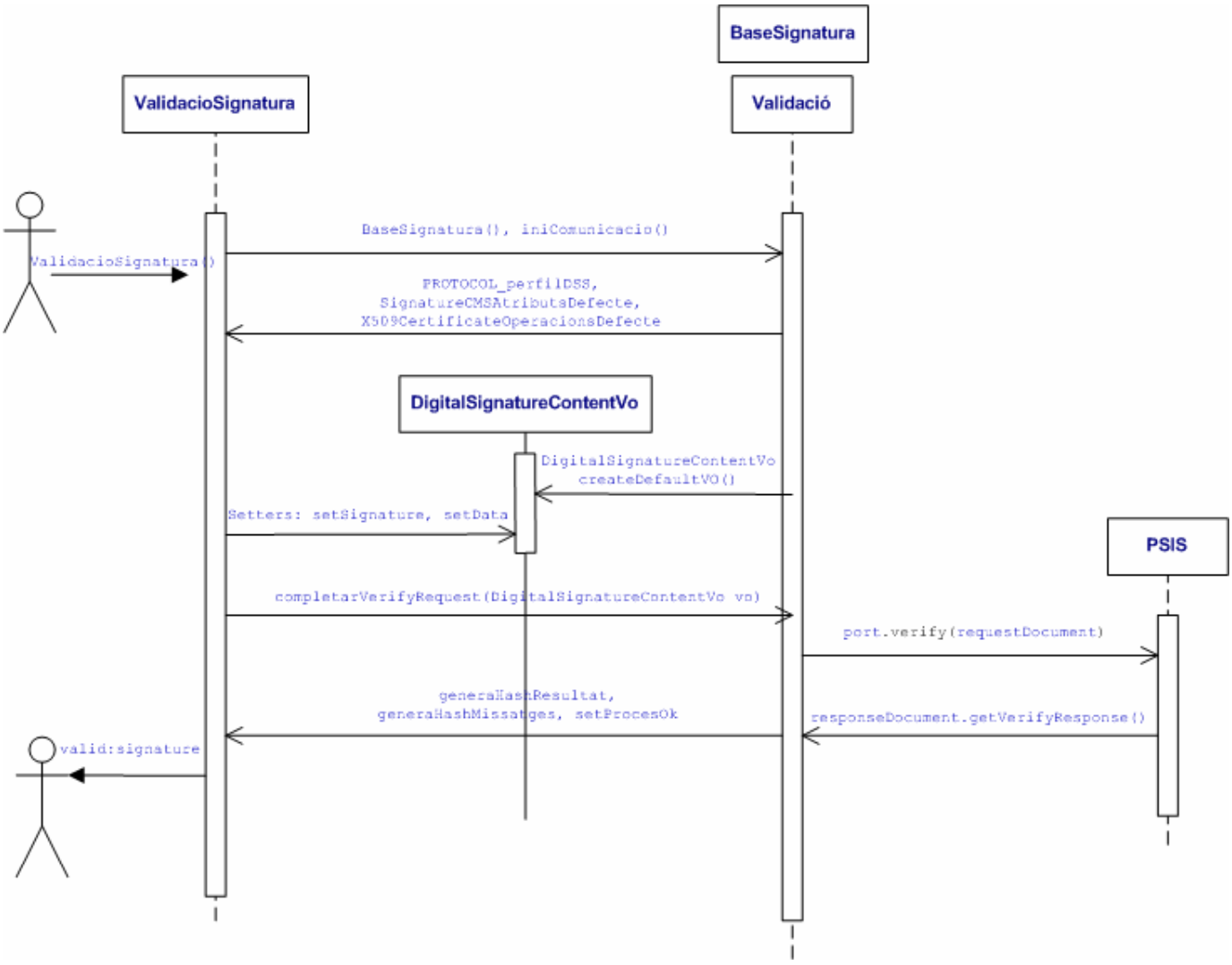
La informació de parametrització i de funcionament de l'applet de signatura de CATCert, s'ha extret i es pot trobar en detall al document: Manual d'ús de l'eina web de signatura-e.pdf referenciat a la bibliografia. Aquesta informació s'ha inclòs per aclarir el funcionament bàsic d'aquest applet, a partir d'aquest apartat procedirem a descriure les variacions i l'ús que s'ha fet al convertir-ho a JWS.

5.4 Capítol 4: Fase III - Construcció

En la fase de construcció ens preocuparem d'explicar el procés de construcció de les peces i mètodes de forma que indicarem les decisions més importants en el desenvolupament i explicarem els desenvolupaments principals. A grans trets els mètodes principals dels serveis PKI els hem explicat a l'apart de l'AO del component ProveidorSignatura, hem detallat els protocols a utilitzar (SOAP, DSS, RMI) i l'estructura de l'intercanvi de missatges (XML, DSS), per tant en aquest apartat aprofundirem en explicar com hem utilitzat aquestes eines per a la construcció de la nostra llibreria **mòdul de serveis de PKI**, **el ProveidorSignatura**.

5.4.1 ProveidorSignaturaCore.





Aquest diagrama de seqüència és un exemple del prototipus de seqüència que és seguirà a l'hora de construir les relacions entre les classes i mètodes principals. En aquest cas es tracta de la validació d'una signatura, per la validació d'un certificat o creació de signatures CMS o un segell de temps seguirem la mateixa dinàmica però amb les classes especifiques pròpies de cada servei. L'estructura de classes s'explicarà en aquest apartat

En aquest apartat entrarem en profunditat en el component ProveidorSignaturaCore que és qui té tota la lògica de negoci implementada per cridar els diferents serveis PKI objecte d'aquest projecte a través de PSIS. La resta de components per crear components de negoci EJB per poder-los invocar remotament via RMI com a components distribuïts es crearan els stubs i estructures necessàries per tenir aquesta possibilitat però no entra en l'abast d'aquest projecte explicar i detallar el seu funcionament ja que es dona per suposat que es coneixen els objectius i estructures d'aquests components EJB.

Aquest component per tant es podrà utilitzar com a llibreria .jar o be com a component distribuït EJB o bé també es pot crear una interfície de servei web.

5.4.1.1 Estructura del projecte ProveidorSignaturaCore:

ProveidorSignaturaCore	
src	Paquet que conté el codi font.
com.steria.digitalsignature.config	Paquet que conté les propietats de configuració.
com.steria.digitalsignature.core	Paquet que conté el nucli de la lògica de negoci d'aquest projecte.
com.steria.digitalsignature.exception	Paquet que conté els tipus d'excepció.
com.steria.digitalsignature.properties	Paquet que conté les propietats de configuració.
com.steria.digitalsignature.vo	Paquet que conté els objectes de transport Value Objects.
oasis.names.tc.dss._1_0.core.wsdl	Paquet que conté els stubs dels webservice a invocar de PSIS.
org.apache.commons.httpclient.contrib	Paquet que conté les llibreries http commóns.
META-INF	
file.properties	Fitxer de texte que conté les propietats de configuració del paquet.
Referenced Libraries	Paquet que conté les llibreries necessàries per l'execució del mòdul.
psis-beans.jar	Paquet que conté les llibreries necessàries de PSIS.
xbean-2.1.0.jar	Paquet que conté un parser XML (XSD).
xfire-all-1.2.2.jar	Paquet que conté un framework per a la connexió de serveis web.
commons-httpclient-3.0.jar	Framework que proporciona una API per comunicacions HTTP
commons-logging-1.0.4.jar	Per registre i formateig de dades i d'informació i log del programari
wsdl4j-1.5.1.jar	Llibreria que ens permet generar els stubs per als WSDL
wss4j-1.5.0.jar	Llibreria per a gestionar la seguretat dels serveis web
jdom-1.0.jar	Llibreria que ens ajudarà a tractar els fitxers XML
JUnit 3	Framework que ens permetrà realizar test i proves unitaries
JRE System Library [jre6]	Paquet bàsic de llibreries de java
bin	Fitxer de texte que conté les propietats de configuració del paquet.
build	Carpeta que conté els fitxers binaris.
classes	Carpeta que temporal de compilació.
inst	Carpeta que conté els fitxers compilats.
lib	Carpeta que conté els paquets resultants com a llibreries independents.
proveidor_build.xml	Carpeta que conté les llibreries necessàries per l'execució del mòdul.
ProveidorSignaturaCore.jardesc	Fitxer de compilació i empaquetament del mòdul ProveïdorSignatura.jar
ws_build.xml	Fitxer de compilació i empaquetament de la llibreria psis-beans.jar

5.4.1.2 Llibreries necessàries:

psis-beans.jar: Les llibreries que utilitzarà el nostre client, per a facilitar la construcció dels missatges que s'enviaran a la plataforma PSIS i la recepció de la resposta: schemas, beans, etc.

oasis.names.tc.dss._1_0.core.wsdl: Aquesta llibreria ens dona l'accés EndPoint a través del Port SOAP per fer les crides webservice a PSIS. Conté les següents classes:

- digitalSignatureServiceClient Factoria de clients de la plataforma PSIS
- digitalSignatureServiceImpl Implementació de la interfície SOAPPort
- SOAPport Interfície de definició de mètodes de la plataforma PSIS

La integració amb el servei web basat en DSS requereix la compilació d'un WSDL (Web Service Definition Language), aquest és el descriptor dels serveis que proporciona PSIS. A partir d'aquest descriptor obtindrem els *stubs* que ens donaran la infrastructura client necessària per a poder realitzar les crides als serveis PSIS de forma transparent (per a nosaltres com una crida a una API qualsevol, com si fos una crida a una llibreria local però en realitat s'encapsula amb SOAP i es realitza una crida remota), la URL del descriptor dels serveis de PSIS és http://psis.catcert.net/wsdl/dss.wsdl.

Per això cal obtenir arxius de compilació WSDL i XSD, generar els stubs primer per poder tenir d'infraestructura necessària per la crida dels serveis de PSIS, aquesta part l'hem obviada i no entrarem en més detall les llibreries resultants són les descrites fins ara. Existeixen unes guies per a la integració amb PSIS que contemplen pas a pas el procés de creació dels Stubs del client a partir del WSDL i aporten diversos exemples de com invocar un gran nombre de les funcionalitats de PSIS (veure Guia bàsica pels integradors de PSIS.pdf apartat de referències).

XMLBeans 2.1.0: XML parser, llibreria per a facilitar el tractament d'arxius XML definits amb XSD.

XFire 1.1.2: Framework per a la connexió i configuració de serveis web. Per tal de poder compilar necessitem un client de WebServices que suporti invocacions amb model Document/Literal, donat que la invocació es fa mitjançant l'enviament de missatges DSS a un port SOAP concret i no fent una invocació a un mètode remot. Dels clients de WebServices que suportin aquesta característica podem fer servir o bé XFire o Axis 2.0, per a implementacions basades en Java, nosaltres escollirem XFire.

Commóns-httpclient-3.0: És un framework que ens proporciona una API per abstraure de les comunicacions sota HTTP i realitza aquestes tasques de comunicació del costat del client.

Commóns-loggin-1.0.4: És una llibreria que ens ajuda a realitzar el registre i formateig de dades i d'informació i log del programari, ja sigui sortida per pantalla o a fitxer.

Wsdl4j-1.5.1: The Web Services Description Language for Java Toolkit (WSDL4J). Aquesta llibreria serveix per a invocar serveis web amb Axis 1.4. És una llibreria que ens permet generar els stubs per als WSDL, crear i manipular els documents WSDL segons l'estàndard JSR110.

Wss4j-1.5.0: Llibreria que implementa l'especificació d'OASIS per a la seguretat dels serveis web (OASIS Web Services Security (WS-Security)). Principalment ens ajudarà a signar I verificar Missatges SOAP amb la informació de seguretat de WS-Security.

Jdom-1.0: És una llibreria que ens ajudarà a tractar els fitxers XML en java.

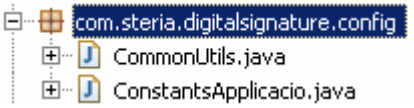
JUnit 3: És un framework que ens permet la validació de la construcció, programant proves unitàries de forma que comprovarem si el funcionament dels mètodes i components es comporta com esperem i segons es va definir funcional i orgànicament. En aquest projecte aquesta llibreria també tindrà una importància rellevant donat que ens servirà per la Fase IV: Verificació de la construcció i lliurament.

JDK v1.5 o superior: Paquet bàsic de desenvolupament Java.

Els serveis de PSIS els disposem en diversos entorns:

- Entorn d'integració:
  - http://psisbeta.catcert.net/psis/catcert-test/dss
  - http://psisbeta.catcert.net/psis/catcert-test/dsspdf
- Entorn d'explotació:
  - http://psis.catcert.net/psis/catcert/dss
  - http://psis.catcert.net/psis/catcert/dsspdf

5.4.1.3 Codi font, classes i mètodes:



ConstantsApplicacio: Conté constants de configuració que permeten referenciar fàcilment els paràmetres complexes noms estàndard DSS o profiles.

- Atributs de Certificat: Es creen constants com per demanar el NIF d'un certificat o el nom de pila de la persona:

```
    // Atributs de Certificat
    public static final String
CERT_NIF="urn:catcert:psis:certificateAttributes:KeyOwnerNIF";
    public static final String
CERT_nomDePilaPersona="Urn:oasis:names:tc:dss:1.0:profiles:XSS:certificat
eAttributes:SubjectDistinguishedName:commónName";
```

- Constants per indicar atributs de protocol, com el perfil que es vol utilitzar:

```
   // Atributs de Protocol
    public static final String
PROTOCOL_perfilDSS="urn:oasis:names:tc:dss:1.0:profiles:XSS";
    public static final String
PROTOCOL_perfilDSS_PDF="urn:oasis:names:tc:dss:1.0:profiles:DSS_PDF";
    public static final String
PROTOCOL_perfilDSS_Timestmaping="urn:oasis:names:tc:dss:1.0:profiles:time
stamping";
```

- Constants per indicar diferents operacions possibles:

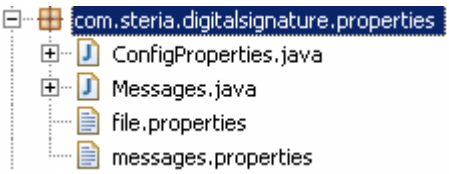
```
    // Operacions
    public static final String
OP_politicaSignatura="xss:SignaturePolicy"; //$NON-NLS-1$
     public static final String
OP_detallsProces="dss:ReturnProcessingDetails";

    public static final String OP_horaSignatura="dss:ReturnSigningTime";
```

- Creem enumeracions constants per indicar quins atributs, operacions, etc pot configurar-se en cada cas, en cada TYPE (cert, sign, etc.)

```
public static final String[] X509CertificateOperacions = new String[] {
            OP_detallsProces, OP_mostrarAtributsCertificat };

 // afegim els atributs opcionals que ens passen per paràmetre sempre i
quant corresponguin als definits - X509Certificat

List<String> listX509CertificateOperacions =
Arrays.asList(X509CertificateOperacions);
```

D'aquesta forma amb un simple *X509CertificateOperacions* afegim totes les operacions estàndard per demanar un servei de validació de certificat X509. També amb un simple "." Tens l'ajuda contextual i pots escollir aquell paràmetre de configuració que vols utilitzar per a parametritzar la teva crida als serveis de PKI.

```
ret.setPerfil(ConstantsApplicacio.PROTOCOL_perfilDSS_PDF);
ret.setPerfil(ConstantsApplicacio.PROTOCOL_perfilDSS_Timestmaping);
```



Aquest paquet ens ajuda a configurar altres constants i paràmetres necessaris per a que el programari funcioni. Principalment hi ha dos fitxers: ConfigProperties.java i file.properties.

ConfigProperties: Conté constants de configuració que permeten referenciar les claus del fitxer properties que configura l'aplicació.

file.properties: Conté els valors específics de la informació que hem d'utilitzar en cada cas, sobre tot conte valors més susceptibles a canviar durant el temps que no la classe de ConstantsApplicacio. Per exemple valors de la ubicació dels fitxers de proves, les diferents urls de treball i proves o be les claus (keys) d'alguns missatges de retorn específics que poden canviar també.

En aquest paquet hem organitzar les classes Vo. Inicialment segons el definit en la fase de disseny orgànic havíem pensat de crear un Vo per cada tipus de signatura, però finalment ho hem organitzat tot amb la mateixa classe Vo. Per tant la classe X509CertificatVo ja no s'utilitza i esta deprecada. Utilitzarem amb aquest fi només la classe DigitalSignatureContentVo i aquesta classe contindrà tots els atributs d'entrada i sortida necessària pels mètodes de PKI i per la invocació dels serveis de PSIS, de manera que amb aquesta classe de transport de dades tindrem totes les dades importants i significatives dels nostres processos de signatura.

DigitalSignatureContentVo: És la classe de transport de dades essencials pels processos de PKI.. Aquesta classe defineix:

- El contingut del Certificat X509 de l'usuari en base64 (X509Certificate) o bé en el cas d'una sol·licitud de segell de temps contindrà el contingut del Certificat de la TSA que volem que ens emeti el segell.

```
protected byte[] certificate;
```

- El contingut de la signatura de tipus CMS (Base64Signature)

```
protected byte[] signature;
```

- El contingut del document pdf amb signatura CMS (Base64Data)
- El resum del document a signar (o posar segell de temps) (DocumentHash)

```
protected byte[] data;
```

- El contingut d'una signatura XML detached (InlineXML)

```
protected String inlineXML;
```

- Una llista amb els atributs opcionals a normalitzar i obtenir que es guardaran com a constants a la classes del component de constants d'aplicació ConstantsApplicacio. Els atributs obligatoris els passarà automàticament el component. Poden anar apareixent nous atributs al estàndard DSS o els seus profiles que ens puguin interessar o noves necessitats que es detectin, només caldrà afegir-ho a la nostra classe de constants ConstantsApplicacio.

```
protected List<String> attributsOpcionals=new ArrayList<String> ();
```

- Una llista amb les operacions addicionals possibles dintre del servei concret de signatura o PKI. Aquestes operacions addicionals i opcionals quedaran registrades també a la classe ConstantsApplicacio.

```
protected List<String> operacionsAddicionals=new ArrayList<String>;
```

- Un hashmap d'atributs on la key serà el nom dels atributs que ens interessen en el procés i el valor el contingut de l'atribut retornat. Per exemple:

DigitalSignatureContentVo.atributs.get(ApplicationConfig.NIF) retornarà el valor del NIF obtingut a la validació. Els atributs obligatoris sempre vindran complimentats o amb valor buit i els atributs opcionals s'afegiran posteriorment.

```
protected HashMap<?, ?> atributs;
```

- Un camp de resultat booleà que indicarà si el procés és correcte (true – s'ha validat positivament la signatura) o ha fallat (false – no s'ha validat la signatura). Pot ser correcte si es valida el certificat, si es valida la signatura, si es retorna segell de temps o pot se fals si dona qualsevol excepció (no hi ha connexió, etc.) o no es valida el certificat, no es valida la signatura, etc.

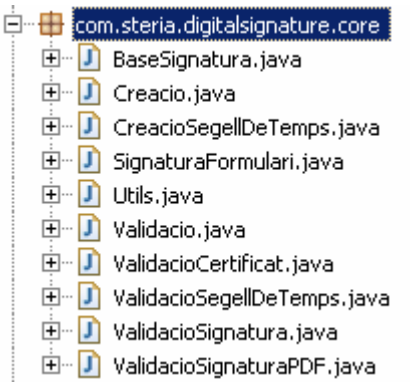
```
protected boolean procesOk;
```

- Un atribut de protocol on s'indicarà el tipus de perfil DSS de treball. Per exemple si treballem amb l'extensió de CATCert XSS, amb el perfil específic de serveis per PDF o be el de segell de temps.

```
protected String perfil;
```

- Un hashmap on afegirem tota la informació de validació i detalls de sortida del procés.

```
protected HashMap<String, String> msg;
```



El core del projecte està organitzat de forma que hi ha una classe Base (pare) de totes, que conté mètodes i atributs aplicables a totes les classes. D'aquesta hereten dues classes una de Creació i un altre de Validació, aquestes dues a la seva hora hereten, sobreescriuen i afegeixen més mètodes específics ja sigui per les classes de creació de signatura com per les de validació respectivament.

BaseSignatura: És la classe de base que heretaran totes les classes d'aquest projecte i conté les funcionalitats base que poden requerir qualsevol classe, com retornar un map amb les respostes de la crida al servei de PSIS, carregar la url de crida al servei de PSIS i inicialitzar la connexió, o carregar el fitxer de propietats de configuració del mòdul i crear els atributs imprescindibles com:

```
protected ConfigProperties propFile = null;
protected XmlOptions options = null;
```

```
protected String soapURL = "";
protected SOAPport port;
```

Creació: És la classe de base que heretaran totes les classes d'aquest projecte que vulguin crear signatures i aquesta heretarà les funcions bàsiques de BaseSignatura i afegirà les especifiques per la creació de la signatura. Aquesta classe defineix:

- Instanciarà la classe que escriurà les sortides de log.
- Definirà atributs necessaris per la creació de signatura:
- Definirà mètodes abstractes que els seus fills hauran d'implementar:

```
    abstract protected void
completarVerifyRequest(DigitalSignatureContentVo vo);
    abstract protected HashMap<String, String>
generaHashResultat(SignResponse response);
```

- Realitzarà les operacions principals que tota classe filla necessitarà:
- Establiment de la comunicació.
- Obtenir una estructura DSS de petició de signatura `signRequest` i iniciarlitzar-lo.
- Inicialització del Vo amb la informació principal per poder començar a treballar.
- El fet de completar el missatge DSS amb la petició de signatura serà responsabilitat de la classe filla que implementi aquest mètode específic: `completarVerifyRequest(vo);`
- Amb aquesta petició complerta, la classe para Creació i BaseSignatura realitzaran la connexió al portSoap i endpoint del servei de PSIS per realitzar la petició.

```
log.info("Crida WS PSIS");
SignResponseDocument responseDocument =
port.sign(this.requestDocument);
```

- Obté la resposta del servei i la parsetja o tracta afegint-lo al hash d'atributs resultants del nostre Vo.
- I finalment afegeix al Vo un chequeix ràpid de si el procés de signatura és correcte o no, mitjançant la variable booleana `procesOk` on espera que el resultat del missatge `dss:ResultMajor` sigui el resultat definit com a correcte `urn\:oasis\:names\:tc\:dss\:1.0\:resultmajor\:Success` (configurable al properties).

