Projecte Fi de Carrera:
Esquema criptogràfic per exàmens electrònics segurs

Aleix Dorca Josa
Enginyeria en Informàtica

Jordi Castellà Roca
Consultor

Data de Lliurament
10 de Gener de 2005

Agraïments i Dedicatòria

El projecte final de carrera és una fita a la que no s'hi arriba tot sol. En el camí fins arribar aquí hi han hagut moltes persones que han influït en el resultat final, és per això que seria una descortesia si no les anomenés aquí.

Primer que ningú, la persona a qui més li dono les gràcies és la Nina, qui sempre, des d'abans que comencés el segon cicle fins avui, ha estat al meu costat, animant-me i amb la paciència suficient per a superar els moments més difícils.

Agraeixo moltíssim a la meva família, Aleix Dorca Bis, Maria Assumpta Josa Mas, Maria Assumpta Dorca Josa i Albert Dorca Josa que són per sobre tot els meus millors amics i les persones sense les quals avui no estaria presentant aquest projecte.

Moltes gràcies a en Jordi Castellà Roca, consultor del Projecte, qui ha estat sempre disposat a resoldre dubtes i a explicar i tornar a explicar tot el que no estava massa clar. L'estructura i la metodologia proposada durant el desenvolupament del Projecte ha estat una de les coses de les que més he après, i que han fet que aquest Projecte comencés com un treball interessant i acabés com una experiència apassionant.

Tot i no tenir res a veure amb el Projecte, molta gent del meu voltant d'alguna manera o altra a ajudat a fer la convivència més afable i és per això que també els hi dono les gràcies. Carlos Lozano, Miguel Espinosa, Jordi Gorgues, Antoni Mestre, Marc Martins, Jaume Ribolleda, Eugeni Serrano, Oriol Garcia, Arjen Anthony Lucassen i Neal Morse.

Moltes gràcies a la Universitat d'Andorra i a la Universitat Oberta de Catalunya per la oportunitat i per l'ajuda prestada.

Dedico aquest projecte als meus pares, Alejo i Marisun.

Durant la vida acadèmica de qualsevol estudiant cal passar per l'etapa dels exàmens. Aquesta tasca tan comuna en universitats presencials planteja un dilema en la formació a distància, essent la Universitat Oberta de Catalunya un exemple perfecte. Si tota la formació es fa a distància, perquè els exàmens en aquest tipus d'ensenyaments no ho són? Probablement hi hauran diverses visions alhora de respondre aquesta pregunta. En qualsevol cas, per poc que pensem en la realització d'exàmens a distància ens sorgeixen un seguit de qüestions a resoldre: com podem assegurar l'autenticitat i integritat de les dades? com podem estar segurs que la persona que fa l'examen és qui diu ser?

En aquest projecte de final de carrera s'ha dissenyat, implementat i provat un sistema que mitjançant criptografia de clau pública, garanteix la correcció en totes les fases d'un procés d'examen electrònic. Entenem que aquest és un pas previ a la realització d'exàmens a distància.

En un procés d'examen electrònic cal garantir com a mínim les propietats següents: l'autenticitat de les dades, la privacitat de les respostes, una correcció legal (el professor no ha de saber qui és l'autor de la resposta), el secret de les respostes durant l'examen, la garantia de lliurament (l'estudiant ha de disposar d'un rebut com a prova que ha realitzat l'examen), i finalment la impossibilitat de còpia.

Garantir aquestes propietats utilitzant un sistema informàtic no és una tasca trivial, i a més a més dels elements de seguretat a nivell de xarxa de comunicacions, fa falta la utilització de mesures concretes a nivell d'aplicació. Per aquest motiu s'ha dissenyat un protocol criptogràfic per cada fase de l'examen. El conjunt d'aquests protocols forma l'esquema criptogràfic que ha estat implementat en aquest sistema per a exàmens electrònics segurs.

El sistema presentat permetria que actualment es conduïssin exàmens electrònics en entorns controlats. El fet que els exàmens estiguin en format digital facilita molt la gestió de les proves. L'estudiant no cal que digui a quina seu realitzarà l'examen, senzillament el dia de l'examen va a una de les seus i respon l'examen. Alhora de corregir els exàmens s'evita la necessitat de passar les respostes dels exàmens en paper a format digital perquè aquestes ja estan en format digital. El consultor obté les respostes i pot corregir-les evitant errors deguts a la possible mala cal·ligrafia de l'estudiant. En el cas exàmens de tipus test la correcció podria ser automàtica.

Podem pensar en la realització d'exàmens en entorns remots no controlats, per exemple des de casa de l'estudiant, com el següent pas lògic. Malauradament, hi ha problemes que difícilment es poden resoldre amb la criptografia. Un exemple seria evitar la còpia. L'esquema ha estat implementat de manera que si en un futur aquests problemes es resolen es pugui aprofitar l'actual implementació.

Índex

Índex de figures.

1. Introducció

1.1 Justificació del Projecte Final de Carrera.

La societat actual exigeix una formació constant per estar al dia en el camp professional, alhora que absorbeix molt temps. La formació a distància és una bona solució perquè permet una formació de qualitat evitant la rigidesa de les classes presencials, i la pèrdua de temps en els desplaçament. És a dir, l'estudiant aprofita al màxim el seu temps disponible.

La proliferació de les tecnologies de la informació, i principalment Internet, han incrementat notablement l'oferta de formació a distància, a la vegada que han permès nous mètodes docents.

Fins ara, l'únic element de la formació a distància que encara es manté presencial són els exàmens. Al final del semestre, quan típicament es fa l'avaluació, l'estudiant s'ha de desplaçar a un centre on, juntament amb la resta d'estudiants, s'examinarà de manera clàssica.

Una primera justificació d'aquest projecte sorgeix de la necessitat d'implantar gradualment un sistema que permeti a un estudiant, sigui un estudiant d'una Universitat presencial o no, el poder realitzar els seus exàmens des d'un emplaçament remot assegurant en tot moment les característiques fonamentals que s'han de complir en qualsevol examen. Tal com s'indica a [3], actualment impedir la còpia d'exàmens no fa possible la realització d'exàmens remots, per aquest motiu el projecte només està destinat a permetre els exàmens remots en entorns controlats.

Una segona justificació és la simplificació de la gestió del procés d'examen en la comunitat de la UOC. La utilització d'un sistema per realitzar exàmens electrònics facilita en gran mesura els punts següents:

- Els estudiants no han d'indicar a quina seu realitzaran l'examen per tal de fer-los–hi arribar els exàmens. Els estudiants senzillament es desplacen a la seu que els hi és més convenient en cada moment.

- No cal escanejar les respostes dels estudiants per tal d'obtenir-les en format digital i d'aquesta manera enviar-les als consultors. Les respostes ja estan en format digital.

- Els consultors no han d'esperar que les respostes estiguin en format digital. Un cop l'examen ha finalitzat ja és possible corregir-lo.

- Els errors o dificultats que es deriven de la mala cal·ligrafia d'alguns estudiants poden evitar-se amb la utilització d'un teclat com a perifèric d'entrada.

- Si l'examen és de tipus test la correcció pot ser automàtica. L'estudiant pot saber la seva nota de forma instantània.

- La utilització d'un històric permet cercar informació de proves passades amb gran facilitat.

1.2 Context en el qual es desenvolupa.

Fa relativament poc temps tot el que aquest Projecte proposa hagués estat una mica fora de lloc, però avui se'ns presenta un marc en el que els exàmens a distància són una possibilitat i un sistema com aquest és necessari.

Les principals raons del perquè fa poc no s'hagués pogut dur a terme són, el gairebé nul accés de les llars a la xarxa, que no es va fer popular fins a finals dels 90. La capacitat de les màquines de dur a terme operacions complexes en un temps raonable també era un problema afegit, ja que és necessari una capacitat de procés important a l'hora de dur a terme tots els càlculs criptogràfics i de comunicació per la xarxa.

Avui en dia, la majoria de les llars del nostre país tenen ordenador i alhora disposen d'una connexió, amb més o menys ample de banda a Internet, però suficient pel que els requeriments del sistema demanen. Aquesta característica fa que la implantació del sistema sigui relativament fàcil de fer.

1.3 Objectius del Projecte Final de Carrera.

L'objectiu del projecte final de carrera és la implementació d'un sistema per la realització d'exàmens electrònics segurs que compleixi les propietats següents.

Característiques que el sistema criptogràfic haurà de complir:

- Autenticitat:
    - L'estudiant ha d'estar segur que l'enunciat de l'examen i la seva correcció ha estat realitzada pel professor.

    - El professor ha d'estar segur que la resposta que corregeix ha estat realitzada per un estudiant i que ningú l'ha modificat un cop lliurada.

- Privacitat:
    - El professor no té per que saber quin estudiant és l'autor de la resposta que està corregint. Només ha de tenir la certesa que ha estat redactada per un estudiant matriculat a l'assignatura.

- Correcció:
    - No s'ha de poder afegir una resposta a l'examen un cop el període per lliurar les respostes ha finalitzat.
    - Un cop les respostes s'han lliurat no s'han de poder modificar.
    - En cas que la resposta a un examen sigui eliminada s'ha de detectar aquesta manipulació.

- Secret de les respostes durant l'examen.
    - Durant la realització de l'examen les respostes d'aquest han de ser secretes.

- Rebut de lliurament:
    - L'estudiant ha d'obtenir un rebut que demostri que ha lliurat la resposta de l'examen.

- Impossibilitat de còpia:
    - Els estudiants no han de poder copiar les seves respostes.

Aquestes característiques s'hauran de mantenir durant tot el cicle de vida d'un examen. Aquest cicle de vida es descompon en cinc fases que són les següents:

- Redacció de l'examen
- Resposta de l'examen
- Correcció de l'examen
- Obtenció del resultat de l'examen
- Revisió de l'examen

En els següents capítols d'aquesta memòria es tractarà cadascuna d'aquestes fases més àmpliament.

A més d'aquests objectius inicials, es vol que el sistema executi un aplicatiu de manera remota i que connecti a un servidor central. Existirà un aplicatiu per l'estudiant i un pel professor. És necessari que els aplicatius s'autentiquin en el moment que volen realitzar alguna operació amb el servidor central.

Aquest servidor central tindrà accés a una base de dades que contindrà tots els enunciats i totes les respostes dels exàmens en format XML[10].

Així doncs, als objectius inicials s'hi afegeixen els següents:

- La implementació d'un sistema per representar les dades en format XML[10] i tornar-les en el format que entenguin les parts.

- La comunicació entre l'aplicatiu utilitzat pel professor i el sistema central, i entre l'estudiant i el sistema central es realitzarà utilitzant RMI[5] de Java[4]. La utilització de RMI[5] reduirà en gran mesura el temps de desenvolupament del projecte.

- Una implementació per dur a terme la gestió de la base de dades. Aquesta gestió inclou, l'entrada de les persones a la base de dades que es realitzarà des del sistema gestor d'exàmens. L'entrada d'enunciats per part del professor, i l'entrada de les respostes per part dels estudiants. Finalment totes les consultes relacionades amb aquestes accions.

- Implementació d'un protocol d'autenticació, que permetrà tenir la certesa de qui demana realitzar una operació, i si hi està autoritzat.

S'ha realitzat un esforç important alhora de dissenyar les classes que implementaran els objectius enumerats, de manera que, si en el futur és necessari modificar alguna funcionalitat de l'aplicatiu, ja sigui per afegir funcionalitats o per modificar les característiques actuals, només sigui necessari modificar el mínim nombre de classes possibles. D'aquí la importància de realitzar un disseny i una implementació incremental, com es mostra a en la planificació del projecte.

1.4 Enfocament i mètode seguit.

Aquest sistema necessita la interacció dels següents mòduls:

- L'esquema criptogràfic és la base del projecte. Les classes s'han dissenyat de manera que poguessin ser reutilitzades en qualsevol dels aplicatius.

- La informació dels exàmens es guarda en documents en XML[10] per tal de fer més senzilla la manipulació de les dades per les diferents parts.

- Els usuaris del sistema s'han de comunicar, i per aquest motiu es necessita una plataforma de comunicació que permeti la transmissió dels exàmens en format XML[10] entre el servidor central i les altres parts del sistema, ja siguin estudiants o professors.

- La informació que rep el gestor d'exàmens es guarda en una base de dades.

- Finalment, cada usuari del sistema necessita una interfície gràfica per dur a terme les funcionalitats del sistema.

Així doncs resumint, el sistema compta amb els següents mòduls:

- Esquema Criptogràfic.

- Representació de les dades en XML[10].

- Comunicació del components utilitzant RMI[5].

- Base de dades.

- Protocol d'autenticació.

- Interfície gràfica.

El mètode que s'ha seguit per a implementar el sistema ha estat el d'anar construint cadascun dels mòduls successivament un darrera l'altre efectuant proves unitàries amb cadascun d'ells. Cada mòdul s'integra al següent i es realitzen proves unitàries de nou. Així el que s'ha fet, pas a pas, és:

- Definició del protocol criptogràfic.

- Implementació del protocol d'exàmens en un entorn local, això inclou tot l'esquema criptogràfic.

- Integració del mòdul a XML[10]. En aquest punt s'emmagatzemen totes les dades dels exàmens que s'utilitzen en el protocol criptogràfic en format XML[10] per tal de facilitar-ne la gestió.

- Implementació de la comunicació dels components. En aquest pas s'envien telemàticament els documents XML[10] entre els entorns locals (aplicatiu professor o aplicatiu estudiant) i el servidor central utilitzant la tecnologia RMI[5] de Java[4].

- Implantació de la base de dades. Els documents rebuts via RMI[5] es guarden a la base de dades MySQL[9].

- Implementació del sistema d'autenticació dels estudiants i dels professors contra el servidor central. S'afegeix aquest mòdul a les parts que ja estaven implementades fins al moment.

- Quan tot aquest sistema funciona, es crea una interfície per l'estudiant i una pel professor per a poder utilitzar les diferents funcions d'una manera intuïtiva i còmoda.

És important destacar que la part de la interfície ha estat un dels objectius menys prioritaris del sistema. S'ha primat el correcte funcionament del sistema d'acord amb els objectius fixats. La interfície podria fer més o menys coses, però això ja podria formar part d'un projecte en sí mateix, augmentant la usabilitat de l'entorn gràfic, sobretot per a persones no familiaritzades en informàtica, etc.

Un altre detall que cal esmentar és el següent. Durant la implementació del projecte, s'ha intentat en tot moment utilitzar software de lliure distribució. D'aquesta manera la implementació s'ha realitzat utilitzant codi Java[4] sota l'entorn de desenvolupament Eclipse[7]. La base de dades utilitzada és MySQL[9]. La única part del projecte que no és de lliure distribució és la llibreria criptogràfica IAIK
[15], tot i que per a fins educatius sí que ho és. Si es decidís posar el sistema en producció caldria adquirir una llicència de la llibreria en qüestió o escollir-ne una altra.

1.5 Planificació del Projecte.

El projecte va començar a l'inici del quadrimestre de tardor de l'any 2004. La planificació que es va fer es detalla a continuació.

Del 14 al 19 de setembre:

- Instal·lació i proves de la llibreria criptogràfica IAIK
- [15].

- Creació dels certificats genèrics per a l'Autoritat de Certificació, l'estudiant i el professor. (veure secció 2. PKI)

Del 20 de setembre al 17 de novembre:

- Disseny, implementació, tests i documentació de l'esquema criptogràfic:
  - Redacció de l'examen.
  - Resposta de l'examen.
  - Correcció de l'examen.
  - Obtenció del resultat de l'examen.
  - Revisió de l'examen.

Del 18 al 31 d'octubre:

- Disseny, implementació, tests i documentació de l'esquema XML[10].

De l'1 al 7 de novembre:

- Disseny, implementació, tests i documentació de la base de comunicació en RMI[5].
  - Part del client.
  - Part del servidor.

Del 8 al 21 de novembre:

- Disseny, implementació, tests i documentació de la interacció del protocol amb la base de comunicació.

Del 22 de novembre al 5 de desembre:

- Instal·lació de la base de dades.

- Creació del model.

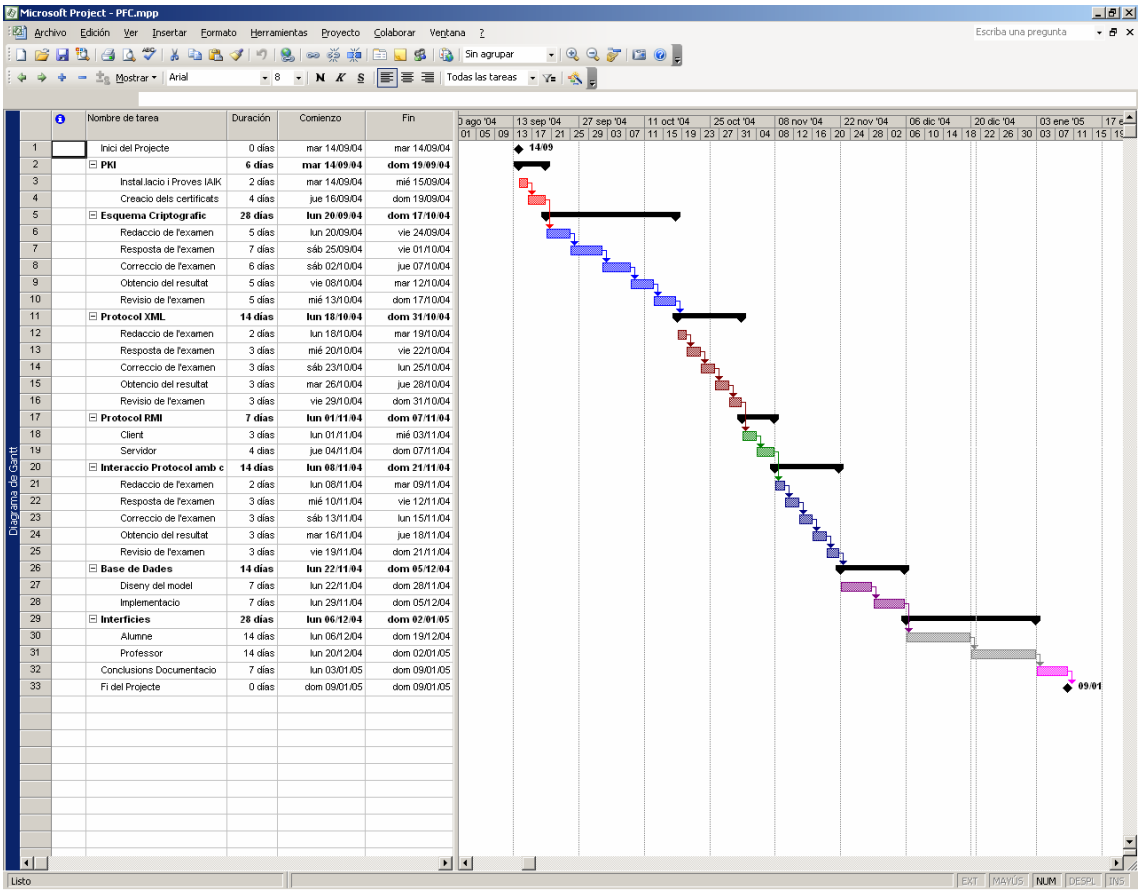
- Proves d'integració amb el sistema actual.

Del 6 de desembre al 19 de desembre:

- Creació de la interfície gràfica.

- Proves d'integració.

A continuació s'inclou un planejament realitzat amb MSProject que mostra de manera gràfica la planificació del Projecte. Aquest document també es presenta adjunt a la memòria.



Imatge 1: Diagrama de la planificació del projecte.

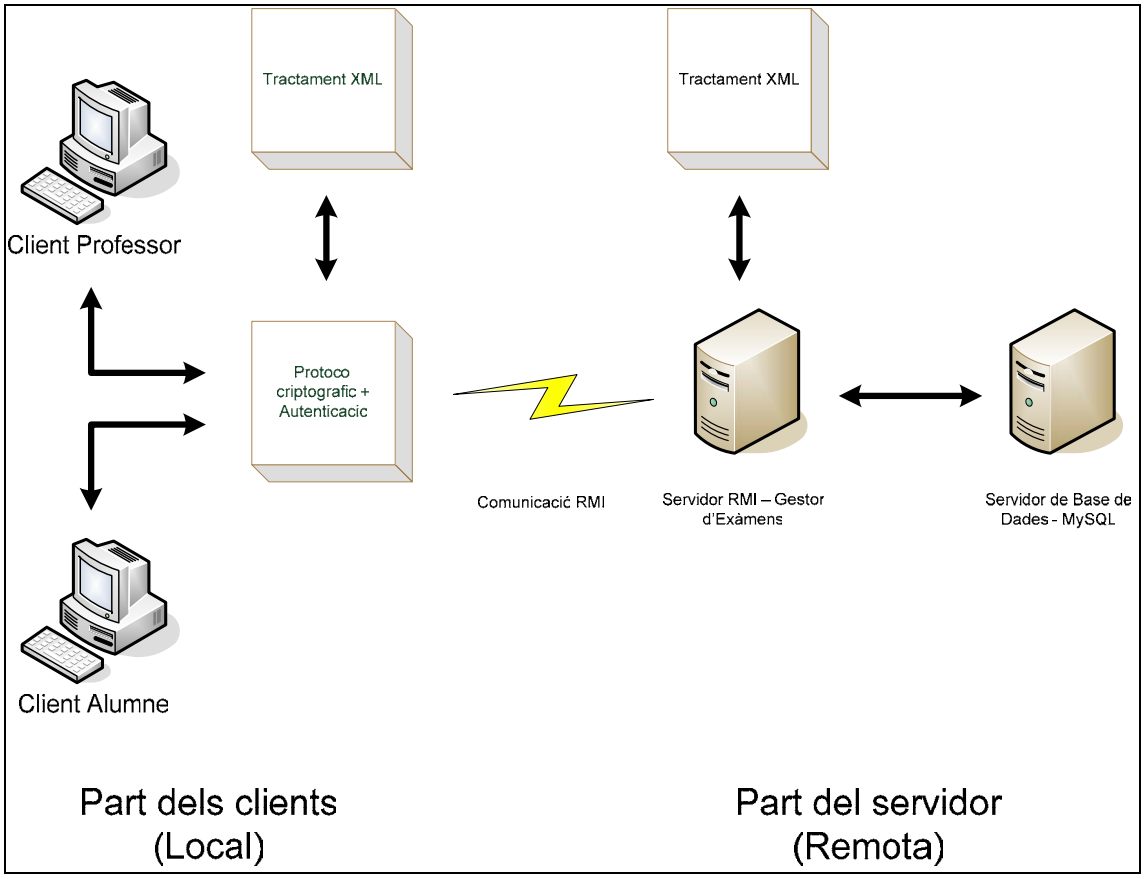
Com es pot veure en la planificació la part que ha rebut més èmfasi i la que ha pres més temps és la de l'estudi i implementació del protocol criptogràfic.

Fins el moment del lliurament del projecte s'han fet revisions del codi per assegurar que sigui òptim i lliure d'errors. Cal destacar que la documentació s'ha anat completant en cada fase donant lloc al document actual.

1.6 Esquema general del projecte.

A continuació s'inclou l'esquema general del funcionament del projecte, aquesta imatge podria servir de resum del que s'ha vingut explicant en aquesta introducció.

És important veure que hi ha una separació clara en els elements del projecte. Es treballa sempre en un entorn de client-servidor. Els aplicatius no tenen accés directe a la base de dades si no és mitjançant el servidor gestor d'exàmens. De la mateixa manera, el protocol criptogràfic l'executen els clients en les seves màquines locals i únicament envien les dades obtingudes via RMI[5]. Seguint aquest model les claus privades no viatgen mai per la xarxa, i són sempre a la màquina del client.



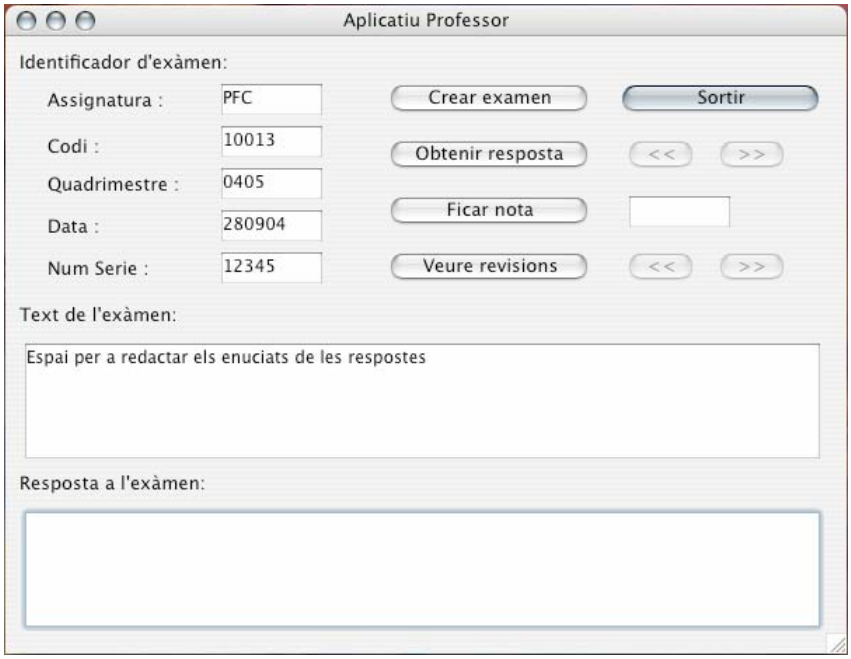
Imatge 2: Esquema general del Projecte.

1.7 Productes obtinguts.

Un cop tot el sistema ha estat implementat s'ha obtingut un producte que està format per diferents components. El sistema complert l'integren tres parts diferenciades:

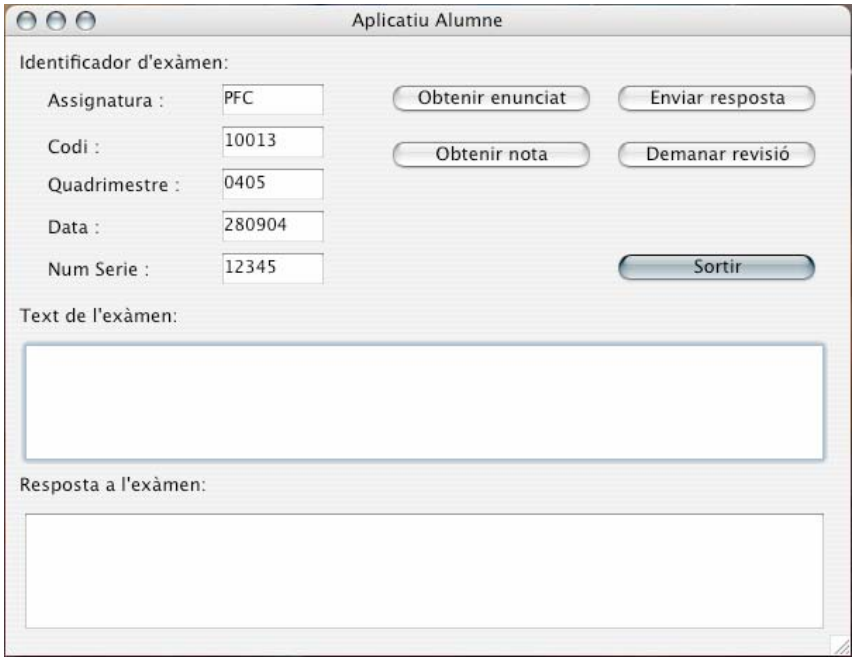
- Sistema gestor d'exàmens. Inclou el servidor RMI[5], que permet que els clients executin codi remot, i també inclou el sistema gestor de bases de dades. A la base de dades només hi té accés el gestor d'exàmens. No hi ha una imatge del servidor en funcionament perquè aquest s'executa com un servei intern de la màquina host del servidor.

- Aplicatiu del professor. El professor utilitza aquest aplicatiu per redactar i corregir els exàmens assignant-los-hi una nota. De la mateixa manera el professor pot veure quins estudiants han demanat una revisió del seu examen.



Imatge 3: Aplicatiu del professor executat en un Mac OSX

- Aplicatiu de l'estudiant. L'estudiant empra aquest aplicatiu per accedir als enunciats dels exàmens i enviar les seves respostes al sistema gestor d'exàmens. De la mateixa manera, un cop els exàmens estan corregits, l'estudiant pot veure la nota que se li ha assignat i demanar una revisió de l'examen si no hi està d'acord.



Imatge 4: Aplicatiu de l'estudiant executat en un Mac OSX

L'explicació detallada d'aquestes interfícies i del seu funcionament concret es pot trobar a la secció 8. Interfície gràfica.

1.8 Descripció dels següents capítols de la memòria.

Els següents capítols de la memòria fan referència a les decisions de disseny i a la implementació dels mòduls necessaris del projecte. L'estructura dels capítols segueix el disseny incremental comentat prèviament.

- PKI
  En el resum del projecte s'ha esmentat que l'esquema es basa en criptografia de clau pública. Per tant, cada usuari del sistema ha de disposar d'una parella de claus. La infrastructura de clau pública necessària per tal d'emetre i gestionar les parelles de claus d'un grup d'usuaris amb els respectius certificats s'anomena PKI. En aquest primer capítol s'explica què és una infrastructura de clau pública, i com s'ha implementat en aquest cas.

- Esquema Criptogràfic.
  A partir del moment en el que el professor redacta l'enunciat de l'examen fins que l'estudiant obté la nota hi ha un seguit de fases que anomenarem cicle de vida. En aquest capítol s'explica les diferents fases de que consta el cicle de vida i el protocol criptogràfic per cadascuna de les respectives fases. El conjunt de protocols és el que s'anomena esquema criptogràfic.