Nota: Com hem vist el hashmap `msg` del `DigitalSignatureContentVo` conté la informació extensa del procés de creació de signatura.

Fixem-nos que hem organitzat el codi de forma que cada classe s'encarregui de la seva responsabilitat i que les classes pares defineixen els atributs, mètodes i infrastructura comuns i només transfereix el control a la classe filla per a implementar i executar aquells mètodes que han d'acabar de completar segons les seves responsabilitats. Com per exemple, quan la classe Base i Creació creen la connexió i port SOAP contra PSIS, creen el missatge DSS, deixant completar el missatge a la classe filla especifica i responsable de la funcionalitat concreta i tornen a agafar el control per enviar la petició a PSIS i tractar la resposta, omplir els resultats en el Vo final i retornar el resultat. Aquesta és part de política de disseny de l'arquitectura on el programador exclusivament haurà de heretar les classes Base per a que pugui només preocupar-se de definir la funcionalitat que vulgui programar de les de PKI, sense preocupar-se gaire de la connexió, atributs obligatoris, operacions obligatòries, o tractaments específics de la informació dels missatges DSS.

Validació: És la classe de base que heretaran totes les classes d'aquest projecte que vulguin validar certificats, signatures o segells de temps i aquesta heretarà les funcions bàsiques de BaseSignatura i afegirà les especifiques per la validacions. Aquesta classe funciona de la mateixa forma que la classe de `Creació` i pràcticament realitza les mateixes accions amb petites variacions com que el missatge DSS que genera és de `verifyRequest` (de verificació de signatura en comptes de `signRequest`, creació de signatura). Per això no entrarem en més detall d'aquesta classe.

ValidacioCertificat: Com hem comentat aquestes classes hereten de les pare que les aporten tota la infrastructura necessària per treballar i per tant la classe ValidacioCertificat es centrarà en la lògica de negoci pròpia per realitzar la petició de validació d'un certificat digital X509. Aquesta classe defineix:

- Completar el missatge DSS de validació de certificat `verifyRequest` i el DigitalSignatureContentVo a través de la implementació d'aquest mètode específic. En aquest s'agafa la informació del Vo com el certificat i atributs addicionals i es crea l'estructura oportuna del missatge DSS de petició de signatura segons la descripció feta d'aquesta en l'apartat de missatges DSS:

```
protected void completarVerifyRequest (DigitalSignatureContentVo)
```

- Amb aquesta petició complerta, la classe para Validació i BaseSignatura realitzaran la connexió al portSoap i endpoint del servei de PSIS per realitzar la petició.

```
log.info("Crida WS PSIS");
SignResponseDocument
responseDocument=port.sign(this.requestDocument);
```

- Obté la resposta del servei i la parsetja o tracta afegint-lo al hash d'atributs resultants del nostre Vo. Un hashmap d'atributs on la key serà el nom dels atributs que ens interessen en el procés i el valor el contingut de l'atribut retornat. Per exemple: DigitalSignatureContentVo.atributs.get(ApplicationConfig.NIF) retornarà el valor del NIF

obtingut a la validació. Els atributs obligatoris sempre vindran complimentats o amb valor buit i els atributs opcionals s'afegiran posteriorment.

```
protected HashMap<String, String> generaHashResultat(VerifyResponse response)
```

- Aquest mètode crea un Vo per defecte que obté informació de prova com un certificat digital de prova i altres paràmetres del fitxer de propietats. En un cas real aquest mètode no s'utilitza i es carreguen dades reals del certificats a traves dels mètodes i atributs del DigitalSignatureContentVo.

```
public DigitalSignatureContentVo createDefaultVO() throws SignaturaInvalidAttributExcepcio, IOException {
```

ValidacioSignatura: Com hem comentat aquestes classes hereten de les pare que les aporten tota la infrastructura necessària per treballar i per tant la classe ValidacioSignatura es centrarà en la lògica de negoci pròpia per realitzar la petició de validació d'una signatura de tipus CMS detached, ja sigui una signatura CMS detached amb el document, sense o bé una signatura CMS detached però del resum del document (del hash del document). Aquesta classe defineix:

- Completar el missatge DSS de petició de signatura `verifyRequest` i el DigitalSignatureContentVo a través de la implementació d'aquest mètode específic. En aquest s'agafa la informació del Vo com el certificat i atributs addicionals i es crea l'estructura oportuna del missatge DSS de petició de signatura segons la descripció feta d'aquesta en l'apartat de missatges DSS:

```
protected void completarVerifyRequest (DigitalSignatureContentVo)
```

- Podem veure que en aquest mètode afegim el tipus de perfil del protocol que volem utilitzar, en aquest cas el perfil estes de CATCert XSS.

```
vo.setPerfil(ConstantsApplicacio.PROTOCOL_perfilDSS);
```

- També afegim a la llista d'atributs els atributs per defecte d'aquest tipus de validació CMS

```
vo.getAttributsOpcionals().addAll(Arrays.asList(ConstantsApplicacio.SignatureCMSAtributsDefecte));
```

- Aquí creem la resta de l'estructura del missatge DSS de validació de signatures CMS, demanant les opcions optatives i la resta d'atributs opcionals que vulguem per l'operació. Aquesta estructura és la del cas SignatureObject. Base64Signature que vàrem veure a capítol de missatges DSS.

- Després comprovant quina de les signatures CMS estem utilitzant inclourem el hash del document, el document o simplement la signatura CMS en el missatge DSS de validació.

- Amb aquesta petició complerta, la classe para Validació i BaseSignatura realitzaran la connexió al portSoap i endpoint del servei de PSIS per realitzar la petició.

```
log.info("Crida WS PSIS");
SignResponseDocument responseDocument=port.sign(this.requestDocument);
```

- Com en la resta de classes, obtenim la resposta del servei i la parsetja o tractem afegint-lo al hash d'atributs resultants del nostre Vo.

  ```
  protected HashMap<String, String> generaHashResultat(VerifyResponse response)
  ```

- També tenim el mètode que crea un Vo per defecte que obté informació de prova, com un signatures de prova per realitzar la petició de validació d'una signatura de tipus CMS detached, ja sigui una signatura CMS detached amb el document, sense o bé una signatura CMS detached però del resum del document (del hash del document) del fitxer de propietats. En un cas real aquest mètode no s'utilitza i es carreguen dades reals de les signatures a traves dels mètodes i atributs del DigitalSignatureContentVo.

  ```
  public DigitalSignatureContentVo createDefaultVO() throws Exception
  ```

ValidacioSignaturaPDF: Com hem comentat aquestes classes hereten de les pare que les aporten tota la infrastructura necessària per treballar i per tant la classe ValidacioSignatura es centrarà en la lògica de negoci pròpia per realitzar la petició de validació de signatures de PDF. Aquest cas és un servei diferent que ofereix CATCert (com vàrem veure a la fase d'anàlisi funcional i orgànica) que permet validar signatures de PDF amb particularitats, com afegir raó de signatura i d'altres, per això utilitzarem el perfil diferent: urn:oasis:names:tc:dss:1.0:profiles:DSS_PDF. La resta de mètodes de la classe funcionen de forma similar al comentats en les classes anteriors.

ValidacioSegellDeTemps: Aquesta classe es centrarà en la lògica de negoci pròpia per realitzar la petició de validació de segell de temps. Aquest cas és un servei diferent que ofereix CATCert (com vàrem veure a la fase d'anàlisi funcional i orgànica) que permet validar segell de temps, per això utilitzarem el perfil diferent: urn:oasis:names:tc:dss:1.0:profiles:timestamping. La resta de mètodes de la classe funcionen de forma similar al comentats en les classes anteriors.

CreacioSegellDeTemps: Aquesta classe es centrarà en la lògica de negoci pròpia per realitzar la petició de creació de segell de temps. Aquest cas és un servei diferent que ofereix CATCert (com vàrem veure a la fase d'anàlisi funcional i orgànica) que permet validar segell de temps, per això utilitzarem el perfil diferent: urn:oasis:names:tc:dss:1.0:profiles:timestamping.

Hi ha una important particularitat en aquesta funcionalitat, en la creació de segell de temps s'ha d'afegir el certificat de la TSA, com CATCert per a que amb aquest es realitzi la signatura que garanteix el moment de temps.

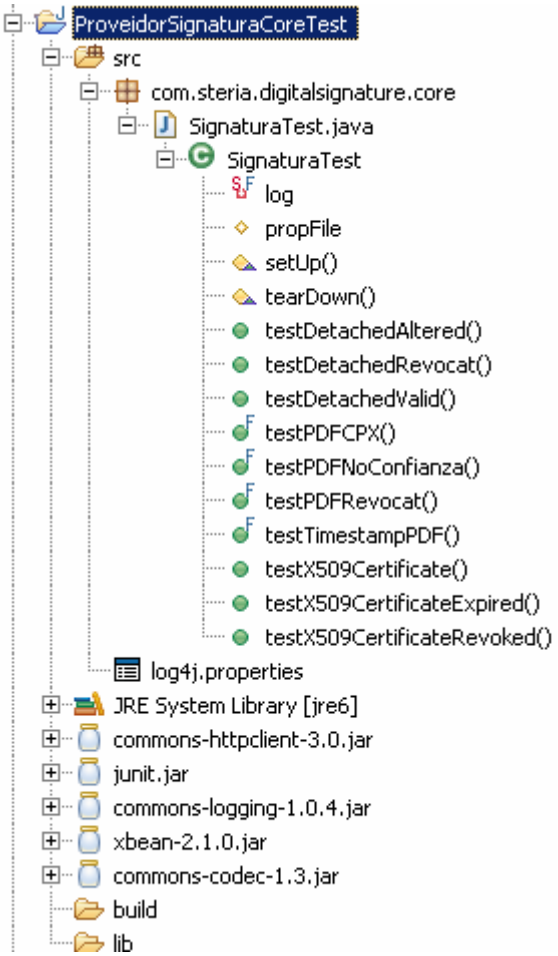
```
   X509DataType x509data = key.addNewX509Data();
        // S'afegeix el certificat de TSA de CATCert en Base64
        XmlBase64Binary b64certificate =
x509data.addNewX509Certificate();

    // Atributs d'Esquema
    public static final String
SCHEMA_XMLtimestamp="oasis:names:tc:dss:1.0:core:schema:XMLTimeStampToken
";
```

La resta de mètodes de la classe funcionen de forma similar al comentats en les classes anteriors.

5.4.2 ProveidorSignaturaCoreTest.

En aquest apartat entrarem en profunditat en el component de verificació ProveidorSignaturaCoreTest que ens permetrà programar casos de prova amb la llibreria JUnit per provar les funcionalitats implementades en el mòdul ProveidorSignaturaCore. Més endavant en la Fase IV: Verificació de la construcció i lliurament, podrem analitzar el resultat de les proves.

5.4.2.1 Estructura del projecte ProveidorSignaturaCoreTest:



Paquet que conté el codi font.

Paquet que conté el nucli de la lògica de negoci del testeig.

5.4.2.2 Llibreries necessàries:

XMLBeans 2.1.0: XML parser, llibreria per a facilitar el tractament d'arxius XML definits amb XSD.

Commóns-httpclient-3.0: És un framework que ens proporciona una API per abstraure de les comunicacions sota HTTP i realitza aquestes tasques de comunicació del costat del client.

Commóns-loggin-1.0.4: És una llibreria que ens ajuda a realitzar el registre i formateig de dades i d'informació i log del programari, ja sigui sortida per pantalla o a fitxer.

JUnit 3: És un framework que ens permet la validació de la construcció, programant proves unitàries de forma que comprovarem si el funcionament dels mètodes i components es comporten com esperem i segons es va definir funcional i orgànicament. En aquest projecte aquesta llibreria també tindrà una importància rellevant donat que ens servirà per la Fase IV: Verificació de la construcció i lliurament.

JDK v1.5 o superior: Paquet bàsic de desenvolupament Java.

5.4.2.3 Codi font, classes i mètodes:

SignaturaTest: Aquesta classe es centrarà en la lògica de negoci pròpia per realitzar les proves unitaries. Tindrem diferents mètodes per a provar cadascun de les primitives PKI implementades.

Hem organitzat aquest paquet ProveidorSignaturaCoreTest com a un paquet independent en comptes d'afegir la classe a ProveidorSignaturaCore perquè així podem realitzar la prova definitiva des de fora del paquet core i crear el ProveidorSignaturaCore.jar, provar d'incloure'l en un altre paquet i veure que es poden realitzar totes les crides de forma correcte i amb les llibreries necessàries.

En la Fase IV "Verificació de la construcció i lliurament" podrem veure com funcionen en detall aquests mètodes.

5.5 Capítol 5: Fase IV - Verificació de la construcció i lliurament

En aquesta fase realitzarem la verificació de la construcció a través del framework JUnit. També indicarem els paquets, llibreries i elements tècnics necessaris per lliurar el codi del projecte, instal·lar-ho i poder-ho utilitzar.

5.5.1 Programari, llibreries i instal·lació

5.5.1.1 Productes obtinguts

1) ProveïdorSignaturaCore.jar: És una llibreria que ens permetrà a partir d'una API simplificada utilitzar serveis de Signatura Digital. Les Constants de Configuració i parametritzacions bàsiques estan predefinides de forma que les funcionalitats bàsiques tenen un fàcil accés i us. Aquesta llibreria també permet funcionalitats més complexes de Signatura afegint les parametritzacions avançades. Aquesta llibreria s'anirà ampliant amb noves funcionalitats.

A) Paquet complet amb llibreries incloses: Per utilitzar aquesta llibreria s'ha de desplegar al servidor d'aplicacions el ProveïdorSignaturaCore.jar que conté totes les llibreries necessàries.

Per fer això seguirem els següents pasos:

- Per obtenir aquesta utilitat hem de llançar l'script ant que està dintre del paquet ProveidorSignaturaCore\proveidor_build.xml
- Aquest ens empaquetarà el projecte i el deixarà preparat per executar a \ProveidorSignaturaCore\inst\ ProveidorSignaturaCore.jar

B) Paquet complet amb llibreries excloses: Si les llibreries incloses poden donar problemes de versions o de classloaders amb d'altres llibreries ja incloses en un projecte global o servidor d'aplicacions, s'hauria de desplegar la llibreria ProveïdorSignaturaCore.jar amb les llibreries necessàries no incloses per a que es puguin incloure al classpath només les necessàries que no col·lisionessin en les del projecte en curs.

Per fer això seguirem els següents pasos:

- Llançar l'script ant que està dintre del paquet ProveidorSignaturaCore\proveidor_libs_build.xml
- Aquest ens empaquetarà el projecte amb totes les llibreries necessaries pel funcionanment incloses en el paquet i amb els classpath necesaris configurats i el deixarà preparat per executar a \ProveidorSignaturaCore\inst\ ProveidorSignaturaCore.jar

Les llibreries necessàries són:

- JDK v1.5 o superior

- psis-beans.jar
- XMLBeans 2.1.0
- XFire 1.1.2
- Commóns-httpclient-3.0
- Commóns-loggin-1.0.4
- Wsdl4j-1.5.1
- Wss4j-1.5.0
- Jdom-1.0

2) ProveidorSignaturaCoreTest: Paquet que ens permet provar les classes de signatura a través de la classe que exten TestCase SignaturaTest. Aquesta classe es centrarà en la lògica de negoci pròpia per realitzar les proves unitaries. Tindrem diferents mètodes per a provar cadascun de les primitives PKI implementades.

Hem organitzat aquest paquet ProveidorSignaturaCoreTest com a un paquet independent en comptes d'afegir la classe a ProveidorSignaturaCore perquè així podem realitzar la prova definitiva des de fora del paquet core i crear el ProveidorSignaturaCore.jar, provar d'incloure'l en un altre paquet i veure que es poden realitzar totes les crides de forma correcte i amb les llibreries necessàries.

Les llibreries necessàries d'aquest paquet són:

- JDK v1.5 o superior:
- XMLBeans 2.1.0
- Commóns-httpclient-3.0
- commóns-codec-1.3
- Commóns-loggin-1.0.4
- JUnit 3

3) ProveïdorSignatura.ear: El component de ProveidorSignatura estarà format per subcomponents o paquets que formaran una gran aplicació (enterprise o EAR) i que implementaran la lògica de negoci necessària per respondre a les necessitats de l'actor (appl. externa) segons els seus casos d'us. S'encarregarà de resoldre els casos d'us de l'actor appl. externa: validar certificat, normalització d'atributs, validar signatura, segell de temps, consultar documents signats i validar-los en un moment donat de temps. Aquest component donarà una interfície per accedir a les seves funcionalitats IProveidor. A aquesta interfície només podran accedir les aplicacions amb els permisos necessaris. Com hem vist en l'apartat anterior lliuràvem un .jar, una llibreria d'utilitats que hom podrà utilitzar al afegir al seu projecte. Si es vol incloure les

mateixes funcionalitats amb un component distribuït o EJB en una plataforma escalable donem la possibilitat de fer-ho convertint la llibreria en aquesta tecnologia.

Els paquets que formaran l'aplicació (EAR):

- **ProveidorSignaturaCore**: Aquest paquet és la llibreria que conté tota la lògica de negoci implementada, és qui realitza les crides a les funcionalitats de PSIS a través de la implementació DSS mitjançant XML i comunicació a través de webservices i SOAP. Aquest component conté les classes de configuració, els objectes Vo a utilitzar i els mètodes principals així com les llibreries i classes d'ajuda (utilities).

  Aquest component es pot utilitzar com a llibreria independent .jar. Com per exemple afegir la llibreria en un altre projecte i utilitzar les seves funcions públiques.

  Per exemple, en el cas de la plataforma de tramitació amb que està relacionat aquest projecte de signatura, aquest component com a llibreria ProveidorSignaturaCore.jar s'utilitzarà en el mòdul de SSO. Servirà per realitzar amb l'Open SSO una autentificació amb certificats digitals de l'usuari en la plataforma de tramitació. De manera que l'usuari es connecta amb certificats digitals s'agafa les seves credencials i s'envien a través d'aquesta llibreria a validar el certificat i normalitzar l'atribut del NIF, la classe "Validacio" tornarà si és vàlid el certificat i quin és el NIF normalitzat del l'usuari en qüestió, d'aquesta forma se li donarà accés a l'usuari amb les credencials que aquest hagi de tenir en la plataforma de contractació. Aquest usuari quan arribi al Porta Signatures, de forma automàtica, només se li permetrà signar amb el mateix certificat de l'usuari autenticat en la plataforma (via les credencials recollides en el SSO).

- **ProveidorSignaturaEJB**: Aquest component s'implementarà amb un EJBs de sessió ProveidorSignaturaEJB seguint per tant el patró facade (session-facade: aconseguim desacoblar la capa del model de la capa de negoci cadascú fa la seva feina/responsabilitat i reduïm notablement el trànsit de xarxa). Aquest component permetrà el desplegament del component entrerprise ProveidorSignatura en un servidor d'aplicacions de forma que les seves funcionalitats públiques estiguin disponibles per realitzar peticions des de qualsevol punt de la plataforma de tramitació de forma distribuïda.

- **ProveidorSignaturaEJBClient**: Interfície del component per a lliurar als programadors o programari que vulgui utilitzar el nostre component de ProveidorSignatura.

- **ProveidorSignaturaEAR**: Aquest paquet ens permetrà empaquetar tots els paquets en un de sol com a únic recurs distribuïble i usable.

4) JWS: En les extensions de futur obtindrem el frontal de Signatura Digital, que està basat en la tecnologia Java, concretament és Java Web Start, vindrà signada en tots els seus components i treballarà amb els recursos necessaris i les comunicacions necessàries amb el servidor (JBoss) de forma segura (HTTPS). Aquesta aplicació té un mòdul que requereix compilació amb JDK 1.6. Per les diverses compilacions, signatura dels diversos mòduls i l'ensamblaje es proveirà d'un script ant que ho manega.

El fet de que la implementació sigui en java obre el ventall de possibilitats de sistemes operatius i navegadors que poden suportar el seu ús.

El JWS constarà de 5 paquets (jar) que es descarreguen amb el control de versions implícit que proporciona el JWS amb l'arquitectura JNLP (Java Network Launching Protocol). Amb aquesta tecnologia podem definir quins recursos necessitem, quines versions de cadascun i quins paràmetres necessiten: fitxers del projecte: JnlpSteria\files\app\signatura.jnlp i JnlpSteria\version.xml. D'aquesta forma automàticament es descarregarà o actualitzarà aquells recursos que siguin necessaris segons les indicacions d'aquests fitxers, també se seleccionarà que el client requereixi la versió de JRE 1.6 que en cas de manca s'instal·larà automàticament.

- JWSSteriax.x.jar – Contindrà un paquet amb la lògica del JWS i les diferents implementacions de signatura i magatzems de certificats. I un segon paquet de llibreries d'Apache Commóns per a connexions http, necessàries per a poder generar segells de temps (RFC3161 o XML). Suportarà comunicació bidireccional, utilitzant sockets segurs amb el servidor d'aplicacions. Des de l'HTML (PortaSignatures) s'enviaran les cookies del procés en que es troba l'usuari i el JWS es comunicarà posteriorment amb el servidor d'aplicacions (PortaSignatures) recuperant la sessió de l'usuari i finalitzant el procés.