- Representació de les dades: XML[10].
  La representació de les dades de manera que puguin ser gestionades i interpretades fàcilment és un punt important. En el sistema presentat les dades intercanviades estan en documents XML[10]. En cada fase del protocol només es passen entre les parts les dades de l'examen que són necessàries. Per tal de dur a terme aquesta tasca s'utilitza la llibreria JDOM[8].

- Comunicació dels components.
  El programari del professor, l'estudiant i el gestor d'exàmens s'executaran en diferents dispositius. La comunicació d'aquests programaris, o components del sistema, és una part important. Per reduir el temps de desenvolupament es va pensar en la utilització del RMI[5] de Java[4]. Aquest capítol explica com s'ha realitzat la comunicació entre els clients i el servidor utilitzant RMI[5].

- Base de Dades.
  La base de dades és necessària per guardar el contingut dels exàmens i les respostes dels estudiants. S'espera que el volum de dades sigui important, de manera que l'ús d'una base de dades es fa indispensable. El sistema gestor de bases de dades (SGBD) que s'ha utilitzat és MySQL[9], perquè és de lliure distribució i cobreix perfectament les necessitats del projecte.

- Autenticació.
  Un cop el sistema anterior està implementat i funcionant, és necessari identificar i autenticar als usuaris. El sistema gestor d'exàmens necessita assegurar-se que els clients que estan fent les peticions són qui diuen ser i tenen els drets per a poder accedir als serveis que ofereix.

- Interfície Gràfica.
  Aquest capítol explica com s'ha dissenyat la interfície gràfica. Aquest era un dels requeriments menys prioritaris del sistema. Malgrat això, cal destacar que al mateix temps que és senzilla d'utilitzar, compleix tots els requeriments de funcionalitat del sistema.

- Jocs de Proves.
  En aquest capítol s'explica amb un exemple, el funcionament de tot el procés; des de el moment de crear els certificats, passant per tot el cicle de vida d'un examen. També es comenta la configuració de la màquina virtual Java[4], RMI[5], la base de dades, etc.

- Diagrames.
  Aquest capítol, amb els diagrames de seqüència, serveix com a resum dels principals passos pels que passen els algorismes per aplicar l'esquema criptogràfic. Inclou també el diagrama de classes complert.

- Treball futur.
  Com en tot sistema complex sempre hi ha coses a millorar, o noves funcionalitats a afegir. En aquest capítol s'expliquen les millores que no s'han acabat implementant perquè no estaven a la planificació inicial, però que, al llarg del projecte, es va veure que serien interessants en el futur per tal d'ampliar les funcionalitats i característiques del sistema.

- Conclusions.
  En aquest capítol s'exposa l'acompliment dels objectius, i com s'ha aconseguit. També es destaquen les consideracions més importants experimentades durant la implementació.

- Bibliografia i Annexos.
  En la bibliografia hi ha les referències a tots els documents consultats, a més a més de les adreces on es poden aconseguir els programes i llibreries necessàries per posar en explotació el sistema. En els annexos s'inclouen arxius de configuració, el glossari, etc.

2. PKI

2.1 Introducció

L'esquema criptogràfic implementat en aquest projecte necessita que les diferents parts (professors, estudiants, gestor d'exàmens) disposin d'una parella de claus i el seu corresponent certificat.

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

Una PKI consta d'una autoritat de certificació notada amb CA, aquestes sigles corresponen al terme en anglès Certification Authority. Un altre component de la PKI són les autoritats de registre, notades amb les sigles RA (Registry Authority). Quan un usuari vol obtenir un certificat normalment realitza els passos següents. En un primer pas crea una parella de claus i realitza una petició de certificat mitjançant una RA. La RA valida la identitat de l'usuari que ha demanat el certificat i envia la petició a la CA. La CA rep les peticions de les RA i emet els certificats. La clau privada de la CA és una peça d'informació molt sensible, i per això està en un entorn amb un alt nivell de seguretat.

Si un usuari veu que la clau privada corresponent al seu certificat ha estat compromesa ha de comunicar-ho a la CA. La CA revoca aquell certificat incloent-lo en una llista de certificats revocats. La llista de certificats revocats la notem amb les sigles CRL del terme anglès Certificate Revocation List.

La verificació de la validesa d'un certificat és un pas important. No es pot acceptar un certificat com a vàlid sense realitzar els passos següents.

Una primera comprovació és saber quina CA ha emès el certificat. Si la CA és de la nostra confiança, per exemple perquè és una entitat de reconegut prestigi, seguim amb el procés de verificació. El segon pas és verificar que el certificat no ha estat revocat. Per fer aquesta comprovació podem preguntar-li directament a la CA utilitzant un protocol dissenyat per fer aquesta consulta com és el OCSP[23], o descarregar-nos la CRL de la CA i buscar-hi si hi ha el certificat.

Per aprofundir en el tema es recomana veure les següents referències [1]

En el projecte s'ha utilitzat la llibreria de lliure distribució Openssl[12] per tal de construir de forma ràpida i senzilla una petita PKI. Openssl està disponible amb totes les versions de Unix, i existeix una versió compilada per a plataformes Windows[13]. En la implementació s'ha utilitzat la versió de Unix (versió 0.9.7b 10-Apr-2003) i els fitxers de comandes que es mostren a continuació. Aquest són executables només en un sistema Linux. Malgrat això es poden adaptar fàcilment per tal de ser utilitzats en una plataforma Windows.

Els certificats emesos segueixen l'estàndard X.509[22]. La clau privada i el corresponent certificat s'han emmagatzemat un fitxer seguint el format estàndard PKCS12[20].

2.2 Passos a seguir per generar els arxius necessaris.

L'esquema implementat necessita que cadascun dels usuaris, inclòs el gestor d'exàmens tingui una parella de claus amb un fitxer en format PKCS12[20]. Aquest arxiu, en la seva essència, és un contenidor amb els elements següents:

- Parella de claus, pública i privada, de l'usuari.
- Certificat de l'usuari emès per una autoritat de certificació (CA).
- Certificat de l'autoritat de certificació.

El primer pas és l'obtenció del certificat de l'autoritat de certificació. Si aquest sistema es decidís d'implantar en organismes on ja es disposés d'una infrastructura de clau pública amb una autoritat de certificació pròpia no suposaria cap canvi substancial. En el cas del projecte no es disposa d'aquesta autoritat i per això el primer pas és crear-ne una.

Els passos per generar el certificat de la CA són:

- Generar la parella de claus de la CA amb una longitud de clau de 2048 bits. Aquesta acció es realitza utilitzant el fitxer de comandes "generarClaus". El fitxer de sortida que conté la parella de claus ha estat anomenat CA.key.
- Generar un certificat autosignat amb la parella de claus de la CA. Aquest és el certificat de la CA. Per emetre aquest certificat s'ha emprat el fitxer de comandes "generaCertificatAutosignat" i la parella de claus del primer punt. L'arxiu del certificat s'ha anomenat CA.crt .

Ara ja es pot passar a crear els arxius dels usuaris del sistema. El primer que crearem serà el de l'estudiant tipus. S'han de seguir els passos que es descriuen a continuació:

- Generar una parella de claus per l'estudiant. Per fer-ho es torna a utilitzar "generarClaus". La longitud de les claus en aquest cas és de 1024 bits.

- Emetre una petició de certificat contra la CA que ja tenim disponible. S'empra el fitxer de comandes "generaPeticioCertificat".

- La CA emet el certificat. Per fer-ho s'empra el fitxer de comandes "generaCertificat". S'ha d'utilitzar el fitxer de configuració "openssl.cnf". Aquest fitxer es pot trobar a l'annex del projecte.

- Generar l'arxiu PKCS12[20]. Aquest arxiu conté la parella de claus de l'estudiant, el seu certificat, i el certificat de la CA. Per construir-lo s'utilitza el fitxer de comandes "generaPKCS12".

El mateix s'ha de realitzar pel professor i pel gestor d'exàmens. Els passos són sempre els mateixos i d'aquesta manera es poden crear tants usuaris com faci falta. Ara mateix, com que només seran necessaris 3 usuaris, la feina no és exagerada, però en el cas de tenir que implantar aquest procediment en un entorn de producció real, podria ser una bona idea el tenir un aplicatiu integrat amb el servidor d'usuaris per a crear els PKCS12[20] de manera més eficient i còmoda.

Al final d'aquest procés s'han de tenir els següents arxius:

- CA.crt
- alumne.p12
- professor.p12
- gestor.p12

Els tres últims són els que s'utilitzaran a l'aplicatiu. Adjunts a la memòria s'inclouen els arxius que es van utilitzar durant el desenvolupament.

Com a detall a implementar en un futur, el fitxer CA.crt, s'hauria d'afegir al codi del protocol criptogràfic per tal que les diferents parts comprovessin que tots els certificats són de confiança.

2.3 Fitxers de comandes de generació de certificats.

A continuació s'inclouen els codis dels fitxers de comandes esmentats en el punt anterior:

Fitxer de comandes GenerarClaus:

```
#!/bin/bash

if [ $# -le 1 ]; then
       echo "Usage: "$0" <key_file> <key_length>"
       echo "or"
       echo "Usage: "$0" <key_file> <key_length> <random_file_length>"
       exit 1
fi
```

```
if [ $# -eq 2 ]; then
       openssl genrsa -des3  -out $1 $2
       exit 0
fi

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

exit 0
```

Fitxer de comandes GeneraCertificatAutosignat:

```
#!/bin/bash

if [ $# -le 2 ]; then
       echo "Usage: "$0" <key_file> <file.crt> <dies>"
       exit 1
fi

openssl req -new -sha1 -x509  -key $1 -out $2 -days $3

exit 0
```

Fitxer de comandes GeneraPeticioCertificat:

```
#!/bin/bash

if [ $# -le 2 ]; then
       echo "Usage: "$0" <key_file> <file.csr> <config_file>"
       exit 1
fi

openssl req -new -sha1 -config $3 -key $1 -out $2

exit 0
```

Fitxer de comandes GeneraCertificat:

```
#!/bin/bash

if [ $# -le 2 ]; then
       echo "Usage: "$0" <file.csr> <file.crt> <config_file>"
       echo "or"
       echo     "Usage:     "$0"     <file.csr>     <file.crt>     <config_file>
<extensions_section>"
       exit 1
fi

if [ $# -le 3 ]; then
       openssl ca -config $3 -out $2 -infiles $1
       exit 0
fi

openssl ca -config $3 -out $2 -extensions $4 -infiles $1

exit 0
```

Fitxer de comandes GeneraPKCS12:

```
#!/bin/bash

if [ $# -le 3 ]; then
       echo "Usage: "$0" <key_file> <file.crt> <CA_certificate> <file.p12>"
       exit 1
fi

openssl pkcs12 -export -in $2 -inkey $1  -certfile $3 -out $4

exit 0
```

Per a fer servir aquests fitxer de comandes és necessari un fitxer de configuració anomenat openssl.conf. El contingut del mateix és força extens i per aquest motiu s'adjunta a l'annex de la memòria.

Un cop la part de la PKI està completada es passa a descriure l'esquema criptogràfic implementat en el projecte.

3. Esquema criptogràfic

3.1 Introducció

Típicament des del moment que el professor redacta l'enunciat de l'examen fins que l'estudiant obté la nota hi ha un seguit de fases. El conjunt d'aquestes fases pot anomenar-se cicle de vida de l'examen. En base a aquest cicle de vida s'ha dissenyat un protocol criptogràfic per cada fase. El conjunt d'aquests protocols rep el nom d'esquema criptogràfic.

En aquest capítol es descriu que és el cicle de vida d'un examen, i cada protocol que hi està associat. Es fa èmfasi en el disseny, el funcionament, i en les decisions de disseny preses durant la implementació.

Per una bona comprensió d'aquest capítol es recomana disposar d'uns coneixements previs de criptografia, o utilitzar [1] com a referència.

3.2 Cicle de vida d'un examen.

Tal i com ja hem apuntat en la introducció d'aquesta memòria s'empra el terme cicle de vida d'un examen per agrupar les diferents etapes per les quals passa un examen, des del moment que es redacta fins que acaba el període de revisió. A continuació es defineixen breument aquestes etapes.

- Redacció de l'examen: la primera fase és la creació del propi examen, és a dir, el professor redacta l'examen via teclat i li assigna un identificador. Aquest es guardarà a la base de dades fins el moment de realitzar-lo.

- Resposta de l'examen: la segona fase és la resposta de l'examen per part de l'estudiant. La durada d'aquesta fase dependrà del temps que hagi establert el professor.

- Correcció de l'examen: un cop el període de l'examen ha finalitzat el professor ja pot corregir l'examen. En aquesta fase, el professor llegeix totes les respostes i les hi assigna una nota.

- Obtenció del resultat de l'examen: un cop el professor ha corregit l'examen l'estudiant ja pot consultar la nota del seu examen.

- Revisió de l'examen: si l'estudiant no està d'acord amb la seva nota pot demanar una revisió de l'examen. En aquest cas, el professor tindrà accés a veure la llista de revisions sol·licitades i podrà modificar-ne les notes.

Cadascuna d'aquestes etapes correspon a una part del protocol de l'esquema criptogràfic que s'ha comentat anteriorment. Donada la complexitat d'evitar la còpia en exàmens electrònics remots, aquest projecte es centra en els exàmens electrònics realitzats en un entorn controlat.

3.3 Notació emprada en el protocol.

En la descripció dels protocols es fa necessària la utilització d'una certa notació criptogràfica, que es mostra a continuació.

- K: Clau d'un sistema criptogràfic.

- $E_K(M)$: Xifratge simètric d'un missatge ***M*** amb la clau ***K***.

- $D_K(M)$: Desxifratge simètric d'un missatge ***M*** amb la clau ***K***.

- $(P_{Entitat}, S_{Entitat})$: Parella de claus asimètriques propietat d'Entitat, on ***P*** correspon a la clau pública i ***S*** a la clau privada.

- $S_{Entitat}(M)$: Signatura d'un missatge ***M*** amb la clau privada ***S*** d'Entitat.

- $P_{Entitat}(M)$: Xifratge del missatge ***M*** amb clau asimètrica pública ***P*** d'Entitat.

- H(M): sortida d'una funció de resum criptogràfica del missatge ***M***, aquestes funcions reben el nom de funcions de hash.

3.4 Actors del Sistema.

El sistema comptarà amb tres actors, tots tres essent persones físiques:

- Professor: encarregat de la redacció i correcció dels exàmens.

- Estudiant: encarregat de respondre els exàmens, consultar la nota i demanar revisions.

- Gestor: encarregat de verificar que el procés es desenvolupa correctament i d'afegir/eliminar els usuaris a la base de dades.

3.5 Redacció de l'examen.

L'aplicatiu del professor realitza els passos següents alhora de dur a terme la redacció i posterior lliurament de l'enunciat d'un examen al sistema gestor d'exàmens.

Professor:

- Crear un identificador únic per l'examen, ***Id***, format per les següents dades:
  - ***As***: Assignatura.
  - ***Cd***: Codi Assignatura.
  - ***Qu***: Quadrimestre.
  - ***Da***: Data.
  - ***Ns***: Número de sèrie.

- Redactar l'enunciat ***E*** de l'examen via teclat.

- Signar amb la clau privada del professor S_p el identificador de l'examen ***Id*** i l'enunciat ***E***. Obtenim ***Es***.

- Autenticar el professor davant del Gestor utilitzant la seva parella de claus.

- Lliurar les següents dades al Gestor: (***Id, E, Es***)

Gestor d'exàmens:

- Autenticar el professor

- Verificar la signatura digital ***Es***.

- Guardar les dades enviades pel professor.

Decisions preses per la implementació:

- L'identificador ***Id*** és una cadena amb la concatenació de les dades. Un exemple vàlid d'identificador d'examen podria ser: "PFC10013040528090412345"
  - ***As***: "PFC"
  - ***Cd***: "10013"
  - ***Qu***: "0405"
  - ***Da***: "280904"
  - ***Ns***: "12345"

- Quan es signen dades, com ara (***Id , E***), aquestes dades són concatenades.

- La tria de l'identificador es pot fer de manera aleatòria sense restriccions. En entorns de producció reals es podria afegir un sistema de control sobre els identificadors de manera que es creessin automàticament.

Les úniques accions que el professor realitza són la tria de l'identificador i la redacció del text de l'examen. La resta es realitza de manera transparent a ell.

Classes necessàries per a la redacció d'un examen.

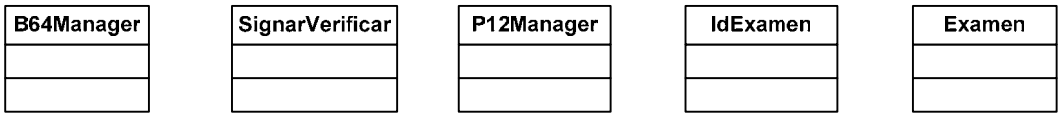
Per a realitzar aquests passos s'ha proposat tenir una classe per signar exàmens i per comprovar la signatura, aquesta classe s'anomenarà SignarVerificar. Aquesta classe té accés a una altra classe que s'encarrega de gestionar els arxius PKCS12[20] per tal d'obtenir les claus privades i públiques. Es recorda que la obtenció d'aquests arxius ha estat explicada a la secció 2. PKI. Encara no s'inclou l'autenticació del professor que es farà més endavant, concretament s'explica a la secció 7. Protocol d'autenticació.

De la mateixa manera fa falta una classe per guardar els exàmens a la base de dades, però això es detalla al capítol corresponent en el moment de la implantació de la base de dades. En aquest punt encara no existeix l'accés a la base de dades, per tant es guarden les dades en una classe buida anomenada Examen.

També es disposa d'una classe anomenada IdExamen. Aquesta classe té l'estructura de l'identificador de l'examen, és a dir, disposa d'un camp per a cadascun dels camps de l'identificador.

S'ha de tenir en compte un altre detall. Les dades que es generen després de realitzar la signatura d'un text no es poden transportar ni tractar com si fossin una cadena de caràcters per problemes de codificació. El que s'ha de fer si es vol emmagatzemar com a cadena és passar les dades a base64. Com que un dels objectius és guardar les dades en format cadena a la base de dades, fa falta una classe que s'encarregui de dur a terme les conversions a base64 i a binari de nou. Aquesta classe s'anomena B64Manager.

Es mostra a continuació una definició simple en UML [14] de les classes esmentades fins el moment:



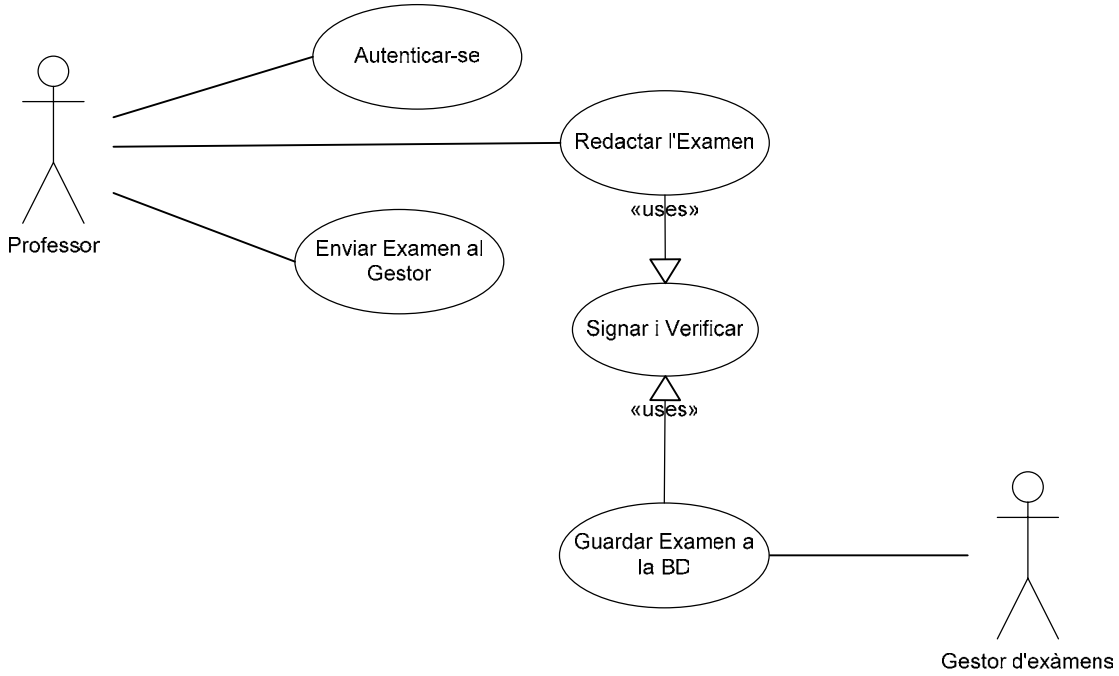
Durant l'evolució del projecte es podrà veure com aquestes classes formen part de l'esquelet del sistema, proveint les noves necessitats d'altres classes com en el cas de la base de dades o la comunicació per RMI[5], per exemple.

Com es pot comprovar en la definició dels casos d'ús que es mostren a continuació, es fa sovint referència a guardar les dades a una base de dades. Donat que en aquest punt del desenvolupament el sistema encara no disposa d'aquest servei, s'entendrà que aquestes dades s'emmagatzemen de manera lògica en la classe Examen.

Si el lector no està familiaritzat amb els diagrames UML [14] es recomana veure [2] com a referència.

Diagrama de casos d'ús d'aquesta fase:

Redacció de l'examen



Descripció dels casos d'ús d'aquesta fase:

Autenticar-se

Cas d'ús: Autenticar-se.
Actors: Professor.
Propòsit: Reconèixer al professor com a usuari vàlid al sistema.
Tipus: Secundari i essencial.

Curs típic dels esdeveniments:

Accions dels actors	***Accions del Sistema***
1. El cas d'ús comença quan l'actor vol realitzar una acció al sistema i se li demana autenticar-se.	
2. L'actor prepara les dades que enviarà al servidor i les envia.	
	3. El servidor rep les dades, les processa i autentica el client.
	4. Envia confirmació i dades auxiliars al client.
5. El client rep la confirmació i les dades auxiliars.	

Cursos alternatius:

Punt 3: El servidor rep les dades que són incorrectes i no autentica el client.

Redactar l'Examen

Cas d'ús: Redactar l'examen.
Actors: Professor.
Propòsit: Introduir un nou examen via teclat.
Tipus: Primari i essencial.

Curs típic dels esdeveniments:

Accions dels actors	***Accions del Sistema***
1. El professor entra les dades d'un Identificador i d'un enunciat via el teclat. 2. Signa amb la clau privada del professor S_p l' identificador de l'examen ***Id*** i l'enunciat ***E***.	

Enviar Examen al Gestor

Cas d'ús: Enviar examen al gestor.
Actors: Professor.
Propòsit: Les dades de l'examen són enviades al servidor per ser verificades i guardades.
Tipus: Primari i essencial.

Curs típic dels esdeveniments:

Accions dels actors	***Accions del Sistema***
1. El cas d'ús comença quan l'actor envia les dades al servidor.	
	2. El servidor rep les dades.
3. L'actor rep confirmació de l'enviament.	

Signar i Verificar

Cas d'ús: Signar i Verificar
Actors: Professor i gestor
Propòsit: Signar les parts dels exàmens i verificar les dades signades..
Tipus: Primari i essencial

Curs típic dels esdeveniments:

Accions dels actors	***Accions del Sistema***
1. L'actor demana signar o verificar dades	
	2.El Sistema retorna les dades signades o verificades.

Cursos alternatius:

Punt 2: El Sistema retorna un error al verificar una signatura.

Guardar un Examen a la base de dades.

Cas d'ús: Guardar un Examen a la base de dades.
Actors: Gestor
Propòsit: Guardar a la base de dades el nou examen redactat.
Tipus: Primari i essencial.

Curs típic dels esdeveniments:

Accions dels actors	***Accions del Sistema***
1. El gestor ha verificat les dades de l'examen i envia a la base de dades el nou examen.	
	2. El servidor rep les dades, les processa i guarda les dades a la base de dades.

Cursos alternatius:

Punt 2: Si l'examen ja existeix a la base de dades es retorna un error.

3.6 Resposta a un examen.

Els actors que intervenen en aquest procés són l'estudiant i el gestor d'exàmens. En aquest protocol l'aplicatiu realitza els passos següents per tal de dur a terme la resposta i posterior lliurament d'un examen respost.

Estudiant:

- Autenticar l'estudiant contra el gestor amb la seva parella de claus.
- Obtenir l'examen signat que s'ha generat en el punt anterior (***Id, E, Es***).
- Verificar la signatura del professor, ***Es***.
- Entrar la resposta ***R*** a l'examen pel teclat.
- Obtenir un identificador de resposta ***Ie.***
- Signar ***Ie***, la Resposta ***R***, i la signatura de l'examen ***Es*** amb la clau privada de l'estudiant. El resultat és ***Rs***.
- Enviar de manera segura al gestor: (***Ie, R i Rs***).

Gestor d'exàmens:

- Comprovar la signatura de ***Rs***.
- Crear la capçalera de lliurament ***T*** que inclou ***Id***, ***Ie*** i ***Tm***. ***Tm*** és l'instant actual en el que el gestor genera la capçalera.
- Generar un rebut de lliurament ***Ts*** a partir de les següents dades: ***Es***, ***R*** i ***T***. ***Ts*** és la signatura d'aquestes tres dades.

- Enviar a l'aplicatiu de l'estudiant ***T*** i ***Ts*** com a prova de que ha realitzat l'examen.

- Guardar la resposta ***R*** en un sobre digital o PKCS7 que anomenem ***X.***

- Signar ***X***, ***R*** i ***Es***, obtenint així ***Xs***.

- El gestor guarda de forma segura: (***Id***, ***E***, ***Es***, ***R***, ***T***, ***Ts***, ***X*** i ***Xs***).

Estudiant:

- Comprovar les dades enviades pel gestor (***T*** i ***Ts***).

- Guardar ***T*** perquè és la prova de que ha realitzat l'examen.

Decisions preses per la implementació:

- De la mateixa manera que s'ha fet abans, totes les signatures es generen a partir de la concatenació dels camps que inclouen.

- ***Ie*** s'obté a partir d'una instància de la classe SecureRandom() i ***Tm*** a partir de la instància d'una classe Date().

Classes necessàries per a la resposta d'un examen.

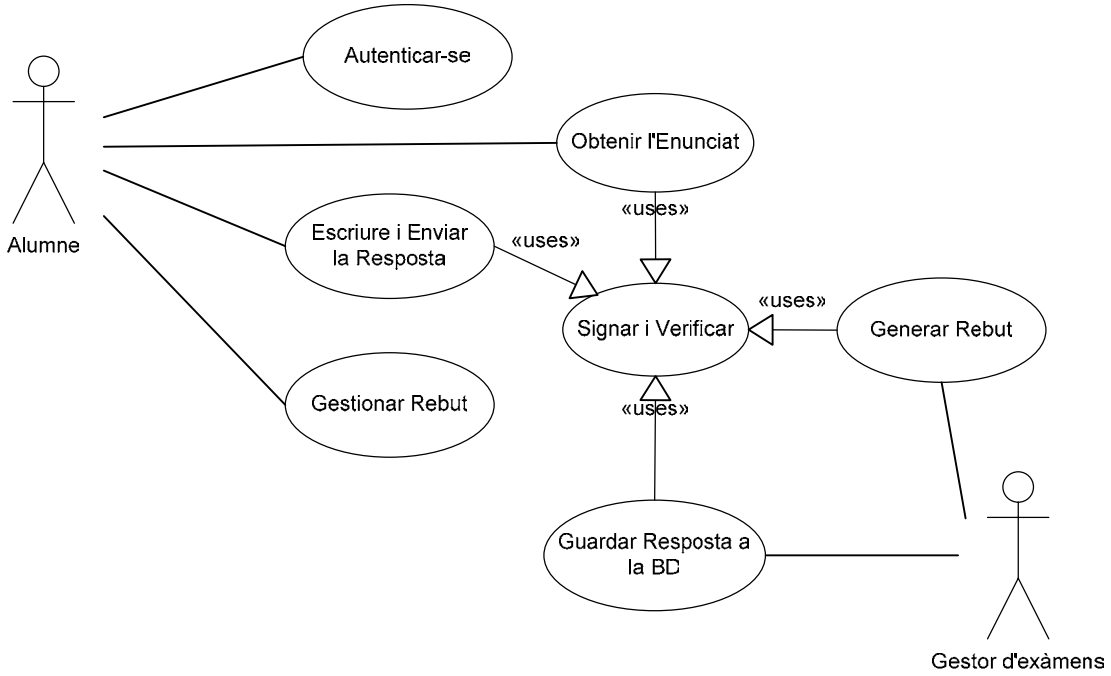
En aquest punt del protocol s'afegeix una nova funcionalitat que fins ara no havia estat necessària. El gestor d'exàmens xifra la resposta i la guarda en format PKCS7[21], és per això que s'afegeix una nova classe que s'anomena Xifrador. El format PKCS7[21] és un contenidor de dades criptogràfiques, com pot ser una signatura digital, una CRL, etc.

Xifrador

Per fer el joc de proves s'ha implementat una classe buida per a guardar els exàmens ("Examen"). Aquesta classe s'anirà modificant fins al moment que s'implementi la part corresponent a XML[10].

Diagrama de casos d'ús d'aquest protocol:

Resposta de l'examen



Descripció dels casos d'ús d'aquesta fase:

Els casos d'ús que no es comenten és degut a que ja han estat comentats en punts anteriors i el funcionament és el mateix, només canvia l'actor.

Obtenir l'Enunciat

Cas d'ús: Obtenir l'enunciat.
Actors: Estudiant.
Propòsit: Demana al sistema que li retorni per pantalla l'enunciat a respondre.
Tipus: Primari i essencial.

Curs típic dels esdeveniments:

Accions dels actors	***Accions del Sistema***
1. El cas d'ús comença quan l'actor vol obtenir un enunciat definit per un identificador.	
2. L'actor envia l'identificador al servidor	
	3. El servidor rep les dades, les processa i accedeix a la base de dades buscant l'enunciat.
	4. Envia l'enunciat a l'actor.
5. L'actor rep l'enunciat per pantalla.	