- CATCertCMSlibx.x.jar – Es tracta d'un paquet reduït de la llibreria de Bouncy-Castle per a la generació de signatures CMS.

- CATCertXMLlibx.x.jar – Conté el paquet d'Apache XML. Un paquet reduït amb petites modificacions de les llibreries d'Apache Security per a la generació de signatures XML.

- CATCertPDFlibx.x.jar – Conté un paquet reduït de la llibreria iText per al tractament de documents PDF. Permetrà incrustar signatures CMS en els documents, i que aquestes siguin vàlides (i visibles) si el document s'obre amb l'eina d'Adobe, l'Acrobat Reader.

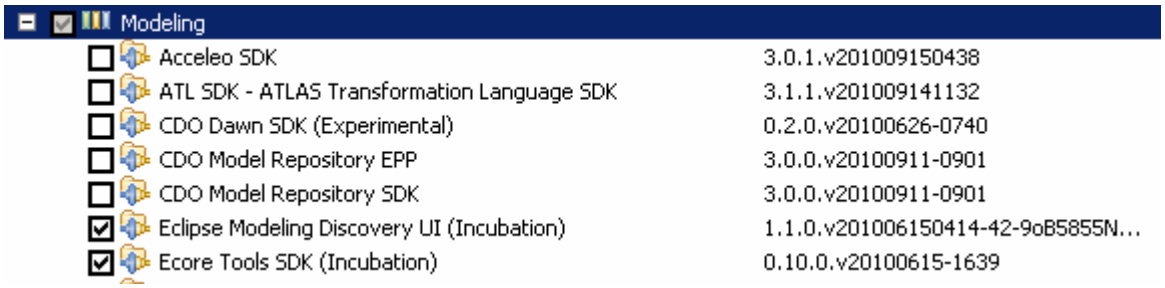
Com hem vist a la documentació es disposarà de multitud de paràmetres que permetran configurar l'eina segons les necessitats. Per tal de facilitar-ne l'ús, es llistaran agrupats segons funcionalitat, donant detalls del que cada un implica i de com s'utilitza.

5.5.1.2 IDE i eines de desenvolupament

Eclipse Helios (3.6.1): Eclipse Java EE IDE for Web Developers. IDE obert de programació.

Per a poder desenvolupar el projecte, la llibreria principal ProveidorSignaturaCore, s'ha utilitzat l'Eclipse Java EE IDE for Web Developers Version: Helios Service Release 1 Build id: 20100917-0705 i el plugin per a poder utilitzar l'Ant inclos en el package de Modelat:

org.eclipse.jdt 3.6.0 proveeix el suport per l'ant



Per a poder afegir aquest projecte al projecte principal de tramitació electrònica i integrar-ho amb les característiques de l'IDE que aquest utilitza s'ha utilitzat:

Xdoclet que ens ha permès crear l'estructura dels paquets d'EJB i generar automàticament les parts clients. També ens permet a partir d'atributs generar fitxers de deploy o descriptors XML com el jboss.xml o les definicions dels EJBs de manera que nosaltres només escrivim les anotacions en el codi font i Xdoclet genera tots els fitxers necessaris per nosaltres.

5.5.2 Verificació de la construcció

JUnit 3: És un framework que ens permet la validació de la construcció, programant proves unitàries de forma que comprovarem si el funcionament dels mètodes i components es comporta com esperem i segons es va definir funcional i orgànicament.

Amb aquesta llibreria comprovarem que els mètodes definits en la fase AF i AO estiguin complint els requeriments inicials i donin els resultats esperats. Ens permetrà controlar l'execució de les classes i mètodes construïts per realitzar els diferents casos de prova, introduint paràmetres d'entrada i esperant els paràmetres de sortida esperats o provocant errors per garantir que estan detectats i tractats els errors possibles.

Per això programarem els JUnit per comprovar els casos d'èxit dels mètodes. Aquests casos seran tant els casos que, per exemple, la signatura s'ha validat correctament com els que la signatura surt invalida o dona un error de que el certificat està revocat o qualsevol altre error esperat en el resultat del procés o mètode PKI definits i inventariats a la documentació adjunta. Per tant un cas d'èxit que en el framework JUnit contemplarem con test correcte és també una fallida o resultat d'error esperada en el resultat del test donat que, per exemple, hem introduït paràmetres d'entrada com un certificat revocat o un document invàlid expressament per comprovar que el mètode tracta de forma correcte aquests casos d'excepció i dona també el resultat esperat.

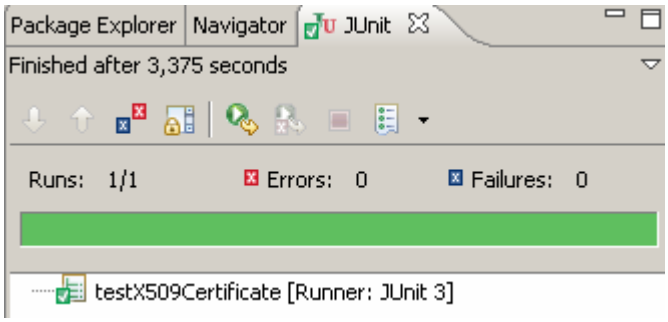
Aquest test garantitzen també que un cop han superat tots amb èxit les proves qualsevol canvi es tornen a llançar els test i d'aquesta forma comprovem els impactes de regressió, és a dir, que un mètode que segueix complint la seva especificació original després de modificacions en aquest o d'altres amb dependències.

El propi framework JUnit inclou formes de visualitzar els resultats (runners) que es poden veure en mode texte, gràfic (AWT o Swing) o com tasca Ant.

L'avantatge d'aquestes eines i IDE com l'Eclipse que permeten a través de plugins com aquest generar plantilles necessàries per la creació de les proves unitaries de les classes java de forma que faciliten al programador que es centri en la prova i el resultat esperat, deixant com hem comentat en d'altres casos tota la infrastructura adjacent en mans de les eines i aprofitant el temps per centrar el programador en les tasques de "valor".

Per utilitzar el JUnit a banda de importar el seu framework hem creat els casos de test estenen la classe de test ValdiacioSignatura de la classe TestCase. En les validacions i test cases podem veure en tot moment tot el procés pas a pas de la creació de la crida DSS de petició de servei a PSIS amb tot el missatge XML/DSS format i de la mateixa forma el seu resultat i totes les comunicacions.

5.5.3 Test de Validació de certificats digitals X509



testX509Certificate: Aquest valida un certificat X509 de manera que comprova que el certificat sigui vàlid, la persona és qui diu ser, no ha estat.

Paràmetres entrada: En aquesta prova passem com a paràmetre d'entrada el certificat personal d'identificació: "signaturadigital/cert/CARLOSVILAMATEOS.crt"

Procés: Per realitzar el test es crida a la classe ValidacioCertificat que realitzarà els passos descrits en l'apartat de construcció d'aquesta classe i al javadoc. S'ompliran la resta de paràmetres d'entrada obligatoris per a formar el missatge de petició de servei de validació de certificat amb tota l'estructura DSS necessària comentada als capítols anteriors.

Resultat: Finalment retornarà l'objecte DigitalSignatureContentVo amb els atributs de sortida:

- Un hashmap d'atributs on la key serà el nom dels atributs que ens interessen en el procés i el valor el contingut de l'atribut retornat. Per exemple: DigitalSignatureContentVo.atributs.get(ApplicationConfig.NIF) retornarà el valor del NIF obtingut a la validació. Els atributs obligatoris sempre vindran complimentats o amb valor buit i els atributs opcionals s'afegiran posteriorment.

```
protected HashMap<?, ?> atributs;
```

- Un camp de resultat booleà que indicarà si el procés és correcte (true – s'ha validat positivament la signatura) o ha fallat (false – no s'ha validat la signatura). Pot ser correcte si es valida el certificat, si es valida la signatura, si es retorna segell de temps o pot se fals si dona qualsevol excepció (no hi ha connexió, etc.) o no es valida el certificat, no es valida la signatura, etc.

```
protected boolean procesOk;
```

- Un hashmap on afegirem tota la informació de validació i detalls de sortida del procés.

```
protected HashMap<String, String> msg;
```

Per verificar que el test es satisfactori, ho comprovarem a través d'assertions que el missatge de detall del resultat de la validació en el format DSS sigui igual a l'expressió que indica que el certificat és vàlid:

```
vo.getMsg().get(ConstantsApplicacio.MSG_ResultMinor);
assertTrue(minor.contains("valid:certificate:Definitive"));
```

Revisem els logs del procés pas a pas:

Pas 1: Petició Validació Certificat

```
INFO: BaseSignatura (constructor) done. URL = http://psisbeta.catcert.net/psis/catcert-test/dss
22-dic-2010 12:42:41 com.steria.digitalsignature.core.ValidacioCertificat createDefaultVO
INFO: createDefaultVO() init
22-dic-2010 12:42:41 com.steria.digitalsignature.core.ValidacioCertificat createDefaultVO
INFO: default certFile=[signaturadigital/cert/CARLOSVILAMATEOS.crt]
22-dic-2010 12:42:41 com.steria.digitalsignature.core.ValidacioCertificat createDefaultVO
INFO: createDefaultVO() done
22-dic-2010 12:42:41 com.steria.digitalsignature.core.SignaturaTest testX509Certificate
INFO: testing certFile=[signaturadigital/cert/CARLOSVILAMATEOS.crt]
22-dic-2010 12:42:41 com.steria.digitalsignature.core.Validacio validar
INFO: validar() init. [PERFIL: urn:oasis:names:tc:dss:1.0:profiles:XSS]
22-dic-2010 12:42:41 com.steria.digitalsignature.core.BaseSignatura iniComunicacio
INFO: iniComunicacio() init
22-dic-2010 12:42:43 com.steria.digitalsignature.core.BaseSignatura iniComunicacio
INFO: iniComunicacio() done
22-dic-2010 12:42:43 com.steria.digitalsignature.core.Validacio validar
INFO: VerifyRequestDocument:
```

1) Iniciem el procés iniciant els paràmetres base (gràcies a la classe BaseSignatura) com la URL de validació del servei de PSIS, en aquest cas l'entorn de test.
2) Creem el Vo per defecte amb els atributs necessaris com el tipus de PerfilDSS.
3) Afegim el certificat a validar
4) Indiquem el perfil de treball XSS.
5) Inicialitzem la comunicació amb el port webservice.
6) Cridem a la validació remota del certificat personal.
7) S'envia el missatge DSS de petició a PSIS.

```
INFO: <VerifyRequest Profile="urn:oasis:names:tc:dss:1.0:profiles:XSS" xmlns="urn:oasis:names:tc:dss:1.0:core:schema"
xmlns:xd="http://www.w3.org/2000/09/xmldsig#">
  <OptionalInputs>
    <ReturnProcessingDetails/>
  </OptionalInputs>
  <SignatureObject>
    <Other>
      <xd:X509Data>

<xd:X509Certificate>LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tDQpNSUlJUmpDQ0J5NmdBd0lCQWdJREFkOStNQTBHQ1NxR1N
JYjNEUUVCQlFVQU1JSUJNVEVMTUFrR0ExVUVCaE1DDQpSVk14T3pBNUJnTlZCQW9UTWtGblpXNWphV0VnUTJGMFlXeGhibUVn
WkdVZ1EyVnlkR2xtYVdOaFkybHZJQ2hPDQpTVVlnVVMwd09EQXhNVGMyTFVrcE1UUXdNZ1lEVlFRSEV5dFFZWE56WVhSblpTQmta
U0JzWVNCRGIyNWpaWEJqDQphVzhnTVRFZ01EZ3dNRGdnUW1GeVkyVnNiMjVoTVM0d0xBWURWUVFMRXlWVFpYSjJaV2x6SUZC
MVlteHBZM01nDQpaR1VnUTJWeWRHbG1hV05oWTJsdklFVkRWaTB5TVRVd013WURWUVFMRXl4V1pXZGxkU0JvZEhSd2N6b3ZMM
2QzDQpkeTVqWVhSalpYSjBMbTVsZEM5MlpY
…
VdOaElHUmxJR05sY25ScFptbGpZV05wYnlCa1
MGRIQTZMeTlsY0hOalpESXVZMkYwWTJWeW
UhwTUlIbUJnd3JCZ0VFQWZWNEFRTUJWZ0V
GNuUXVibVYwTDNabGNrbEVRMkYwDQpNSU
1pwWTJGMElIQmxjbk52DQpibUZzSUVsRVEwRlVMQ0J5WldOdmJtVm5kWFFnWkNkcFpHVnVkR2xtYVdOaFkybnpMQ0J6YVdkdVlYUj
FjbUVnDQphU0I0YVdaeVlYUWdaR1VnWTJ4aGMzTmxJREInYVc1a2FYWnBaSFZoYkM0Z1ZtVm5aWFVnYUhSMGNITTZMeTkzDQpk
M2N1WTJGMFkyVnlkQzV1WlhRdmRtVnlTVVJEWVhRd0xRWURWUjBKQkNZd0pEQVFCZ2dyQmdFRkJRY0pCREVFDQpFd0pGVXpB
UUJnZ3JCZ0VGQlFjSkJURUVFd0pGVXpBTkJna3Foa2lHOXcwQkFRVUZBQU9DQVFFQWRyaVNDV09EDQpUdklEc3FmeUk4UFF6WX
NMeUwrSW9LT1JjT2ZUbWZETnc0N2tZNEpGeWo0am53TFR6L1BPcmluYWFXcHBtTGxVDQo0MWVHajFQWk1VcXdTZTRiaGE5ZEZP
emJqcEtBdU9ReEZLV3ZkdzhJYXBna0NDVy84N3RoWHNhZFZYQm8xUnRrDQprQzlCN1MwVGJ1SERrL3NySzJuOXAzMXAxOFRQWX
l2ZU9RTEpmT01Lc3RxdGZXdmo1YTBneDNFRGpwK2RpV1diDQpoNVBZbk1NWHRHemh0S2ZFc3dpdkpBM1NiOEM5NlFxNGVyT0M1
aXMrTi9hTEpKSjJJVkRsQ1poZ0gxLzROUHprDQpwcFdTTjNlQ3NwWVd1SExFeGJuRDZjWnIrMldueHVEaXo2SkFmK1g1Wkx0SXpHaits
Ry9EY2VrVE1kbldRT0xxDQoxRnZlNStxZDlScm5hZz09DQotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0tDQo=</xd:X509Certificate>
      </xd:X509Data>
    </Other>
  </SignatureObject>
</VerifyRequest>
```

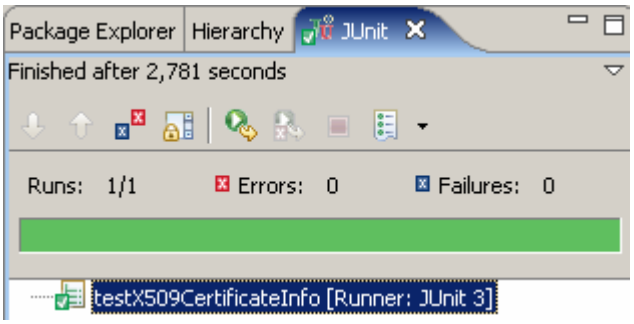
Formació del missatge DSS enviat de petició de validació de certificat.
El missatge es de tipus VerifyRequest en un objecte de tipus SignatureObject al tag Other com vàrem veure a l'apartat de formació de missatges DSS. En aquest tag Other\X509Data inseríem el nostre certificat codificat en base 64

Pas 2: Resposta Validació Certificat

1) Podem veure el resultat de la petició de validació a PSIS en el missatge DSS de resposta VerifyResponseDocument.
2) En el detall del missatge de resposta podem veure que el certificat és vàlid això s'emmagatzemarà al DigitalSignatureContentVo HashMap key ResultMinor

```
22-dic-2010 12:42:43 com.steria.digitalsignature.core.Validacio validar
INFO: Crida WS PSIS
22-dic-2010 12:42:45 com.steria.digitalsignature.core.Validacio validar
INFO: VerifyResponseDocument:
22-dic-2010 12:42:45 com.steria.digitalsignature.core.Validacio validar
INFO: <dss:VerifyResponse Profile="urn:oasis:names:tc:dss:1.0:profiles:XSS" xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <dss:Result>
    <dss:ResultMajor>urn:oasis:names:tc:dss:1.0:resultmajor:Success</dss:ResultMajor>
    <dss:ResultMinor>urn:oasis:names:tc:dss:1.0:profiles:XSS:resultminor:valid:certificate:Definitive</dss:ResultMinor>
  </dss:Result>
  <dss:OptionalOutputs>
    <dss:ProcessingDetails>
      <dss:ValidDetail Type="urn:oasis:names:tc:dss:1.0:detail:ValidityInterval">
        <dss:Message xml:lang="en">The signing key is inside its static validity interval.</dss:Message>
      </dss:ValidDetail>
      <dss:ValidDetail Type="urn:oasis:names:tc:dss:1.0:detail:IssuerTrust">
        <dss:Message xml:lang="en">The issuer of the given key is trusted.</dss:Message>
      </dss:ValidDetail>
      <dss:ValidDetail Type="urn:oasis:names:tc:dss:1.0:detail:RevocationStatus">
        <dss:Message xml:lang="en">The signing key is not revoked.</dss:Message>
      </dss:ValidDetail>
    </dss:ProcessingDetails>
  </dss:OptionalOutputs>
</dss:VerifyResponse>
22-dic-2010 12:42:45 com.steria.digitalsignature.core.Validacio validar
INFO: Validació realitzada, resposta rebuda.
22-dic-2010 12:42:45 com.steria.digitalsignature.core.Validacio validar
INFO: Hash Atributs: {}
22-dic-2010 12:42:45 com.steria.digitalsignature.core.Validacio validar
INFO: Hash Msg(returned): {dss:ResultMinor=urn:oasis:names:tc:dss:1.0:profiles:XSS:resultminor:valid:certificate:Definitive,
dss:ResultMajor=urn:oasis:names:tc:dss:1.0:resultmajor:Success}
22-dic-2010 12:42:45 com.steria.digitalsignature.core.Validacio validar
INFO: validar() done. [PERFIL: urn:oasis:names:tc:dss:1.0:profiles:XSS]
```

Podem veure també més detalls dels missatges de validació, com que es confia en el certificat i que no està revocat.



testX509CertificateInfo: Aquest valida un certificat X509 de manera que comprova que el certificat sigui vàlid, la persona és qui diu ser, no ha estat revocat. Però a banda demanem les funcionalitats esteses de la normalització d'atributs del certificat i per tant que ens retorni els atributs que demanem com per exemple al Nif amb el que podríem associar al login de la plataforma juntament amb el certificat digital i així realitzar una validació SSO amb certificat.

Paràmetres entrada: En aquesta prova passem com a paràmetre d'entrada el certificat personal d'identificació: "signaturadigital/cert/CARLOSVILAMATEOS.crt"

Procés: El procés serà similar al test anterior.

Resultat: El resultat es guardarà als mateixos objectes que els definits en la resta de test al objecte DigitalSignatureContentVo.

Per verificar que el test es satisfactori, ho comprovarem a través d'assertions, en aquest cas comprovarem que el missatge de detall del resultat de la validació en el format DSS sigui igual a l'expressió que indica que el certificat és vàlid:

```
vo.getMsg().get(ConstantsApplicacio.MSG_ResultMinor);
assertTrue(minor.contains("valid:certificate:Definitive"));
```

I també comprovarem que ens han retornat atributs com el NIF:

```
String nif =(String)(vo.getAtributs().get(ConstantsApplicacio.CERT_NIF));
assertTrue(nif.contains("46358509P"));
```

Revisem els logs del procés pas a pas:

```
INFO: BaseSignatura (constructor) done. URL = http://psisbeta.catcert.net/psis/catcert-test/dss
22-dic-2010 12:42:41 com.steria.digitalsignature.core.ValidacioCertificat createDefaultVO
INFO: createDefaultVO() init
22-dic-2010 12:42:41 com.steria.digitalsignature.core.ValidacioCertificat createDefaultVO
INFO: default certFile=[signaturadigital/cert/CARLOSVILAMATEOS.crt]
22-dic-2010 12:42:41 com.steria.digitalsignature.core.ValidacioCertificat createDefaultVO
INFO: createDefaultVO() done
22-dic-2010 12:42:41 com.steria.digitalsignature.core.SignaturaTest testX509Certificate
INFO: testing certFile=[signaturadigital/cert/CARLOSVILAMATEOS.crt]
22-dic-2010 12:42:41 com.steria.digitalsignature.core.Validacio validar
INFO: validar() init. [PERFIL: urn:oasis:names:tc:dss:1.0:profiles:XSS]
22-dic-2010 12:42:41 com.steria.digitalsignature.core.BaseSignatura iniComunicacio
INFO: iniComunicacio() init
22-dic-2010 12:42:43 com.steria.digitalsignature.core.BaseSignatura iniComunicacio
INFO: iniComunicacio() done
22-dic-2010 12:42:43 com.steria.digitalsignature.core.Validacio validar
INFO: VerifyRequestDocument:
22-dic-2010 12:42:43 com.steria.digitalsignature.core.Validacio validar
```

1) Iniciem el procés iniciant els paràmetres base (gràcies a la classe BaseSignatura) com la URL de validació del servei de PSIS, en aquest cas l'entorn de test.
2) Creem el Vo per defecte amb els atributs necessaris com el tipus de PerfilDSS.
3) Afegim el certificat a validar
4) Indiquem el perfil de treball XSS.
5) Inicialitzem la comunicació amb el port webservice.
6) Cridem a la validació remota del certificat personal.
7) S'envia el missatge DSS de petició a PSIS.