Cursos alternatius:

Punt 3: El servidor rep l'identificador d'un enunciat que no existeix a la base de dades.

Escriure i Enviar la resposta

Cas d'ús: Escriure i enviar la resposta.
Actors: Estudiant.
Propòsit: Entra pel teclat la resposta a l'examen i l'envia al servidor.
Tipus: Primari i essencial.

Curs típic dels esdeveniments:

Accions dels actors	***Accions del Sistema***
1. El cas d'ús comença quan l'actor entra la resposta a un enunciat.	
2. L'estudiant escriu la resposta.	
3. L'estudiant demana totes les signatures necessàries per enviar la resposta.	
	4. El sistema retorna les signatures
5. L'estudiant envia la resposta i les dades auxiliars al servidor	

Generar Rebut

Cas d'ús: Generar Rebut.
Actors: Gestor.
Propòsit: El gestor crear el rebut de la resposta i l'envia a l'estudiant.
Tipus: Primari i essencial.

Curs típic dels esdeveniments:

Accions dels actors	***Accions del Sistema***
1. El cas d'ús comença quan l'actor ha rebut una resposta i vol enviar un rebut a l'origen.	
2. El gestor demana un rebut.	
	3. Es retorna el rebut creat.
4. El gestor ja pot enviar el rebut a l'estudiant.	

Gestionar Rebut

Cas d'ús: Gestionar Rebut.
Actors: Estudiant.
Propòsit: L'estudiant guarda el rebut de manera segura.
Tipus: Primari i essencial.

Curs típic dels esdeveniments:

Accions dels actors	***Accions del Sistema***
1. El cas d'ús comença quan l'actor rep un rebut a la resposta que ha enviat.	
2. L'estudiant guarda de manera segura el rebut de la resposta.	

Guardar una Resposta a la base de dades.

Cas d'ús: Guardar una resposta a la base de dades.
Actors: Gestor.
Propòsit: Guardar a la base de dades la resposta rebuda.
Tipus: Primari i essencial.

Curs típic dels esdeveniments:

Accions dels actors	***Accions del Sistema***
1. El gestor demana la verificació de les dades de la resposta.	
	2. El sistema comprova les signatures.
3. El gestor envia les dades a la base de dades.	
	4. El servidor rep les dades, les processa i guarda les dades a la base de dades.

Cursos alternatius:

Punt 2: Si la signatura és incorrecta es retorna un error.

Punt 4: Si la Resposta ja existeix a la base de dades es retorna un error.

3.7 Correcció d'un examen.

Aquesta és la part del protocol corresponent a la correcció i la posterior assignació d'una nota. Els actors que intervenen en aquest procés són el professor i el gestor d'exàmens.

Professor:

- Autenticar al professor contra el gestor amb la seva parella de claus.
- Obtenir les dades de l'examen juntament amb la resposta (***Id, E, Es, R, X, Xs***)
- Verificar les signatures de ***Es*** i ***Xs***.
- Corregir la resposta ***R*** i ficar una nota ***N***.
- Signar la resposta de l'examen amb la nota ***N***: $\boldsymbol{Ns = S_p [Es, Ts, N].}$
- Enviar ***N*** i ***Ns*** al Gestor.

Gestor:

- Verificar la signatura de ***Ns.***
- Desxifrar ***X*** per obtenir ***Rs***.
- Emmagatzemar de forma segura: (***Id, E, Es, R, Rs, T, Ts, N, Ns***).

Decisions preses per la implementació:

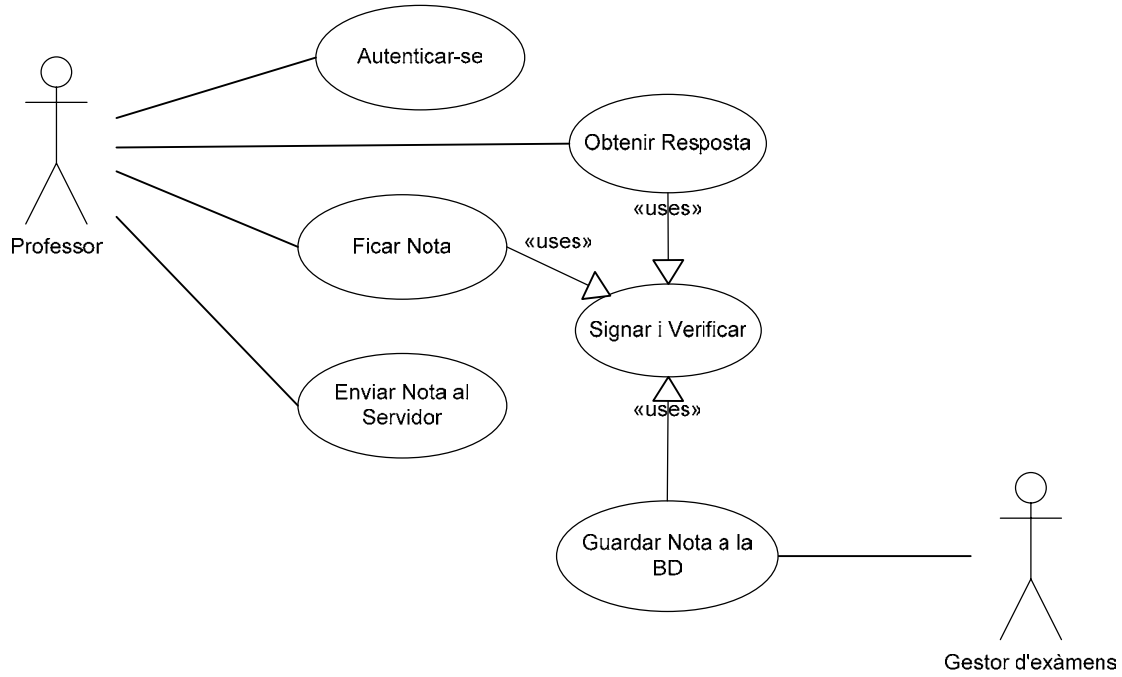
- Les signatures es generen a partir de la concatenació dels camps que inclouen.
- La nota de l'examen s'emmagatzemarà temporalment en un atribut en la classe Examen.

Classes necessàries per a la correcció d'un examen.

El protocol específic d'aquesta fase no introdueix cap classe nova.

Diagrama de casos d'ús d'aquest protocol:

Correcció de l'examen



Descripció dels casos d'ús d'aquesta fase:

Guardar una Nota a la base de dades.

Cas d'ús: Guardar una Nota a la base de dades.
Actors: Gestor.
Propòsit: Guardar a la base de dades la nota assignada.
Tipus: Primari i essencial.

Curs típic dels esdeveniments:

Accions dels actors	***Accions del Sistema***
1. El gestor ha verificat les dades de la nota i l'envia a la base de dades.	
	2. El servidor rep les dades, les processa i les guarda a la base de dades.

Ficar Nota

Cas d'ús: Ficar Nota.
Actors: Professor.
Propòsit: El professor assigna una nota a una resposta.
Tipus: Primari i essencial.

Curs típic dels esdeveniments:

Accions dels actors	***Accions del Sistema***
1. El professor ha llegit una resposta obtinguda.	
2. El professor assigna una nota i signa les dades necessàries per a enviar-les al servidor.	
	3. El sistema retorna les dades signades.
4. El professor té les dades preparades per enviar-les al servidor.	

Obtenir la Resposta

Cas d'ús: Obtenir la Resposta.
Actors: Professor.
Propòsit: Demana al sistema que li retorni per pantalla la resposta a un enunciat.
Tipus: Primari i essencial.

Curs típic dels esdeveniments:

Accions dels actors	***Accions del Sistema***
1. El cas d'ús comença quan l'actor vol obtenir una resposta a un enunciat definit per un identificador.	
2. El professor envia l'identificador al servidor	
	3. El servidor rep les dades, les processa i accedeix a la base de dades buscant la resposta.
	4. Envia la resposta al professor.
5. El professor rep la resposta per pantalla.	

Cursos alternatius:

Punt 3: El servidor rep l'identificador d'un enunciat que no conté respostes a la base de dades. En aquest cas es retorna un error.

Enviar Nota al servidor

Cas d'ús: Enviar Nota al servidor.
Actors: Professor.
Propòsit: El professor envia la correcció de la resposta al servidor.
Tipus: Primari i essencial.

Curs típic dels esdeveniments:

Accions dels actors	***Accions del Sistema***
1. El professor ha assignat una nota i ha generat les signatures pertinents. 2. El professor enviar les dades al servidor.	
	3, El servidor rep les dades i envia una confirmació.
4. El professor rep la confirmació de l'enviament.	

3.8 Obtenció de la nota d'un examen.

Un estudiant segueix el protocol següent per tal d'obtenir la seva nota d'un examen que ja ha estat corregit. Els actors que intervenen en aquest procés són l'estudiant i el gestor d'exàmens.

Estudiant:

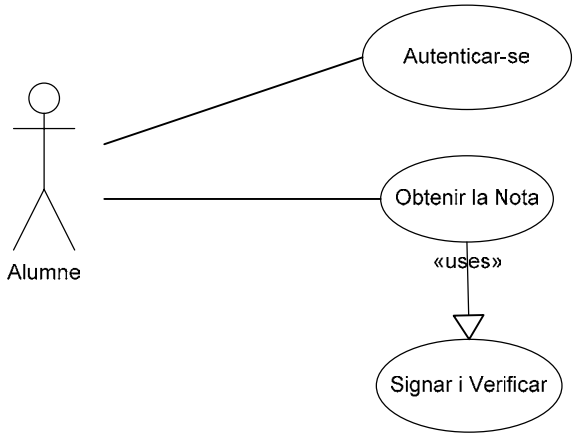
- Autenticar l'estudiant contra el gestor amb la seva parella de claus.
- L'estudiant demana la nota ***N*** per a un identificador ***Id***.
- Verificar la signatura de ***Ns***.
- El Gestor envia ***N*** al aplicatiu de l'estudiant.

Classes Necessàries per a l'obtenció de la Nota d'un Examen

Aquesta és la part de l'esquema més senzilla i no implica cap modificació a l'estructura de classes.

Diagrama de casos d'ús d'aquest protocol:

Obtenció de la Nota



Descripció dels casos d'ús d'aquesta fase:

Obtenir la Nota

Cas d'ús: Obtenir la Nota.
Actors: Estudiant.
Propòsit: Demana al sistema que li retorni per pantalla la nota a un identificador.
Tipus: Primari i essencial.

Curs típic dels esdeveniments:

Accions dels actors	***Accions del Sistema***
1. El cas d'ús comença quan l'actor estudiant vol obtenir una nota a un enunciat definit per un identificador.	
2. L'estudiant envia l'identificador al servidor	
	3. El servidor rep les dades, les processa i accedeix a la base de dades buscant la Nota.
	4. Envia la nota a l'estudiant.
5. L'estudiant verifica les dades rebudes.	
6. L'estudiant veu la nota per pantalla.	

Cursos alternatius:

Punt 3: El servidor rep l'identificador d'un enunciat que no té cap nota assignada a la base de dades. En aquest cas retorna un error.

Punt 5: La verificació de la signatura és incorrecta, es retorna un error.

3.9 Revisió d'un examen.

Aquesta és la part del protocol que cal seguir per tal de fer una petició de revisió d'un examen corregit. Els actors que intervenen en aquest procés són l'estudiant, el professor i el gestor d'exàmens.

Estudiant:

- Autenticar l'estudiant contra el gestor amb la seva parella de claus.
- Obtenir un identificador aleatori ***Ir*** per la revisió.
- Signar ***Ir*** conjuntament amb el rebut de l'examen que l'estudiant posseeix: ***Rv = S_a[Ir, Ts].***
- Enviar ***Ir*** i ***Rv*** al gestor d'exàmens.

Gestor:

- El Gestor verifica la signatura de ***Rv***.
- Emmagatzema la informació (***Ir, Rv***).

Professor:

- Autenticar al professor contra el gestor amb la seva parella de claus.
- Obtenir totes les peticions de revisió per a un ***Id*** que el professor ha indicat.

Decisions preses per la implementació:

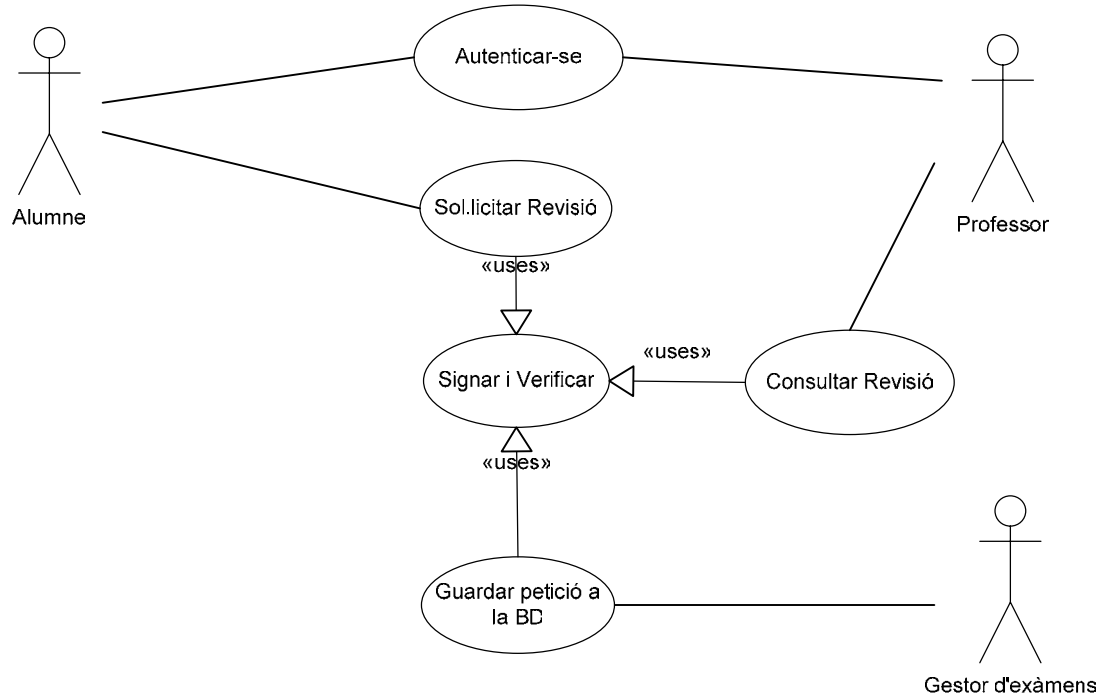
- ***Ir*** s'obté a partir d'una instància de la classe SecureRandom().

Classes necessàries per a la revisió d'un Examen.

La revisió d'un examen no implica la modificació de l'estructura de classes que ja estava definida fins aquest moment.

Diagrama de casos d'ús d'aquest protocol:

Revisió de l'examen



Descripció dels casos d'ús d'aquesta fase:

Sol·licitar Revisió

Cas d'ús: Sol·licitar Revisió.
Actors: Estudiant.
Propòsit: Demana al sistema que insereixi una petició de revisió per a un identificador.
Tipus: Primari i essencial.

Curs típic dels esdeveniments:

Accions dels actors	***Accions del Sistema***
1. El cas d'ús comença quan l'actor estudiant vol demanar una revisió a un enunciat definit per un identificador.	
2. L'estudiant envia la petició al servidor	
	3. El servidor rep les dades, les processa i accedeix a la base de dades per inserir la petició.
	4. Envia la confirmació a l'estudiant.
5. L'estudiant rep la confirmació.	

Cursos alternatius:

Punt 3: El servidor rep l'identificador d'un enunciat que ja té una revisió assignada en la base de dades. En aquest cas retornarà un error.

Punt 3: El servidor rep l'identificador d'un enunciat que encara no té una nota assignada en la base de dades. No es pot demanar una revisió a un examen que no està corregit. En aquest cas es retorna un error.

Consultar Revisió

Cas d'ús: Consultar Revisió.
Actors: Professor.
Propòsit: Demana al sistema que li retorni per pantalla la demanda de revisió a un enunciat.
Tipus: Primari i essencial.

Curs típic dels esdeveniments:

Accions dels actors	***Accions del Sistema***
1. El cas d'ús comença quan l'actor vol obtenir una revisió a un enunciat definit per un identificador.	
2. L'actor envia l'identificador al servidor	
	3. El servidor rep les dades, les processa i accedeix a la base de dades buscant la petició de revisió
	4. Envia la resposta al professor.
5. El professor rep la resposta per pantalla.	

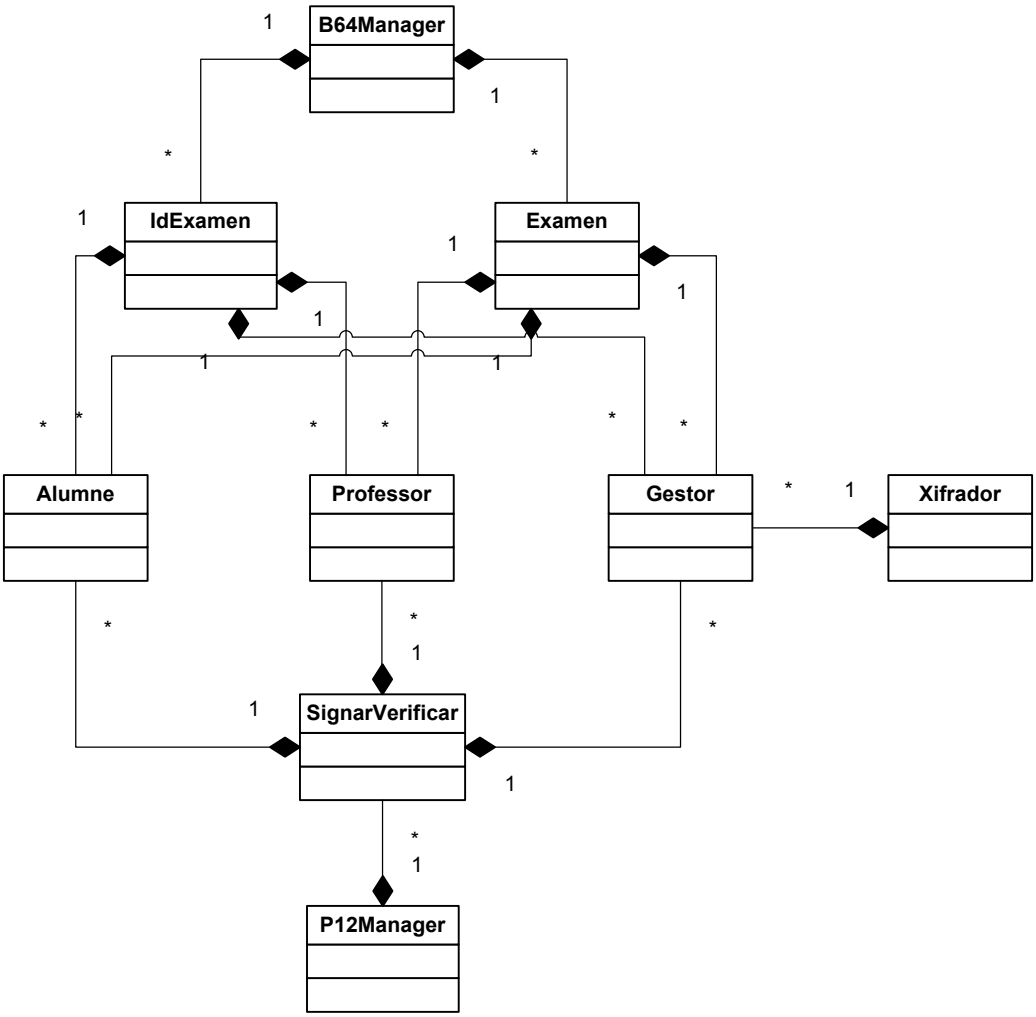
Cursos alternatius:

Punt 3: El servidor rep l'identificador d'un enunciat que no conté demandes de revisió a la base de dades. En aquest cas es retorna un error.

En els casos d'ús s'han comentat procediments que encara no estan definits, tot i que el protocol criptogràfic els necessita. Les classes necessàries per a dur a terme les tasques comentades es defineixen en els capítols corresponents. Per a realitzar les tasques per les que es no disposa encara la seva classe s'han utilitzat procediments lògics tal i com es veurà en l'apartat de les proves realitzades.

3.10 Diagrama de classes de l'esquema criptogràfic.

Com s'ha vist, al final de cada punt del protocol s'anaven comentant les classes que s'utilitzaven. A continuació es mostra el diagrama de classes complert de l'esquema criptogràfic. S'inclou, a més, una classe per a cadascun dels actors.



Com a detalls a comentar d'aquest diagrama, es fa notar que:

- Cada classe d'usuari utilitzarà una i només una classe per signar i verificar (que utilitzarà durant tota l'execució de l'aplicatiu).

- La única classe que té accés al Xifrador és el gestor.

3.11 Proves realitzades.

Per tal de dur a terme un joc de proves sobre el protocol, s'ha utilitzat una classe de prova que conté un exemple del cicle de vida d'un examen. El codi d'aquesta classe no inclou, com és d'esperar en aquest moment del disseny incremental del projecte, ni accés a mètodes remots utilitzant RMI[5], ni accés a base de dades, ni tampoc la implementació del mètode per autenticar els diferents clients contra el gestor d'exàmens. Aquest codi es pot veure en un annex d'aquesta memòria.

4. Representació de les dades: XML.

4.1 Introducció

XML[10] és l'acrònim de eXtensible Markup Language. Des de que va aparèixer aquesta forma de representar les dades s'ha imposat com una de les formes més eficients per intercanviar i emmagatzemar dades entre aplicacions i/o protocols.

En el projecte s'ha utilitzat XML[10] per a fer les transferències dels exàmens i les respostes corresponents entre el gestor d'exàmens i els clients, tan del professor com de l'estudiant. De la mateixa manera també s'utilitza XML[10] per a fer la transferència de dades durant el protocol d'autenticació entre el client i el servidor, però com aquesta tasca és molt especifica se li dedica un capítol.

La idea general és que cada cop que s'enviïn dades entre les parts només s'enviarà la part necessària segons la fase del protocol en la que s'estigui duent a terme la comunicació. Per aquest fet és normal veure que en l'especificació de la classe s'obtenen força mètodes que són característics de punts concrets d'un protocol. Així per exemple, si en la fase de l'obtenció de la nota l'estudiant obté la nota i ha de verificar la signatura, només rebrà la part del document XML[10] que conté les dades necessàries per a verificar la signatura.

Per a realitzar la feina de conversió a XML[10] i viceversa s'ha utilitzat la llibreria pública JDOM[8]. Si en un futur es decideix integrar aquesta aplicació amb una altra, el fet de tenir les dades en XML[10] facilitarà el procés.

4.2 Estructura del document XML.

L'estructura del document XML[10] utilitzada per a un examen és la següent:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE InformacioExamen SYSTEM "examen.dtd">
<Examen>

	<IdExamen>
		<As/>
		<Cd/>
		<Qu/>
		<Da/>
		<NumS/>
	</IdExamen>

	<Alumne/>
	<Enunciat/>
	<Resposta/>
	<Es/>
	<Ie/>
	<Rs/>
	<Tm/>
	<T/>
	<Ts/>
	<X/>
	<Xs/>
	<Nota/>
	<Ns/>
	<Rv/>
	<Ir/>

</Examen>
```

L'explicació de cadascun dels camps de l'XML[10] i del que contenen és com segueix:

- <Examen> És l'arrel del document, aquí es guarda tota la informació de l'examen.

- <IdExamen> Estructura que guardarà la informació referent a l'identificador de l'examen, aquesta part es subdivideix en una sèrie de camps que es desglossen així:
  - <As> Nom de l'assignatura.
  - <Cd> Codi de l'assignatura.
  - <Qu> Quadrimestre en el que s'imparteix.
  - <Da> Data de l'examen.
  - <NumS> Número de sèrie de l'examen.

- <Alumne> Es guarda aquí l'identificador de l'estudiant que ha realitzat l'examen.

- <Enunciat> Guarda el text de l'enunciat de l'examen.

- <Resposta> Aquí es guarda la resposta de l'estudiant.

- <Es> Inclou la signatura de l'examen formada pels camps ***IdExamen*** i ***E***.

- <Ie> Identificador de resposta ***Ie***, generat aleatòriament.

- <Rs> Inclou la signatura de la resposta a l'examen formada pels camps ***Ie***, ***R*** i ***Es***.
- <Tm> Instant en el que el Gestor d'exàmens rep la resposta a un examen.
- <T> Capçalera de lliurament que inclou els camps ***IdExamen***, ***Ie*** i ***Tm***.
- <Ts> És el rebut de l'examen compost pels camps ***Es***, ***R*** i ***T***.
- <X> Conté el xifratge de la Resposta ***R*** (sobre digital).
- <Xs> Signatura del xifratge anterior que addicionalment conté a més els camps ***X***, ***R*** i ***Es***.
- <Nota> Aquest camp guarda la nota de l'examen.
- <Ns> Signatura de ***Es***, ***Ts*** i ***Nota.***
- <Ir> Identificador de Revisió d'examen ***Ir***, generat aleatòriament.
- <Rv> Signatura d'una revisió que inclou els camp ***Ir*** i ***Ts***.

4.3 DTD del document XML.

DTD són les sigles de Document Type Definition i el seu propòsit és definir la legalitat dels blocs d'un document XML[10]. Bàsicament defineix l'estructura d'un document amb una llista dels elements legals.

El DTD del document XML[10] emprat per guardar la informació dels exàmens és el següent:

```
<?xml version="1.0"?>
<!DOCTYPE InformacioExamen [
  <!ELEMENT Examen        (IdExamen, Alumne, Enunciat, Resposta, Es, Ie, Rs,
Tm, T,  Ts, X, Xs, Nota, Ns, Rv, Ir)>
  <!ELEMENT IdExamen      (As, Cd, Qu, Da, NumS)>
  <!ELEMENT As            (#PCDATA)>
  <!ELEMENT Cd            (#PCDATA)>
  <!ELEMENT Qu            (#PCDATA)>
  <!ELEMENT Da            (#PCDATA)>
  <!ELEMENT NumS          (#PCDATA)>
  <!ELEMENT Alumne        (#PCDATA)>
  <!ELEMENT Enunciat      (#PCDATA)>
  <!ELEMENT Resposta      (#PCDATA)>
  <!ELEMENT Es            (#PCDATA)>
  <!ELEMENT Ie            (#PCDATA)>
  <!ELEMENT Rs            (#PCDATA)>
  <!ELEMENT Tm            (#PCDATA)>
  <!ELEMENT T             (#PCDATA)>
  <!ELEMENT Ts            (#PCDATA)>
  <!ELEMENT X             (#PCDATA)>
  <!ELEMENT Xs            (#PCDATA)>
  <!ELEMENT Nota          (#PCDATA)>
  <!ELEMENT Ns            (#PCDATA)>
  <!ELEMENT Rv            (#PCDATA)>
  <!ELEMENT Ir            (#PCDATA)>
]>
```

Ja que sovint moltes parts dels documents estaran buides o incompletes no s'han afegit restriccions de contingut per a facilitar la flexibilitat i evitar errors.

4.4 Funcionament de la representació de dades mitjançant XML.

En el disseny s'ha considerat el fet que posteriorment s'utilitzarà una base de dades per tal de guardar aquests documents, de manera que el gestor d'exàmens ha de poder accedir de manera senzilla i ràpida. Per tant es guarda un document per cada un dels exàmens i per a cada resposta d'un examen concret.

Quan els aplicatius demanen informació sobre l'XML[10] es poden trobar en punts molt diferents de l'execució. No interessa que en tot moment el document XML[10] estigui essent enviat del gestor cap a les altres parts i al revés. En cada moment de l'aplicació únicament s'envia la part del document XML[10] que sigui necessària per a realitzar les tasques demanades.

La manera de dur a terme aquest procés és la següent, cada part de l'aplicatiu té accés a crear documents XML[10] temporals on afegeix i modifica la informació que va utilitzant. Si necessita informació extra, la part informa al gestor d'exàmens de la fase en la que es troba i aquest li retornarà només els camps que li fan falta. Un cop la tasca ha estat realitzada el document XML[10] temporal s'envia al gestor d'exàmens, qui s'encarrega d'analitzar-lo i afegir-lo a la resta de dades que ja tenia.

Utilitzant aquest protocol de transmissió s'assegura una transferència mínima de dades en les comunicacions de les dues parts, evitant la carrega de les línies de comunicació.

Decisions preses per la implementació:

- La classe XMLManager és l'encarregada de passar les dades a base64 i al revés. Aquest procés es realitza d'una manera transparent a la resta de classes. Només s'han de passar les dades en binari i aquesta classe s'encarregarà de guardar-les en base64 i de retornar-les en binari.

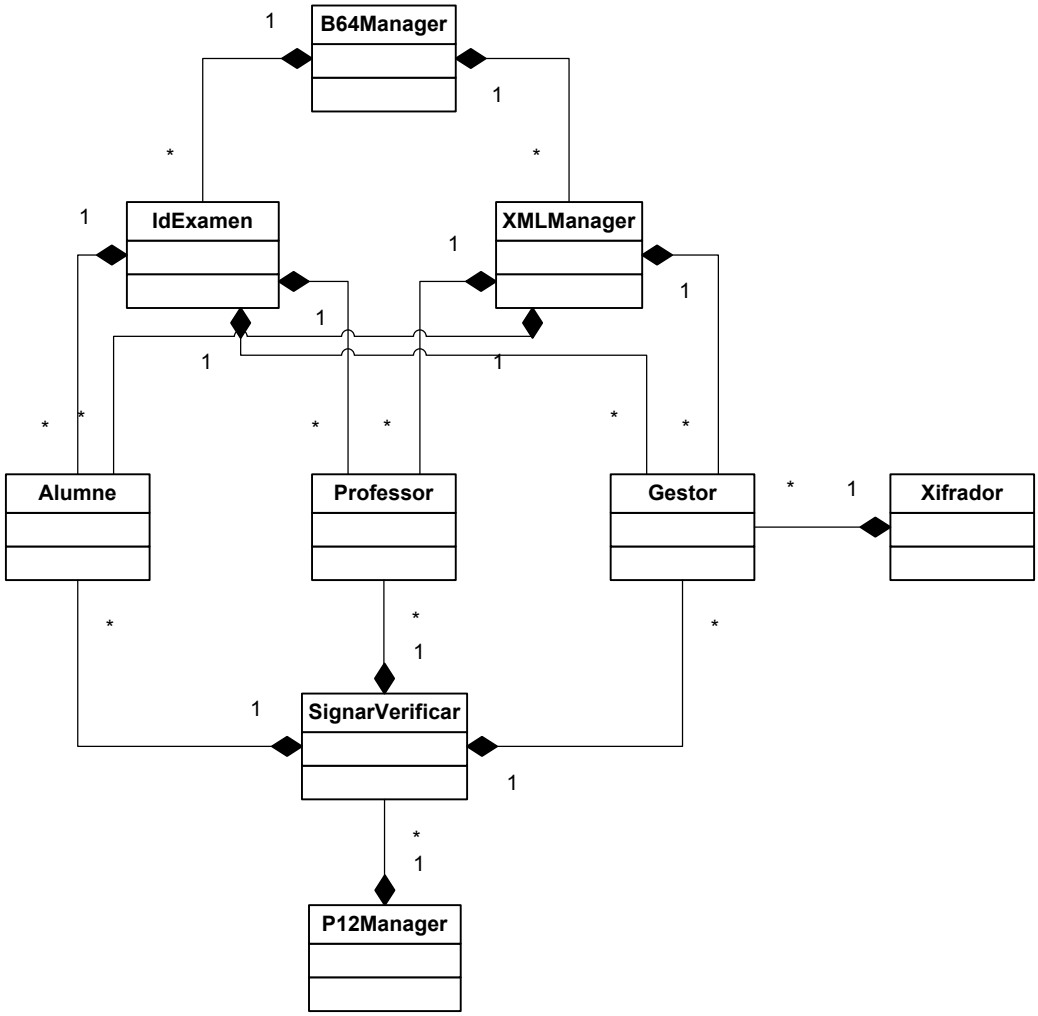
- Sovint és necessari que es retorni, junt amb les dades, les capçaleres XML[10], per tal de guardar tota aquesta informació a la base de dades. Aquesta tasca també la realitza la classe XMLManager.

- Aquesta classe és l'encarregada de dur a terme el pas de una cadena tipus String a document XML[10] utilitzant l'objecte SAXBuilder. L'anàlisi posterior del document generat utilitza recursivitat per navegar per tot l'arbre.

- A partir d'aquest punt la classe Examen deixa de tenir sentit. No passa el mateix amb la classe IdExamen que encara s'utilitza perquè disposa de mètodes útils per tal de concatenar les dades d'un Identificador i retornar estructures fàcilment utilitzables amb XML[10].

4.5 Diagrama de classes de la representació de dades mitjançant XML.

En el diagrama de classes del capítol anterior hem substituït la classe Examen per la classe XMLManager:



Aquesta classe és especialment extensa perquè conté els atributs per a cadascun dels elements que componen el document XML[10] i un mètode associat a cada atribut que retorna el contingut XML[10]. En el contingut XML[10] també hi ha la capçalera, i les dades estan codificades en base64 per a ser tractades posteriorment a la base de dades.

S'ha implementat un mètode per a cada etapa del protocol criptogràfic, de manera que a partir de totes les dades que pot contenir l'XML[10] només retorna aquelles necessàries per una fase en concret. Per posar un exemple, el mètode `get_IdEEs_Redaccio()` només s'utilitzarà durant la redacció de l'examen i retornarà la cadena formada pels camps ***Id***, ***E*** i ***Es***.

Finalment també es poden destacar els mètodes necessaris per tal de calcular les signatures i les posteriors verificacions. Per exemple, les dades per a signar un examen serien obtingudes amb `getEASignar()`. En aquest cas, la classe XMLManager s'encarregarà de concatenar les dades necessàries, essent aquest un pas més transparent per l'usuari final.

En temes del disseny es pot afegir que aquesta classe serà utilitzada per tots tres actors. Més endavant es veurà que les classes que implementaran els protocols de cadascun d'aquests actors, aconseguint així l'abstracció de les vistes, seran les úniques que accediran a aquesta classe.

4.6 Proves realitzades.

Igual que en el capítol anterior, per tal de provar que les dades es converteixen correctament al format XML[10] i al revés, s'ha modificat l'arxiu de test. El codi modificat i que ja utilitza la nova classe es pot veure en l'annex Annex G: Codi del joc de proves de la representació de les dades en XML. d'aquesta memòria.

5. Comunicació del components.

5.1 Introducció

La comunicació del diferents components és una part essencial del projecte. El professor ha de poder enviar els enunciats dels exàmens de forma remota. Els estudiants han de poder accedir des de les diferents seus als enunciats i enviar les respostes. La comunicació dels diferents components del sistema tradicionalment suposaria el disseny d'un protocol o mecanisme de comunicació. Per evitar la sobrecàrrega de feina, i donat que la part essencial és l'esquema criptogràfic es va optar per la utilització de la tecnologia RMI[5].

RMI[5] són les sigles de Remote Method Invocation. Java[4] incorpora aquesta tecnologia a l'API estàndard. RMI[5] consta d'un servidor on s'executen diferents instàncies de les classes servidores que es necessiten. Les aplicacions que volen emprar els mètodes remots únicament necessiten saber la interfície del servidor, és a dir, els mètodes que ofereix la classe que està al servidor. La implementació d'aquesta interfície està oculta i el client no arriba mai a saber que és el que s'està executant.

En el projecte s'ha utilitzat RMI[5] per a comunicar els aplicatius de l'estudiant i del professor amb el gestor d'exàmens. Tota la informació que s'envia entre les parts es fa utilitzant RMI[5].

RMI[5] ha reduït notablement el temps de desenvolupament.

5.2 Funcionament de la comunicació amb RMI.

El funcionament de la tecnologia RMI[5] és el següent. El servidor RMI[5] estableix un contracte amb les aplicacions remotes. Això vol dir que el servidor informa de les classes i mètodes que estaran disponibles. Aquest pas es fa creant una interfície que serà coneguda per tothom. Alhora el servidor RMI[5] té la implementació de la interfície.

Quan la classe està implementada s'ha de procedir a crear els punts de connexió que utilitzaran els aplicatius remots. El resultat d'aquest procediment crea dues classes que els clients han de conèixer. Aquestes es coneixen com Skeleton i Stub. Un cop preparat, les aplicacions remotes ja poden accedir a fer les instanciacions dels objectes remots. S'ha de tenir en compte que els objectes remots són persistents.

5.3 Implantació de RMI al Sistema.

Utilitzant RMI[5] podem dur a terme la gestió dels arxius XML[10] i l'accés a la base de dades d'una manera centralitzada, de manera que els usuaris remots només coneixen el nom del mètode que invoquen, però realment no saben el que està passant en l'altre extrem de la comunicació.

Per tal de dur a terme aquesta feina fa falta la creació de noves classes i d'una interfície remota. La interfície s'anomena IMetodes i inclou la descripció dels mètodes que es posen a disposició de tercers a través del servidor. Com tota interfície és necessari tenir una classe que implementi els mètodes descrits, doncs bé, aquesta classe s'anomena IMetodesImpl. Aquesta conté la implementació dels mètodes que bàsicament el que fan és gestionar les connexions a la base de dades i la gestió dels documents XML[10]. Finalment es necessita una classe que instancii un servidor i el faci públic a la xarxa. Aquesta classe s'anomena Servidor.

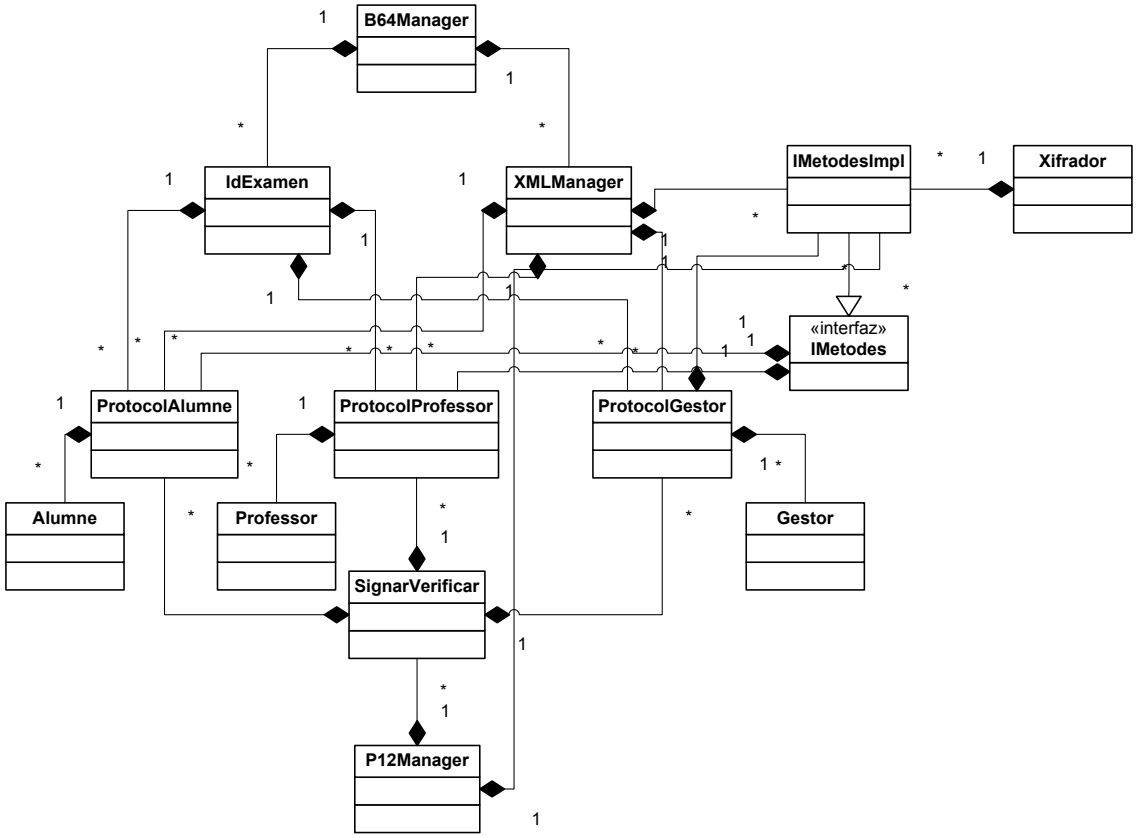
Les classes remotes que accedeixen a aquest servidor són dues, una pel professor i una altra per l'estudiant. Més endavant, quan s'implementa el protocol d'autenticació es crea una tercera classe per comunicar amb el servidor. Aquesta s'encarrega del protocol de seguretat. Cadascuna d'aquestes classes crea un vincle de comunicació amb el servidor que permet fer les crides a tots els mètodes públics. Fins ara, en les proves, aquestes crides es feien de manera local.

Decisions preses per la implementació:

- Utilitzar RMI[5] perquè redueix el temps de desenvolupament d'aplicacions remotes. Els tests han verificat la portabilitat, tant del codi Java[4], com l'execució remota en diferents sistemes operatius, com ara Win32, Linux i Mac OS X.

5.4 Diagrama de classes de la comunicació dels components.

El diagrama de classes modificat és:



Poc a poc es veu com es van relegant feines cap al servidor. En la part de la base de dades es veurà com s'introdueixen cadascun dels protocols pels actors per separat amb una abstracció més general.

La classe Servidor només serveix per a fer públic a través de la xarxa els mètodes descrits en la interfície IMetodes, per això no surt en el diagrama de classes.

5.5 Proves realitzades.

Per tal de posar a prova el sistema de RMI[5] el que s'ha fet és crear dues classes, una per l'estudiant i l'altra pel professor. El codi que conté cadascuna d'elles és el mateix que ja s'ha comentat en capítols anteriors amb la única diferencia que ara ja es fan les crides necessàries als mètodes remots.

Així per exemple, quan el professor té la redacció de l'examen llesta i la vol enviar al gestor d'exàmens utilitza el codi següent:

```
int Resultat = server.enviar(xml.getRedaccio());
```

El resultat que retorna aquesta crida va en funció de l'execució que es realitza en el servidor. El que el servidor realitza en cada moment depèn de la fase en la que es troba l'execució, però en general la feina en aquest test es limita a verificar les

signatures de les dades enviades, ja que el sistema encara no disposa d'una base de dades. Per veure aquest fet amb més detall, el codi que executa el servidor en la fase de redacció és el següent:

```
System.out.println("El gestor ha rebut Es i procedeix a fer la verificacio.");

SignarVerificar SVGestor = new SignarVerificar("gestor.p12", "uoc2004");
System.out.println("Classe SignarVerificar pel Gestor instanciada.");

SVGestor.verifySignature(xml.getEs(), xml.getEASignar().getBytes());
System.out.println("Signatura verificada pel gestor d'examens.");
```

El mateix es realitza per la resta de fases. No s'inclou la resta del codi ja que és una mica extens i no aporta més informació útil.

5.6 Evolució del Protocol.

Per tal de no ficar un capítol extra després de tots, a continuació s'explica quin és el resultat final de la implementació RMI[5] del projecte. Un cop la base de dades i la interfície han estat implementades és necessari tenir una classe tant pel professor com per l'estudiant que continguin el protocol criptogràfic per fases, i permetin l'abstracció dels mètodes de la interfície.

Aquestes dues classes més la que implementa el protocol del servidor són les encarregades en última instància d'accedir, tal com s'ha comentat en el capítol anterior, a la classe XMLManager.

Durant la implementació d'aquest capítol, a més, s'ha utilitzat una nova classe que s'anomena LogManager. L'objectiu d'aquesta classe és el d'anar anotant en un arxiu totes les incidències de l'execució de l'aplicació. La utilització d'aquesta classe es va fer necessària un cop es va veure que el contingut de log mostrat per pantalla era massa extens. Com que només és una classe auxiliar es comenta amb més detall en un dels annexos. Tampoc no es mostrarà en els diagrames de classes ja que gairebé la totalitat de les classes hi accedeixen de manera que dificultaria la lectura.

6. Base de Dades.

6.1 Introducció

Els test que s'anaven realitzant fins aquest punt necessitaven d'un enunciat i/o una resposta per funcionar. Per tant, cada cop que s'iniciava el programa de test les dades es creaven de nou estant aquestes incloses en el codi.

La utilització d'una base de dades permet emmagatzemar els enunciats dels exàmens i les seves respostes de forma persistent, que és un dels objectius principals d'aquest projecte.

El Sistema Gestor de Bases de Dades utilitzat és MySQL[9]. Aquest, com la gran majoria de programari utilitzat, és de lliure distribució. A més, se'n pot trobar una implementació tant per a plataformes MS Win32, Linux o MacOSX, cosa que facilita la instal·lació en tot tipus de plataformes.

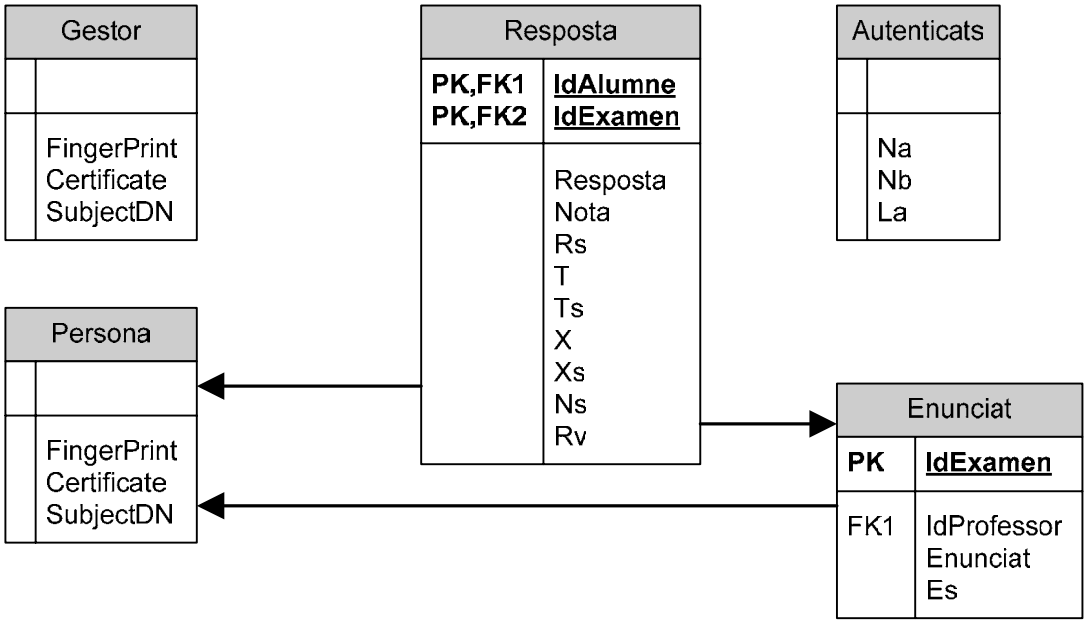
6.2 Utilitat de la base de dades.

A la base de dades s'hi guarden totes les dades referents al sistema, des de les dades dels certificats, la identitat dels actors del sistema, així com les dades dels exàmens i finalment les dades referents als clients autenticats.

L'encarregat d'accedir a la base de dades és exclusivament el gestor d'exàmens, que rebrà les peticions dels clients. Aquest era un dels objectius inicials del projecte, tal i com es pot veure en l'esquema de la introducció.

6.3 Model de la base de dades.

A continuació es mostra un esquema del model de la base de dades:



Les relacions són les següent:

- Persona → Resposta: Relació de 1 a *. Relaciona els estudiants de la taula Persona amb les respostes de la taula Resposta. Pot haver-hi més d'una resposta per cada estudiant.

- Persona → Enunciat: Relació de 1 a *. Relaciona els professors de la taula Persona amb els enunciats de la taula Enunciat. Pot haver-hi més d'un enunciat per cada professor.

- Enunciat → Resposta: Relació de 1 a *. Relaciona les respostes de la taula Respostes amb els enunciats de la taula Enunciat Hi pot haver més d'una resposta per cada enunciat.

Com es pot veure en la figura anterior, es relacionen les dues taules Enunciat i Resposta amb la taula Persona, això és degut a que la taula Persona conté tant els professors com els estudiants. S'ha optat per aquesta solució per ser la més simple i la que a priori dona menys problemes. Si es creu que per raons de seguretat és millor tenir els Estudiants i els Professors en taules separades, els canvis que s'han de realitzar són mínims.

6.4 Descripció de les taules de la base de dades..

Taula Autenticats:

Aquesta taula conté les dades referents als clients autenticats durant el procés d'autenticació per accedir a les opcions del sistema, i té els camps següents:

- La: funció de Hash sobre el certificat del client en base64 (Fingerprint).
- Na: número aleatori generat pel client.
- Nb: número aleatori generat pel servidor. Aquest camp serveix per a comprovar que el client és qui diu ser i que té permisos per a realitzar les accions sol·licitades.

Taula Persona:

Aquesta taula emmagatzema totes les dades, tant dels estudiants com dels professors. Realment el que inclou aquesta taula són els certificats i els fingerprints dels mateixos. A part s'hi afegeixen les dades relatives al certificat que poden servir en el futur. Els camps que conté són:

- Fingerprint: funció de Hash sobre el certificat de la persona en base64.
- Certificate: el certificat X509 de la persona.
- SubjectDN: dades auxiliars sobre el certificat de les persones.

Taula Gestor:

Aquesta taula és idèntica al anterior, però per qüestions de seguretat emmagatzemem els gestors d'exàmens en un espai apart. Els camps que conté són:

- FingerPrint: funció de Hash sobre el certificat de la persona en base64
- Certificate: el certificat X509 del Gestor.
- SubjectDN: Dades auxiliars sobre el certificat dels gestors.

Taula Enunciat:

La taula enunciat s'utilitza per guardar els enunciats dels exàmens. A més a més inclou la signatura digital de l'examen i el camp identificador que serveix per a relacionar-lo amb les respostes. La descripció dels camps que conté són els següents:

- IdProfessor: fingerprint del professor, per a futures comprovacions.
- IdExamen: estructura XML[10] de la concatenació dels camps del Identificador d'un examen. Les dades dels camps estan en base64.
- Enunciat: estructura XML[10] de l'enunciat de l'examen, també en base64.
- Es: signatura digital de l'enunciat de l'examen. Està en format XML[10] i en base64

Taula Resposta:

Aquesta taula s'utilitza per guardar les respostes dels exàmens. La taula inclou tots els camps relacionats amb el protocol criptogràfic. El camp IdAlumne és el que ens servirà per a relacionar-los amb els enunciats. Totes les dades a l'interior de l'estructura XML[10] estan en base64.

La descripció dels camps que conté és la següent:

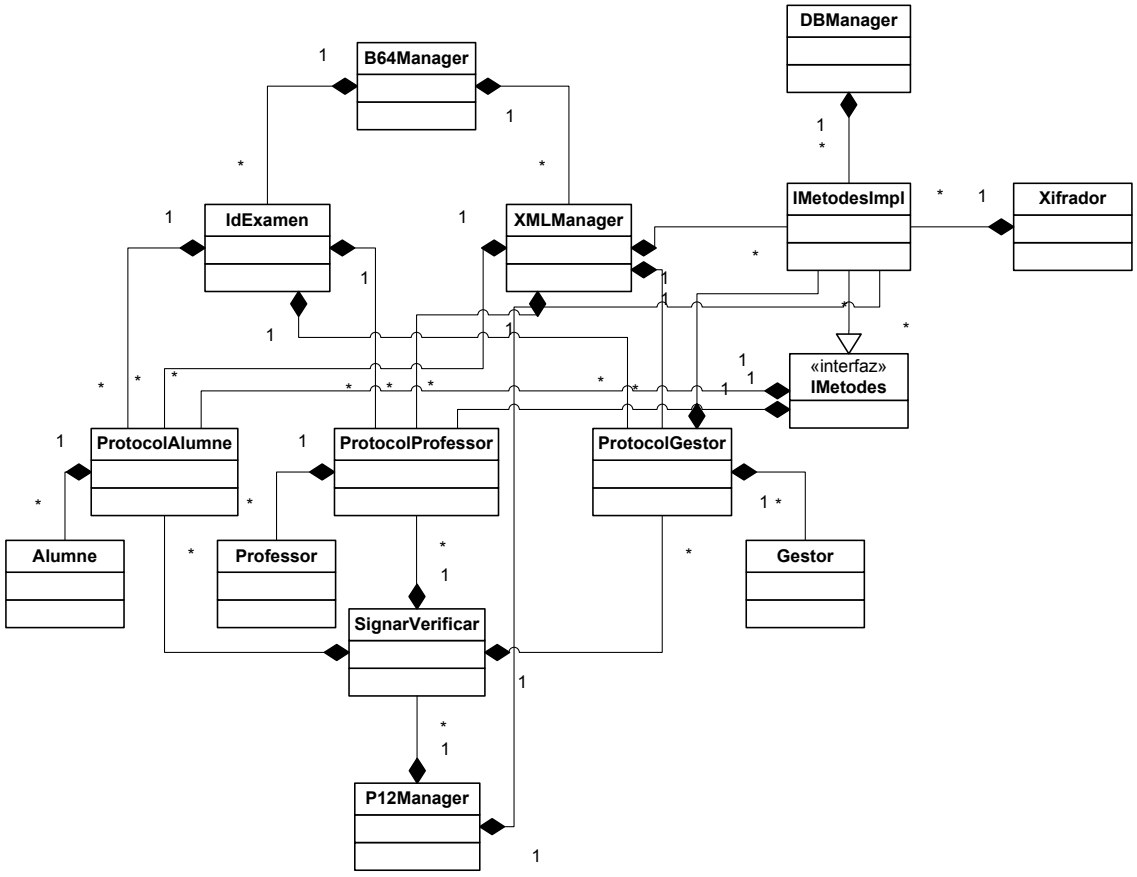
- IdAlumne: fingerprint de l'estudiant. Identificador únic.
- IdExamen: concatenació dels camps de l'identificador d'un examen.
- Resposta: la resposta de l'examen.
- Nota: nota assignada a l'examen.
- Rs: signatura digital de l'enunciat de l'examen.
- T: capçalera de lliurament que inclou els camps ***IdExamen***, ***Ie*** i ***Tm***.
- Ts: rebut de l'examen.
- X: conté el xifratge de la Resposta
- Xs: signatura del xifratge anterior que conté a més els camps ***X***, ***R*** i ***Es***.
- Ns signatura de ***Es***, ***Ts*** i ***Nota***
- Rv signatura d'una revisió que inclou els camp ***Ir*** i ***Ts***.

6.5 Classe responsable d'accés a la base de dades.

La classe responsable d'accedir a la base de dades és DBManager. La seva feina inclou totes les operacions, tant d'inserció de persones, com les consultes a enunciats i respostes, alhora que gestiona l'entrada i sortida de clients autenticats.

En el diagrama de classes de l'aplicació, aquesta és només utilitzada per la classe definida en el capítol anterior anomenada ProtocolGestor.

6.6 Diagrama de Classes de la part de la base de dades.



6.7 Parametrització de l'accés a la base de dades.

Per tal de facilitar l'accés a la base de dades des de qualsevol localització en la que s'instal·li l'aplicatiu, s'ha optat per tenir un arxiu de configuració que permeti entrar les dades bàsiques sobre la situació de la BD i el compte d'accés a la mateixa. L'arxiu es troba situat en l'arrel de l'aplicació i s'anomena "host.txt". Aquest que es mostra a continuació és un exemple de configuració amb els següents camps:

- Nom de la base de dades a connectar (pfc per defecte)
- Adreça IP del host de la base de dades.
- Nom del superusuari en la base de dades.
- Contrasenya del superusuari.

Exemple de "hosts.txt":

```
pfc
192.168.0.8
gestor
uoc2004
```

7. Protocol d’autenticació.

7.1 Introducció

Fins aquest punt s'ha descrit l'esquema criptogràfic, la representació de les dades, la comunicació dels components, i la base de dades. Ara bé, un punt essencial és poder identificar i autenticar els usuaris del sistema perquè segons qui sigui cada usuari podrà realitzar unes o altres operacions. Aquest pas és un dels primer que es realitza quan s'inicia la interfície gràfica.

El protocol d'autenticació que s'utilitza en el projecte està basat en el protocol modificat segur de Needham – Schroeder
[17].

7.2 Funcionament del protocol.

El protocol de Needham – Schroeder en si és força senzill d'implementar. S'ha creat una classe per fer aquesta autenticació que disposa de dos mètodes que implementen aquest funcionament. Per una banda, hi ha un mètode local en cadascun dels aplicatius, tant del professor com de l'estudiant, que s'encarrega de fer la feina per la part dels usuaris. A més a més, en la interfície pública dels mètodes compartits que ja s'ha comentat anteriorment en el capítol del protocol RMI[5], s'han afegit els mètodes necessaris per tal de dur a terme l'autenticació.

El protocol estableix la comunicació entre dues parts, que en aquesta descripció reben el nom de client i servidor per tal de fer la comprensió més senzilla.

La notació que s'utilitza és la mateixa que s'ha emprat en la descripció de l'esquema criptogràfic, tot i que a més a més s'utilitzen les expressions següents:

- Nx: número aleatori, tindrem N_{client} i $N_{servidor}$.

- Lx: fingerprint o Hash del certificat de la part x, n'hi haurà dos, un pel client i un pel servidor

Els passos a seguir són els següents:

Part a executar en el client:

- $E_{servidor}(N_{Client}, L_{Client})$
  Es xifra amb la clau pública del servidor un número aleatori i el hash de certificat del client.

- S'envia el resultat del pas anterior al servidor.

Part a executar en el servidor:

- $D_{servidor}(E_{servidor}(N_{client}, L_{client}))$
  El servidor obté N_{client} i L_{client} desxifrant el que ha rebut del client.

- $E_{client}(N_{client}, N_{servidor}, L_{servidor})$
  El servidor xifra utilitzant la clau pública del client els dos números aleatoris i el hash del certificat propi del servidor.

- S'envia el resultat del pas anterior al client.

Part a executar en el client:

- $D_{client}(E_{client}(N_{client}, N_{servidor}, L_{servidor}))$
  El client desxifra amb la seva clau privada i obté $N_{servidor}$ i $L_{servidor}$.

- $E_{servidor}(N_{servidor})$
  El client xifra $N_{servidor}$ amb la clau pública del servidor.

- S'envia el resultat del pas anterior al servidor

Part a executar en el servidor:

- $D_{servidor}(E_{client}(N_{servidor}))$
  El servidor desxifra amb la seva clau privada el missatge rebut.

- Comprova si $N_{servidor}$ que ha en última instància és igual al que ell va enviar. Si són iguals, aleshores el protocol està acabat i les dues parts es consideren autenticades.

7.3 Implantació del protocol al Projecte.

Per implantar el protocol explicat al projecte, i tal i com ja hem comentat, s'ha afegit una classe amb dos mètodes, un de local i un altre de remot per simular cadascuna de

les parts. Seguint les decisions de disseny del projecte, les dades que s'intercanvien el client i el servidor estan en format XML[10].

La integració del protocol anterior varia una mica de la implementació que es pot trobar en el projecte. Per tal d'estalviar una transferència de dades es va fer que un cop el client ha realitzat el primer pas i envia les dades al servidor, aquest crea una entrada a la base de dades conforme el client ha iniciat el procés d'autenticació. Amb les dades que envia, el servidor pot estar segur de la seva autenticitat.

Un cop les dades estan introduïdes en la base de dades, el servidor envia la resposta al client, de manera que únicament el client legítim pot obtenir el valor $N_{Servidor}$ que li permetrà autenticar-se més tard contra el servidor.

Per fer això, en el moment en el que el client ha de realitzar alguna acció que requereix l'autenticació envia una petició al servidor i adjunta el valor $N_{Servidor}$ que havia obtingut anteriorment. El que fa el servidor es comprovar si a la base de dades existeix algun client autenticat prèviament on el valor $N_{Servidor}$ coincideixi amb el que acaba de rebre. D'alguna manera podríem dir que s'estan fent els dos últims passos del protocol d'autenticació però en el moment de realitzar la petició i no abans. En el cas que el servidor trobi el valor a la base de dades procedeix a donar pas al client per realitzar les accions pertinents alhora que esborra de la base de dades el registre del client autenticat. D'aquesta manera si el client requereix d'una altra acció, ha de tornar a realitzar tot el procés de nou. Això evita els atacs de reutilització de valors (reply attacks).

Les dades que es passen entre el client i el servidor durant aquesta fase estan emmagatzemats dins d'una classe que s'ha declarat serialitzable per a que pugui ser passada a través de RMI[5] sense problemes.

7.4 Document XML per l'autenticació.

Les dades de l'autenticació es passen constantment entre els clients i el servidor. Per tal de seguir la mateixa política de transferència de dades entre les parts, aquestes es passaran dins d'un document XML[10].

L'esquema del document XML[10] per a l'autenticació és el següent:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE InformacioAutenticacio SYSTEM "autenticacio.dtd">
<Autenticacio>
      <Na/>
      <La/>
      <Nb/>
      <Lb/>
</Autenticacio>
```

L'explicació de cadascun dels camps del document és la següent:

- <Na> número aleatori del client (estudiant o professor).
- <La> hash (fingerprint) del client.
- <Nb> número aleatori del servidor.
- <Lb> hash (fingerprint) del servidor.

El DTD d'aquesta estructura XML[10] és el següent:

```
<?xml version="1.0"?>
<!DOCTYPE InformacioAutenticacio [
  <!ELEMENT Autenticacio  (Na, Nb, La, Lb)>
  <!ELEMENT Na            (#PCDATA)>
  <!ELEMENT Nb            (#PCDATA)>
  <!ELEMENT La            (#PCDATA)>
  <!ELEMENT Lb            (#PCDATA)>
]>
```

De la mateixa manera que en els documents XML[10] dels exàmens es disposava d'una classe que s'encarregava de gestionar-los, en aquests es farà el mateix. La classe que s'encarregarà de gestionar els documents XML[10] de l'autenticació serà XMLAuthManager. A aquesta classe hi tindran accés tant els usuaris dels aplicatius remots com el gestor d'exàmens.

8. Interfície gràfica.

8.1 Introducció

La interfície gràfica d'un aplicatiu acostuma a ser una de les parts més crítiques d'implementar. L'usuari final passarà moltes hores davant de l'aplicatiu i una interfície mal dissenyada o feixuga d'utilitzar pot condemnar l'aplicatiu al fracàs.

En el projecte, la interfície era un dels requeriments menys importants, es tractava de realitzar-ne una que fos prou simple d'utilitzar i que alhora donés la possibilitat de realitzar totes les opcions del sistema. S'ha optat pel més simple, una única pantalla tant per l'estudiant com pel professor.

8.2 Llibreria utilitzada: SWT

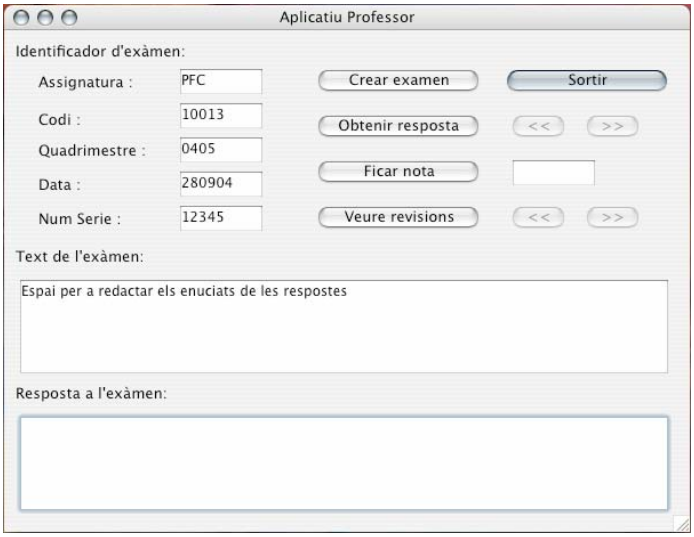
SWT són les sigles de Standard Widget Toolkit, i permet dissenyar interfícies d'una manera senzilla alhora que efectiva. El paquet SWT és de lliure distribució i ve amb les distribucions de l'entorn de desenvolupament Eclipse[7], que és el que s'ha utilitzat per desenvolupar el programari. La seva instal·lació i procediments de configuració per a ser utilitzats per primera vegada no són del més intuïtiu però.

Aquest procés està comentat a la secció 9. Joc de Proves., on com ja s'ha comentat, hi ha un exemple complert de l'execució del projecte, incloent tots els passos de configuració.

8.3 Aplicatiu del professor

Com s'apuntava en la introducció de la memòria, es disposa de dues pantalles, la primera d'elles és la de l'aplicatiu del professor.

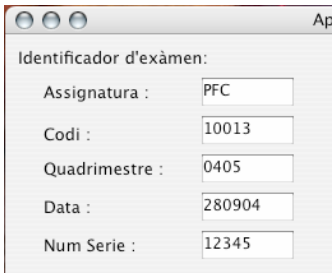
Aplicatiu del professor



Imatge 5: Aplicatiu del professor en un Mac OSX

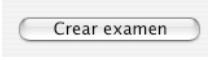
Aquesta pantalla està dividida en tres zones importants:

- Identificador d'examen: en aquests cinc camps de text el professor ha de subministrar el codi de l'examen que es vol crear o corregir. Els cinc camps són obligatoris i en cas de haver-n'hi algun buit, es produirà un error.



Imatge 6: Zona de l'identificador de l'examen

- Zona de comandes: aquests botons serveixen per a dur a terme les tasques del protocol del professor:
    - Crear Examen: executa la fase de redacció. És necessari que tots els camps de l'identificador i el camp de l'enunciat de l'examen estiguin correctament omplerts.



Imatge 7: Botó "Crear examen"

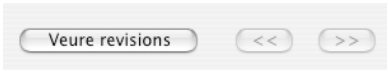
- Obtenir Resposta: en aquest cas, s'executa la fase 3 del protocol, la correcció de l'examen. El professor, com sempre, subministra l'identificador de l'examen i rep les respostes que hi ha per l'examen. Un text indicador ens informarà del nombre de respostes. Si aquest és major que un, aleshores s'activaran els botons que permeten navegar entre les respostes.

Imatge 8: Botó "Obtenir resposta" i botons de navegació.

- Ficar Nota: aquesta acció finalitza la fase de correcció. Assigna una nota a un examen seleccionat via el pas anterior. Si el camp de la nota no està correctament omplert mostrarà un missatge d'error.

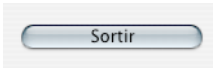
Imatge 9: Botó "Ficar nota" i camp per entrar el valor.

- Veure Revisions: de la mateixa manera que amb les respostes de l'examen, amb aquesta acció, el professor activa la fase de revisió del protocol. Obtindrà, una a una, les peticions de revisió que els estudiants hagin sol·licitat des del seu aplicatiu. Si n'hi ha més d'una aleshores els botons de navegació s'activaran.



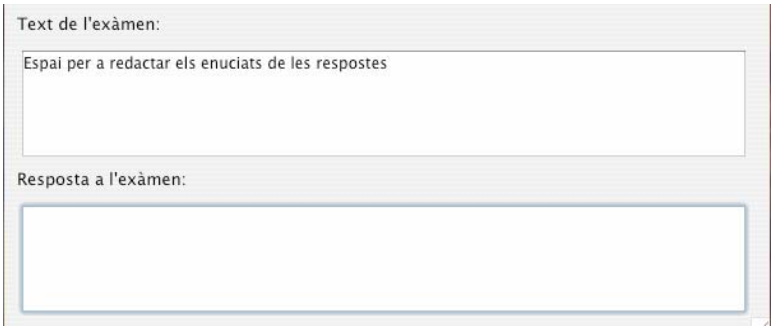
Imatge 10: Botó "Veure revisions" i botons de navegació.

- Sortir: aquest botó acaba l'execució del programari.



Imatge 11: Botó "Sortir".

- Camps per l'enunciat i les respostes: en aquests dos camps, als que el professor només té accés a editar el de l'enunciat, apareixeran els enunciats i les respostes dels estudiants.



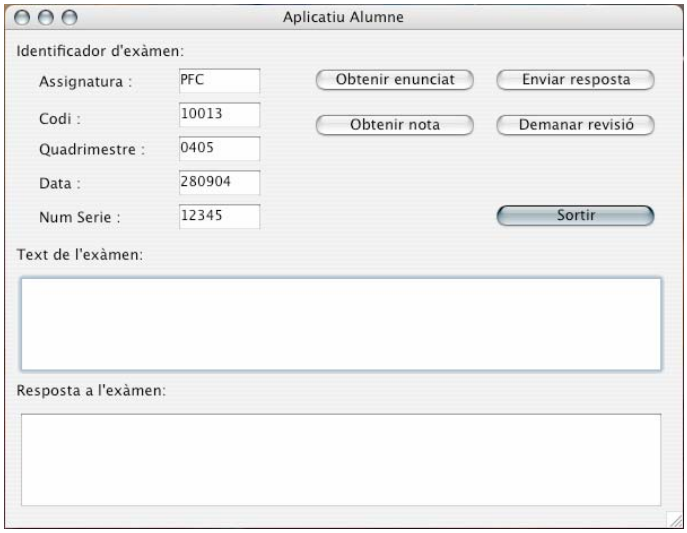
Imatge 12: Camps de text per l'enunciat i la resposta.

8.4 Aplicatiu de l'estudiant.

De la mateixa manera, l'estudiant disposa de la seva pròpia interfície gràfica. L'estudiant disposa d'una interfície que dona peu a menys opcions, ja que simplement pot rebre un enunciat, veure la nota de l'examen corregit i si ho creu oportú, pot demanar una revisió de l'examen. Aquí es cobreixen les fases 2, 4, i la primera part de la 5 de l'esquema criptogràfic.

La interfície de l'estudiant és tal i com es mostra a continuació:

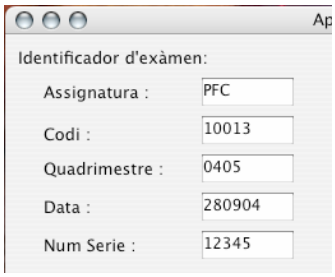
Aplicatiu de l'estudiant



Imatge 13: Aplicatiu de l'estudiant en un Mac OSX

Aquesta pantalla està dividida en tres zones característiques, igual que en l'aplicatiu del professor:

- Identificador d'examen: en aquests cinc camps de text l'estudiant ha de subministrar el codi de l'examen que vol contestar, del que en vol saber la nota, o del que en vol demanar una revisió. Els cinc camps són obligatoris i en cas de haver-n'hi algun buit, es mostra un missatge d'error.



Imatge 14: Zona de l'identificador de l'examen

- Zona de comandes: aquests botons serveixen per a dur a terme les tasques del protocol de l'estudiant:

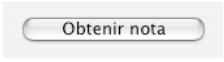
    - Obtenir Enunciat: aquest botó executa la fase de resposta del protocol. En aquest punt, l'estudiant veurà a l'espai del Text de l'examen, l'enunciat proposat pel professor. Si els cinc camps de l'identificador no estan correctament emplenats es produirà un error.

Imatge 15: Botó "Obtenir enunciat"

    - Enviar Resposta: prement aquest botó es finalitza la fase de resposta del protocol. En aquest cas, són necessaris els camps de l'identificador i el camp de la resposta. En cas de que algun d'aquest estigui malament es produirà un error.

Imatge 16: Botó "Enviar resposta"

    - Obtenir Nota: aquest botó dona accés a la fase d'obtenció de la nota del protocol. L'estudiant rebrà un missatge en el que se li informa de la seva nota si aquesta ja ha estat assignada. Cas de no ser així rebrà un missatge igualment on se li informa que la nota encara no està assignada.

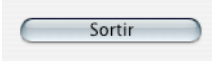


Imatge 17: Botó "Obtenir nota"

    - Demanar Revisió: aquest botó executa la fase de revisió del protocol. Introduirà a la base de dades la petició de revisió per a l'identificador en concret.

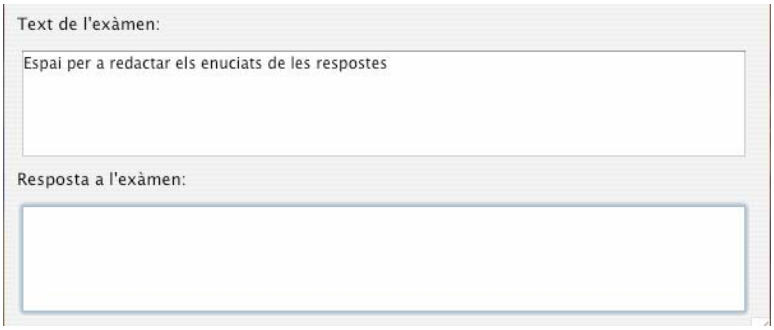
Imatge 18: Botó "Obtenir revisió"

    - Sortir: aquest botó acaba l'execució del programari.



Imatge 19: Botó "Sortir"

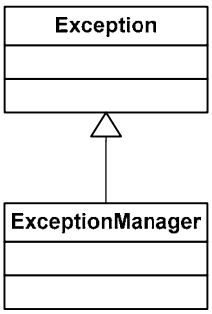
- Camps per l'enunciat i les respostes: en aquests dos camps apareixeran els enunciats dels exàmens i l'estudiant podrà escriure la resposta.



Imatge 20: Camps de text per l'enunciat i la resposta.

8.5 Gestió d'errors.

La interfície gràfica és l'encarregada en última instància d'informar dels errors que es puguin presentar durant l'execució del programari. Per tal de dur aquesta tasca a terme, s'ha implementat una classe específica. El nom d'aquesta classe és ExceptionManager i és una extensió de la classe Exception de Java[4].



La classe conté un atribut que s'anomena codiError. Cada cop que es produeix un error en l'aplicatiu es llença una nova excepció amb el codi de l'error corresponent. La llista dels error possibles es troba en l'annex H d'aquesta memòria.

El fet d'utilitzar aquest sistema per a notar els errors pot permetre, en un futur, traduir l'aplicatiu sense cap problema, assegurant així la internacionalització de l'eina, alhora que permet anar afegint errors a mesura que es detecten.

9. Joc de Proves.

9.1 Introducció

L'objectiu del joc de proves és donar un exemple de la configuració i l'execució complerta del sistema presentat. En aquest capítol es mostra com crear els certificats dels usuaris, la configuració de la base de dades, l'execució del servidor RMI[5] i un exemple complert del cicle de vida d'un examen utilitzant els aplicatius corresponents.

9.2 Generació dels certificats.

El primer que fa falta és preparar els arxius necessaris pels usuaris de l'aplicatiu. El primer que s'ha de preparar és el certificat autosignat de l'entitat certificadora. Per a dur a terme aquesta tasca s'empraran els arxius de comandes que s'han comentat en el Capítol 2. Sempre que s'hagi d'entrar una contrasenya s'emprarà "uoc2004", així és més senzill recordar-ho i per a dur a terme aquest exemple és suficient.

Adjunt a la memòria s'ha subministrat un directori anomenat PKI on hi ha una estructura de carpetes preparada per a dur a terme aquesta tasca. S'anirà explicant a mesura que l'execució avanci. Es recomana situar-se dins del directori /bin, per a executar més còmodament els scripts.

D'una banda s'han de crear les claus per a la CA amb una longitud de 2048 bits i després es generarà un certificat autosignat. S'ha de fer el següent:

```
./generarClaus CA.key 2048
```

El resultat és un arxiu anomenat CA.key que conté la parella de claus de la CA. A continuació s'ha de generar un certificat autosignat per la CA. S'ha de fer el següent:

```
./generaCertificatAutosignat CA.key CA.crt 365
```

El paràmetre 365 es refereix als dies que el certificat serà vàlid. Durant l'execució d'aquesta última comanda es demana a més que s'introdueixin totes les dades per a identificar la CA.

Per a aquest exemple s'han utilitzat les següents:

- Country Name: AD
- State or Province Name: Andorra
- Locality Name: Andorra
- Organization Name: UOC
- Organitzational Unit Name: UOC
- Common Name: Entitat Certificadora
- Email: camp deixat en blanc.

El resultat de les accions es veu a continuació:

```
IBook:~/PKI/bin aleix$ ./generarClaus CA.key 2048
Generating RSA private key, 2048 bit long modulus
.....................................................................
.....................................................................
.....................................+++
.....+++
e is 65537 (0x10001)
Enter pass phrase for CA.key:
Verifying - Enter pass phrase for CA.key:

IBook:~/PKI/bin aleix$ ./generaCertificatAutosignat CA.key CA.crt 365
Enter pass phrase for CA.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:AD
State or Province Name (full name) [Some-State]:Andorra
Locality Name (eg, city) []:Andorra
Organization Name (eg, company) [Internet Widgits Pty Ltd]:UOC
Organizational Unit Name (eg, section) []:UOC
Common Name (eg, YOUR name) []:Entitat Certificadora
Email Address []:
```

El resultat és l'arxiu CA.crt, que és el que feia falta per fer la resta.

Ara fa falta la generació dels arxius pels usuaris. El primer que és fa és crear la parella de claus pel gestor d'exàmens. Per dur a terme aquesta tasca fa falta executar els següents fitxers de comandes:

Generar les claus pel gestor, s'ha de tornar a utilitzar el codi següent (aquest cop la longitud és de 1024 bits):

```
./generarClaus gestor.key 1024
```

Seguidament s'ha de generar una petició de certificat per la CA que s'ha creat en l'apartat anterior. S'ha d'emprar la següent comanda:

```
./generaPeticioCertificat gestor.key gestor.csr ../openssl.cnf
```

Durant l'execució d'aquesta última comanda s'han d'introduir totes de dades per a identificar el gestor.

Per a aquest exemple s'han utilitzat les següents:

- Country Name: AD
- State or Province Name: Andorra
- Locality Name: Andorra
- Organization Name: UOC
- Organitzational Unit Name: Gestors
- Common Name: Gestor
- La resta de camps es poden deixar en blanc.

Ara s'han de moure els arxius creats a noves ubicacions per a no crear problemes de localització. Principalment s'ha de ficar l'arxiu CA.key a la carpeta PKI/CAPFC/private. L'arxiu CA.crt el col·loquem en PKI/CAPFC.

Ara s'ha de fer que la CA emeti el certificat, això es realitza amb la següent comanda, per evitar problemes amb el fitxer de configuració es recomanable situar-se ara a l'arrel de PKI:

```
./bin/generaCertificat ./bin/gestor.csr ./bin/gestor.crt openssl.cnf
```

Finalment s'ha de generar l'arxiu .p12, que és el que l'aplicatiu farà servir. S'ha d'utilitzar la següent comanda:

```
./bin/generaPKCS12 ./bin/gestor.key ./bin/gestor.crt ./CAPFC/CA.crt gestor.p12
```

El resultat de les accions es pot veure a continuació:

```
IBook:~/PKI/bin aleix$ ./generarClaus gestor.key 1024
Generating RSA private key, 1024 bit long modulus
....++++++
.......++++++
e is 65537 (0x10001)
Enter pass phrase for gestor.key:
Verifying - Enter pass phrase for gestor.key:

IBook:~/PKI/bin aleix$ ./generaPeticioCertificat gestor.key gestor.csr
../openssl.cnf
Enter pass phrase for gestor.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [ES]:AD
State or Province Name (full name) [Catalunya]:Andorra
Locality Name (eg, city) [Barcelona]:Andorra
Organization Name (eg, company) [Universitat Oberta de Catalunya]:UOC
Organizational Unit Name (eg, section) [Consultors]:Gestors
Common Name (eg, YOUR name) []:Gestor
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

IBook:~/PKI/bin aleix$ cd ..

IBook:~/PKI aleix$ ./bin/generaCertificat ./bin/gestor.csr ./bin/gestor.crt
openssl.cnf
Using configuration from openssl.cnf
Enter pass phrase for ./CAPFC/private/CA.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'AD'
stateOrProvinceName   :PRINTABLE:'Andorra'
localityName          :PRINTABLE:'Andorra'
organizationName      :PRINTABLE:'UOC'
organizationalUnitName:PRINTABLE:'Gestors'
commonName            :PRINTABLE:'Gestor'
Certificate is to be certified until Dec 24 09:30:14 2005 GMT (365 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated

IBook:~/PKI aleix$ ./bin/generaPKCS12 ./bin/gestor.key ./bin/gestor.crt
./CAPFC/CA.crt gestor.p12
Enter pass phrase for ./bin/gestor.key:
Enter Export Password:
Verifying - Enter Export Password:
```

El resultat és l'arxiu gestor.p12.

Ara és necessari fer el mateix pel professor i per l'estudiant tipus que s'utilitzen en l'exemple. L'execució és la mateixa que s'ha realitzat pel gestor. Es tracta de fer el mateix que s'ha fet tenint en compte el següent:

- A tot arreu on s'ha ficat gestor s'ha de modificar per estudiant o professor.
- A les dades del certificat s'han de canviar els següents camps:
  - Organitzational Unit Name: Estudiants o Professors depenent del cas.
  - Common Name: Estudiant o Professor depenent del cas.

Executat tot això s'haurien de tenir tres arxius .p12 a l'arrel del directori PKI:

alumne.p12
professor.p12
gestor.p12

Es poden guardar aquests arxius en la carpeta que més interessi, serà necessari recordar-la per a l'execució dels programaris.

Aquesta fase està completada. S'explicarà ara com preparar la base de dades.

9.3 Preparació de la base de dades.

Es parteix de la base que es disposa d'una màquina dedicada a allotjar el servidor de base de dades, en la que s'ha instal·lat prèviament MySQL[9] (es recomana la versió 4.1.7). Per a la gestió de la base de dades s'ha utilitzat el programa de lliure distribució SQLyog
[16] (versió 4.0 indispensable per a treballar amb MySQL 4.x).

El primer que s'ha d'executar és el fitxer que es pot trobar en l'annex C. Aquest fitxer crea la base de dades pfc (nom per defecte, però es pot triar el que es vulgui) i les taules necessàries. Seguidament és indispensable donar permisos totals a l'usuari que es connectarà des de l'aplicatiu del gestor d'exàmens a la base de dades que s'ha creat. Això és important fer-ho correctament ja que si, per exemple, el nom de l'usuari és gestor, no és el mateix donar permisos a gestor@localhost que a gestor@192.168.0.1, ja que són usuaris que accedeixen des de localitzacions diferents i per tant poden tenir permisos diferents.

Per fer l'accés a la base de dades de manera senzilla s'ha habilitat la possibilitat de tenir un fitxer que anomenat "host.txt", que es troba en l'arrel de l'aplicatiu del gestor, que conté les dades de l'usuari que es connecta a la base de dades.

hosts.txt:

```
pfc
192.168.0.8
gestor
uoc2004
```

En aquest exemple s'estaria dient que la base de dades s'anomena pfc, que està localitzada en el servidor amb adreça IP 192.168.0.8 i que l'usuari s'anomena gestor amb contrasenya uoc2004. Per tant, segons la configuració del sistema en concret cal

donar permisos a l'usuari gestor amb la adreça IP de la màquina que executa el servidor gestor d'exàmens. (gestor@192.168.0.8)

Un cop fet això es pot donar per configurada la base de dades, que ja està preparada per a acceptar les peticions que el gestor d'exàmens realitzi. La primera d'aquestes peticions és la d'introduir les dades dels certificats que s'han generat en el primer pas.

9.4 Inserció dels usuaris a la base de dades.

Aquest pas és crític i sovint és fàcil oblidar-lo. Si la base de dades no compta amb les dades dels certificats no pot comprovar que les dades que els clients li estan enviant en el moment de la autenticació són correctes.

Per a fer aquesta inserció es disposa d'una aplicació realitzada en Java[4]. Aquesta aplicació s'anomena CertManager.java i necessita de les classes que el gestor disposa en el seu directori, concretament utilitza DBManager, ExceptionManager i P12Manager. L'execució es realitza de la següent forma:

```
java CertManager filename.p12 password -g|-p
```

L'últim paràmetre indica si l'usuari que s'entra és una persona que utilitzarà l'aplicatiu o be és un gestor. El tracte pels dos és diferent. L'execució per a introduir el gestor seria:

```
java CertManager gestor.p12 uoc2004 –g
```

L'execució pels altres dos usuaris, estudiant i professor seria:

```
java CertManager professor.p12 uoc2004 –p
java CertManager alumne.p12 uoc2004 –p
```

Un cop realitzades aquestes execucions, les dades dels actors del sistema estan correctament entrades a la base de dades.

En aquest punt el servidor està gairebé preparat. Només falta arrencar el servidor RMI[5] que el deixarà en disposició de rebre peticions dels clients remots, abans però, s'ha de configurar la màquina virtual Java[4] per a que reconegui les llibreries externes que s'han utilitzat[6].

9.5 Configuració per a l'execució en Java.

Per tal de que tot es pugui executar sense problemes amb l'entorn Java[4], és necessari modificar alguns arxius de la instal·lació base de Java[4] i alhora s'han d'afegir les llibreries pròpies que s'han utilitzat.

La principal llibreria és la del IAIK
[15]. L'arxiu s'anomena *iaik_jce_full.jar* i la instal·lació d'aquesta llibreria porta implícita la modificació de dos arxius de seguretat que es troben en %JAVA_HOME%\jre\lib\security. Els dos arxius a modificar s'inclouen en la distribució d'aquesta memòria. Concretament s'han de sobrescriure els arxius:

- local_policy.jar
- US_export_policy.jar

L'arxiu *iaik_jce_full.jar* s'ha de copiar a %JAVA_HOME%\jre\lib\ext.

De la mateixa manera, faran falta les llibreries de JDOM[8] per a tractar documents XML[10] i la llibreria d'accés a bases de dades MySQL[9]. Els arxius s'anomenen *jdom.jar* i *mysql-connector-java-3.0.15-ga-bin.jar* respectivament. Tots dos s'han de copiar a la mateixa ubicació que l'anterior (en %JAVA_HOME%\jre\lib\ext). La instal·lació de les llibreries gràfiques es comentarà en l'apartat dels clients.

9.6 Execució del servidor RMI.

El servidor RMI[5] depèn de que Java[4] estigui correctament instal·lat, cosa que no depèn d'aquest projecte i alhora de que no hi hagi un firewall que bloquegi les connexions de xarxa, evidentment. El port que utilitza RMI[5], en general, és el 1099. Un cop està assegurat que aquest port està obert podem passar al següent pas. D'una banda és necessari compilar el codi font i després és necessari preparar les estructures que permetran a RMI[5] acceptar connexions.

Suposant que ja es té el codi compilat s'hauran d'executar les següents comandes:

```
rmic IMetodesImpl
rmiregistry &
java Servidor gestor.p12 uoc2004 &
```

La primera comanda prepara les estructures que ja s'han comentat en el capítol de RMI[5], en concret crearà dos arxius que són indispensables:

- IMetodesImpl_Skel.class
- IMetodesImpl_Stub.class

Els aplicatius del client han de tenir accés a aquests arxius, ja sigui via les variables d'entorn o per que es troben en la mateixa carpeta de l'aplicatiu.

La segona comanda executa com a servei el rmiregistry, que obre el port necessari i es posa a l'escolta. Finalment, la última comanda executa el servidor. Com es pot veure es passa com a paràmetres el nom i contrasenya del gestor d'exàmens. Una de les millores proposades pel projecte és crear una interfície per a dur a terme aquesta tasca de manera més còmoda.

Si es treballa en un entorn Linux/Unix veurem com en els processos del sistema s'executa el servidor si es fa servir la comanda `ps`:

```
IBook:~/Desktop/eclipse/workspace/PFCv1.1 aleix$ ps
  PID  TT  STAT      TIME COMMAND
  502 std  S      0:00.05 -bash
  517 std  S      0:00.34 rmiregistry
  518 std  S      0:02.98 java Servidor gestor.p12 uoc2004
```

La part del servidor està llesta.

9.7 Execució de la interfície gràfica del professor.

Com ja s'ha comentat en el seu capítol, per a les interfícies gràfiques s'ha utilitzat la llibreria que ve amb les distribucions de l'Eclipse[7], anomenada SWT. Aquesta, depenent del sistema operatiu que utilitzem, ve amb un o dos paquets (.jar). En el cas de plataformes Win32, ve en un únic arxiu anomenat *swt.jar*.

Aquest arxiu s'haurà de col·locar en alguna carpeta accessible des del CLASSPATH de la màquina virtual Java[4]. Per norma general i per comoditat, totes les llibreries utilitzades en aquest Projecte s'han instal·lat en el directori %JAVA_HOME%\jre\lib\ext.

De la mateixa manera és necessari afegir una llibreria (que en plataforma win32 és un arxiu .dll) a la ruta de cerca del Java[4] de les llibreries dinàmiques. En general això es pot afegir en el moment que és fa la mateixa crida al java, afegint la directiva d'execució: -Djava.library.path={runtime-library-path}. De la mateixa manera, es poden copiar les dll a una carpeta ja inclosa en aquesta ruta de cerca (%JAVA_HOME%\jre\bin, per exemple). De qualsevol de les dues maneres funcionarà correctament.