Pas 1: Petició Validació Certificat i normalització d'atributs de certificat

Podem veure com es forma un missatge dss de VerifyRequest.
Podem veure que els namespaces utilitzen els prefixes de dss i xss, d'aquesta forma podem utilitzar les estructures esteses de CATCert per a sol·licitar informació addicional (tag dss:ReturnProcessingDetails) com la normalització de certs atributs del certificat, com per exemple el NIF.

```
INFO: <VerifyRequest Profile="urn:oasis:names:tc:dss:1.0:profiles:XSS" xmlns="urn:oasis:names:tc:dss:1.0:core:schema"
xmlns:urn="urn:oasis:names:tc:dss:1.0:profiles:XSS" xmlns:xd="http://www.w3.org/2000/09/xmldsig#">
  <OptionalInputs>
    <ReturnProcessingDetails/>
    <urn:ReturnX509CertificateInfo>
      <urn:AttributeDesignator Name="urn:catcert:psis:certificateAttributes:KeyOwnerNIF"/>
      <urn:AttributeDesignator Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:Version"/>
      <urn:AttributeDesignator Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SerialNumber"/>
      <urn:AttributeDesignator Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:Signature"/>
      <urn:AttributeDesignator Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SignatureAlgorithm"/>
      <urn:AttributeDesignator Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:IssuerDistinguishedName"/>
      <urn:AttributeDesignator Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SubjectDistinguishedName:givenName"/>
      <urn:AttributeDesignator Name="Urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SubjectDistinguishedName:surname"/>
      <urn:AttributeDesignator Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SubjectPublicKeyAlgorithm"/>
      <urn:AttributeDesignator Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SubjectPublicKey"/>
      <urn:AttributeDesignator Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SubjectDistinguishedName:organizationName"/>
    </urn:ReturnX509CertificateInfo>
  </OptionalInputs>
  <SignatureObject>
    <Other>
      <xd:X509Data>
<xd:X509Certificate>LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tDQpNSUlJUmpDQ0J5NmdBd0lCQWdJREFkOStNQTBHQ1NxR1NJYjNEUUV
CQlFVQU1JSUJNVEVMTUFrR0ExVUVCaE1DDQpSVk14T3pBNUJnTlZCQW9UTWtGblpXNWphV0VnUTJGMFlXeGhibUVnWkdVZ1EyVnlkR2xtYV
dOaFkybHZJQ2hPDQpTVVlnVVMwd09EQXhNVGMyTFVrcE1UUXdNZ1lEVlFRSEV5dFFZWE56WVhSblpTQmtaU0JzWVNCRGIyNWpaWEJqDQp
hVzhnTVRFZ01EZ3dNRGdnUW1GeVkyVnNiMjVoTVM0d0xBWURWUVFMRXlWVFpYSjJaV2x6SUZCMVlteHBZM01nDQpaR1VnUTJWeWRHbG1h
V05oWTJsdklF
...
...
LT1JjT2ZUbWZETnc0N2tZNEpGeWo0am53TFF                                                                                V3
ZkdzhJYXBna0NDVy84N3RoWHNhZFZYQm8xU                                                                                 mo1
YTBneDNFRGpwK2RpV1diDQpoNVBZbk1NWH                                                                                  Hpr
DQpwcFdTTjNlQ3NwWVd1SExFeGJuRDZjWnIr                                                                                09D
QotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0tDQo=</xd:X509Certificate>
      </xd:X509Data>
    </Other>
  </SignatureObject>
</VerifyRequest>
```

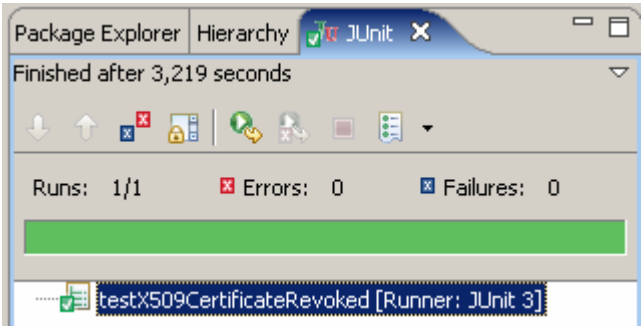
Formació del missatge DSS enviat de petició de validació de certificat.
El missatge es de tipus VerifyRequest en un objecte de tipus SignatureObject al tag Other com vàrem veure a l'apartat de formació de missatges DSS. En aquest tag Other\X509Data inserïem el nostre certificat codificat en base 64

Pas 2: Resposta Validació Certificat i normalització d'atributs de certificat

1) Podem veure el resultat de la petició de validació a PSIS en el missatge DSS de resposta VerifyResponseDocument.
2) En el detall del missatge de resposta podem veure que el certificat és vàlid això s'emmagatzemarà al DigitalSignatureContentVo HashMap key ResultMinor

INFO: <dss:VerifyResponse Profile="urn:oasis:names:tc:dss:1.0:profiles:XSS" xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<dss:Result>
<dss:ResultMajor>urn:oasis:names:tc:dss:1.0:resultmajor:Success</dss:ResultMajor>
<dss:ResultMinor>urn:oasis:names:tc:dss:1.0:profiles:XSS:resultminor:valid:certificate:Definitive</dss:ResultMinor>
</dss:Result>
<dss:OptionalOutputs>
<dss:ProcessingDetails>
<dss:ValidDetail Type="urn:oasis:names:tc:dss:1.0:detail:ValidityInterval">
<dss:Message xml:lang="en">The signing key is inside its static validity interval.</dss:Message>
</dss:ValidDetail>
<dss:ValidDetail Type="urn:oasis:names:tc:dss:1.0:detail:IssuerTrust">
<dss:Message xml:lang="en">The issuer of the given key is trusted.</dss:Message>
</dss:ValidDetail>
<dss:ValidDetail Type="urn:oasis:names:tc:dss:1.0:detail:RevocationStatus">
<dss:Message xml:lang="en">The signing key is not revoked.</dss:Message>
</dss:ValidDetail>
</dss:ProcessingDetails>
<urn:X509CertificateInfo xmlns:urn="urn:oasis:names:tc:dss:1.0:profiles:XSS">
<urn:Attribute Name="urn:catcert:psis:certificateAttributes:KeyOwnerNIF">
<urn1:AttributeValue xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">46358509P</urn1:AttributeValue>
</urn:Attribute>
<urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:Version">
<urn1:AttributeValue xsi:type="xs:integer" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">3</urn1:AttributeValue>
</urn:Attribute>
<urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SerialNumber">
<urn1:AttributeValue xsi:type="xs:integer" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">122750</urn1:AttributeValue>
</urn:Attribute>
<urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:Signature">
<urn1:AttributeValue xsi:type="xs:base64Binary" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">driSCWODTvIDsqfyI8PQzYsLyL+IoKORcOfTmfDNw47kY4JFyj4jnwLTz/POrinaaWppmLlU41eGj1PZMUqwSe4bha9dFOzbjpKAuOQxFKWvdw8IapgkCCW/87thXsadVXBo1RtkkC9B7S0TbuHDk/srK2n9p31p18TPYyveOQLJfOMKstqtfWvj5a0gx3EDjp+diWWbh5PYnMMXtGzhtKfEswivJA3Sb8C96Qq4erOC5is+N/aLJJJ2IVDlCZhgH1/4NPzkppWSN3eCspYWuHLExbnD6cZr+2WnxuDiz6JAf+X5ZLtIzGj+lG/DcekTMdnWQOLq1Fve5+qd9Rrnag==</urn1:AttributeValue>
</urn:Attribute>
<urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SignatureAlgorithm">
<urn1:AttributeValue xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">1.2.840.113549.1.1.5</urn1:AttributeValue>
</urn:Attribute>
<urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:IssuerDistinguishedName">
<urn1:AttributeValue xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">CN=EC-IDCat,OU=Entitat publica de certificacio de ciutadans,OU=Vegeu https://www.catcert.net/verCIC-2 (c)03,OU=Serveis Publics de Certificacio ECV-2,L=Passatge de la Concepcio 11 08008 Barcelona,O=Agencia Catalana de Certificacio (NIF Q-0801176-I),C=ES</urn1:AttributeValue>
</urn:Attribute>
<urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SubjectDistinguishedName:givenName">
<urn1:AttributeValue xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">CARLOS</urn1:AttributeValue>
</urn:Attribute>
<urn:Attribute Name="Urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SubjectDistinguishedName:surname"/>
<urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SubjectPublicKeyAlgorithm">
<urn1:AttributeValue xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">RSA</urn1:AttributeValue>
</urn:Attribute>
<urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SubjectPublicKey">
<urn1:AttributeValue xsi:type="xs:base64Binary" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQD5tCEfHWjfu05hLQh2zSOvHIywYSNYbJ5mUL0iDGp36aj+fOoYPixYYSINJwHQfvREZNtvdoz895J79oROsOIVJ4KjygQmIi6YomSp/UYaUizHuGgKr+ZHwLdCViNrWhTBWwx7rCdYI0zgJxVC2uXqHpCrCM06UvIxHqdjvTErVwIDAQAB</urn1:AttributeValue>
</urn:Attribute>
<urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:certificateAttributes:SubjectDistinguishedName:organizationName"/>
</urn:X509CertificateInfo>
</dss:OptionalOutputs>
</dss:VerifyResponse>

En aquest missatge de resposta podem veure l'estructura XSS X509CertificateInfo on ens retornarà tots els atributs del certificat que hem demanat normalitzats i aquests els inclourem en el hash d'atributs de resposta. Un cop hem rebut la resposta del servei de PSIS en format XML (DSS), tractarem la resposta com s'explica a l'apartat de construcció a través del mètode ValidacioSignatura.generaHashResultat() aquest afegirà tota la informació resultant en els claus/valors definits en la classe ConstantsApplicacio en el hash d'atributs de resposta DigitalSignatureContentVo.atributs.get(ApplicationConfig.CERT_NIF) per a que hom que utilitzi la nostra api pugui obtenir fàcilment aquesta informació.



testX509CertificateRevoked: Aquest valida un certificat X509 de manera que comprova que el certificat sigui vàlid, la persona és qui diu ser, no ha estat revocat. En aquest cas comprovem que es detecta un cas de certificat revocat i per tant invàlid.

Paràmetres entrada: En aquesta prova passem com a paràmetre d'entrada el certificat personal revocat d'identificació: "signaturadigital/cert/RevocatCARLOSVILAMATEOS.crt"

Procés: El procés serà similar al test anterior.

Resultat: El resultat es guardarà als mateixos objectes que els definits en la resta de test al objecte DigitalSignatureContentVo.

Per verificar que el test és satisfactori, ho comprovarem a través d'assertions, en aquest cas comprovarem que el missatge de detall del resultat de la validació en el format DSS sigui igual a l'expressió que indica que el certificat està revocat:

```
vo.getMsg().get(ConstantsApplicacio.MSG_ResultMinor);
assertTrue(minor.contains("invalid:certificate:Revoked"));
```

Revisem els logs del procés pas a pas:

```
22-dic-2010 15:24:21 com.steria.digitalsignature.core.BaseSignatura <init>
INFO: BaseSignatura (constructor) init. URL = http://psisbeta.catcert.net/psis/catcert-test/dss
22-dic-2010 15:24:21 com.steria.digitalsignature.core.BaseSignatura <init>
INFO: BaseSignatura (constructor) done. URL = http://psisbeta.catcert.net/psis/catcert-test/dss
22-dic-2010 15:24:21 com.steria.digitalsignature.core.ValidacioCertificat createDefaultVO
INFO: createDefaultVO() init
22-dic-2010 15:24:21 com.steria.digitalsignature.core.ValidacioCertificat createDefaultVO
INFO: default certFile=[signaturadigital/cert/CARLOSVILAMATEOS.crt]
22-dic-2010 15:24:21 com.steria.digitalsignature.core.ValidacioCertificat createDefaultVO
INFO: createDefaultVO() done
22-dic-2010 15:24:21 com.steria.digitalsignature.core.SignaturaTest
testX509CertificateRevoked
INFO: testing certFile=[signaturadigital/cert/RevocatCARLOSVILAMATEOS.crt]
22-dic-2010 15:24:21 com.steria.digitalsignature.core.Validacio validar
INFO: validar() init. [PERFIL: urn:oasis:names:tc:dss:1.0:profiles:XSS]
22-dic-2010 15:24:21 com.steria.digitalsignature.core.BaseSignatura iniComunicacio
INFO: iniComunicacio() init
22-dic-2010 15:24:23 com.steria.digitalsignature.core.BaseSignatura iniComunicacio
INFO: iniComunicacio() done
22-dic-2010 15:24:23 com.steria.digitalsignature.core.Validacio validar
INFO: VerifyRequestDocument:
22-dic-2010 15:24:23 com.steria.digitalsignature.core.Validacio validar
```

1) Iniciem el procés iniciant els paràmetres base (gràcies a la classe BaseSignatura) com la URL de validació del servei de PSIS, en aquest cas l'entorn de test.
2) Creem el Vo per defecte amb els atributs necessaris com el tipus de PerfilDSS.
3) Afegim el certificat a validar
4) Indiquem el perfil de treball XSS.
5) Inicialitzem la comunicació amb el port webservice.
6) Cridem a la validació remota del certificat personal.
7) S'envia el missatge DSS de petició a PSIS.

Pas 1: Petició Validació Certificat

INFO: <VerifyRequest Profile="urn:oasis:names:tc:dss:1.0:profiles:XSS" xmlns="urn:oasis:names:tc:dss:1.0:core:schema"
xmlns:xd="http://www.w3.org/2000/09/xmldsig#">
<OptionalInputs>
<ReturnProcessingDetails/>
</OptionalInputs>
<SignatureObject>
<Other>
<xd:X509Data>

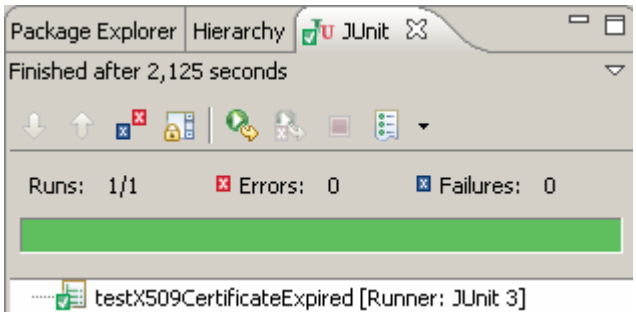
<xd:X509Certificate>LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tDQpNSUlJR1RDQ0J3R2dBd0lCQWdJQ0dCVXdEUVlKS29aSWh2
Y05BUUVGQlFBd2dnRXhNUXN3Q1FZRFZRUUdFd0pGDQpVekU3TURrR0ExVUVDaE15UVdkbGJtTnBZU0JEWVhSaGJHRnVZU0JrW
lNCRFpYSjBhV1pwWTJGamFXOGdLRTVKDQpSaUJSTFRBNE1ERXhOell0U1NreE5EQXlCZ05WQkFjVEsxQmhjM05oZEdkbElHUmxJz
VExGWmxaMlYxSUdoMGRIQnpPaTh2ZDNkM0xtTmhkR05sY25RdWJtVjBMM1psDQpja2xFUTJGMElDaGpLVEF6TVJRd0VnWURWUV
FFRXd0V1NVeEJJRTFCVkVWUFV6RVBNQTBHQTFV
eEd6QVpCZ05WQkFNVEVrTkJVa3hQVXlCV1NVeEJJF
WWtDZ1lFQTNiQi9Kb0g1emVTYmk1Z3VESVlZDQpwW
DQpCUVVlQ1FReEJCTUNSVk13RUFZSUt3WUJCUVV
...
...
lveXpybFNrSXJxSzlBcVNheEZYZmdEemZHRW1ybU9MMlVRRktSDQpCanVLaGN4TkFVcFhyWWgvZkZKcjRHNVlZdVUvU21RYzRJVW
5EMVd1ZzRwbmZLU2pVZ0taMjh0d1crZUQ2c3psDQpyTU9DMzNCcURzcno3Y0h3OS95Kzd1UC9xaWc0VGU1Nk8rUUdEeWRuWjBob
mIwdHB3VkYxa0dUTnlKVjdFSXhODQpOTUhLYkxWSUkxcURzMUFIMkg0UUlGd1NlL2VNNkxXRWd1Mmp1L0FpaWtpdkY5L3RoVzlsM
WxuMndxV2s2SjdHDQpOczFtWVR3MDh4bFUwdUZ6S3c9PQ0KLS0tLS1FTkQgQ0VSVElGSUNBVEUtLS0tLQ0K</xd:X509Certificate>
</xd:X509Data>
</Other>
</SignatureObject>
</VerifyRequest>

Formació del missatge DSS enviat de petició de validació de certificat.
El missatge es de tipus VerifyRequest en un objecte de tipus SignatureObject al tag Other com vàrem veure a l'apartat de formació de missatges DSS. En aquest tag Other\X509Data inseríem el nostre certificat codificat en base 64

Pas 2: Resposta Validació Certificat

1) Podem veure el resultat de la petició de validació a PSIS en el missatge DSS de resposta VerifyResponseDocument.
2) En el detall del missatge de resposta podem veure que el certificat no es vàlid i que està revocat, això s'emmagatzemarà al DigitalSignatureContentVo HashMap key ResultMinor

INFO: <dss:VerifyResponse Profile="urn:oasis:names:tc:dss:1.0:profiles:XSS" xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<dss:Result>
<dss:ResultMajor>urn:oasis:names:tc:dss:1.0:resultmajor:Success</dss:ResultMajor>
<dss:ResultMinor>urn:oasis:names:tc:dss:1.0:profiles:XSS:resultminor:invalid:certificate:Revoked</dss:ResultMinor>
</dss:Result>
<dss:OptionalOutputs>
<dss:ProcessingDetails>
<dss:ValidDetail Type="urn:oasis:names:tc:dss:1.0:detail:ValidityInterval">
<dss:Message xml:lang="en">The signing key is inside its static validity interval.</dss:Message>
</dss:ValidDetail>
<dss:ValidDetail Type="urn:oasis:names:tc:dss:1.0:detail:IssuerTrust">
<dss:Message xml:lang="en">The issuer of the given key is trusted.</dss:Message>
</dss:ValidDetail>
<dss:InvalidDetail Type="urn:oasis:names:tc:dss:1.0:detail:RevocationStatus">
<dss:Message xml:lang="en">The signing key is revoked.</dss:Message>
</dss:InvalidDetail>
</dss:ProcessingDetails>
</dss:OptionalOutputs>
</dss:VerifyResponse>
22-dic-2010 15:24:24 com.steria.digitalsignature.core.Validacio validar
INFO: Validació realitzada, resposta rebuda.
22-dic-2010 15:24:24 com.steria.digitalsignature.core.Validacio validar
INFO: Hash Atributs: {}
22-dic-2010 15:24:24 com.steria.digitalsignature.core.Validacio validar
INFO: Hash Msg(returned): {dss:ResultMinor=urn:oasis:names:tc:dss:1.0:profiles:XSS:resultminor:invalid:certificate:Revoked,
dss:ResultMajor=urn:oasis:names:tc:dss:1.0:resultmajor:Success}
22-dic-2010 15:24:24 com.steria.digitalsignature.core.Validacio validar
INFO: validar() done. [PERFIL: urn:oasis:names:tc:dss:1.0:profiles:XSS]



testX509CertificateExpired: Aquest valida un certificat X509 de manera que comprova que el certificat sigui vàlid, la persona és qui diu ser. En aquest cas comprovem que es detecta un cas de certificat que va expirar i per tant invàlid.

Paràmetres entrada: En aquesta prova passem com a paràmetre d'entrada el certificat personal d'identificació expirat: "signaturadigital/cert/cpisr_griselda.crt"

Procés: El procés serà similar al test anterior.

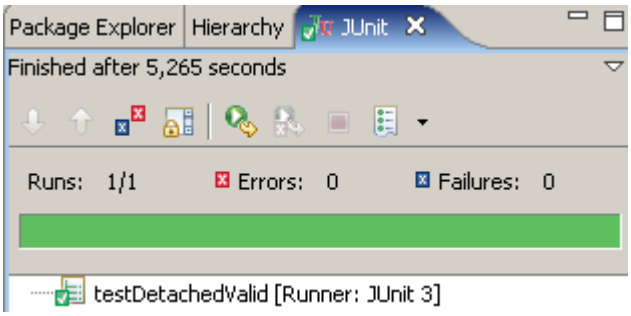
Resultat: El resultat es guardarà als mateixos objectes que els definits en la resta de test al objecte DigitalSignatureContentVo.