Resumint:

- S'ha de copiar *swt.jar* (i *swt-pi.jar* en Mac OSX) a %JAVA_HOME%\jre\lib\ext
- S'ha de copiar les .dll a %JAVA_HOME%\jre\bin

Per resoldre dubtes d'instal·lació i d'execució es recomana enèrgicament la consulta de l'adreça següent:

```
dev.eclipse.org/viewcvs/index.cgi/platform-swt-home/faq.html?rev=HEAD
```

És important tenir en compte que les llibreries es guarden en les carpetes del "Runtime Environment". Cada instal.lació de Java les pot tenir en ubicacions diferents. Es recomana consultar la documentació per a col·locar-les en el lloc adequat.

Un cop fet tot això la instal·lació de les llibreries necessàries estarà completada i es pot passar a l'execució dels aplicatius. La classe amb la interfície del professor s'anomena SWTProfessor i per executar-la es pot fer servir la següent comanda:

```
java SWTProfessor
```

Apareixerà una pantalla de benvinguda com la següent:

Imatge 21: Pantalla de benvinguda.

Un cop carregat el programa es demanarà per primer i únic cop que l'usuari s'identifiqui. Per tal de fer això s'ha de localitzar en el disc l'arxiu professor.p12 que s'ha creat en el primer punt (creació dels certificats) i entrar la contrasenya ("uoc2004" en aquest cas). Això es veu així:

Imatge 22: Pantalla d'autenticació.

Si la identificació no provoca cap error, aleshores es presenta la pantalla principal del professor.

9.8 Execució de la interfície gràfica de l'estudiant.

Els passos per executar la interfície de l'estudiant són els mateixos amb la diferencia que en aquest cas la classe s'anomena SWTAlumne i s'executa amb:

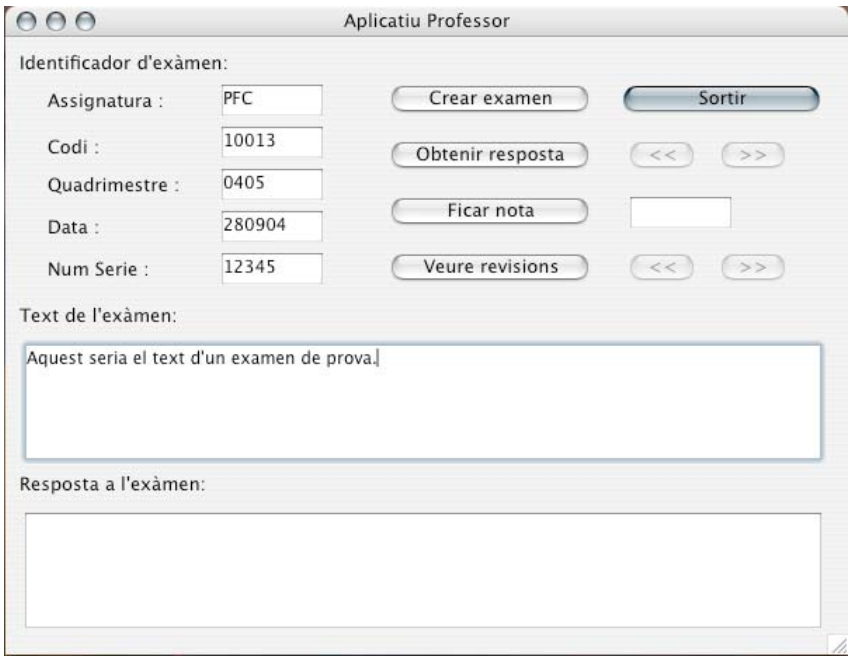
```
java SWTAlumne
```

Apareixerà una pantalla de benvinguda idèntica a l'anterior i es demanarà igualment que l'usuari s'identifiqui. En aquest cas s'ha de localitzar l'arxiu alumne.p12 i entrar la contrasenya "uoc2004".

Un cop les dues interfícies estan funcionant es pot passar a realitzar el cicle de vida que inclou l'esquema criptogràfic comentat en el Capítol 3 d'aquesta memòria.

9.9 Creació d'un Examen.

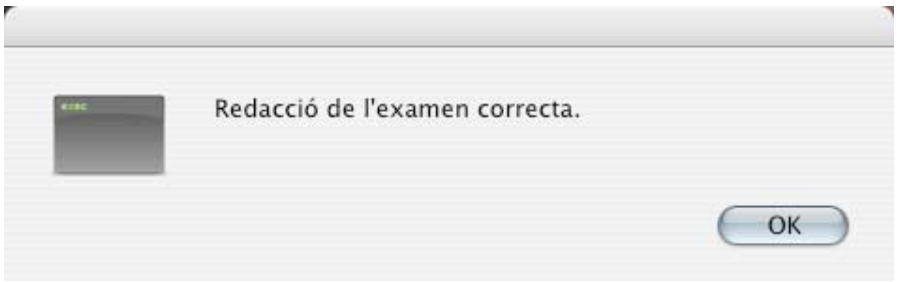
La pantalla principal del professor és la que ja s'ha mostrat anteriorment:



Imatge 23: Aplicatiu del professor.

Com es pot veure en la imatge ja s'ha omplert el camp del text de l'examen. Aquest és el que s'utilitza per tal de crear els exàmens. Com es pot veure també els camps de l'identificador de l'examen també contenen valors. Un cop aquests camps estan correctament emplenats es pot prémer el botó "Crear Examen".

Si tot ha anat bé, es mostrarà el següent missatge:



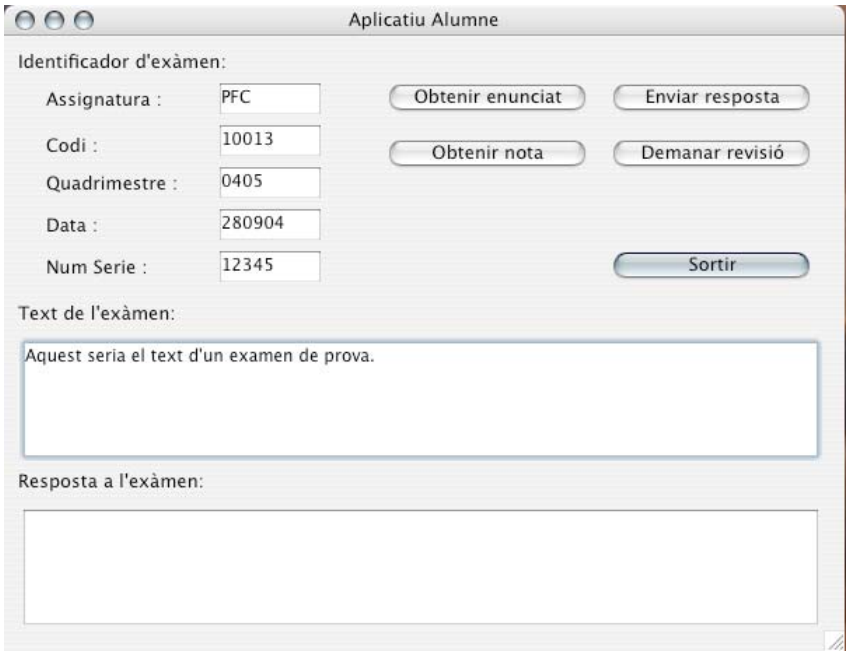
Imatge 24: Redacció correcta.

Amb això es pot donar per completada la fase de creació d'un examen. S'ha de tenir en compte que no es poden tenir dos exàmens a la base de dades amb el mateix identificador.

9.10 Resposta a un Examen.

La pantalla inicial de l'estudiant és la que també s'ha mostrat anteriorment, en aquest cas s'han emplenat els camps de l'identificador de l'examen amb el mateix valor que

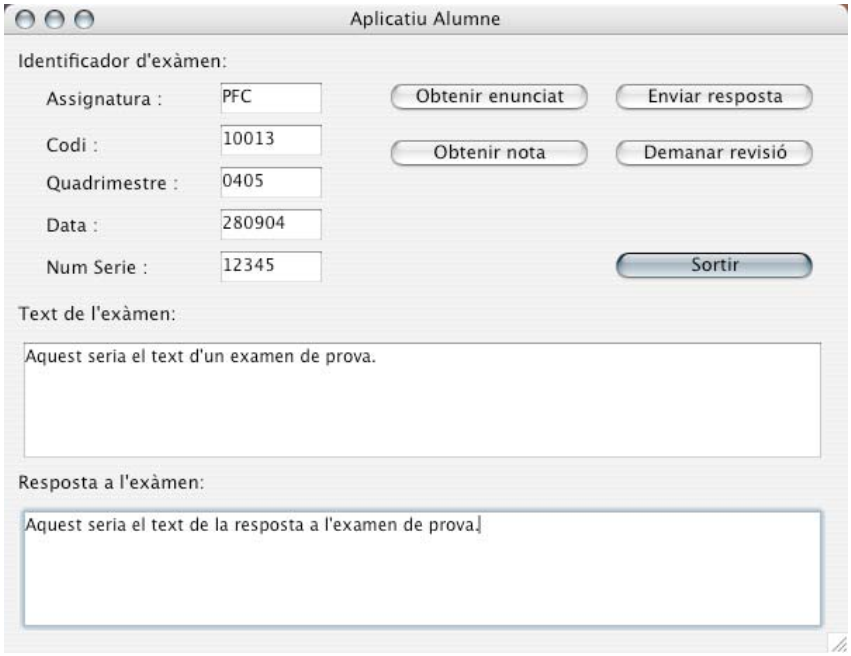
en l'aplicatiu del professor i s'ha premut el botó Obtenir Examen. El resultat és el següent:



Imatge 25: Aplicatiu de l'estudiant.

Com es pot apreciar el camp de l'enunciat de l'examen conté el text que el professor ha introduït en el seu aplicatiu.

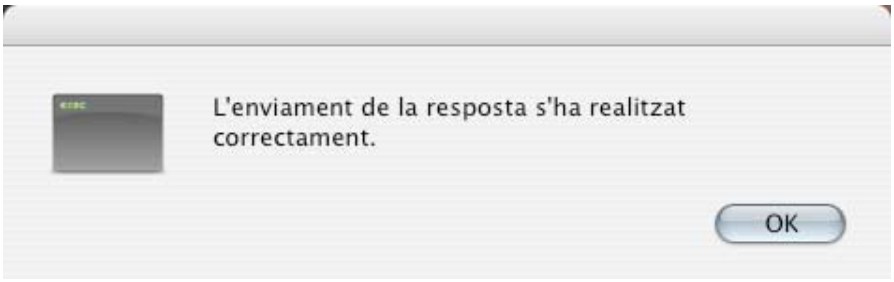
El pas següent a realitzar per l'estudiant seria el de respondre l'examen:



Imatge 26: L'estudiant ha respost l'examen.

Un cop el camp de la resposta està omplert simplement s'ha de prémer el botó Enviar Resposta. S'ha de tenir en compte que per a cada examen només hi pot haver una resposta per estudiant a la base de dades.

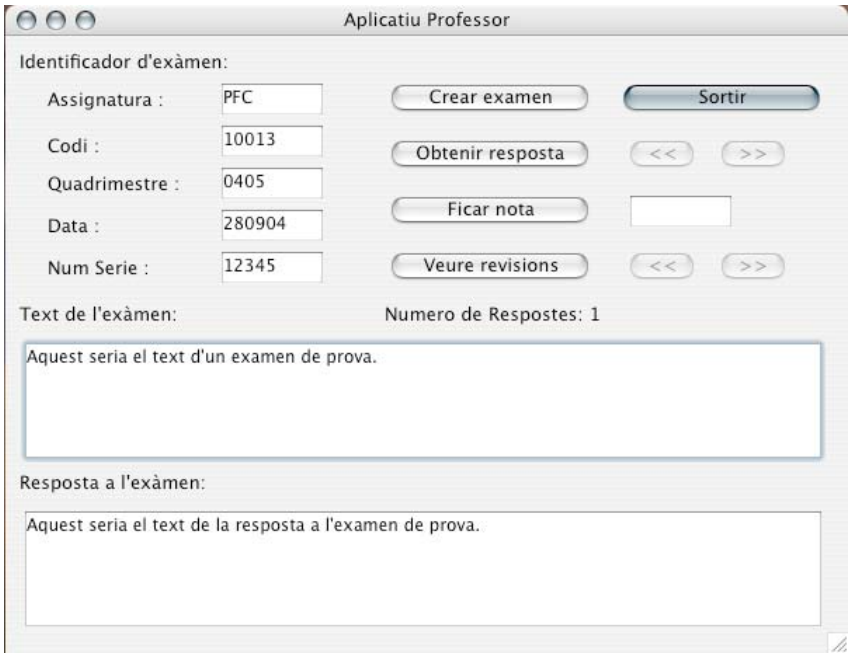
Si la comunicació funciona correctament es mostrarà aquest missatge amb el que la fase es dóna per completada:



Imatge 27: Procés de Resposta correcte.

9.11 Correcció d'un Examen.

De nou en l'aplicatiu del professor s'ha de prémer el botó Obtenir Resposta. Com que en el aquest exemple només hi ha una resposta a l'examen definit per l'identificador els dos botons de navegació per respostes no estan activats. El professor veu la següent pantalla:



Imatge 28: Aplicatiu del professor amb la resposta a l'examen.

Ara el professor ha de navegar entre les respostes (una en aquest cas) i assignar una nota a cadascuna d'elles. Per assignar una nota s'ha d'emplenar el camp que es troba a la dreta del botó Ficar Nota i posteriorment prémer aquest botó.

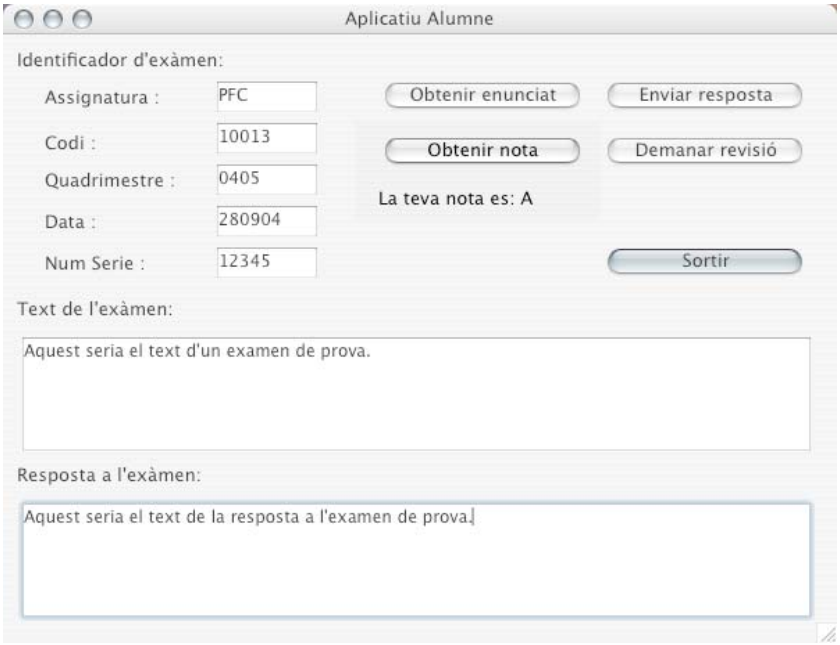
Imatge 29: Assignar una Nota a l'examen.

Si tot ha anat bé, el professor rebrà un missatge de confirmació:

Imatge 30: Nota assignada correctament.

9.12 Obtenir la nota d'un Examen.

Un altre cop, és feina de l'estudiant realitzar aquesta tasca. Simplement s'ha d'emplenar els camps obligatoris de l'identificador de l'examen i prémer el botó Obtenir Nota. Hi ha dues possibilitats. En cas de que no hagi estat assignada una nota es notificarà. En canvi si la nota ha estat assignada es mostrarà un missatge com el següent en l'aplicatiu de l'estudiant:



Imatge 31: Nota obtinguda en l'aplicatiu de l'estudiant.

Amb aquesta acció la fase de l'obtenció de la nota es dóna per completada.

9.13 Revisió d'un Examen.

Si l'estudiant després de veure la seva nota no està d'acord amb el resultat pot demanar una revisió a l'examen. Simplement s'ha de prémer el botó Demanar Revisió. Si la demanda es realitza correctament, l'estudiant veu el següent missatge:

Imatge 32: Missatge confirmant la petició de revisió.

La feina de l'estudiant acaba aquí.

Finalment des de l'aplicatiu del professor, aquest podrà prémer el botó Veure Revisions. El funcionament és el mateix que el de prémer el botó Obtenir Resposta, però en aquest cas, només es rebran aquelles respostes marcades per a ser revisades. De la mateixa manera, si hi ha més d'una petició pot navegar entre les revisions utilitzant els corresponents botons de navegació.

Un cop el professor s'ha rellegit la resposta pot tornar a assignar una nota a l'examen. En aquest cas la nova nota substituirà a l'anterior. El professor rep els missatges de confirmació que ja s'han mostrat.

En aquest punt acaba el protocol criptogràfic. Només queda explicar com apagar el servidor i el joc de proves es donarà per completat.

9.14 Apagar el Sistema.

Per sortir dels aplicatius del professor i de l'estudiant en qualsevol moment es pot prémer el botó Sortir i aquest acabarà. El client en qüestió només ha de arrancar de nou i podrà recuperar la feina en la fase en la que estès ja que tot s'emmagatzema en la base de dades. No és necessari fer de nou tot el cicle de vida de dalt a baix, evidentment.

Respecte a la parada del servidor, com que encara no es disposa d'una interfície pel gestor d'exàmens, aquesta s'ha de fer manualment. És a dir, s'ha d'executar el visualitzador de processos del sistema operatiu en qüestió i acabar les dues operacions que s'havien arrencat al principi.

En primer lloc s'ha d'aturar el procés "rmiregistry" i posteriorment s'ha d'aturar el procés "java Servidor nomfitxer.p12 password".

Amb aquests darrers passos el joc de proves es dóna per finalitzat.

10. Diagrames.

10.1 Introducció

Durant l'explicació dels capítols anteriors s'ha anat afegint poc a poc les classes que conformaven el sistema complert, de manera que s'anava veient com l'estructura del diagrama de classes s'assemblava a l'esquema general de l'aplicació.

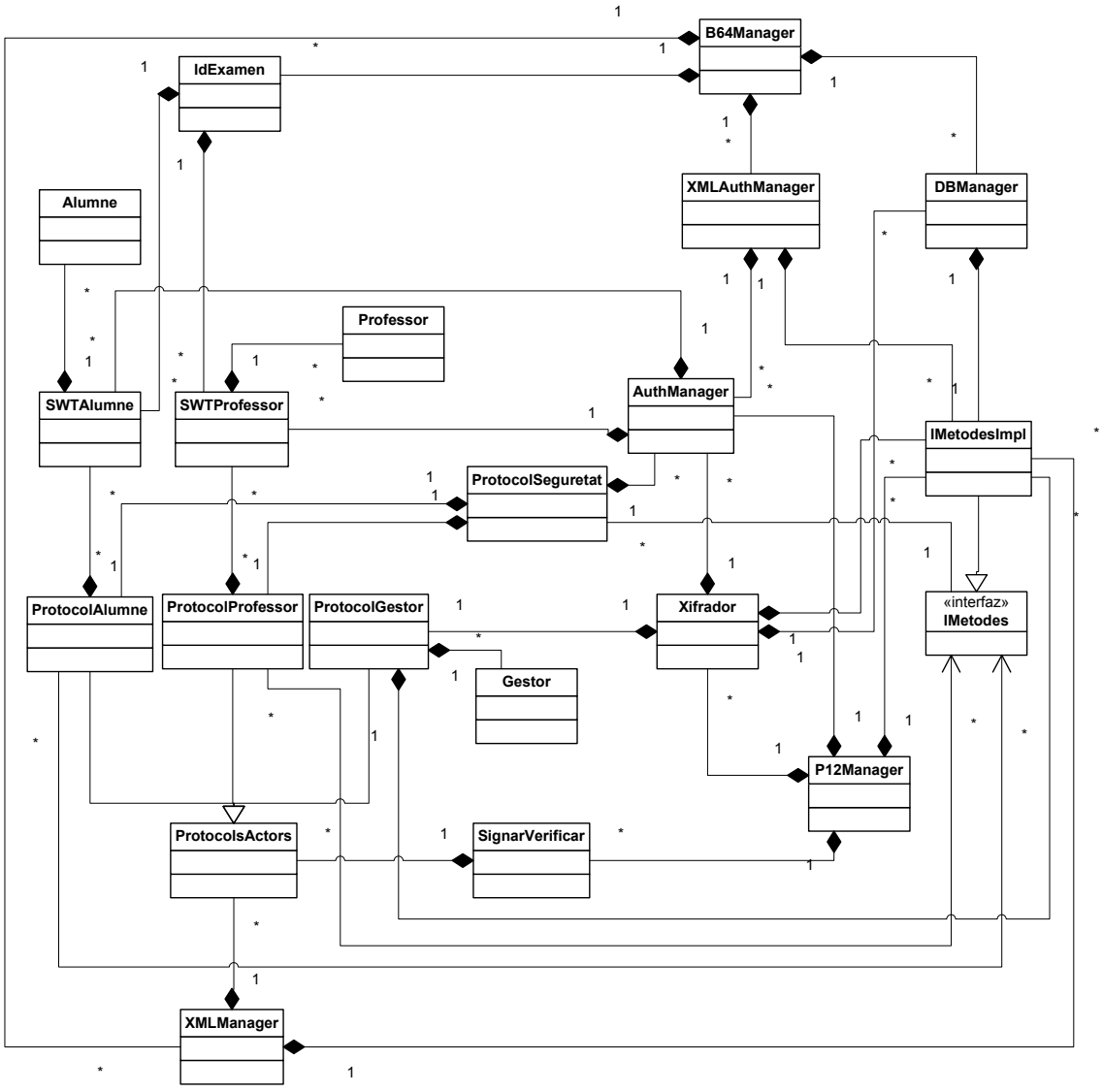
En aquest capítol es mostra el diagrama de classes complert a la vegada que s'inclouen els diagrames de seqüència complerts per a l'execució del cicle de vida d'un examen.

10.2 Diagrama de Classes.

El diagrama de classes[2] que es presenta a continuació inclou la gran part de les classes que s'ha utilitzat en el Projecte. Les que no hi són, és per que són classes auxiliars que no modifiquen l'execució del sistema. Com a exemple d'aquestes classes auxiliars podríem anomenar la classe LogManager, descrita en els annexos o la classe Servidor, que només serveix per fer pública la classe IMetodes etc.

Altres classes, tot i ser importants s'ha preferit deixar fora del diagrama per no complicar-lo. Un clar exemple d'aquest tipus podria ser la classe ClientAutenticat, ja que només és una estructura que s'utilitza per passar dades entre classes de manera més eficaç. Aquesta classe és usada per la majoria de classes i dificulta força la lectura del diagrama que es presenta a continuació:

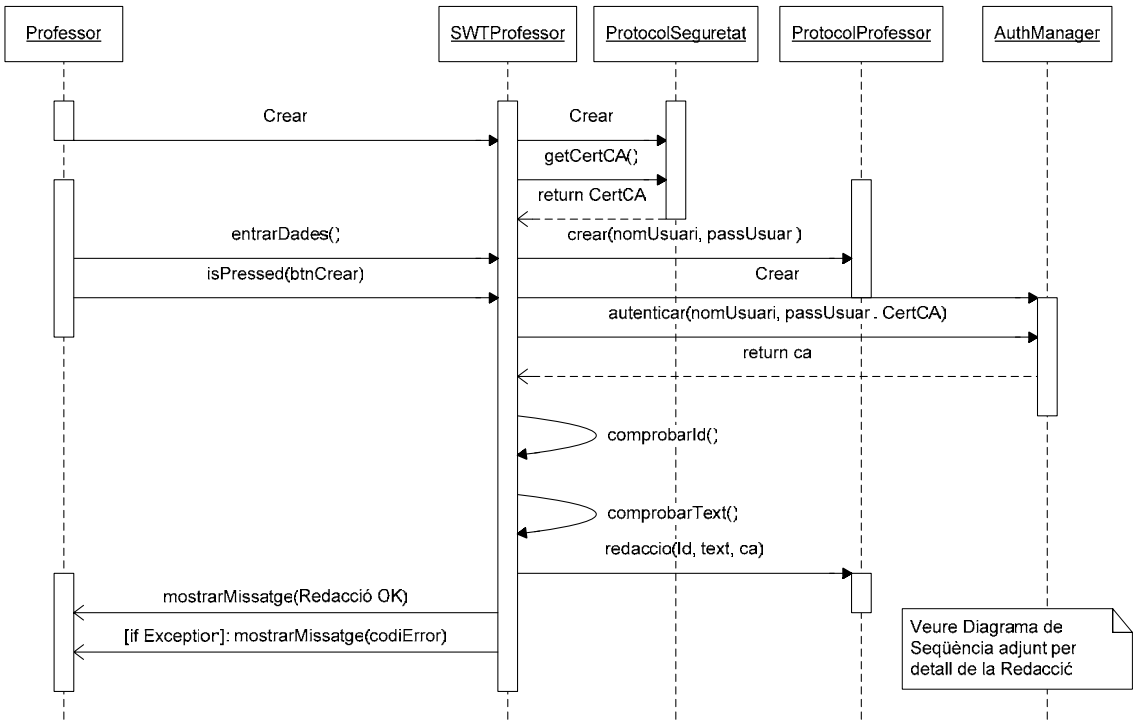
Diagrama de classes



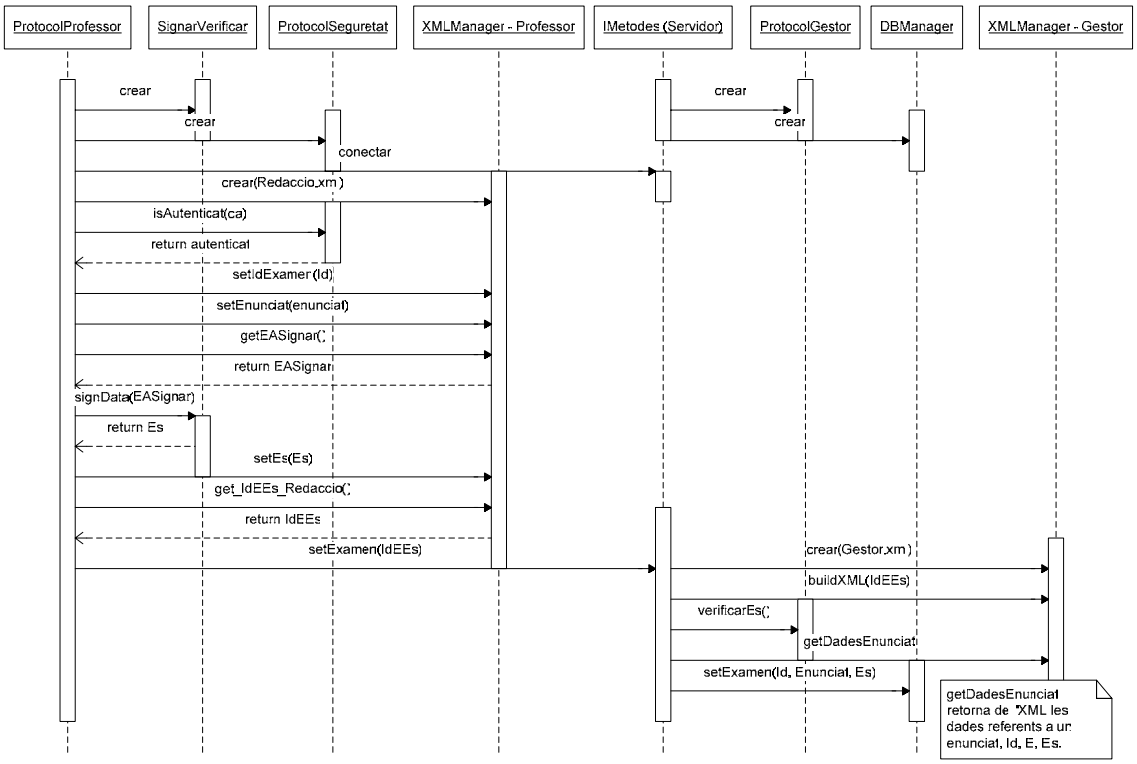
10.3 Diagrames de Seqüència.

Es mostren a continuació els diagrames de seqüència de l'execució del cicle de vida. Per tal de no fer els diagrames interminables i il·legibles s'ha limitat la profunditat dels mateixos només a aquelles crides realment significatives. El que es mostra a continuació són els diagrames de seqüència tant de l'execució de les Interfícies gràfiques, com els de les 5 fases de l'esquema criptogràfic, executats pels protocols de l'estudiant i del professor.

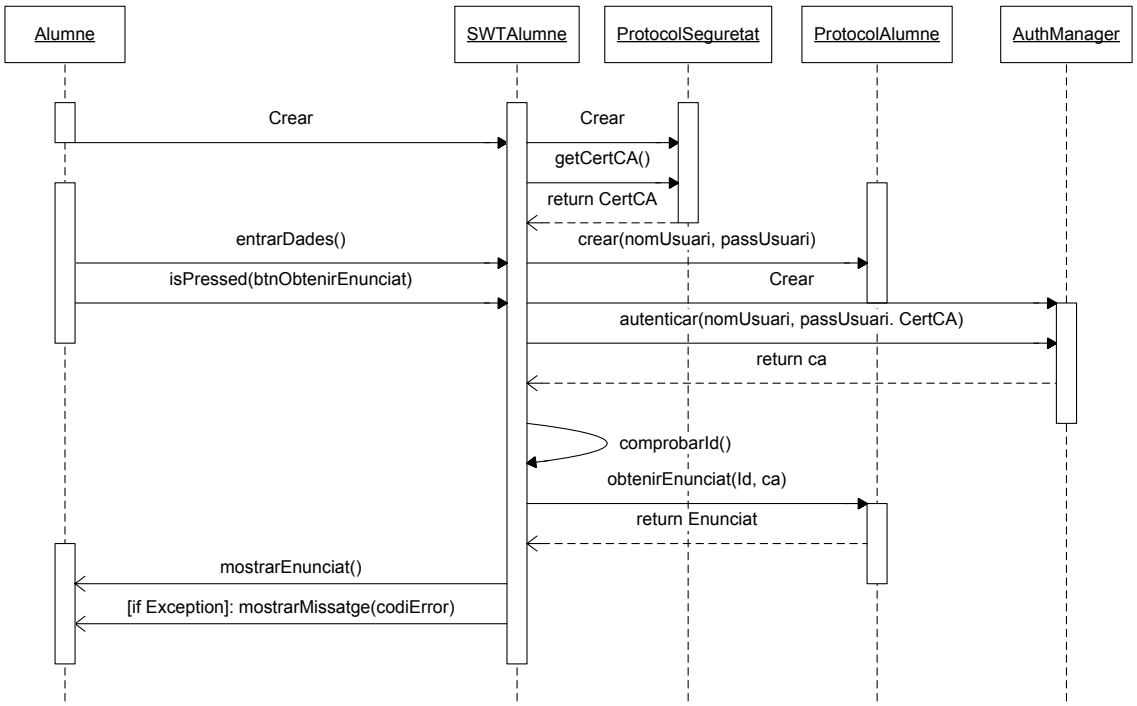
Fase 1: Redacció de l'examen.



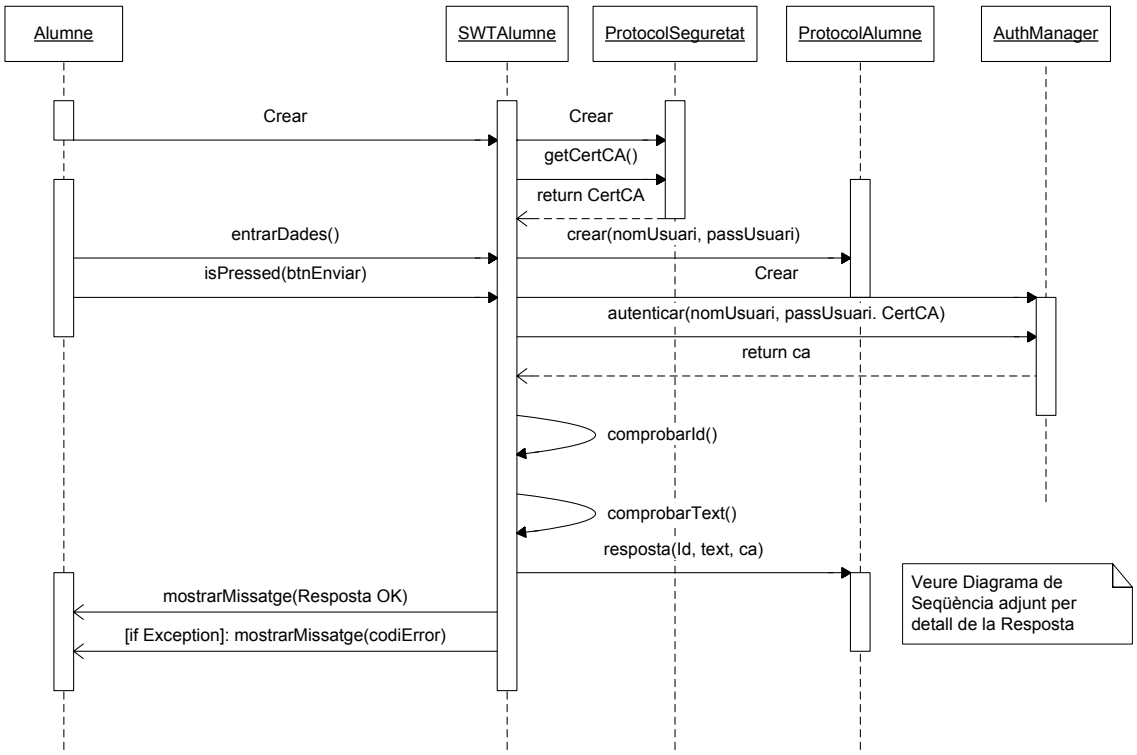
Detall de la crida a "redacció(Id, text, ca)" en la fase de redacció de l'examen:



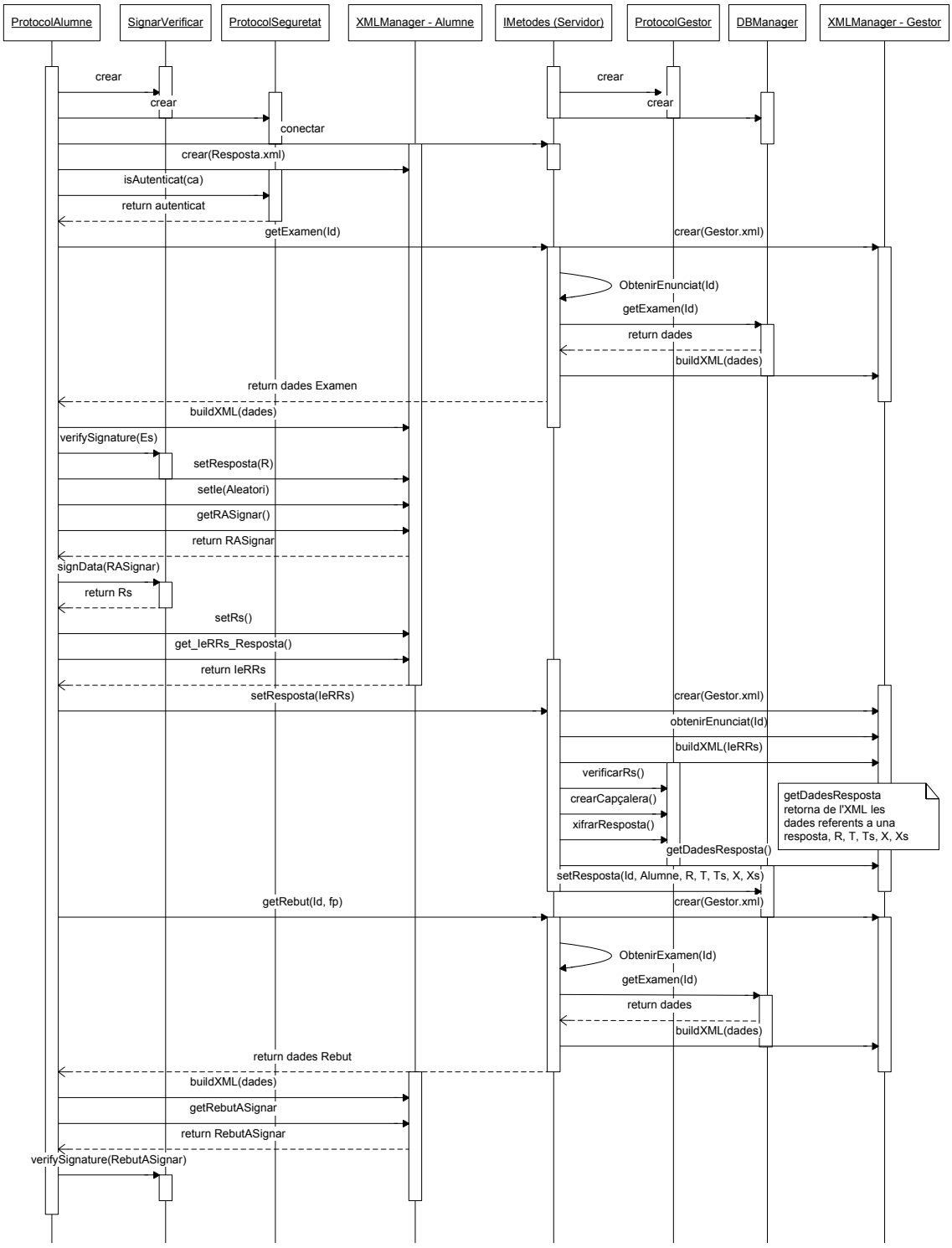
Fase 2(a): Obtenir l'enunciat de l'examen.



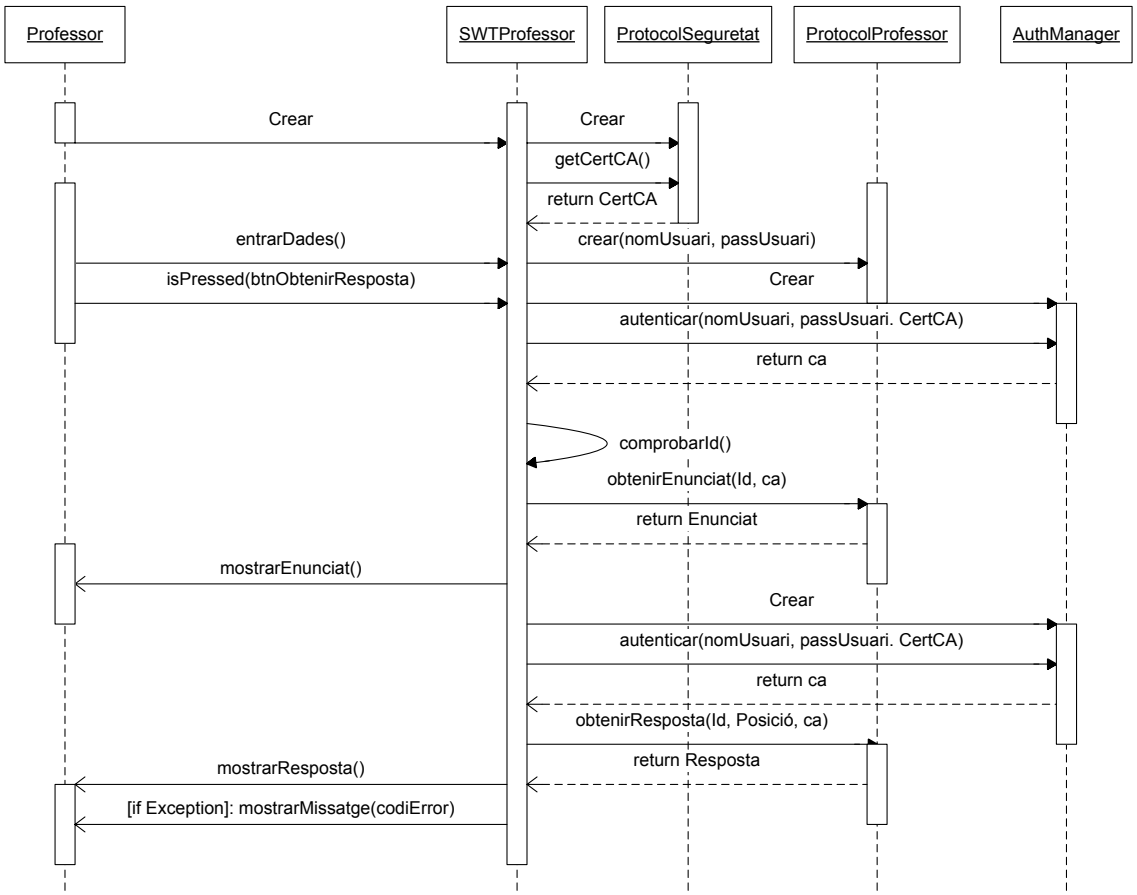
Fase 2(b): Enviar la resposta a l'enunciat.



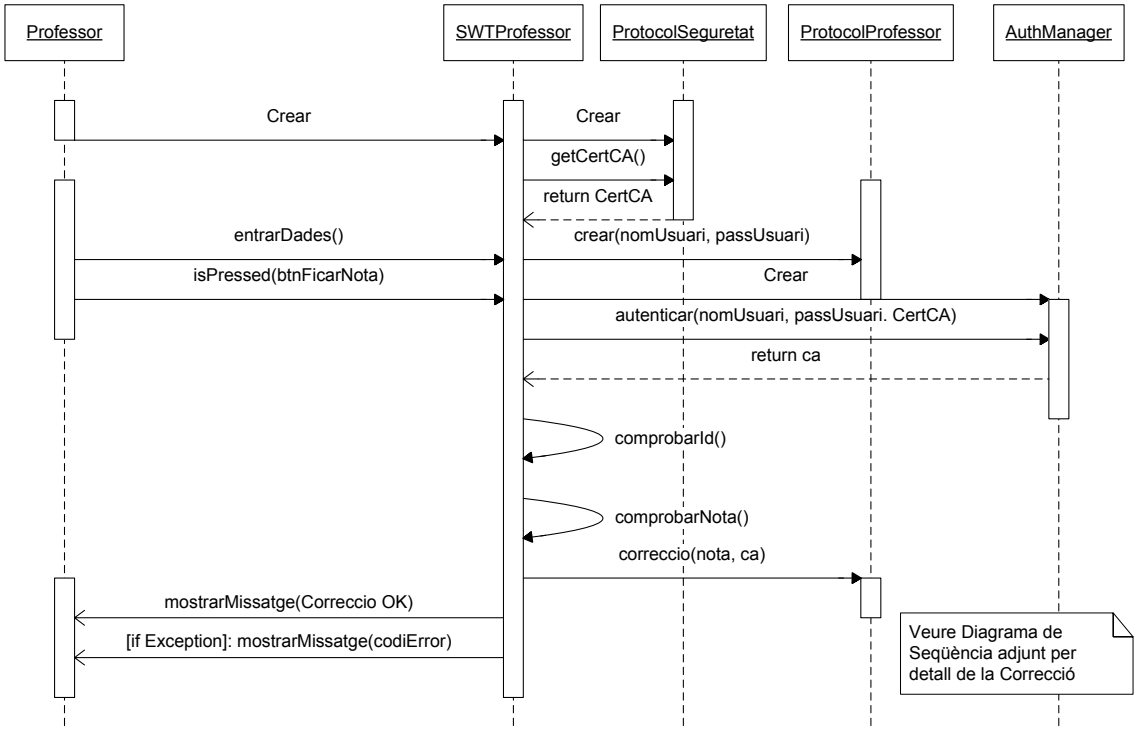
Detall de la crida a “resposta(Id, text, ca)” en la fase de la resposta a un examen:



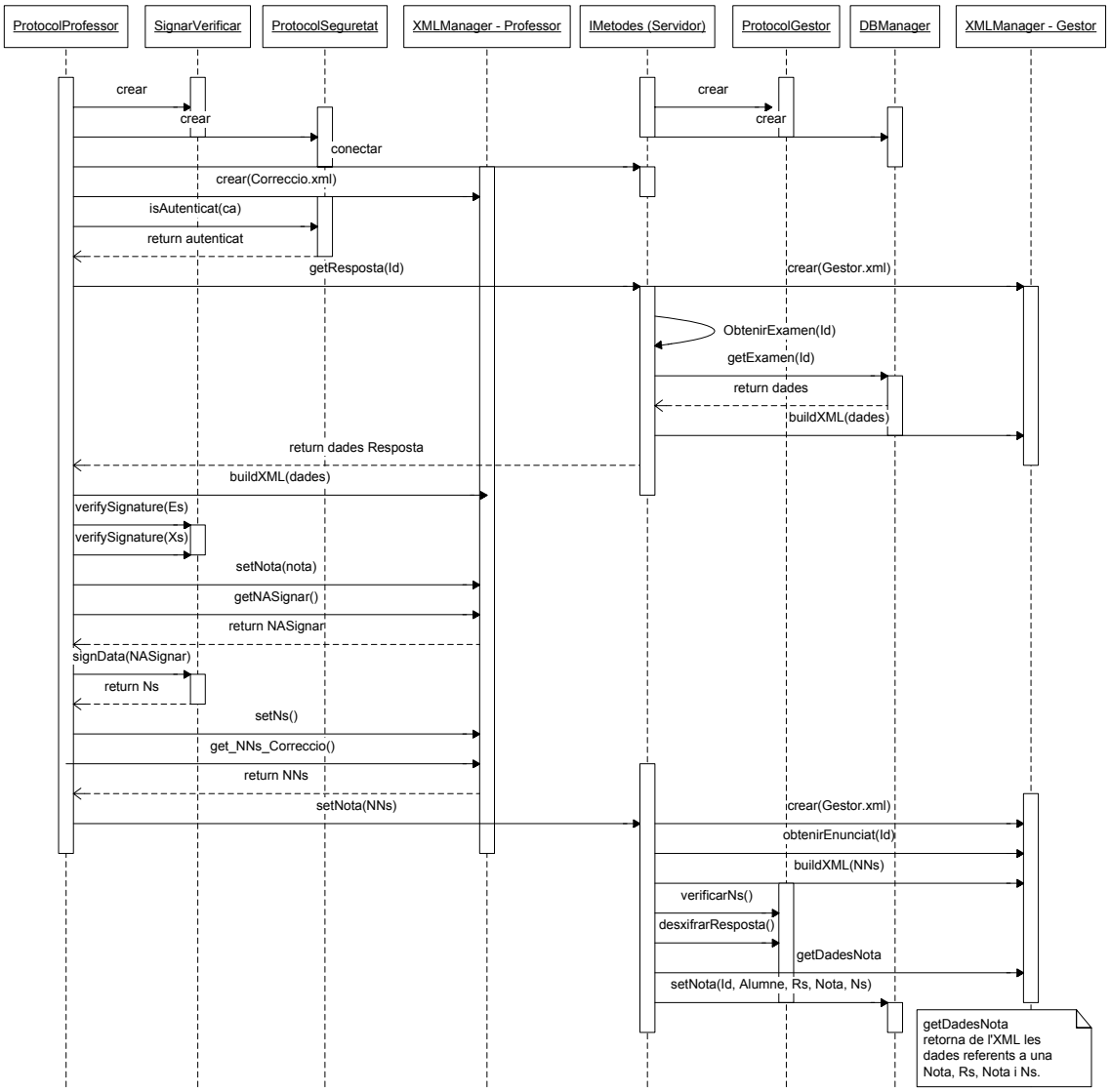
Fase 3(a): Obtenció de la resposta a l'examen.



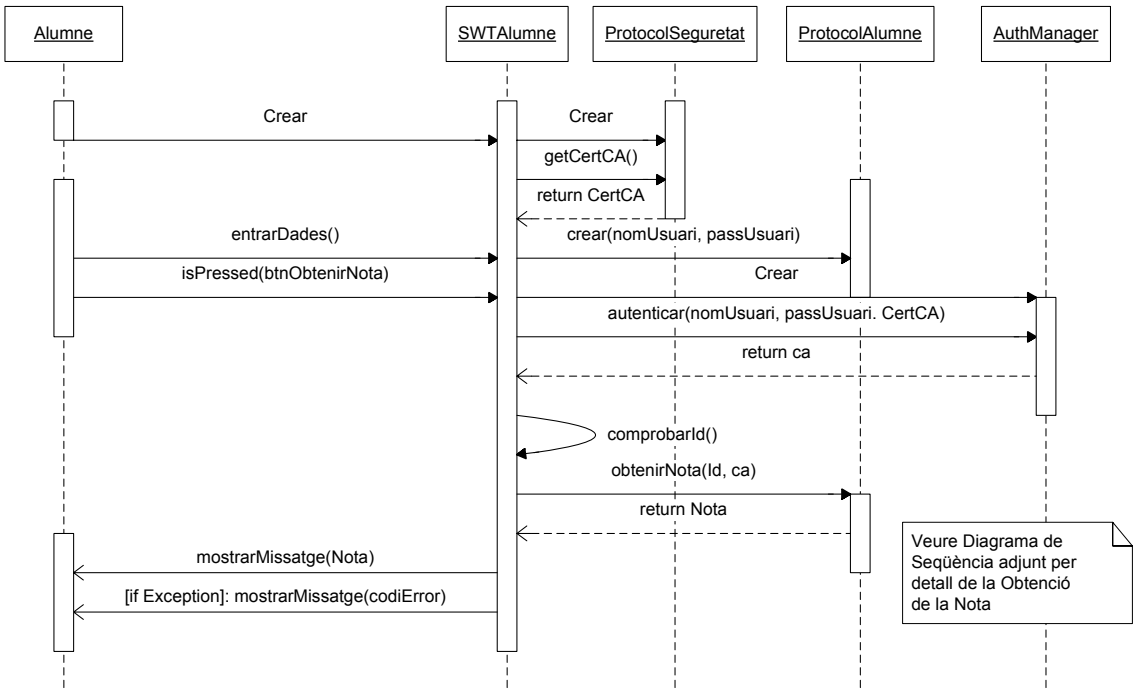
Fase 3(b): Assignar una nota a una resposta.



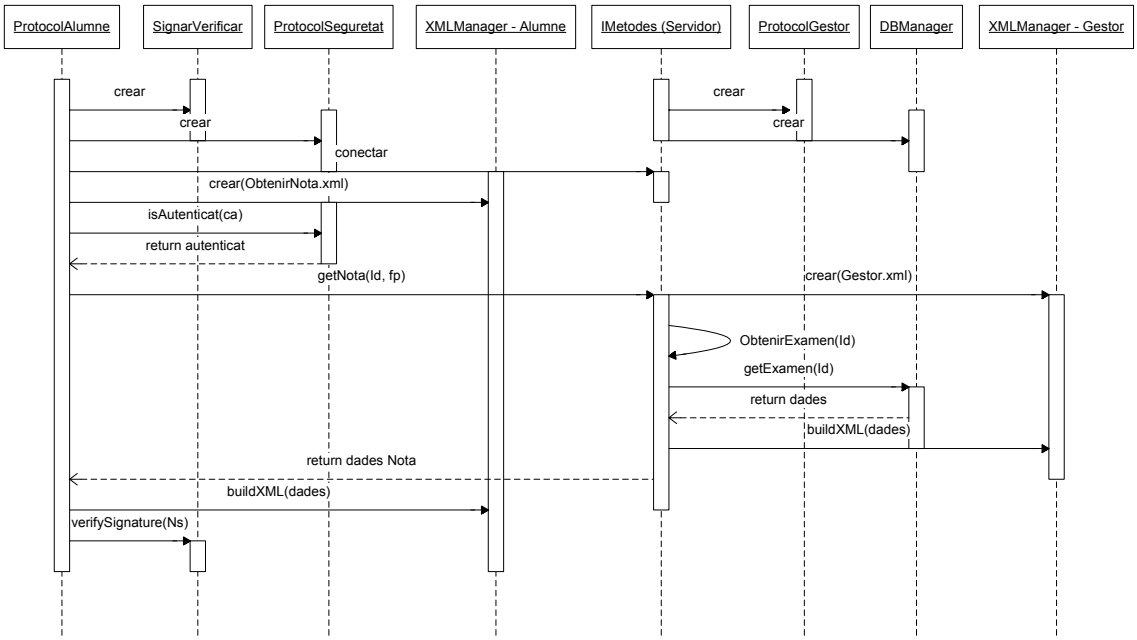
Detall de la crida a “correccio(nota, ca)” en la fase de la correcció d’un examen:



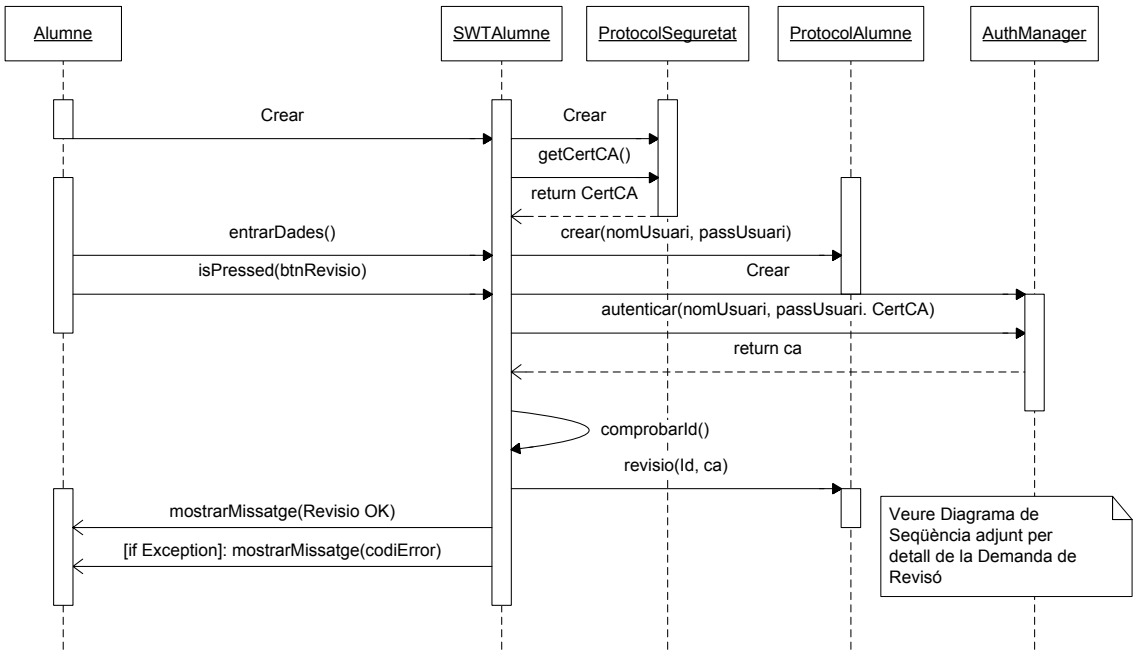
Fase 4: Obtenir la nota d'un examen.



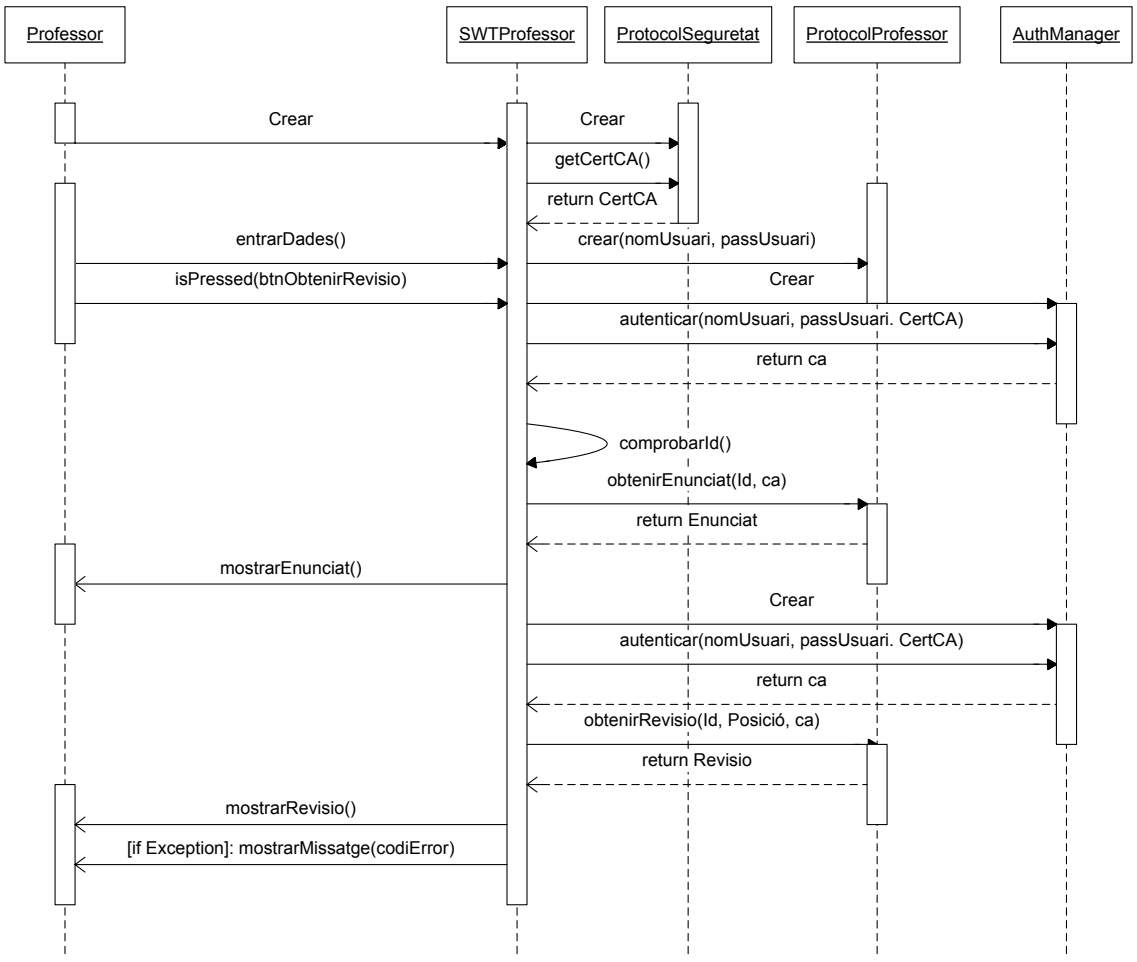
Detall de la crida a "obtenirNota(Id, ca)" en la fase de l'obtenció de la Nota:



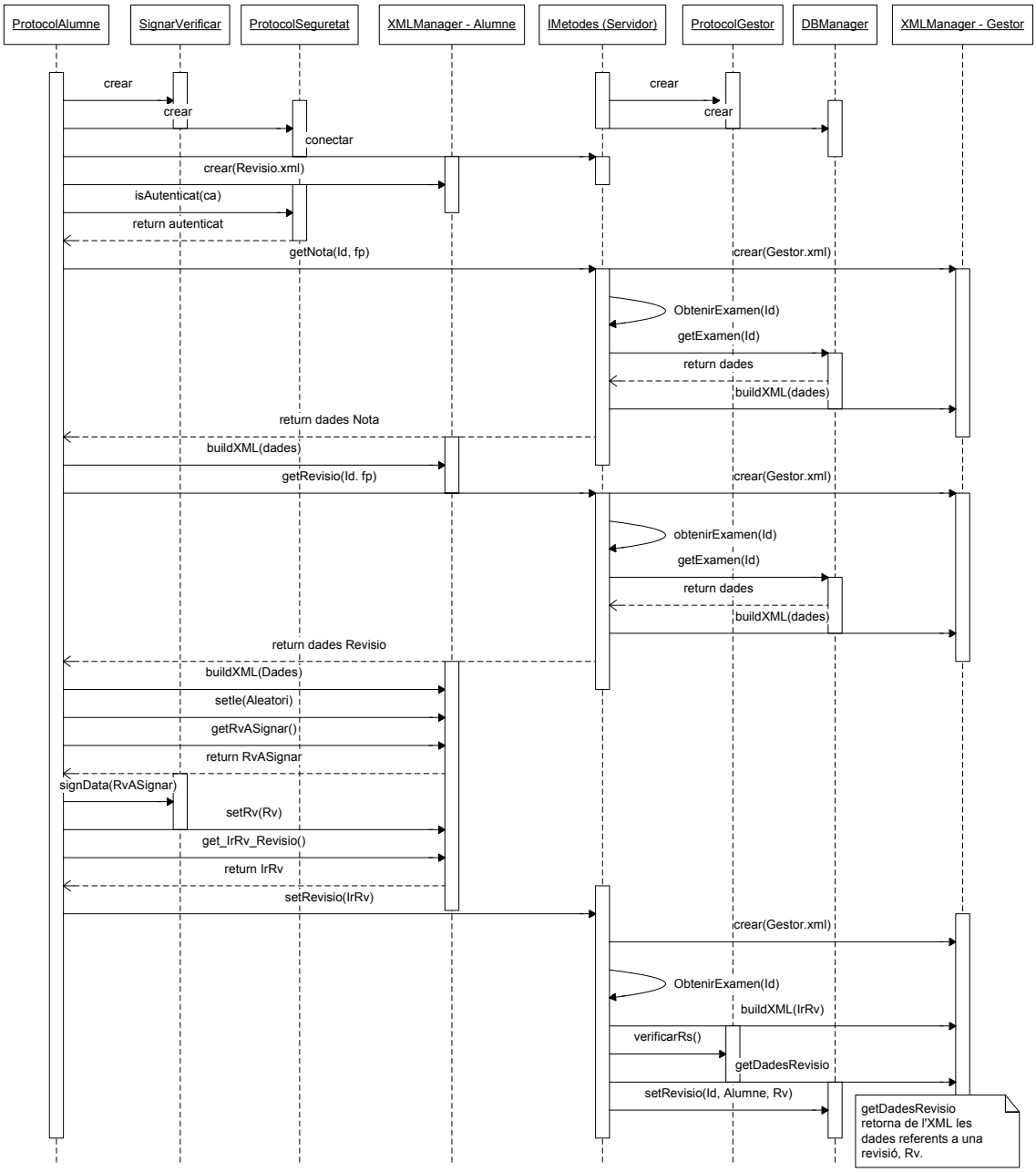
Fase 5(a): Demanar una revisió a un examen corregit.