Per verificar que el test és satisfactori, ho comprovarem a través d'assertions, en aquest cas comprovarem que el missatge de detall del resultat de la validació en el format DSS sigui igual a l'expressió que indica que el certificat està revocat:

```
vo.getMsg().get(ConstantsApplicacio.MSG_ResultMinor);
assertTrue(minor.contains("invalid:certificate:Expired"));
```

```
INFO: <dss:VerifyResponse Profile="urn:oasis:names:tc:dss:1.0:profiles:XSS" xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <dss:Result>
    <dss:ResultMajor>urn:oasis:names:tc:dss:1.0:resultmajor:Success</dss:ResultMajor>
    <dss:ResultMinor>urn:oasis:names:tc:dss:1.0:profiles:XSS:resultminor:invalid:certificate:Expired</dss:ResultMinor>
  </dss:Result>
  <dss:OptionalOutputs>
    <dss:ProcessingDetails>
      <dss:IndeterminateDetail Type="urn:oasis:names:tc:dss:1.0:detail:IssuerTrust">
        <dss:Message xml:lang="en">Cannot determine whether key is trusted or not.</dss:Message>
      </dss:IndeterminateDetail>
      <dss:IndeterminateDetail Type="urn:oasis:names:tc:dss:1.0:detail:RevocationStatus">
        <dss:Message xml:lang="en">Cannot determine revocation status of the signing key.</dss:Message>
      </dss:IndeterminateDetail>
      <dss:InvalidDetail Type="urn:oasis:names:tc:dss:1.0:detail:ValidityInterval">
        <dss:Message xml:lang="en">The signing key is outside its static validity interval.</dss:Message>
      </dss:InvalidDetail>
    </dss:ProcessingDetails>
  </dss:OptionalOutputs>
</dss:VerifyResponse>
28-dic-2010 11:22:42 com.steria.digitalsignature.core.Validacio validar
INFO: Validació realitzada, resposta rebuda.
28-dic-2010 11:22:42 com.steria.digitalsignature.core.Validacio validar
INFO: Hash Atributs: {}
28-dic-2010 11:22:42 com.steria.digitalsignature.core.Validacio validar
INFO: Hash Msg(returned): {dss:ResultMinor=urn:oasis:names:tc:dss:1.0:profiles:XSS:resultminor:invalid:certificate:Expired,
dss:ResultMajor=urn:oasis:names:tc:dss:1.0:resultmajor:Success}
28-dic-2010 11:22:42 com.steria.digitalsignature.core.Validacio validar
INFO: validar() done. [PERFIL: urn:oasis:names:tc:dss:1.0:profiles:XSS]
```

5.5.4 Test de Validació de Signatures CMS



testDetachedValid: Aquest test valida un una signatura CMS detached a partir del document original codificat en Base64. Com a document signat de base utilitzarem la portada d'aquesta memòria "cvilama_platreball_portada.pdf", el servei validarà si el document realment està signat per qui diu ser, si és íntegre i per tant no ha estat modificat des del moment de la signatura, si és així la persona que ha signat no ho podrà repudiar i podem assegurar que és qui ha signat el document i que aquest no ha estat modificat des d'aleshores.

Paràmetres d'entrada: En aquesta prova passem com a paràmetre d'entrada la signatura del document en format CMS detached i també passem el propi document original però alterat de forma que no és el mateix document que es va signar.

- Signatura CMS detached: "signaturadigital/CPISR-1/ cvilama_platreball_portada(cms.signature Actiu).dat"
- Document original alterat: "signaturadigital/CPISR-1/ cvilama_platreball_portada(cms.docActiu).dat""

Procés: Per realitzar el test es crida a la classe ValidacioSignatura que realitzarà els passos descrits en l'apartat de construcció d'aquesta classe i al javadoc. S'ompliran la resta de paràmetres d'entrada obligatoris per a formar el missatge de petició de servei de validació de signatura amb tota l'estructura DSS necessària comentada als capítols anteriors.

Resultat: El resultat es guardarà als mateixos objectes que els definits en la resta de test al objecte DigitalSignatureContentVo.

Per verificar que el test és satisfactori, ho comprovarem a través d'assertions, en aquest cas comprovarem que el missatge de detall del resultat de la validació en el format DSS sigui igual a l'expressió que indica que la signatura és correcte:

```
vo.getMsg().get(ConstantsApplicacio.MSG_ResultMinor);
assertTrue(minor.contains("valid:signature:onAllDocuments"));
```

Revisem els logs del procés pas a pas:

```
22-dic-2010 16:34:45 com.steria.digitalsignature.core.BaseSignatura <init>
INFO: BaseSignatura (constructor) init. URL = http://psisbeta.catcert.net/psis/catcert-test/dss
22-dic-2010 16:34:45 com.steria.digitalsignature.core.BaseSignatura <init>
INFO: BaseSignatura (constructor) done. URL = http://psisbeta.catcert.net/psis/catcert-test/dss
22-dic-2010 16:34:45 com.steria.digitalsignature.core.ValidacioSignatura <init>
INFO: constructor ValidacioSignatura() init
22-dic-2010 16:34:45 com.steria.digitalsignature.core.ValidacioSignatura <init>
INFO: soapURL=[http://psisbeta.catcert.net/psis/catcert-test/dss]
22-dic-2010 16:34:45 com.steria.digitalsignature.core.ValidacioSignatura <init>
INFO: constructor ValidacioSignatura() done
22-dic-2010 16:34:45 com.steria.digitalsignature.core.ValidacioSignatura createDefaultVO
INFO: createDefaultVO() init
22-dic-2010 16:34:45 com.steria.digitalsignature.core.ValidacioSignatura createDefaultVO
INFO: createDefaultVO() done
22-dic-2010 16:34:46 com.steria.digitalsignature.core.Validacio validar
INFO: validar() init. [PERFIL: urn:oasis:names:tc:dss:1.0:profiles:XSS]
22-dic-2010 16:34:46 com.steria.digitalsignature.core.BaseSignatura iniComunicacio
INFO: iniComunicacio() init
22-dic-2010 16:34:47 com.steria.digitalsignature.core.Validacio validar
INFO: VerifyRequestDocument:
22-dic-2010 16:34:48 com.steria.digitalsignature.core.Validacio validar
```

1) Iniciem el procés iniciant els paràmetres base (gràcies a la classe BaseSignatura) com la URL de validació del servei de PSIS, en aquest cas l'entorn de test.
2) Creem el Vo per defecte amb els atributs necessaris com el tipus de PerfilDSS.
3) Afegim la signatura a validar
4) Indiquem el perfil de treball XSS.
5) Inicialitzem la comunicació amb el port webservice.
6) Cridem a la validació remota del certificat personal.
7) S'envia el missatge DSS de petició a PSIS.

Pas 1: Petició Validació Signatura

Formació del missatge DSS enviat de petició de validació de signatura.
El missatge és de tipus VerifyRequest a l'estructura ImputDocuments afegim el document en codificació Base64 (hem retallat l'exemple per no omplir pàgines de caràcters en base64), a l'estructura SignatureObject\Base64Signature afegim la signatura en Base64. Aquestes son les dades que s'enviaran a validar.


```
INFO: <VerifyRequest Profile="urn:oasis:names:tc:dss:1.0:profiles:XSS" xmlns="urn:oasis:names:tc:dss:1.0:core:schema"
xmlns:urn="urn:oasis:names:tc:dss:1.0:profiles:XSS">
  <OptionalInputs>
    <ReturnProcessingDetails/>
    <urn:ReturnSignatureInfo>
      <urn:AttributeDesignator Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:DigestAlgorithm"/>
      <urn:AttributeDesignator Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:DigestEncryptionAlgorithm"/>
      <urn:AttributeDesignator Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:SignatureAlgorithm"/>
      <urn:AttributeDesignator Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:SignatureValue"/>
    </urn:ReturnSignatureInfo>
  </OptionalInputs>
  <InputDocuments>
    <Document>
<Base64Data>JVBERi0xLjQNJeLjz9MNCjYgMCBvYmoNPDwvTGluZWFyaXplZCAxL0wgMjIxMTEvTyA4L0UgMTc4MTkvTiAxL1QgMjE5
CAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIC
AgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICA
gICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgI
CAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgCjw/eHBhY
…
…
AwMDAxNzkyOCAwMDAwMCBuDQowMDAwMDIxNjkwIDAwMDAwIG4NCnRyYWlsZXINCjw8L1NpemUgNj4+DQpzdGFydHhyZWYNCj
ExNg0KJSVFT0YNCg==</Base64Data>
    </Document>
  </InputDocuments>
  <SignatureObject>
    <Base64Signature
Type="urn:ietf:rfc:3852">MIAGCSqGSIb3DQEHAqCAMIACAQExCzAJBgUrDgMCGgUAMIAGCSqGSIb3DQEHAaCAJIAEggzIJVBERi0x
LjQNJeLjz9MNCjYgMCBvYmoNPDwvTGluZWFyaXplZCAxL0wgMjIxMTEvTyA4L0UgMTc4MTkvTiAxL1QgMjE5NDUvSCBbIDYzNiAxNzN
CAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgIC
AgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICA
gICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAKPD94cGFja2V0IGVuZD0idyI/Pg0KZW5kc3RyZWFtDWVuZG9iag
...
...
G9w0BAQEFAASBgCrstDYYKlmgHpdsDxNfMRp0QhPRIUQtZus1F6O3RfenSbNqWe6DqiPWZG6rw+1+WJscY+ZVCs6PduSPOO24AB
+WfE7sY7L3MjtaXW5BTEh9D/wp00joPT/XQm9m+OXgoc5BsAxsHPHdDtFkbRY19CuTDur4G3Ve6vGeHb0R/L0yAAAAAAAA</Base64S
ignature>
  </SignatureObject>
</VerifyRequest>
```

Pas 2: Resposta Validació Signatura

1) Podem veure el resultat de la petició de validació a PSIS en el missatge DSS de resposta VerifyResponseDocument.
2) Com que hem demanat OptionalOutputs ens retornen la informació de amb que algoritme hem realitzat el resum (SHA1) o l'encriptació del resum (RSA) o de la signatura (SHA1withRSA), etc. **(Aquesta informació per practicitat dels exemples no és tornarà a mostrar en els següents tests)**
3) Podem veure que el resultat es que la signatura és valida: "valid:signature:onAllDocuments"

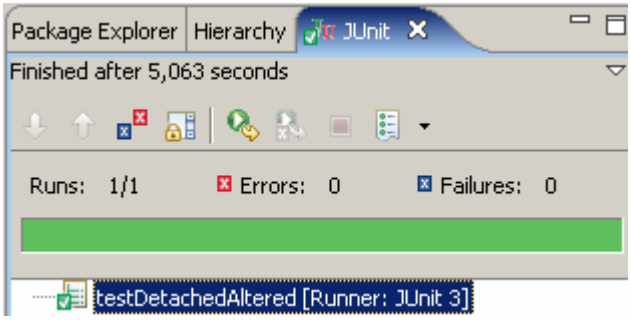

```
INFO: <dss:VerifyResponse Profile="urn:oasis:names:tc:dss:1.0:profiles:XSS" xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <dss:Result>
    <dss:ResultMajor>urn:oasis:names:tc:dss:1.0:resultmajor:Success</dss:ResultMajor>
    <dss:ResultMinor>urn:oasis:names:tc:dss:1.0:resultminor:valid:signature:onAllDocuments</dss:ResultMinor>
  </dss:Result>
  <dss:OptionalOutputs>
    <urn:SignatureInfo xmlns:urn="urn:oasis:names:tc:dss:1.0:profiles:XSS">
      <urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:DigestAlgorithm">
        <urn1:AttributeValue xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">SHA1</urn1:AttributeValue>
      </urn:Attribute>
      <urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:DigestEncryptionAlgorithm">
        <urn1:AttributeValue xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">RSA</urn1:AttributeValue>
      </urn:Attribute>
      <urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:SignatureAlgorithm">
        <urn1:AttributeValue xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">SHA1withRSA</urn1:AttributeValue>
      </urn:Attribute>
      <urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:SignatureValue">
        <urn1:AttributeValue xsi:type="xs:base64Binary" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">9+Plo6M2t6hOZ7VPWRGBARGqWvR6Uq6fWjd5XyghuW8/sg7LX14BB0LkxUpkZ6DeuBtixdM1uJTZ7MDrMKJV0zVneklUHXC
At1uFR7Iu/ejh731Yvi7QbazGvVOupMpaMkc4JxTWEbFn1Gv8GILUhGM4zSCw6zykJcF9lgdmHPY=</urn1:AttributeValue>
      </urn:Attribute>
    </urn:SignatureInfo>
    <dss:ProcessingDetails>
      <dss:ValidDetail Type="urn:oasis:names:tc:dss:1.0:detail:Signature">
        <dss:Message xml:lang="en">The signature is valid.</dss:Message>
      </dss:ValidDetail>
      <dss:ValidDetail Type="urn:oasis:names:tc:dss:1.0:detail:ValidityInterval">
        <dss:Message xml:lang="en">The signing key is inside its static validity interval.</dss:Message>
      </dss:ValidDetail>
      <dss:ValidDetail Type="urn:oasis:names:tc:dss:1.0:detail:IssuerTrust">
        <dss:Message xml:lang="en">The issuer of the given key is trusted.</dss:Message>
      </dss:ValidDetail>
      <dss:ValidDetail Type="urn:oasis:names:tc:dss:1.0:detail:RevocationStatus">
        <dss:Message xml:lang="en">The signing key is not revoked.</dss:Message>
      </dss:ValidDetail>
      <dss:ValidDetail Type="urn:oasis:names:tc:dss:1.0:detail:ValidationTime">
        <dss:Message xml:lang="en">The signature is validated at: Tue Dec 14 13:31:18 CET 2010</dss:Message>
      </dss:ValidDetail>
    </dss:ProcessingDetails>
  </dss:OptionalOutputs>
</dss:VerifyResponse>
```


3) En el detall del missatge de resposta podem veure que el certificat és vàlid això s'emmagatzemarà al DigitalSignatureContentVo HashMap key ResultMinor
Podem veure també més detalls dels missatges de validació, com que es confia en el certificat de signatura i que no està revocat.

Pas 3: Tractament Resposta Validació Signatura

Podem veure els logs del mètode generaHashResultat: Un cop hem rebut la resposta del servei de PSIS en format XML (DSS), tractarem la resposta com s'explica a l'apartat de construcció a través del mètode ValidacioSignatura.generaHashResultat() aquest afegirà tota la informació resultant en els claus/valors definits en la classe ConstantsApplicacio en el hash d'atributs de resposta DigitalSignatureContentVo.atributs.get(ApplicationConfig.SIGN_funcioResumAlgoritme) per a que hom que utilitzi la nostra api pugui obtenir fàcilment aquesta informació.
(Aquest mètode s'utilitza per a tots els mètodes per això aquest Pas3 i aquesta informació per praticitat dels exemples no és tornarà a mostrar en els següents tests)

```
22-dic-2010 17:12:36 com.steria.digitalsignature.core.Validacio validar
INFO: Validació realitzada, resposta rebuda.
22-dic-2010 17:12:36 com.steria.digitalsignature.core.ValidacioSignatura generaHashResultat
INFO: xmlObject: <xml-fragment xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:urn="urn:oasis:names:tc:dss:1.0:profiles:XSS" xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">SHA1</xml-fragment>
22-dic-2010 17:12:36 com.steria.digitalsignature.core.ValidacioSignatura generaHashResultat
INFO: String: SHA1
22-dic-2010 17:12:36 com.steria.digitalsignature.core.ValidacioSignatura generaHashResultat
INFO: xmlObject: <xml-fragment xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:urn="urn:oasis:names:tc:dss:1.0:profiles:XSS" xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">RSA</xml-fragment>
22-dic-2010 17:12:36 com.steria.digitalsignature.core.ValidacioSignatura generaHashResultat
INFO: String: RSA
22-dic-2010 17:12:36 com.steria.digitalsignature.core.ValidacioSignatura generaHashResultat
INFO: xmlObject: <xml-fragment xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:urn="urn:oasis:names:tc:dss:1.0:profiles:XSS" xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">SHA1withRSA</xml-fragment>
22-dic-2010 17:12:36 com.steria.digitalsignature.core.ValidacioSignatura generaHashResultat
INFO: String: SHA1withRSA
22-dic-2010 17:12:36 com.steria.digitalsignature.core.ValidacioSignatura generaHashResultat
INFO: xmlObject: <xml-fragment xsi:type="xs:base64Binary" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:urn="urn:oasis:names:tc:dss:1.0:profiles:XSS" xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">9+Plo6M2t6hOZ7VPWRGBARGqWvR6Uq6fWjd5XyghuW8/sg7LX14BB0Lkx
UpkZ6DeuBtixdM1uJTZ7MDrMKJV0zVneklUHXCAt1uFR7Iu/ejh731Yvi7QbazGvVOupMpaMkc4JxTWEbFn1Gv8GILUhGM4zSCw6zykJcF9l
gdmHPY=</xml-fragment>
22-dic-2010 17:12:36 com.steria.digitalsignature.core.Validacio validar
INFO: Hash Atributs: {urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:DigestAlgorithm=SHA1,
urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:SignatureAlgorithm=SHA1withRSA,
urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:DigestEncryptionAlgorithm=RSA}
22-dic-2010 17:12:36 com.steria.digitalsignature.core.Validacio validar
INFO: Hash Msg(returned): {dss:ResultMinor=urn:oasis:names:tc:dss:1.0:resultminor:valid:signature:onAllDocuments,
dss:ResultMajor=urn:oasis:names:tc:dss:1.0:resultmajor:Success}
22-dic-2010 17:12:36 com.steria.digitalsignature.core.Validacio validar
INFO: validar() done. [PERFIL: urn:oasis:names:tc:dss:1.0:profiles:XSS]
```



testDetachedAltered: Aquest test valida una signatura CMS detached a partir del document original però alterat després del moment de la signatura. Com a document signat de base utilitzarem la portada d'aquesta memòria "cvilama_platreball_portada.pdf", el servei validarà si el document realment està signat per qui diu ser, si és íntegre i per tant no ha estat modificat des del moment de la signatura, si és així la persona que ha signat no ho podrà repudiar i podem assegurar que és qui ha signat el document i que aquest no ha estat modificat des d'aleshores. En aquest cas hem alterat el document signat i per tant el test correcte és detectar aquesta alteració.

Paràmetres d'entrada: En aquesta prova passem com a paràmetre d'entrada la signatura del document en format CMS detached i també passem el propi document original però alterat de forma que no és el mateix document que es va signar.

- Signatura CMS detached: "signaturadigital/CPISR-1/ cvilama_platreball_portada(cms.signature Actiu).dat"
- Document original alterat: "signaturadigital/CPISR-1/ cvilama_platreball_portada(Alterat).pdf"

Procés: El procés serà similar al test anterior.

Resultat: El resultat es guardarà als mateixos objectes que els definits en la resta de test al objecte DigitalSignatureContentVo.

Per verificar que el test és satisfactori, ho comprovarem a través d'assertions, en aquest cas comprovarem que el missatge de detall del resultat de la validació en el format DSS sigui igual a l'expressió que indica que la signatura no és correcte:

```
vo.getMsg().get(ConstantsApplicacio.MSG_ResultMinor);
assertTrue(minor.contains("invalid:incorrectSignature"));
```

Revisem els logs del procés pas a pas:

L'inici del procés segueix els mateixos passos que els altres tests.

Pas 1: Petició Validació Signatura

La formació del missatge DSS enviat de petició de validació de signatura que és el mateix cas que el test anterior amb la diferencia que enviem un document alterat en Base64.
El missatge és de tipus VerifyRequest a l'estructura ImputDocuments afegim el document en codificació Base64 (hem retallat l'exemple per no omplir pàgines de caràcters en base64), a l'estructura SignatureObject\Base64Signature afegim la signatura en Base64. Aquestes son

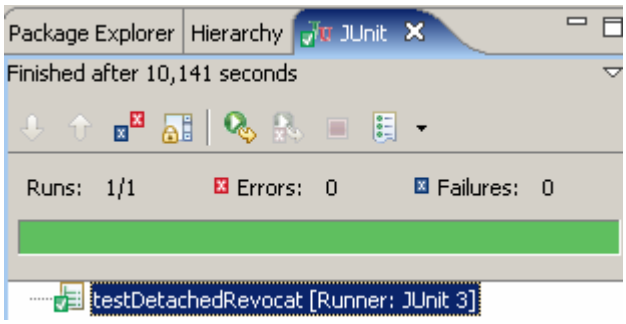
```
INFO: <VerifyRequest Profile="urn:oasis:names:tc:dss:1.0:profiles:XSS" xmlns="urn:oasis:names:tc:dss:1.0:core:schema"
xmlns:urn="urn:oasis:names:tc:dss:1.0:profiles:XSS">
  <OptionalInputs>
    <ReturnProcessingDetails/>
    <urn:ReturnSignatureInfo>
      <urn:AttributeDesignator Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:DigestAlgorithm"/>
      <urn:AttributeDesignator Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:DigestEncryptionAlgorithm"/>
      <urn:AttributeDesignator Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:SignatureAlgorithm"/>
      <urn:AttributeDesignator Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:SignatureValue"/>
    </urn:ReturnSignatureInfo>
  </OptionalInputs>
  <InputDocuments>
    <Document>
<Base64Data>JVBERi0xLjQNJeLjz9MNCjYgMCBvYmoNPDwvTGluZWFyaXplZCAxL0wgMjIxMTEvTyA4L0UgMTc4MTkvTiAxL1QgMjE5
YWorKDByY1wtJEk1SjF2RFVTZ0ZeLys4TD03Xj80aUpIW0lcTU5PSltcXV5fVWZnaGlqa2xtbm9ic3R1dnd4eXp7fH/9oADAMBAAIRAxEA/
/WDL6DnsyaHE0OIGTRPtezvp+8OxXtlN1d9TLqjurtaHsd4tcJBXz8eF7r0Ci3H6JgUXfzlePU14PIIaNFHIA0KYuikkolyLI/mLP6jvyLyT/Fz/A
OKzH/4q7/qV63kfzFn9R35F5J/i5/8AFZj/APF3f9Snw+WaDuH2BMnTJiXwPqX/AClmf8fb/wBW5em/4rv/ABOWf+GrP+pYvMupf8pZn/H2/d
…
…
AwMDAxNzkyOCAwMDAwMCBuDQowMDAwMDIxNjkwIDAwMDAwIG4NCnRyYWlsZXINCjw8L1NpemUgNj4+DQpzdGFydHhyZWYNCj
ExNg0KJSVFT0YNCg==</Base64Data>
    </Document>
  </InputDocuments>
  <SignatureObject>
    <Base64Signature
Type="urn:ietf:rfc:3852">MIAGCSqGSIb3DQEHAqCAMIACAQExCzAJBgUrDgMCGgUAMIAGCSqGSIb3DQEHAaCAJIAEggzIJVBERi0x
LjQNJeLjz9MNCjYgMCBvYmoNPDwvTGluZWFyaXplZCAxL0wgMjIxMTEvTyA4L0UgMTc4MTkvTiAxL1QgMjE5NDUvSCBbIDYzNiAxNzN
...
...
kqhkiG9w0BAQEFAASBgCrstDYYKlmgHpdsDxNfMRp0QhPRIUQtZus1F6O3RfenSbNqWe6DqiPWZG6rw+1+WJscY+ZVCs6PduSPOO2
4AB+WfE7sY7L3MjtaXW5BTEh9D/wp00joPT/XQm9m+OXgoc5BsAxsHPHdDtFkbRY19CuTDur4G3Ve6vGeHb0R/L0yAAAAAAAA</Base
64Signature>
  </SignatureObject>
</VerifyRequest>
```

Pas 2: Resposta Validació Signatura

1) Podem veure el resultat de la petició de validació a PSIS en el missatge DSS de resposta VerifyResponseDocument.
2) Podem veure que el resultat es que la signatura no és valida: “incorrectSignature”


```
INFO: <dss:VerifyResponse Profile="urn:oasis:names:tc:dss:1.0:profiles:XSS" xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <dss:Result>
    <dss:ResultMajor>urn:oasis:names:tc:dss:1.0:resultmajor:Success</dss:ResultMajor>
    <dss:ResultMinor>urn:oasis:names:tc:dss:1.0:resultminor:invalid:incorrectSignature</dss:ResultMinor>
    <dss:ResultMessage xml:lang="en">Signature pdu is not valid</dss:ResultMessage>
  </dss:Result>
  <dss:OptionalOutputs>
    <urn:SignatureInfo xmlns:urn="urn:oasis:names:tc:dss:1.0:profiles:XSS">
      <urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:DigestAlgorithm">
        <urn1:AttributeValue xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">SHA1</urn1:AttributeValue>
      </urn:Attribute>
      <urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:DigestEncryptionAlgorithm">
        <urn1:AttributeValue xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">RSA</urn1:AttributeValue>
      </urn:Attribute>
      <urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:SignatureAlgorithm">
        <urn1:AttributeValue xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">SHA1withRSA</urn1:AttributeValue>
      </urn:Attribute>
      <urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:SignatureValue">
        <urn1:AttributeValue xsi:type="xs:base64Binary" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">9+Plo6M2t6hOZ7VPWRGBARGqWvR6Uq6fWjd5XyghuW8/sg7LX14BB0LkxUpkZ6DeuBtixdM1uJTZ7MDrMKJV0zVneklUHXC
At1uFR7Iu/ejh731Yvi7QbazGvVOupMpaMkc4JxTWEbFn1Gv8GILUhGM4zSCw6zykJcF9lgdmHPY=</urn1:AttributeValue>
      </urn:Attribute>
    </urn:SignatureInfo>
    <dss:ProcessingDetails>
      <dss:InvalidDetail Type="urn:oasis:names:tc:dss:1.0:detail:Signature">
        <dss:Message xml:lang="en">The signature is not valid.</dss:Message>
      </dss:InvalidDetail>
    </dss:ProcessingDetails>
  </dss:OptionalOutputs>
</dss:VerifyResponse>
22-dic-2010 19:08:31 com.steria.digitalsignature.core.Validacio validar
INFO: Validació realitzada, resposta rebuda.
22-dic-2010 19:08:31 com.steria.digitalsignature.core.Validacio validar
INFO: Hash Atributs: {urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:DigestAlgorithm=SHA1,
urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:SignatureAlgorithm=SHA1withRSA,
urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:DigestEncryptionAlgorithm=RSA}
22-dic-2010 19:08:31 com.steria.digitalsignature.core.Validacio validar
INFO: Hash Msg(returned): {dss:ResultMessage=Signature pdu is not valid,
dss:ResultMinor=urn:oasis:names:tc:dss:1.0:resultminor:invalid:incorrectSignature,
dss:ResultMajor=urn:oasis:names:tc:dss:1.0:resultmajor:Success}
22-dic-2010 19:08:31 com.steria.digitalsignature.core.Validacio validar
```


3) En el detall del missatge de resposta podem veure que el certificat no és vàlid això s’emmagatzemarà al DigitalSignatureContentVo HashMap key ResultMinor



testDetachedRevocat: Aquest test valida una signatura CMS detached a partir del document original. Com a document signat de base utilitzarem la portada d'aquesta memòria "cvilama_platreball_portada.pdf", el servei validarà si el document realment està signat per qui diu ser, si és íntegre i per tant no ha estat modificat des del moment de la signatura, si és així la persona que ha signat no ho podrà repudiar i podem assegurar que és qui ha signat el document i que aquest no ha estat modificat des d'aleshores. En aquest cas hem signat amb un certificat que està revocat i per tant el test correcte és detectar aquesta revocació, perquè en el fons la signatura és correcte i el document està correctament signat i no ha sigut alterat des d'aleshores.

Paràmetres d'entrada: En aquesta prova passem com a paràmetre d'entrada la signatura del document en format CMS detached i també passem el propi document original.

- Signatura CMS detached: "signaturadigital/CPISR-1/ cvilama_platreball_portada (Revocat).p7b"
- Document original alterat: "signaturadigital/CPISR-1/ cvilama_platreball_portada (cms.doc Actiu).dat"

Procés: El procés serà similar al test anterior.

Resultat: El resultat es guardarà als mateixos objectes que els definits en la resta de test al objecte DigitalSignatureContentVo.

Per verificar que el test és satisfactori, ho comprovarem a través d'assertions, en aquest cas comprovarem que el missatge de detall del resultat de la validació en el format DSS sigui igual a l'expressió que indica que la signatura no és correcte:

```
vo.getMsg().get(ConstantsApplicacio.MSG_ResultMinor);
assertTrue(minor.contains("invalid:untrustedKey"));
```

Revisem els logs del procés pas a pas:

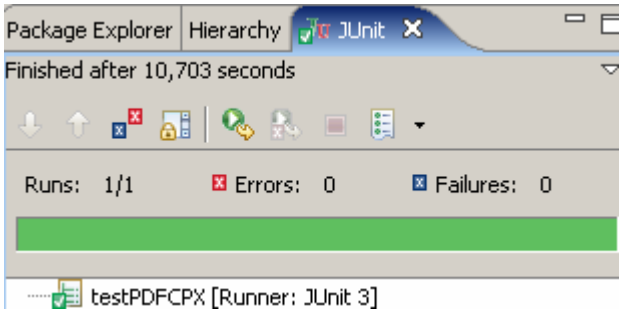
L'inici del procés segueix els mateixos passos que els altres tests.

La formació del missatge DSS enviat de petició de validació de signatura que és el mateix cas que el test anterior amb la diferencia que enviem una signatura revocada.

1) Podem veure el resultat de la petició de validació a PSIS en el missatge DSS de resposta VerifyResponse.
2) Podem veure que el resultat es que la signatura no és confiable: “invalid:untrustedKey”

INFO: <dss:VerifyResponse Profile="urn:oasis:names:tc:dss:1.0:profiles:XSS" xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<dss:Result>
<dss:ResultMajor>urn:oasis:names:tc:dss:1.0:resultmajor:Success</dss:ResultMajor>
<dss:ResultMinor>urn:oasis:names:tc:dss:1.0:resultminor:invalid:untrustedKey</dss:ResultMinor>
<dss:ResultMessage xml:lang="en">Invàlid signing certificate</dss:ResultMessage>
</dss:Result>
<dss:OptionalOutputs>
<urn:SignatureInfo xmlns:urn="urn:oasis:names:tc:dss:1.0:profiles:XSS">
<urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:DigestAlgorithm">
<urn1:AttributeValue xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">SHA1</urn1:AttributeValue>
</urn:Attribute>
<urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:DigestEncryptionAlgorithm">
<urn1:AttributeValue xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">RSA</urn1:AttributeValue>
</urn:Attribute>
<urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:SignatureAlgorithm">
<urn1:AttributeValue xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">SHA1withRSA</urn1:AttributeValue>
</urn:Attribute>
<urn:Attribute Name="urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:SignatureValue">
<urn1:AttributeValue xsi:type="xs:base64Binary" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:urn1="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">wabWwGBEwNXN87MSw9dqVBvhItaVgxxQfBrrLbpADt1cK0XKA7f1YABB9+oovRhyYqLIs69kfGWpTX2181JiyKOkJVPPYOZo
KotWP+860fxbDlNXHE2Vkhrfbzh8Vl2wGoJQi/Z9saB4qsYeIYkmoO5B2i/rHYwuQUBx7YDEPc4=</urn1:AttributeValue>
</urn:Attribute>
</urn:SignatureInfo>
<dss:ProcessingDetails>
<dss:ValidDetail Type="urn:oasis:names:tc:dss:1.0:detail:Signature">
<dss:Message xml:lang="en">The signature is valid.</dss:Message>
</dss:ValidDetail>
<dss:ValidDetail Type="urn:oasis:names:tc:dss:1.0:detail:ValidityInterval">
<dss:Message xml:lang="en">The signing key is inside its static validity interval.</dss:Message>
</dss:ValidDetail>
<dss:ValidDetail Type="urn:oasis:names:tc:dss:1.0:detail:IssuerTrust">
<dss:Message xml:lang="en">The issuer of the given key is trusted.</dss:Message>
</dss:ValidDetail>
<dss:ValidDetail Type="urn:oasis:names:tc:dss:1.0:detail:ValidationTime">
<dss:Message xml:lang="en">The signature is validated at: Wed Dec 22 19:13:40 CET 2010</dss:Message>
</dss:ValidDetail>
<dss:InvalidDetail Type="urn:oasis:names:tc:dss:1.0:detail:RevocationStatus">
<dss:Message xml:lang="en">The signing key is revoked.</dss:Message>
</dss:InvalidDetail>
</dss:ProcessingDetails>
</dss:OptionalOutputs>
</dss:VerifyResponse>
22-dic-2010 19:13:44 com.steria.digitalsignature.core.Validacio validar
INFO: Validació realitzada, resposta rebuda.
22-dic-2010 19:13:44 com.steria.digitalsignature.core.Validacio validar
INFO: Hash Atributs: {urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:DigestAlgorithm=SHA1, urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:SignatureAlgorithm=SHA1withRSA, urn:oasis:names:tc:dss:1.0:profiles:XSS:signatureAttributes:DigestEncryptionAlgorithm=RSA}
22-dic-2010 19:13:44 com.steria.digitalsignature.core.Validacio validar
INFO: Hash Msg(returned): {dss:ResultMessage=Invalid signing certificate, dss:ResultMinor=urn:oasis:names:tc:dss:1.0:resultminor:invalid:untrustedKey, dss:ResultMajor=urn:oasis:names:tc:dss:1.0:resultmajor:Success}
22-dic-2010 19:13:44 com.steria.digitalsignature.core.Validacio validar
INFO: validar() done. [PERFIL: urn:oasis:names:tc:dss:1.0:profiles:XSS]INFO: validar() done. [PERFIL: urn:oasis:names:tc:dss:1.0:profiles:XSS]

5.5.5 Test de Validació de Signatures en PDF



testPDFCPX: Aquest test valida un una signatura CMS detached incrustada en un PDF i amb el servei especial de CATCert per validar fitxers PDF, utilitzant el perfil "urn:oasis:names:tc:dss:1.0:profiles:DSS_PDF". Com a document signat de base utilitzarem la portada d'aquesta memòria "cvilama_platreball_portada.pdf", el servei validarà si el document realment està signat per qui diu ser, si és íntegre i per tant no ha estat modificat des del moment de la signatura, si és així la persona que ha signat no ho podrà repudiar i podem assegurar que és qui ha signat el document i que aquest no ha estat modificat des d'aleshores.

Paràmetres d'entrada:

- Document PDF signat amb la signatura incrustada: "signaturadigital/CPISR-1/ cvilama_platreball_portada_signada (Actiu).pdf"

Procés: Per realitzar el test es crida a la classe ValidacioSignaturaPDF que realitzarà els passos descrits en l'apartat de construcció d'aquesta classe i al javadoc. S'ompliran la resta de paràmetres d'entrada obligatoris per a formar el missatge de petició de servei de validació de signatura amb tota l'estructura DSS necessària comentada als capítols anteriors.

Resultat: El resultat es guardarà als mateixos objectes que els definits en la resta de test al objecte DigitalSignatureContentVo.

Per verificar que el test és satisfactori, ho comprovarem a través d'assertions, en aquest cas comprovarem que el missatge de detall del resultat de la validació en el format DSS sigui igual a l'expressió que indica que la signatura és correcte:

```
vo.getMsg().get(ConstantsApplicacio.MSG_ResultMinor);
assertTrue(minor.contains("valid:signature:onAllDocuments"));
```

Revisem els logs del procés pas a pas:

L'inici del procés segueix els mateixos passos que els altres tests, però utilitzant la classe ValidacioSignaturaPDF explicada a la fase de construcció i utilitzant el perfil especial per validar PDF.

Pas 1: Petició Validació Signatura

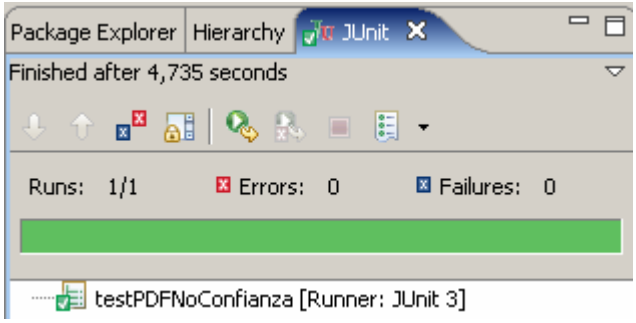
Formació del missatge DSS enviat de petició de validació de signatura.
El missatge és de tipus VerifyRequest a l'estructura ImputDocuments afegim el document PDF en codificació Base64 (hem retallat l'exemple per no omplir pàgines de caràcters en base64).
Podem comprovar que el perfil que s'utilitza és l'específic pel servei de PSIS de validació de documents PDF.

INFO: <VerifyRequest Profile="urn:oasis:names:tc:dss:1.0:profiles:DSS_PDF" xmlns="urn:oasis:names:tc:dss:1.0:core:schema"
xmlns:urn="urn:oasis:names:tc:dss:1.0:profiles:DSS_PDF">
<OptionalInputs>
<urn:ReturnSignatureReason/>
<ReturnSignerIdentity/>
</OptionalInputs>
<InputDocuments>
<Document>
<Base64Data>JVBERi0xLjYNJeLjz9MNCjEwIDAgb2JqDTw8L0xpbmVhcml6ZWQgMS9MIDk1NjU5L08gMTMvRSA4OTkwNS9OIDEvVC
2VzL0tpZHNbMTMgMCBSXT4+DWVuZG9iag05IDAgb2JqDTw8L0NyZWF0aW9uRGF0ZShEOjIwMTAxMTExMTkxMDM5KzAxJzAwJykv
QXV0aG9yKGN2aWxhKS9DcmVhdG9yKFBTY3JpcHQ1LmRsbCBWZXJzaW9uIDUuMikvUHJvZHVjZXIoQWNyb2JhdCBEaXN0aWxsZXI
…
…
gOC4xLjAgXChXaW5kb3dzXCkpL01vZERhdGUoRDoyMDEwMTExMTE5MjM0NyswMScwMCcpL1RpdGxlKE1pY3Jvc29mdCBXb3JkIC0
gUGxhIGRlIFRyZWJhbGwgQ1ZNIE9jdHVicmUgMjAxMC5kb2MpPj4NZW5kb2JqDXhyZWYNCjAgMTANCjAwMDAwMDAwMDAgNjU1Mz
UgZg0KMDAwMDA4OTkwNSAwMDAwMCBuDQowMDAwMDg5OTI5IDAwMDAwIG4NCjAwMDAwOTExMjAgMDAwMDAgbg0KMDAwMDA5MTIwOCAwMDAwMCBuDQowMDAwMDkxMjg0IDAwMDAwIG4NCjAwMDAwOTUwNDYgMDAwMDAgbg0KMDAwMDA5NTA4MCAwMDAwMCBuDQowMDAwMDk1MTA0IDAwMDAwIG4NCjAwMDAwOTUxNTYgMDAwMDAgbg0KdHJhaWxlcg0KPDwvU2l6ZSAxMD4+DQpzdGFydHhyZWYNCjExNg0KJSVFT0YNCg==</Base64Data>
</Document>
</InputDocuments>
</VerifyRequest>

Pas 2: Resposta Validació Signatura

1) Podem veure el resultat de la petició de validació a PSIS en el missatge DSS de resposta VerifyResponseDocument.
2) Podem veure que el resultat es que la signatura és valida: "valid:signature:onAllDocuments"

INFO: <dss:VerifyResponse Profile="urn:oasis:names:tc:dss:1.0:profiles:DSS_PDF" xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<dss:Result>
<dss:ResultMajor>urn:oasis:names:tc:dss:1.0:resultmajor:Success</dss:ResultMajor>
<dss:ResultMinor>urn:oasis:names:tc:dss:1.0:resultminor:valid:signature:onAllDocuments</dss:ResultMinor>
</dss:Result>
<dss:OptionalOutputs>
<dss:SignerIdentity Format="urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName">CN=CARLOS VILA
MATEOS,2.5.4.5=\#130934363335383503950,G=CARLOS,SN=VILA MATEOS,OU=Vegeu https://www.catcert.net/verIDCat
(c)03,C=ES</dss:SignerIdentity>
</dss:OptionalOutputs>
</dss:VerifyResponse>
22-dic-2010 19:26:16 com.steria.digitalsignature.core.Validacio validar
INFO: Validació realitzada, resposta rebuda.
22-dic-2010 19:26:16 com.steria.digitalsignature.core.Validacio validar
INFO: Hash Atributs: {}
22-dic-2010 19:26:16 com.steria.digitalsignature.core.Validacio validar
INFO: Hash Msg(returned): {dss:ResultMinor=urn:oasis:names:tc:dss:1.0:resultminor:valid:signature:onAllDocuments,
dss:ResultMajor=urn:oasis:names:tc:dss:1.0:resultmajor:Success}
22-dic-2010 19:26:16 com.steria.digitalsignature.core.Validacio validar
INFO: validar() done. [PERFIL: urn:oasis:names:tc:dss:1.0:profiles:DSS_PDF]



testPDFNoConfianza: Aquest test valida un una signatura CMS detached incrustada en un PDF i amb el servei especial de CATCert per validar fitxers PDF, utilitzant el perfil "urn:oasis:names:tc:dss:1.0:profiles:DSS_PDF". Com a document signat de base utilitzarem la portada d'aquesta memòria "cvilama_platreball_portada.pdf", el servei validarà si el document realment està signat per qui diu ser, si és íntegre i per tant no ha estat modificat des del moment de la signatura, si és així la persona que ha signat no ho podrà repudiar i podem assegurar que és qui ha signat el document i que aquest no ha estat modificat des d'aleshores.

En aquest cas em signat el document amb un certificat emes per nosaltres mateixos i que per tant no és de confiança ni reconegut per les CAs arrel.

Paràmetres d'entrada:

- Document PDF signat amb la signatura incrustada: "signaturadigital/CPISR-1/ cvilama_platreball_portada_signada (NoConfianca).pdf"

Procés: Per realitzar el test es crida a la classe ValidacioSignaturaPDF que realitzarà els passos descrits en l'apartat de construcció d'aquesta classe i al javadoc. S'ompliran la resta de paràmetres d'entrada obligatoris per a formar el missatge de petició de servei de validació de signatura amb tota l'estructura DSS necessària comentada als capítols anteriors.

Resultat: El resultat es guardarà als mateixos objectes que els definits en la resta de test al objecte DigitalSignatureContentVo.

Per verificar que el test és satisfactori, ho comprovarem a través d'assertions, en aquest cas comprovarem que el missatge de detall del resultat de la validació en el format DSS sigui igual a l'expressió que indica que la signatura no és de confiança:

```
vo.getMsg().get(ConstantsApplicacio.MSG_ResultMinor);
assertTrue(minor.contains("invalid:indeterminateKey"));
```

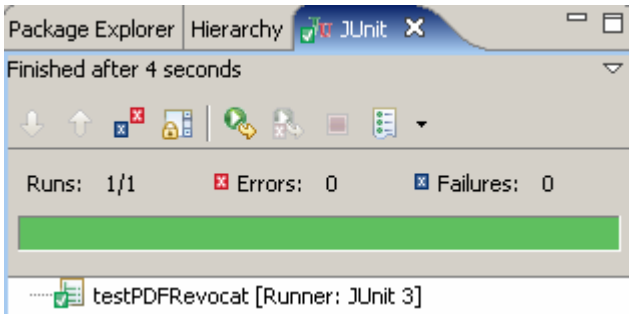
Revisem els logs del procés pas a pas:

L'inici del procés segueix els mateixos passos que els altres tests, però utilitzant la classe ValidacioSignaturaPDF explicada a la fase de construcció i utilitzant el perfil especial per validar PDF.

La formació del missatge DSS enviat de petició de validació de signatura d'un fitxer PDF que és el mateix cas que el test anterior amb la diferencia que enviem una signatura desconeguda.

1) Podem veure el resultat de la petició de validació a PSIS en el missatge DSS de resposta VerifyResponseDocument.
2) Podem veure que el resultat es que la signatura no és valida: "invalid:indeterminateKey"

```
INFO: VerifyResponseDocument:
22-dic-2010 19:41:25 com.steria.digitalsignature.core.Validacio validar
INFO: <dss:VerifyResponse Profile="urn:oasis:names:tc:dss:1.0:profiles:DSS_PDF" xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <dss:Result>
    <dss:ResultMajor>urn:oasis:names:tc:dss:1.0:resultmajor:Success</dss:ResultMajor>
    <dss:ResultMinor>urn:oasis:names:tc:dss:1.0:resultminor:invalid:indeterminateKey</dss:ResultMinor>
    <dss:ResultMessage xml:lang="en">Cannot determinate signing certificate status</dss:ResultMessage>
  </dss:Result>
  <dss:OptionalOutputs/>
</dss:VerifyResponse>
22-dic-2010 19:41:25 com.steria.digitalsignature.core.Validacio validar
INFO: Validació realitzada, resposta rebuda.
22-dic-2010 19:41:25 com.steria.digitalsignature.core.Validacio validar
INFO: Hash Atributs: {}
22-dic-2010 19:41:25 com.steria.digitalsignature.core.Validacio validar
INFO: Hash Msg(returned): {dss:ResultMessage=Cannot determinate signing certificate status,
dss:ResultMinor=urn:oasis:names:tc:dss:1.0:resultminor:invalid:indeterminateKey,
dss:ResultMajor=urn:oasis:names:tc:dss:1.0:resultmajor:Success}
22-dic-2010 19:41:25 com.steria.digitalsignature.core.Validacio validar
INFO: validar() done. [PERFIL : urn:oasis:names:tc:dss:1.0:profiles:DSS_PDF]
```



testPDFRevocat: Aquest test valida un una signatura CMS detached incrustada en un PDF i amb el servei especial de CATCert per validar fitxers PDF, utilitzant el perfil "urn:oasis:names:tc:dss:1.0:profiles:DSS_PDF". Com a document signat de base utilitzarem la portada d'aquesta memòria "cvilama_platreball_portada.pdf", el servei validarà si el document realment està signat per qui diu ser, si és íntegre i per tant no ha estat modificat des del moment de la signatura, si és així la persona que ha signat no ho podrà repudiar i podem assegurar que és qui ha signat el document i que aquest no ha estat modificat des d'aleshores.

En aquest cas em signat el document amb un certificat revocat.

Paràmetres d'entrada:

- Document PDF signat amb la signatura incrustada: "signaturadigital/CPISR-1/ cvilama_platreball_portada (Revocat).pdf"

Procés: Per realitzar el test es crida a la classe ValidacioSignaturaPDF que realitzarà els passos descrits en l'apartat de construcció d'aquesta classe i al javadoc. S'ompliran la resta de paràmetres d'entrada obligatoris per a formar el missatge de petició de servei de validació de signatura amb tota l'estructura DSS necessària comentada als capítols anteriors.

Resultat: El resultat es guardarà als mateixos objectes que els definits en la resta de test al objecte DigitalSignatureContentVo.

Per verificar que el test és satisfactori, ho comprovarem a través d'assertions, en aquest cas comprovarem que el missatge de detall del resultat de la validació en el format DSS sigui igual a l'expressió que indica que la signatura no és de confiança:

```
vo.getMsg().get(ConstantsApplicacio.MSG_ResultMinor);
assertTrue(minor.contains("invalid:untrustedKey"));
```

Revisem els logs del procés pas a pas:

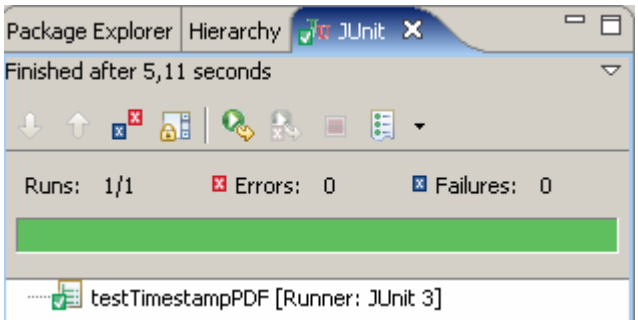
L'inici del procés segueix els mateixos passos que els altres tests, però utilitzant la classe ValidacioSignaturaPDF explicada a la fase de construcció i utilitzant el perfil especial per validar PDF.

La formació del missatge DSS enviat de petició de validació de signatura d'un fitxer PDF que és el mateix cas que el test anterior amb la diferencia que enviem una signatura revocada.

1) Podem veure el resultat de la petició de validació a PSIS en el missatge DSS de resposta VerifyResponse.
2) Podem veure que el resultat es que la signatura està revocada: “invalid:untrustedKey”

```
INFO: <dss:VerifyResponse Profile="urn:oasis:names:tc:dss:1.0:profiles:DSS_PDF" xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <dss:Result>
    <dss:ResultMajor>urn:oasis:names:tc:dss:1.0:resultmajor:Success</dss:ResultMajor>
    <dss:ResultMinor>urn:oasis:names:tc:dss:1.0:resultminor:invalid:untrustedKey</dss:ResultMinor>
    <dss:ResultMessage xml:lang="en">Invalid signing certificate</dss:ResultMessage>
  </dss:Result>
  <dss:OptionalOutputs/>
</dss:VerifyResponse>
22-dic-2010 19:48:03 com.steria.digitalsignature.core.Validacio validar
INFO: Validació realitzada, resposta rebuda.
22-dic-2010 19:48:03 com.steria.digitalsignature.core.Validacio validar
INFO: Hash Atributs: {}
22-dic-2010 19:48:03 com.steria.digitalsignature.core.Validacio validar
INFO: Hash Msg(returned): {dss:ResultMessage=Invalid signing certificate,
dss:ResultMinor=urn:oasis:names:tc:dss:1.0:resultminor:invalid:untrustedKey,
dss:ResultMajor=urn:oasis:names:tc:dss:1.0:resultmajor:Success}
22-dic-2010 19:48:03 com.steria.digitalsignature.core.Validacio validar
INFO: validar() done. [PERFIL: urn:oasis:names:tc:dss:1.0:profiles:DSS_PDF]
```

5.5.6 Test de Signatura CMS i Segell de Temps



testTimestampPDF: **:** Aquest test realitzar una signatura CMS de tipus segell de temps (RFC3161) i posteriorment valida la signatura CMS o segell de temps i el document signat, per realitzar això utilitzar el perfil de segell de temps "urn:oasis:names:tc:dss:1.0:profiles:timestamping". Com a document signat de base utilitzarem la portada d'aquesta memòria "cvilama_platreball_portada.pdf" però aquest portarà la signatura d'una TSA (CATCert) certificant el moment de temps en que es va signar el document. El servei validarà si el document realment està signat per qui diu ser, si s'ha signat en quin moment de temps i si qui afegeix la signatura de temps és fiable i reconegut (la TSA), també seguirà comprovant si el document és íntegre i per tant no ha estat modificat des del moment de la signatura, si és així la persona que ha signat no ho podrà repudiar i podem assegurar que és qui ha signat el document i en quin moment ho va fer i que aquest no ha estat modificat des d'aleshores.

Paràmetres d'entrada:

- Document PDF signat amb la signatura incrustada: "signaturadigital/CPISR-1/ cvilama_platreball_portada_signat (Segell).pdf"
- Fitxer de Certificat de l'autoritat de certificació TSA que ens emet i signa el segell de temps confiable: "signaturadigital/timestamp/timestamp-certificate.dat

Procés: Per realitzar el test es crida a la classe CreacioSegellDeTemps que hereta de la classe Creació com varem veure a l'AO, aquesta classe inicialitzarà tots els valors necessaris pel proces de creació de segell de temps o signatura CMS. El més important serà que el document a signar amb un segell de temps sigui vàlid i que aleshores es faci un resum del document per signar amb el certificat de la TSA. Un cop preparats aquests parámetres, s'ompliran la resta de paràmetres d'entrada obligatoris per a formar el missatge de petició de servei de creació de signatura (segell de temps) amb tota l'estructura DSS necessària comentada als capítols anteriors.

Després pasarem el procés a la classe ValidacioSegellDeTemps que realitzarà els passos descrits en l'apartat de construcció d'aquesta classe i al javadoc. S'ompliran la resta de paràmetres d'entrada obligatoris per a formar el missatge de petició de servei de validació de signatura amb tota l'estructura DSS necessària comentada als capítols anteriors.

Resultat: El resultat es guardarà als mateixos objectes que els definits en la resta de test al objecte DigitalSignatureContentVo.

Per verificar que el test és satisfactori, ho comprovarem a través d'assertions, en aquest cas comprovarem que el missatge del resultat del segell de temps i de la validació és correcte:

```
vo.getMsg().get(ConstantsApplicacio.MSG_ResultMajor);
            assertTrue(minor.contains("Success"));
            minor = (String)
vo.getMsg().get(ConstantsApplicacio.MSG_ResultMajor);
            assertTrue(minor.contains("Success"));
```

Revisem els logs del procés pas a pas:

```
28-dic-2010 9:27:37 com.steria.digitalsignature.core.BaseSignatura <init>
INFO: BaseSignatura (constructor) init. URL = http://psisbeta.catcert.net/psis/catcert-test/dss
28-dic-2010 9:27:37 com.steria.digitalsignature.core.BaseSignatura <init>
INFO: BaseSignatura (constructor) done. URL = http://psisbeta.catcert.net/psis/catcert-test/dss
28-dic-2010 9:27:37 com.steria.digitalsignature.core.CreacioSegellDeTemps <init>
INFO: constructor CreacioSegellDeTemps() init
28-dic-2010 9:27:37 com.steria.digitalsignature.core.CreacioSegellDeTemps <init>
INFO: soapURL=[http://psisbeta.catcert.net/psis/catcert-test/dss]
28-dic-2010 9:27:37 com.steria.digitalsignature.core.CreacioSegellDeTemps <init>
INFO: constructor CreacioSegellDeTemps() done
28-dic-2010 9:27:37 com.steria.digitalsignature.core.Creacio crear
INFO: crear() init. [PERFIL: null]
28-dic-2010 9:27:37 com.steria.digitalsignature.core.BaseSignatura iniComunicacio
INFO: iniComunicacio() init
28-dic-2010 9:27:38 com.steria.digitalsignature.core.BaseSignatura iniComunicacio
INFO: iniComunicacio() done
28-dic-2010 9:27:39 com.steria.digitalsignature.core.Creacio crear
INFO: VerifyRequestDocument:
28-dic-2010 9:27:39 com.steria.digitalsignature.core.Creacio crear
```

1) Iniciem el procés iniciant els paràmetres base (gràcies a la classe BaseSignatura) com la URL de validació del servei de PSIS, en aquest cas l'entorn de test.
2) Creem el Vo per defecte amb els atributs necessaris com el tipus de PerfilDSS.
3) Afegim el segell de temps a realitzar.
4) Indiquem el perfil de treball XSS.
5) Inicialitzem la comunicació amb el port webservice.
6) Cridem a la creació remota del segell de temps.
7) S'envia el missatge DSS de petició a PSIS.

Pas 1: Petició Signatura

1) Podem comprovar que en aquest cas utilitzem una estructura DSS diferent, el **SignRequest**. Aquest cas és el que utilitzarem en el futur per l'ampliació de funcionalitats quan volguem **signar documents CMS** i XMLDsig.

2) Formació del missatge DSS enviat de petició de creació de signatura CMS, segell de temps. En l'estructura X509Certificate inclourem el certificat de la TSA amb el que signarem el segell de temps.
En l'estructura InputDocuments/DocumentHash inclourem el resum del document que volem signar o afegir el segell de temps (signatura CMS) i el metode de resum (DigestMethod).

```
INFO: <SignRequest xmlns="ur
  <OptionalInputs>
    <KeySelector>
      <xd:KeyInfo>
        <xd:X509Data>

<xd:X509Certificate>MIIHIDCCBgigAwIBAgIQbg5vDBUDLv1ELR24xDVsZzANBgkqhkiG9w0BAQUFADCB8zELMAkGA1UEBhMCRVMx
OzA5BgNVBAoTMkFnZW5jaWEgQ2F0YWxhbmEgZGUgQ2VydGlmaWNhY2lvICChOSUYgUS0wODAxMTc2LUkpMSgwJgYDVQQLEx9T
ZXJ2ZWlzIFB1YmxpY3MgZGUgQ2VydGlmaWNhY2lvMTUwMwYDVQQLEyxWZWdldSBodHRwczovL3d3dy5jYXRjZXJ0Lm5ldC92ZXJhc
...
...
5jYXRjZXJ0Lm5ldC9jcmwvZWMtYWNjLmNybIYoaHR0cDovL2Vwc2NkMi5jYXRjZXJ0Lm5ldC9jcmwvZWMtYWNjLmNybDANBgkqhkiG9w
0BAQUFAAOCAQEAOb4kSwvzbM51c1weMj5Udw8jE9HwV8NUVjHTOu+WT2f5cqRUu8oo3F4REK59hnPbxLaVZ/8zDp2afqKoGOAd9e
8TCKY3Nx7tOijnd+jnEIYe/BTisMTXqx8+o0eYoI9uFX/iMQmms569KsPXGnVdbyuys5LE6iCfeeOOVwz9ruKeDwX6f+MQw9mkTguh7vFeb
CNpyfxIzjbDHXKy1NOeVdd1UYs2tgPhOqHBHXZLepU8aRZx3ixKF7TQaipnQc4PLrKUqPrPkQCMtN73b1RrOtfc+dO9C8ZtLIwVEld9vbY
Ap9n0TvdxbWrHSHvFfkFWjq8HRUD+ptXHDZaWBxmuUQ==</xd:X509Certificate>
        </xd:X509Data>
      </xd:KeyInfo>
    </KeySelector>
    <SignatureType>oasis:names:tc:dss:1.0:core:schema:XMLTimeStampToken</SignatureType>
    <IncludeObject ObjId="Doc1" WhichDocument="Doc1" hasObjectTagsAndAttributesSet="false" createReference="true"/>
  </OptionalInputs>
  <InputDocuments>
    <DocumentHash ID="Doc1">
      <xd:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <xd:DigestValue>A96Ovg/94oa1n5SJ3vZi1Quhkfo=</xd:DigestValue>
    </DocumentHash>
  </InputDocuments>
</SignRequest>
```

Pas 2: Resposta Signatura

1) Podem veure el resultat de la petició de signatura a PSIS en el missatge DSS de resposta **SignResponse**.
2) Com que hem demanat OptionalOutputs ens retornen la informació de amb que algoritme hem realitzat la signatura (RSA-SHA1) o el resum (SHA1) o el valor del resum (DigestValue).
3) Podem veure que el resultat és que la signatura CMS o segell de temps s'ha creat correctament i per tant és vàlida: "Signature created"

```
INFO: <dss:SignResponse Profile="urn:oasis:names:tc:dss:1.0:core:schema" xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
 <dss:Result>
  <dss:ResultMajor>urn:oasis:names:tc:dss:1.0:resultmajor:Success</dss:ResultMajor>
  <dss:ResultMessage xml:lang="en">Signature created</dss:ResultMessage>
 </dss:Result>
 <dss:OptionalOutputs/>
 <dss:SignatureObject>
  <dss:Timestamp>
   <ds:Signature Id="id-afd66e8e-48f1-4ef2-8e23-67434475f7df" xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:SignedInfo Id="id-6f569e7e-606c-4e6c-96d5-6c8674cd9bc9">
     <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
     <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
     <ds:Reference URI="#TSTInfo-id-5a232c6e-b6ad-4932-93b4-3c3990c0b35c" Id="id-8120c66e-cee3-4c46-ac3d-a9297300a90f"
Type="urn:oasis:names:tc:dss:1.0:core:schema:XMLTimeStampToken">
      <ds:Transforms>
       <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
      </ds:Transforms>
      <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <ds:DigestValue>Qv4rUyDg/i6b5Ohn5f84Mg12Rqw=</ds:DigestValue>
     </ds:Reference>
     <ds:Reference Id="id-318da3d6-9f7b-4e77-99e7-ed1fdc1afd10">
      <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <ds:DigestValue>A96Ovg/94oa1n5SJ3vZi1Quhkfo=</ds:DigestValue>
     </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>ROGkWPXLkTGwg4CwdGO+rOaQwB5JGP31m1lVCRPYNPUtopVisd0a/GyrtQFh3haKYcgs4lIWE2lZ
HchOlCQ/OkHzoTEyaWQPEVPQIgVbHEVmb4VbT2zmL6cHKmJmxtIocR0G2jNO1kDub/fnJooAJb6U
anntSJ8tkWEmAKmJVVs=</ds:SignatureValue>
    <ds:KeyInfo>
     <ds:X509Data>

<ds:X509Certificate>MIIHIDCCBgigAwIBAgIQRJGqGlQxs1lKdr1fkBo2UjANBgkqhkiG9w0BAQUFADCB8zELMAkGA1UEBhMCRVMxOz
A5BgNVBAoTMkFnZW5jaWEgQ2F0YWxhbmEgZGUgQ2VydGlmaWNhY2lvIChOSUYgUS0wODAxMTc2LUkpMSgwJgYDVQQLEx9TZX
J2ZWlzIFB1YmxpY3MgZGUgQ2VydGlmaWNhY2lvMTUwMwYDVQQLEyxWZWdldSBodHRwczovL3d3dy5jYXRjZXJ0Lm5ldC92ZXJhcnJl
...
AOCAQEAgdRwMoh110JPkLZCrUe8WDxBbfynTUGnsvqLcn9f2V53BvhKPRsjqLau3J40o834/Y/c9HhQgjS5k9CZYBwz2ApI4hDwQyIB/n
y9IP/6nseegRpGZUp/WuHjAg4HC6MhyvbidvL2o2Au2/8scyHmIRj9om5+rEL41+rqQnsNiOfgqXdeHdTZpuhsJ55l1i6gGikFrqubHO6ns/l3X
76/wCnhQu1MQCGhDHJZIB5W4zZe/e+QEVLo12A6gwBxfelcAS/I4TVv+xkEOPLlEaf9DkuxBhxABoFppcUZX7uxd3H1lifrVyVS6ySRxc52
dD9uSOaP7CWZo6QDlMUfhhxP1w==</ds:X509Certificate>
     </ds:X509Data>
     <ds:KeyValue>
      <ds:RSAKeyValue>

<ds:Modulus>AN0p/gklo9HpvgtKEU+spB8GDC/geIBsjyiIvy+aWEPNEytxnFnhSXYvncYBbOvmOBgEVNCk4O5ht9EjVH6txbTTt9IOLvI5C
GpW8Q2s3Eq7BJ7Stzn23aEo0Z/gL4Ga4IPVNEsQAWI8VrQ8H5Bg3lnoCOiZp1FFySfl8KUpuKYt</ds:Modulus>
       <ds:Exponent>AQAB</ds:Exponent>
      </ds:RSAKeyValue>
     </ds:KeyValue>
    </ds:KeyInfo>
    <ds:Object Id="TSTInfo-id-5a232c6e-b6ad-4932-93b4-3c3990c0b35c" MimeType="application/xml">
     <dss:TstInfo>
      <dss:SerialNumber>627173490610980675645508370432570539655644812023</dss:SerialNumber>
      <dss:CreationTime>2010-12-28T08:27:41.117Z</dss:CreationTime>
      <dss:Policy>urn:oid:0.4.0.2023.1.1</dss:Policy>
      <dss:ErrorBound>PT1S</dss:ErrorBound>
      <dss:Ordered>true</dss:Ordered>
      <dss:TSA Format="urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName">OU=Serveis Publics de Certificacio CIT-
1,OU=Vegeu https://www.catcert.net/verCIT-1 (c)05,OU=Jerarquia Entitats de Certificacio Catalanes,O=Agencia Catalana de Certificacio
(NIF Q-0801176-I),C=ES,CN=Servei de segellat de temps de PSIS INTEGRACIO</dss:TSA>
     </dss:TstInfo>
    </ds:Object>
   </ds:Signature>
  </dss:Timestamp>
 </dss:SignatureObject>
```

4) Podem veure el certificat digital de la TSA amb que s'ha signat i les seves careacteristiques.
5) Així com comprovar el serveis de segellat de temps de PSIS.

Pas 3: Tractament resposta signatura

1) Podem veure que el segell de temps s'ha creat corretament com resposta a la primera crida cap als serveis de PSIS.
2) Podem veure els logs del mètode generaHashResultat: Un cop hem rebut la resposta del servei de PSIS en format XML (DSS), tractarem la resposta com s'explica a l'apartat de construcció a través del mètode ValidacioSignatura.generaHashResultat() aquest afegirà tota la informació resultant en els claus/valors definits en la classe ConstantsApplicacio en el hash d'atributs de resposta **DigitalSignatureContentVo.atributs.get(ApplicationConfig.SIGN_signatura)** per a que hom que utilitzi la nostra api pugui obtenir fàcilment aquesta informació. Per exemple aquest mètode et retorna el valor de la signatura realitzada.

```
28-dic-2010 9:27:41 com.steria.digitalsignature.core.Creacio crear
INFO: Segell Creat!
28-dic-2010 9:27:41 com.steria.digitalsignature.core.Creacio crear
INFO: Hash Atributs: {dss:SignatureObject=<?xml version="1.0" encoding="UTF-8"?>
<ds:Signature Id="id-afd66e8e-48f1-4ef2-8e23-67434475f7df" xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<ds:SignedInfo Id="id-6f569e7e-606c-4e6c-96d5-6c8674cd9bc9">
<ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
<ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
<ds:Reference URI="#TSTInfo-id-5a232c6e-b6ad-4932-93b4-3c3990c0b35c" Id="id-8120c66e-cee3-4c46-ac3d-a9297300a90f"
Type="urn:oasis:names:tc:dss:1.0:core:schema:XMLTimeStampToken">
<ds:Transforms>
<ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
</ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<ds:DigestValue>Qv4rUyDg/i6b5Ohn5f84Mg12Rqw=</ds:DigestValue>
</ds:Reference>
<ds:Reference Id="id-318da3d6-9f7b-4e77-99e7-ed1fdc1afd10">
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<ds:DigestValue>A96Ovg/94oa1n5SJ3vZi1Quhkfo=</ds:DigestValue>
</ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>
ROGkWPXLkTGwg4CwdGO+rOaQwB5JGP31m1lVCRPYNPUtopVisd0a/GyrtQFh3haKYcgs4lIWE2lZ
HchOlCQ/OkHzoTEyaWQPEVPQIgVbHEVmb4VbT2zmL6cHKmJmxtIocR0G2jNO1kDub/fnJooAJb6U
anntSJ8tkWEmAKmJVVs=
</ds:SignatureValue>
<ds:KeyInfo><ds:X509Data><ds:X509Certificate>MIIHIDCCBgigAwIBAgIQRJGqGlQxs1lKdr1fkBo2UjANBgkqhkiG9w0BAQUFADCB8zELMA
kGA1UEBhMCRVMxOzA5BgNVBAoTMkFnZW5jaWEgQ2F0YWxhbmEgZGUgQ2VydGlmaWNhY2lvIChOSUYgUS0wODAxMTc2LUkpMSgw
...
JgYDVQQLEx9TZXJ2ZWlzIFB1YmxpY3MgZGUgQ2VydGlmaWNhY2lvMTUwMwYDVQQLEyxWZWdldSBodHRwczovL3d3dy5jYXRjZXJ0Lm
5ldC92ZXJhcnJlbCAoYykwMzE1MDMGA1UECxMsSmVyYXJxdWlhIEVudGl0YXRzIGRlIENlcnRpZmljYWNpbyBDYXRhbGFuZXMxDzANBg
cate></ds:X509Data><ds:KeyValue><ds:RSAKeyValue><ds:Modulus>AN0p/gklo9HpvgtKEU+spB8GDC/geIBsjyiIvy+aWEPNEytxnFnhSXYv
ncYBbOvmOBgEVNCk4O5ht9EjVH6txbTTt9IOLvI5CGpW8Q2s3Eq7BJ7Stzn23aEo0Z/gL4Ga4IPVNEsQAWI8VrQ8H5Bg3lnoCOiZp1FFySfl8
KUpuKYt</ds:Modulus><ds:Exponent>AQAB</ds:Exponent></ds:RSAKeyValue></ds:KeyValue></ds:KeyInfo><ds:Object Id="TSTInfo-id-
5a232c6e-b6ad-4932-93b4-3c3990c0b35c"
MimeType="application/xml"><dss:TstInfo><dss:SerialNumber>627173490610980675645508370432570539655644812023</dss:SerialNum
ber><dss:CreationTime>2010-12-
28T08:27:41.117Z</dss:CreationTime><dss:Policy>urn:oid:0.4.0.2023.1.1</dss:Policy><dss:ErrorBound>PT1S</dss:ErrorBound><dss:Orde
red>true</dss:Ordered><dss:TSA Format="urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName">OU=Serveis Publics de
Certificacio CIT-1,OU=Vegeu https://www.catcert.net/verCIT-1 (c)05,OU=Jerarquia Entitats de Certificacio Catalanes,O=Agencia Catalana de
Certificacio (NIF Q-0801176-I),C=ES,CN=Servei de segellat de temps de PSIS INTEGRACIO</dss:TSA></dss:TstInfo></ds:Object>
</ds:Signature>}
28-dic-2010 9:27:41 com.steria.digitalsignature.core.Creacio crear
INFO: crear() done. [PERFIL: urn:oasis:names:tc:dss:1.0:profiles:timestamping]
28-dic-2010 9:27:41 com.steria.digitalsignature.core.BaseSignatura <init>
INFO: BaseSignatura (constructor) init. URL = http://psisbeta.catcert.net/psis/catcert-test/dss
28-dic-2010 9:27:41 com.steria.digitalsignature.core.BaseSignatura <init>
INFO: BaseSignatura (constructor) done. URL = http://psisbeta.catcert.net/psis/catcert-test/dss
28-dic-2010 9:27:41 com.steria.digitalsignature.core.ValidacioSegellDeTemps <init>
INFO: constructor ValidacioSegellDeTemps() init
28-dic-2010 9:27:41 com.steria.digitalsignature.core.ValidacioSegellDeTemps <init>
INFO: soapURL=[http://psisbeta.catcert.net/psis/catcert-test/dss]
28-dic-2010 9:27:41 com.steria.digitalsignature.core.ValidacioSegellDeTemps <init>
INFO: constructor ValidacioSegellDeTemps() done
28-dic-2010 9:27:41 com.steria.digitalsignature.core.Validacio validar
INFO: validar() init. [PERFIL: null]
28-dic-2010 9:27:41 com.steria.digitalsignature.core.BaseSignatura iniComunicacio
INFO: iniComunicacio() init
28-dic-2010 9:27:41 com.steria.digitalsignature.core.BaseSignatura iniComunicacio
INFO: iniComunicacio() done
28-dic-2010 9:27:42 com.steria.digitalsignature.core.Validacio validar
INFO: VerifyRequestDocument:
28-dic-2010 9:27:42 com.steria.digitalsignature.core.Validacio validar
```


3) En aquest cas tornem a cridar a un altre Server de PSIS per a validar el document i segell de temps realitzar en el pas anterior.

Pas 4: Petició Validació Signatura Segell de temps

Formació del missatge DSS enviat de petició de validació de signatura.
El missatge és de tipus VerifyRequest a l'estructura d'aquest missatge incloem el document signat amb el segell de temps i el certificat amb el que varem signar.

```
INFO: <VerifyRequest xmlns="urn:oasis:names:tc:dss:1.0:core:schema" xmlns:xd="http://www.w3.org/2000/09/xmldsig#">
  <OptionalInputs>
    <ReturnProcessingDetails/>
  </OptionalInputs>
  <InputDocuments>
    <DocumentHash ID="Doc1">
      <xd:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <xd:DigestValue>A96Ovg/94oa1n5SJ3vZi1Quhkfo=</xd:DigestValue>
    </DocumentHash>
  </InputDocuments>
  <SignatureObject>
    <ds:Signature Id="id-afd66e8e-48f1-4ef2-8e23-67434475f7df" xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
      <ds:SignedInfo Id="id-6f569e7e-606c-4e6c-96d5-6c8674cd9bc9">
        <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
        <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
        <ds:Reference URI="#TSTInfo-id-5a232c6e-b6ad-4932-93b4-3c3990c0b35c" Id="id-8120c66e-cee3-4c46-ac3d-a9297300a90f"
Type="urn:oasis:names:tc:dss:1.0:core:schema:XMLTimeStampToken">
          <ds:Transforms>
            <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
          </ds:Transforms>
          <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
          <ds:DigestValue>Qv4rUyDg/i6b5Ohn5f84Mg12Rqw=</ds:DigestValue>
        </ds:Reference>
        <ds:Reference Id="id-318da3d6-9f7b-4e77-99e7-ed1fdc1afd10">
          <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
          <ds:DigestValue>A96Ovg/94oa1n5SJ3vZi1Quhkfo=</ds:DigestValue>
        </ds:Reference>
      </ds:SignedInfo>
      <ds:SignatureValue>ROGkWPXLkTGwg4CwdGO+rOaQwB5JGP31m1lVCRPYNPUtopVisd0a/GyrtQFh3haKYcgs4lIWE2lZ
HchOlCQ/OkHzoTEyaWQPEVPQIgVbHEVmb4VbT2zmL6cHKmJmxtIocR0G2jNO1kDub/fnJooAJb6U
anntSJ8tkWEmAKmJVVs=</ds:SignatureValue>
      <ds:KeyInfo>
        <ds:X509Data>

<ds:X509Certificate>MIIHIDCCBgigAwIBAgIQRJGqGlQxs1lKdr1fkBo2UjANBgkqhkiG9w0BAQUFADCB8zELMAkGA1UEBhMCRVMxOz
A5BgNVBAoTMkFnZW5jaWEgQ2F0YWxhbmEgZGUgQ2VydGlmaWNhY2lvIChOSUYgUS0wODAxMTc2LUkpMSgwJgYDVQQLEx9TZX
...
J2ZWlzIFB1YmxpY3MgZGUgQ2VydGlmaWNhY2lvMTUwMwYDVQQLEyxWZWdldSBodHRwczovL3d3dy5jYXRjZXJ0Lm5ldC92ZXJhcnJl
76/wCnhQu1MQCGhDHJZIB5W4zZe/e+QEVLo12A6gwBxfelcAS/I4TVv+xkEOPLlEaf9DkuxBhxABoFppcUZX7uxd3H1lifrVyVS6ySRxc52
dD9uSOaP7CWZo6QDlMUfhhxP1w==</ds:X509Certificate>
        </ds:X509Data>
        <ds:KeyValue>
          <ds:RSAKeyValue>

<ds:Modulus>AN0p/gklo9HpvgtKEU+spB8GDC/geIBsjyiIvy+aWEPNEytxnFnhSXYvncYBbOvmOBgEVNCk4O5ht9EjVH6txbTTt9IOLvI5C
GpW8Q2s3Eq7BJ7Stzn23aEo0Z/gL4Ga4IPVNEsQAWI8VrQ8H5Bg3lnoCOiZp1FFySfl8KUpuKYt</ds:Modulus>
            <ds:Exponent>AQAB</ds:Exponent>
          </ds:RSAKeyValue>
        </ds:KeyValue>
      </ds:KeyInfo>
      <ds:Object Id="TSTInfo-id-5a232c6e-b6ad-4932-93b4-3c3990c0b35c" MimeType="application/xml">
        <dss:TstInfo>
          <dss:SerialNumber>627173490610980675645508370432570539655644812023</dss:SerialNumber>
          <dss:CreationTime>2010-12-28T08:27:41.117Z</dss:CreationTime>
          <dss:Policy>urn:oid:0.4.0.2023.1.1</dss:Policy>
          <dss:ErrorBound>PT1S</dss:ErrorBound>
          <dss:Ordered>true</dss:Ordered>
          <dss:TSA Format="urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName">OU=Serveis Publics de Certificacio CIT-
1,OU=Vegeu https://www.catcert.net/verCIT-1 (c)05,OU=Jerarquia Entitats de Certificacio Catalanes,O=Agencia Catalana de Certificacio
(NIF Q-0801176-I),C=ES,CN=Servei de segellat de temps de PSIS INTEGRACIO</dss:TSA>
        </dss:TstInfo>
      </ds:Object>
    </ds:Signature>
  </SignatureObject>
</VerifyRequest>
```

Pas 5: Resposta Validació Signatura

1) Podem veure el resultat de la petició de validació a PSIS en el missatge DSS de resposta VerifyResponse.
2) Podem veure que el resultat es que la signatura és valida: “valid:signature:onAllDocuments”
3) Com que hem demanat OptionalOutputs ens retornen la informació adicional de la validació, com per exemple en **el moment en que la signatura es va realitzar i per tant el moment en que era vàlida**.

```
INFO: <dss:VerifyResponse Profile="urn:oasis:names:tc:dss:1.0:core:schema" xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <dss:Result>
    <dss:ResultMajor>urn:oasis:names:tc:dss:1.0:resultmajor:Success</dss:ResultMajor>
    <dss:ResultMinor>urn:oasis:names:tc:dss:1.0:resultminor:valid:signature:onAllDocuments</dss:ResultMinor>
  </dss:Result>
  <dss:OptionalOutputs>
    <dss:ProcessingDetails>
      <dss:ValidDetail Type="urn:oasis:names:tc:dss:1.0:detail:Signature">
        <dss:Message xml:lang="en">The signature is valid.</dss:Message>
      </dss:ValidDetail>
      <dss:ValidDetail Type="urn:oasis:names:tc:dss:1.0:detail:ValidityInterval">
        <dss:Message xml:lang="en">The signing key is inside its static validity interval.</dss:Message>
      </dss:ValidDetail>
      <dss:ValidDetail Type="urn:oasis:names:tc:dss:1.0:detail:IssuerTrust">
        <dss:Message xml:lang="en">The issuer of the given key is trusted.</dss:Message>
      </dss:ValidDetail>
      <dss:ValidDetail Type="urn:oasis:names:tc:dss:1.0:detail:RevocationStatus">
        <dss:Message xml:lang="en">The signing key is not revoked.</dss:Message>
      </dss:ValidDetail>
      <dss:ValidDetail Type="urn:oasis:names:tc:dss:1.0:detail:ValidationTime">
        <dss:Message xml:lang="en">The signature is validated at: Tue Dec 28 09:27:42 CET 2010</dss:Message>
      </dss:ValidDetail>
    </dss:ProcessingDetails>
  </dss:OptionalOutputs>
</dss:VerifyResponse>
28-dic-2010 9:27:42 com.steria.digitalsignature.core.Validacio validar
INFO: Validació realitzada, resposta rebuda.
28-dic-2010 9:27:42 com.steria.digitalsignature.core.Validacio validar
INFO: Hash Atributs: {}
28-dic-2010 9:27:42 com.steria.digitalsignature.core.Validacio validar
INFO: Hash Msg(returned): {dss:ResultMinor=urn:oasis:names:tc:dss:1.0:resultminor:valid:signature:onAllDocuments,
dss:ResultMajor=urn:oasis:names:tc:dss:1.0:resultmajor:Success}
28-dic-2010 9:27:42 com.steria.digitalsignature.core.Validacio validar
INFO: validar() done. [PERFIL: urn:oasis:names:tc:dss:1.0:profiles:timestamping]
```

Pas 6: Tractament de la resposta Validació Signatura

4) En el detall del missatge de resposta podem veure que el certificat és vàlid i aquesta resposta amb tots els parametres de la resposta s’emmagatzemarà al DigitalSignatureContentVo HashMap key ResultMinor i HashMap atributs.

5.6 Capítol 6: Conclusions i extensions de futur

Aquest projecte ens ha servit per a conèixer el món de la Signatura Digital i utilitzar les primitives bàsiques de PKI a través de proveïdors fiables de serveis de Signatura com CATCert.

En aquest hem pogut comprovar com les eines informàtiques poden ajudar a resoldre problemes competitius a les empreses i ha facilitar l'acces digital a les persones.

Hem vist quines eines eficaces existeixen pels serveis de Signatura Digital o PKI, sobre quins estàndards es recolzen aquesta infrastructura (DSS, XSS, XML, X509, XMLDSig, CMS) i quins proveïdors d'aquests serveis podem utilitzar. En aquest sentit hem entrat en detall en els serveis que proporciona CATCert a través de la seva plataforma de serveis d'identificació i signatura reconeguda, PSIS. A partir d'aquest punt hem analitzat, dissenyat i implementat una llibreria que proporciona abstracció a una futura plataforma de tramitació electrònica que a partir d'aquest punt anirà creixent i evolucionant juntament amb la Societat Digital i l'Era de la Informació.

El futur de la Signatura reconeguda passa per l'estandardització de les signatures i el format XML i per tant la signatura XMLDsig té molts punts d'una bona evolució donat que aquest format més nou que el CMS, té grans avantatges com:

- XML: format estàndard interoperable.
- Tot es pot plasmar en el format XML, qualsevol tipus de document, i queda representat d'una forma homogènia, autocontinguda i estàndard i sempre interpretable en el temps sense dependre d'eines exteriors.
- Fàcilment interpretable, autocontingut, es poden realitzar manipulacions del contingut i canviar formats de codis fàcilment (per exemple amb XPath, XSD, XSLT, etc.).

La llibreria creada en aquest projecte posa les bases d'una llibreria creada per utilitzar aquests serveis a través d'estàndards (DSS, CMS, XMLDsig, etc.) a partir d'aquest punt s'anirà ampliant les possibilitats i funcionalitats de la llibreria: amb nous serveis i ampliació dels existents, Realitzar Signatures XMLDsig, Polítiques de Signatura, arxiu electrònic (i-Arxiu), Signatura de Formularis (XMLDsig), Etc.

També s'ampliarà el mòdul amb un accesor a BBDD i un model de BBDD preparat per a guardar evidències, signatures detached CMS o XMLDsig. Aquest suport persistent (taules a BD) al component de ProveidorSignatura ens permetrà registrar les signatures del formulari o d'altres necessàries i afegir funcionalitats de recuperació i validació d'aquestes signatures juntament amb un frontal on l'usuari pugui consultar i validar la sol·licitud o formulari que va realitzar.

En les extensions de futur també obtindrem el frontal de Signatura Digital, que està basat en la tecnologia Java, concretament és Java Web Start, vindrà signada en tots els seus components i treballarà amb els recursos necessaris i les comunicacions necessàries amb el servidor (JBoss) de forma segura (HTTPS). Aquesta aplicació té un mòdul que requereix compilació amb JDK 1.6. Per les diverses compilacions, signatura dels diversos mòduls i l'assemblatge es proveirà d'un script ant que ho manega. Aquest component ens permetrà també disposar d'un frontal de signatura que interactuarà amb l'usuari final.

6 Glossari:

Online Pure-players	Model de negoci on l'empresa es dedica només al negoci pel canal on-line
Clicks-and-bricks	Model de negoci on l'empresa utilitza el canal presencial i el canal on-line pel negoci.
AF	Anàlisi Funcional
AO	Anàlisi Orgànic
PKI	Public Key Infrastructure
AE	Administració Electrònica
DSS	Digital Signature Services
WBS	Work Breakdown Structure
PSIS	Plataforma de Sistemes d'Identificació i Signatura
CA	Certificate Authority
CAdES	CMS Advanced Electronic Signatures
CMS	Cryptographic Message Syntax (RFC 3852)
DSS	Digital Signature Services
JDK	Java Developer Kit
JRE	Java Runtime Environment
JCA/JCE	Java Cryptography Architecture & JavaCryptography Extensions
JSSE	Java Secure-Sockets Extension
JAAS	Java Authentication & AuhorizationService
JWS	Java Web Start Technology
OASIS	Organization for the Advancement of Structured Information Standards
PDF	Portable Document Format
PKCS7	Public Key Cryptography Standards
RFC	Request For Comments
W3C	World Wide Web Consortium
SOAP	Simple Object Access Protocol
SSL	Secure Sockets Layer

TLS	Transport Layer Security
TSA	Transportation Safety Administration
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTF	Universal Transformation Format
VA	Autoritat de Validació
XSS	eXtended Signature Services (XSS) Profile of the OASIS Digital Signature Service (DSS)
WSDL	Web Service Definition Language
XAdES	XML Advanced Electronic Signatures
XML	Extensible Markup Language
XMLDsig	XML Digital Signatures
XSD	XML Schema Definition
CRL	Certificate Revocation List
CATCert	Agència Catalana de Certificació
PKCS	Public-Key Cryptography Standards
PFX	Personal Information Exchange File
LDAP	Lightweight Directory Access Protocol
OCSP	Online Certificate Status Protocol
TSA	Time Stamp Authority
ASN.1	Abstract Syntax Notation
SOA	Simple Object Access Protocol
J2EE	The Java 2 Platform, Enterprise Edition
HTTPS	Hypertext Transfer Protocol Secure
XSD	XML Schema Definition
CSS	Cascading Style Sheets
W3C	World Wide Web Consortium
POJO	Plain Old Java Object
JTA	Java Transaction Service

JMS	Java Message Service
JAF	Activation Framework
RMI	Remote Method Invocation
IIOP	Internet Inter-Object Protcol
JNDI	Java Naming and Directory Interface
NIS+	Network Information Service Plus
JDBC	Java DataBase Connectivity
MVC	Model View Controler
JSP	Java Server Pages
Taglibs	Tag Libraries
EJB	Enterprise Java Beans
JWS	Java Web Start
EAR	Enterprise Application Resource
WAR	Web Application Resource
JAR	Java Application Resource
SSO	Single Sing-On
VO	Value Object

7 Bibliografia i referències

Fitxer	Títol
Abc De La Signatura Electrònica	Abc De La Signatura Electrònica Autor: Alamillo, Ignacio Isbn: 9788439367956 Any: 2005
dss-v1[1].0-spec-cd-Core-r03.pdf	Digital Signature Service Core Protocols, Elements, and Bindings
oasis-dss-1.0-core-profiles-XSS-spec-wd02.doc	eXtended Signature Services (XSS) Profile of the OASIS Digital Signature Service (DSS)
Guia_Segellat_Temps_v1.8.pdf	Guia del servei de segellat de temps de CATCert
signaturaAcrobat7_v1.pdf	Ús de la signatura electrònica amb Adobe Acrobat 7.0
	Documentació amb les especificacions del Protocol de missatgeria DSS (Digital Signatura Standard) i els esquemes per a qualsevol implementació. DSS defineix una interfície XML a una interface XML amb la finalitat de processar signatures digitals per a serveis Web i altres aplicacions.
Guia bàsica pels integradors de PSIS_20071015.pdf	Guia d'integradors de PSIS. Per la creació de clients des de WSDL. Inclou exemples en codi font Java 1.5, .Net i Visual Basic 6
Manual d'ús de l'eina web de signatura-e v1.9.5.pdf	Eina de frontal de signatura desenvolupada per CATCert.

URL	Descripció
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=dss	Pàgina web oficial d'estandardització del protocol DSS
http://www.webservices.org/	Pàgina web d'informació general referent als serveis web.

http://www.w3.org/TR/soap/	Pàgina web d'estandardització del protocol SOAP
http://www.w3.org/TR/2003/NOTE-XAdES-20030220/	Pàgina web d'estandardització de firmes digitals XadES.
http://www.w3.org/TR/xmldsig-core/	Pàgina web d'estandardització de firmes digitals DSIG
http://ws.apache.org/axis/	Pàgina web oficial d'AXIS
http://xmlbeans.apache.org/	Pàgina web oficial de XMLBeans
http://xfire.codehaus.org/	Pàgina web oficial de XFire