Fase 5(b): Obtenir les revisions.



Detall de la crida a “revisio(Id, ca)” en la fase de la Revisió d’un Examen:



11. Treball Futur.

11.1 Introducció.

Un projecte de software rares vegades es pot donar per finalitzat, sempre hi algun aspecte que no ha estat cobert o alguna millora que s'hi voldria afegir. Aquest capítol està dedicat a les possibles ampliacions que es podrien implementar en el sistema.

11.2 Millores a implementar.

El que s'ha implementat en aquest projecte hauria de servir de base per a que les Universitats que desitgin implementar el sistema tinguin un punt de partida funcional per a permetre els exàmens remots.

Com a punts importants sobre el treball a realitzar en el futur destaquen:

- Millora de la interfície gràfica adaptant-la a les necessitats de cada Universitat.

- Afegir una interfície per al gestor d'exàmens, ja que ara mateix les dades dels usuaris s'han d'introduir manualment a la base de dades. De la mateixa manera, aquesta interfície hauria de donar accés a la base de dades per a gestionar els exàmens i la resta de dades com a superusuari. Per exemple, ara mateix no hi ha manera d'eliminar un registre de la base de dades, si no és, és clar, utilitzant el SGDB.

- Actualment cada examen i cada resposta són bàsicament una cadena de caràcters i la nota de l'examen s'assigna a un examen en la seva totalitat. Una millora podria ser la possibilitat de tenir, per a cada pregunta de l'examen, una resposta associada i de la mateixa manera, per a cada resposta, una nota respectiva. El resultat final de l'examen, aleshores es podria calcular com a la mitja ponderada de les notes o, fins i tot, podríem afegir la possibilitat de que el professor assignes pesos diferents a cada pregunta i que el sistema calcules la nota final a partir d'aquests pesos. Per dur a terme aquesta modificació hauríem de modificar tant el model XML[10] de l'aplicatiu com el model de la base de dades.

- Permetre la comprovació dels certificats amb el certificat de la CA. Això voldria dir incloure un repositori de CA's de confiança. Aquesta tasca no s'ha acabat implementant en aquest projecte.

- Estudiar la possibilitat d'utilitzar Smart Cards per a realitzar la entrada i la autenticació al sistema. D'aquesta manera l'usuari disposa d'una tarja amb la que no ha d'anar amunt i avall amb els seus arxius de PKCS12[20]. Realment seria còmode d'utilitzar. Cal destacar que l'estructuració en mòduls diferenciats, permetria incloure aquesta possibilitat amb senzilles modificacions.

- Actualment s'ha utilitzat la PKI de lliure distribució Openssl. En un sistema en producció seria necessària la utilització d'una PKI per la gestió dels certificats dels usuaris del sistema.

- Relacionat amb el punt anterior, i per millorar el procediment de la creació dels PKCS12[20] es podria implementar, en la mateixa aplicació, un sistema per crear els arxius de manera automàtica i còmoda. Es podria, fins i tot, mirar de sincronitzar amb el servidor de persones de l'empresa o institució.

- De la mateixa manera que en moltes aplicacions, on existeix la possibilitat de fer-lo anar en diferents idiomes, es podria estudiar la possibilitat de traduir l'aplicatiu a diversos idiomes i que depenent de la màquina host de cada usuari se'n fes servir un o altre, millorant així la comoditat de l'usuari.

- De la mateixa manera que s'ha fet amb la base de dades, on s'ha permès la utilització d'un fitxer de configuració per a localitzar el servidor MySQL[9], es podria facilitar l'accés al servidor amb un fitxer de configuració pel servidor RMI[5]. Així en cas de canvi de màquina no s'hauria de recompilar el codi.

- Es podria afegir una funcionalitat al projecte per a fer extensiu l'ús de l'aplicatiu sense requerir de la instal·lació de programari específic en les màquines dels estudiants. Es podria afegir una implementació del programari dels clients sobre una interfície Web fent que el servidor Web s'encarregués de la gran part de la feina o utilitzar servlets per a distribuir la carrega.

12. Conclusions.

Un cop arribats a aquest punt és el moment de fer una valoració de la feina requerida i de la feina presentada. En aquest capítol es demostra com els requisits que es van demanar a l'enunciat del projecte han estat satisfactòriament assolits.

D'entrada i com a punt més important, es disposa d'un esquema criptogràfic que ens assegura el principal objectiu d'aquest projecte: la possibilitat de realitzar exàmens on-line de manera remota, protegint i assegurant en tot moment la identitat dels usuaris.

Si es repassen les línies principals de l'enunciat es pot veure com, a poc a poc, tots els punt crítics s'han anat assolint.

Punt 1: Creació d'un esquema criptogràfic.

Com ja s'ha comentat, i tal com es citava en els objectius inicials de la memòria s'ha aconseguit que l'esquema criptogràfic compleixi el següent:

- Autenticitat.
- Privacitat.
- Correcció.
- Secret de les respostes durant l'examen.
- Rebut de lliurament.
- Impossibilitat de còpia.

De la mateixa manera es pot comprovar com s'ha aconseguit que el sistema guardi coherència amb el cicle de vida que s'apuntava a l'inici.

- Redacció de l'examen.
- Resposta de l'examen.
- Correcció de l'examen.
- Obtenció del resultat de l'examen.
- Revisió de l'examen.

Punt 2: Representació de les dades XML

El sistema gaudeix d'un sistema de representació de les dades que ens permet de guardar les dades en documents XML[10] per fer més senzilla la portabilitat entre els diferents aplicatius del projecte. Aquest objectiu ha estat assolit per complet, tant per la informació dels exàmens, com per la que circula durant les autenticacions.

Punt 3: Comunicació dels components.

El sistema funciona amb una base per a la comunicació entre les diferents parts muntada sobre l'API RMI[5] que Java[4] facilita. Aquest era un dels objectius primordials, ja que de no funcionar així, s'hagués hagut d'optar per un sistema Web on la gran part de la feina la faria el servidor en comptes d'estar repartida com ara.

Punt 4: Base de dades

El servidor compta amb un sistema gestor de bases de dades en el que es pot emmagatzemar tota la informació dels exàmens i de les respostes, a més de la informació relativa a les persones que poden accedir al sistema. La base de dades s'executa utilitzant MySQL[9].

Punt 5: Protocol d'autenticació.

Finalment, abans de passar a desenvolupar la interfície gràfica s'ha aconseguit implementar el protocol d'autenticació que permet als usuaris de realitzar les accions que sol·liciten si i només si el servidor comprova que són qui diuen ser.

Punt 6: Interfície gràfica.

S'ha intentat fer la interfície gràfica el més simple i intuïtiva possible facilitant la interacció amb el sistema. Cal destacar, que a primer cop d'ull al veure la interfície pot semblar que el programa en si és molt simple i que només es cerca en una base de dades un registre marcat per un identificador i prou. No obstant no és així, la quantitat de feina interna que es realitza per a cada una de les opcions que la interfície permet fer és força important.

Opinió Personal.

La meva opinió al respecte del projecte no podria ser més positiva. Ha estat molt gratificant poder unir totes les eines i formes de treballar que havia utilitzat per separat en tantes assignatures durant la llicenciatura. La veritat és que no ha estat un simple projecte en el que només he tingut que aplicar tot allò que ja sabia, sinó que per a cada punt he tingut que aprofundir en la matèria per descobrir el veritable funcionament i la millor manera d'aplicar-ho sense que la resta del projecte en sortís ressentit, sinó al contrari, beneficiat.

El que més m'ha agradat ha estat el treballar utilitzant un sistema de disseny i implementació incremental, cosa que permet que un cop una part ja funciona es pot gairebé oblidar per centrar-te en la següent que funciona per sobre. La veritat és que normalment m'hagués imaginat que la part donada per bona s'hauria d'anar modificant un cop darrere l'altre per adaptar-se a les noves funcionalitats i la veritat és que no ha estat així, cosa que realment admiro.

De la mateixa manera, l'ajuda constant prestada pel tutor del projecte, Jordi Castellà, ha estat excel·lent i li agraeixo molt. Sovint és complicat trobar consultors que donin tant per a que un projecte arribi a tan bon port.

Si ara tornes a començar el projecte des de zero hi hauria molts detalls que funcionarien d'una altra manera i probablement el resultat seria una mica millor. Això passa sempre, vull pensar, i hi ha moltes coses que fins que no t'hi trobes no penses que succeiran. Suposo que aquesta situació és fa cada cop menys usual a mesura que passen els anys i la experiència t'ajuda a planificar millor els projectes.

Bibliografia

[1] Apunts de Criptografia de la UOC. Versió de l'any 2003.

[2] Apunts d'Enginyeria del Software III de la UOC Versió de l'any 2003.

[3] Jordi Herrera-Joancomartí, Josep Prieto-Blázquez, Jordi Castellà-Roca: A Secure Electronic Examination Protocol using Wireless Networks. ITCC (2) 2004: 263-268.

[4] The J2SE Development Kit (JDK), java.sun.com/downloads

[5] Java Remote Method Invocation (Java RMI), http://java.sun.com/products/jdk/rmi/.

[6] Java 2 Standard Edition Development Kit 5.0, Installation Notes, http://java.sun.com/j2se/1.5.0/install.html.

[7] Eclipse universal tool platform, www.eclipse.org
(download www.eclipse.org/downloads/index.php).

[8] The JDOM XML API, www.jdom.org/docs/apidocs/index.html
(web page www.jdom.org).

[9] The MySQL database server, www.mysql.com/documentation/index.html, www.mysql.com/doc/en/index.html, (download www.mysql.com/downloads/index.html).

[10] Extensible Markup Language (XML), www.w3.org/XML

[11] XML Tutorial www.w3schools.com/xml, www.w3schools.com/dtd

[12] Openssl: The open source toolkit for SSL/TLS, www.openssl.org, www.openssl.org/support/faq.html

[13] The Win32 OpenSSL Installation Project, www.slproweb.com/products/Win32OpenSSL.html

[14] The Unified Modeling Language: UML, www.uml.org

[15] The "Institute for Applied Information Processing and Communication", jce.iaik.tugraz.at/aboutus/index.php
(download jce.iaik.tugraz.at/download/evaluation/index.php#iaik-jce).

[16] A graphical tool to manage your MySQL database anywhere in the world: SQLyog, www.webyog.com/sqlyog/index.php,
(download www.webyog.com/sqlyog/download2.html).

[17] Protocol d'autenticació Needham-Schroeder, dimacs.rutgers.edu/Workshops/Security/program2/boyd/node14.html

[18] Menezes A.J, van Oorschot P.C, Vanstone S.A., Handbook of applied cryptography, CRC Press, ISBN 0-8493-8523-7.

[19] Anderson R., Security Engineering – A guide to building dependable distributed systems, John Wiley & Sons, Inc, ISBN 0-471-38922-6.

[20] PKCS #12: Personal Information Exchange Syntax Standard, http://www.rsasecurity.com/rsalabs/node.asp?id=2138

[21] PKCS #7: Cryptographic Message Syntax Standard, http://www.rsasecurity.com/rsalabs/node.asp?id=2129

[22] RFC 2510 - Internet X.509 Public Key Infrastructure Certificate Management Protocols, http://www.faqs.org/rfcs/rfc2510.html

[23] RFC 2560 - X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP, http://www.faqs.org/rfcs/rfc2560.html

Annexos

Annex A: Glossari de termes.

API: Sigles de Application Programming Interface. Interfície a través de la qual un programa accedeix als serveis del sistema operatiu i d'altres. Una API proveeix un nivell d'abstracció entre l'aplicació i el kernel per tal d'assegurar la portabilitat del codi.

Aplicatiu: En el nostre cas, l'aplicatiu són el conjunt de classes que conformen cadascuna de les eines que els actors del sistema utilitzen.

Arquitectura client – servidor: Sistema a través del qual, un usuari fa peticions que són transmeses al servidor, que les tracta i les reenvia a l'usuari. En general, les màquines del client i del servidor són diferents, però no es descarta que pugui funcionar sobre la mateixa.

Autoritat de Certificació: Entitat que emet certificats digitals d'usuaris o companyies, de manera que aquests es poden identificar davant d'un tercer. És de vital importància que l'Autoritat de Certificació comprovi que la part que demana un certificat és realment qui diu ser.

Base 64: Codificació que empra només 6 bits per caràcter. D'aquesta manera tenim 64 possibles valors. En el nostre cas permet transmetre cadenes que contenen les signatures com a cadena de caràcters sense fer malbé el contingut de les mateixes. Base64 també s'utilitza molt en comunicacions amb dades binàries provinents de text, com ara en el correu electrònic.

Base de Dades: Un o més conjunts estructurats de dades persistents, normalment associades a programaris que les consulten i actualitzen. Una base de dades és un component d'un Sistema Gestor de base de dades.

Cas d'ús: Diagrama pertanyent a les especificacions UML que permet veure gràficament i per a cada actor del sistema, les accions que pot dur a terme i les relacions d'aquestes accions amb el sistema.

Certificat: Arxiu que conté les dades que donen fe de l'autenticitat de la persona o entitat que el presenta.

Clau: Peça d'informació que s'utilitza en criptografia simètrica per a xifrar i desxifrar un missatge. En criptografia asimètrica la clau pot ser pública o privada. La clau pública s'utilitza per xifrar missatges o verificar una signatura. La clau privada s'utilitza per desxifrar o per signar unes dades. La longitud en bits de la clau sovint ens dóna una idea de la robustesa del sistema que emprem.

Document Type Definition: Definició d'un document XML o SGML. Consisteix en unes regles per interpretar els documents i per establir les regles de construcció dels mateixos.

DTD: Veure Document Type Definition.

Entorn de desenvolupament: Conjunt d'eines que permeten el desenvolupament d'un programari de manera integrada, des de la redacció del codi, fins la posterior compilació i l'execució a passos (debugging) per verificar els errors.

Fitxer de configuració: Arxiu accessible per l'usuari amb opció a ser modificat i que permet que el sistema s'adapti a un altre entorn d'execució sense necessitat de

recompilar el codi font. En aquest projecte els fitxers de configuració serveixen per a configurar l'accés a la base de dades.

Fingerprint: Funció de hash que s'aplica sobre un certificat d'un usuari per a obtenir un identificador únic i més còmode d'utilitzar. En el projecte s'utilitzem fingerprints per identificar de manera única als actors del sistema a partir dels seus certificats.

Funció de Hash: Resum amb pèrdua que dona lloc a una seqüència de longitud fixa a partir d'unes dades sense importar la longitud d'aquestes. Les seves propietats permeten la seva utilització alhora de verificar la integritat de les dades. Les funcions de hash també són utilitzades per assignar posicions en temes de cerca i ordenació.

IAIK: Sigles de "Institute for Applied Information Processing and Communication". Aquests són els desenvolupadors de la llibreria criptogràfica amb el mateix nom.

Interfície: Punt d'interacció i/o comunicació entre un ordenador i una altra entitat, ja sigui persona o un altre equip.

Java: Llenguatge de programació multi-plataforma, robust, interpretat, distribuït, orientat a objectes, portable, desenvolupat per Sun Microsystems a mitjans dels 90.

JDOM: Sigles de Java Document Object Model. Solució complerta basada en Java per accedir i modificar documents XML des de codi Java.

Lliure distribució: Se'n diu així del programari que s'ofereix lliurement sense la necessitat de que l'usuari final aboni una quantitat de diners per a la seva utilització. Normalment, amb la distribució s'inclou el codi font al que l'usuari hi te accés a modificar-lo i redistribuir-lo si així ho desitja.

Log: En informàtica, es coneix el log, com la zona del sistema on s'anoten les incidències que van ocorrent. Serveix com a informació i alhora com a guia per a detectar i solucionar problemes d'aplicacions i sistemes.

Longitud de clau: En termes de criptografia indica el nombre de bits de la clau que utilitzem per a xifrar i desxifrar les dades. Les claus simètriques i privades s'han de mantenir en llocs segurs.

Màquina Virtual: Màquina abstracta per a la que existeix un intèrpret. En general s'utilitza en sistemes operatius per a assegurar la portabilitat del aplicatius entre els mateixos. Existeixen màquines virtuals per a moltíssims llenguatges de programació essent Java el més popular avui dia.

MySQL: Sistema Gestor de base de dades de lliure distribució.

Número aleatori: Número generat sense que l'usuari tingui contacte en el procés. Generalment en el nostre cas, i en criptografia, s'utilitza per a poder demostrar que entre sessions un usuari és qui diu ser, ja que només les dues parts implicades coneixen el número generat.

PKCS: Sigles de Public-Key Cryptography Standards. Són un conjunt d'estàndards definits pels laboratoris RSA que especifiquen els estàndards de clau pública.

PKI: Sigles de Public Key Infrastructure corresponent a infrastructura de clau pública.

Port: Camí lògic o extrem d'un canal en un sistema de comunicació. Els protocols TCP i UDP utilitzats en Ethernet utilitzen els ports per a demultiplexar canals lògics en la mateixa interfície de xarxa d'una computadora.

Protocol d'autenticació: Conjunt d'operacions duen a terme dues o més parts de manera que al final almenys una de les part queda autenticada davant de la resta.

Proves d'integració: Conjunt de test que es duen a terme sobre un aplicatiu en fase de desenvolupament per tal d'assegurar el seu correcte funcionament amb d'altres aplicatius o sistemes amb els que interactua.

RMI: Sigles de Remote Method Invocation. API propietària de Java que permet a les aplicacions locals executar codi que es troba allotjat en una altra màquina remota. Aquesta última posa a disposició uns mètodes públics que seran accessibles a través d'una interfície.

Signatura: Document generat a partir d'un missatge i la clau privada d'un usuari. Al xifrar les dades del missatge amb la clau privada es genera un missatge xifrat que una tercera persona pot desxifrar amb la clau pública i comprovar que és el mateix que les dades originals. D'aquesta manera s'assegura que l'origen de les dades no ha estat modificat i que la persona que l'envia és qui diu ser, ja que només ella te accés a la seva clau privada.

Smart Cards: Targeta que disposa d'un xip electrònic segur contra manipulacions.

Sobre digital: Mètode emprat en la criptografia que utilitza els dos tipus de criptosistemes. D'una banda el missatge a transmetre es xifra utilitzant un xifratge de clau simètrica. A continuació, la clau que s'ha emprat per dur a terme aquest xifratge es xifra utilitzant un sistema de xifratge asimètric. El resultat de les dues operacions és el document que s'envia a l'altra part. L'avantatge d'utilitzar aquest sistema és que redueix dràsticament el temps emprat en xifrar el document original ja que el xifratge simètric és molt més ràpid que l'asimètric.

Script: Conjunt d'operacions que s'agrupen en un arxiu que les executa una darrere l'altra per a facilitar l'execució de tasques repetitives. En el nostre cas utilitzem scripts per a la generació dels certificats. Durant el desenvolupament també els hem usat per a arrencar el servidor.

Sistema Gestor de base de dades: Conjunt d'aplicacions que normalment porten control d'un conjunt de dades persistents, oferint alhora facilitat d'accés i consulta als usuaris finals.

SGBD: Veure Sistema Gestor de base de dades.

SWT: Sigles de Standard Widget Toolkit. Llibreria per a la creació d'interfícies gràfiques desenvolupada pel projecte Eclipse, i que és de lliure distribució.

UML: Sigles de Unified Model Language. Llenguatge no propietari d'especificació. És un mètode utilitzat per a especificar, visualitzar, construir i documentar un sistema orientat a objectes en fase de desenvolupament.

www: Xarxa de computadores que contenen llocs d'Internet que ofereixen text i imatges, so, animacions a través del protocol de xarxa HTTP (HyperText Transfer Protocol)

Xifratge simètric: Mètode emprat en la criptografia que utilitza la mateixa clau per a xifrar i desxifrar un missatge.

Xifratge asimètric: Mètode emprat en la criptografia que utilitza dues claus, una pública i una privada, per dur a terme el xifratge i posterior desxifratge d'un missatge. En general, s'utilitza la clau pública de l'usuari destí per xifrar i aquest utilitza la clau privada per desxifrar.

XML: Metallenguatge escrit en SGML que permet a un usuari de crear el seu propi llenguatge de tags. Freqüentment utilitzat per a facilitar l'intercanvi de documents en entorns de xarxa.

Annex B: Fitxer de configuració per a PKI.

```
#
# OpenSSL example configuration file.
# This is mostly being used for generation of certificate requests.
#

# This definition stops the following lines choking if HOME isn't
# defined.
HOME                    = .
RANDFILE                = $ENV::HOME/.rnd

# Extra OBJECT IDENTIFIER info:
#oid_file               = $ENV::HOME/.oid
oid_section             = new_oids

# To use this configuration file with the "-extfile" option of the
# "openssl x509" utility, name here the section containing the
# X.509v3 extensions to use:
# extensions            =
# (Alternatively, use a configuration file that has only
# X.509v3 extensions in its main [= default] section.)

[ new_oids ]

# We can add new OIDs in here for use by 'ca' and 'req'.
# Add a simple OID like this:
# testoid1=1.2.3.4
# Or use config file substitution like this:
# testoid2=${testoid1}.5.6

####################################################################
[ ca ]
default_ca    = CA_default        # The default ca section

####################################################################
[ CA_default ]

dir           = ./CAPFC           # Where everything is kept
certs         = $dir/certs        # Where the issued certs are kept
crl_dir               = $dir/crl          # Where the issued crl are kept
database      = $dir/index.txt    # database index file.
new_certs_dir= $dir/newcerts              # default place for new certs.

certificate   = $dir/CA.crt # The CA certificate
serial        = $dir/serial               # The current serial number
crl           = $dir/crl.pem              # The current CRL
private_key   = $dir/private/CA.key # The private key
RANDFILE      = $dir/private/.rand        # private random number file

x509_extensions     = usr_cert          # The extentions to add to the cert

# Extensions to add to a CRL. Note: Netscape communicator chokes on V2 CRLs
# so this is commented out by default to leave a V1 CRL.
# crl_extensions    = crl_ext

default_days = 365              # how long to certify for
default_crl_days= 30                    # how long before next CRL
default_md      = sha1                    # which md to use.
preserve      = no                # keep passed DN ordering

# A few difference way of specifying how similar the request should look
# For type CA, the listed attributes must be the same, and the optional
# and supplied fields are just that :-)
policy        = policy_match
```

```
# For the CA policy
[ policy_match ]
countryName         = match
stateOrProvinceName = optional
organizationName    = match
organizationalUnitName    = optional
commonName          = supplied
emailAddress        = optional

# For the 'anything' policy
# At this point in time, you must list all acceptable 'object'
# types.
[ policy_anything ]
countryName         = optional
stateOrProvinceName = optional
localityName        = optional
organizationName    = optional
organizationalUnitName    = optional
commonName          = supplied
emailAddress        = optional

####################################################################
[ req ]
default_bits        = 1024
default_keyfile     = privkey.pem
distinguished_name  = req_distinguished_name
attributes          = req_attributes
x509_extensions     = v3_ca      # The extentions to add to the self signed
cert

# Passwords for private keys if not present they will be prompted for
# input_password = secret
# output_password = secret

# This sets a mask for permitted string types. There are several options.
# default: PrintableString, T61String, BMPString.
# pkix  : PrintableString, BMPString.
# utf8only: only UTF8Strings.
# nombstr : PrintableString, T61String (no BMPStrings or UTF8Strings).
# MASK:XXXX a literal mask value.
# WARNING: current versions of Netscape crash on BMPStrings or UTF8Strings
# so use this option with caution!
string_mask = nombstr

# req_extensions = v3_req # The extensions to add to a certificate request

[ req_distinguished_name ]
countryName                 = Country Name (2 letter code)
countryName_default             = ES
countryName_min                 = 2
countryName_max                 = 2

stateOrProvinceName         = State or Province Name (full name)
stateOrProvinceName_default     = Catalunya

localityName                = Locality Name (eg, city)
localityName_default            = Barcelona

0.organizationName          = Organization Name (eg, company)
0.organizationName_default= Universitat Oberta de Catalunya

# we can do this but it is not needed normally :-)
#1.organizationName         = Second Organization Name (eg, company)
#1.organizationName_default     = World Wide Web Pty Ltd

organizationalUnitName          = Organizational Unit Name (eg, section)
organizationalUnitName_default  = Consultors
```

```
commonName                = Common Name (eg, YOUR name)
commonName_max                  = 64

emailAddress              = Email Address
emailAddress_max          = 40

# SET-ex3                 = SET extension number 3

[ req_attributes ]
challengePassword         = A challenge password
challengePassword_min           = 4
challengePassword_max           = 20

unstructuredName          = An optional company name

[ usr_cert ]

# These extensions are added when 'ca' signs a request.

# This goes against PKIX guidelines but some CAs do it and some software
# requires this to avoid interpreting an end user certificate as a CA.

basicConstraints=CA:FALSE

# Here are some examples of the usage of nsCertType. If it is omitted
# the certificate can be used for anything *except* object signing.

# This is OK for an SSL server.
# nsCertType              = server

# For an object signing certificate this would be used.
# nsCertType = objsign

# For normal client use this is typical
nsCertType = client, email

# and for everything including object signing:
# nsCertType = client, email, objsign

# This is typical in keyUsage for a client certificate.
keyUsage = nonRepudiation, digitalSignature, keyEncipherment

# This will be displayed in Netscape's comment listbox.
nsComment                 = "Seguretat en Xarxes de Computadors"

# PKIX recommendations harmless if included in all certificates.
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer:always

# This stuff is for subjectAltName and issuerAltname.
# Import the email address.
subjectAltName=email:copy

# Copy subject details
issuerAltName=issuer:copy

#nsCaRevocationUrl        = http://www.domain.dom/ca-crl.pem
#nsBaseUrl
#nsRevocationUrl
#nsRenewalUrl
#nsCaPolicyUrl
#nsSslServerName

[ v3_req ]

# Extensions to add to a certificate request

basicConstraints = CA:FALSE
```

```
keyUsage = nonRepudiation, digitalSignature, keyEncipherment

[ v3_ca ]


# Extensions for a typical CA


# PKIX recommendation.

subjectKeyIdentifier=hash

authorityKeyIdentifier=keyid:always,issuer:always

# This is what PKIX recommends but some broken software chokes on critical
# extensions.
#basicConstraints = critical,CA:true
# So we do this instead.
basicConstraints = CA:true

# Key usage: this is typical for a CA certificate. However since it will
# prevent it being used as an test self-signed certificate it is best
# left out by default.
# keyUsage = cRLSign, keyCertSign

# Some might want this also
# nsCertType = sslCA, emailCA

# Include email address in subject alt name: another PKIX recommendation
# subjectAltName=email:copy
# Copy issuer details
# issuerAltName=issuer:copy

# DER hex encoding of an extension: beware experts only!
# obj=DER:02:03
# Where 'obj' is a standard or added object
# You can even override a supported extension:
# basicConstraints= critical, DER:30:03:01:01:FF

[ crl_ext ]

# CRL extensions.
# Only issuerAltName and authorityKeyIdentifier make any sense in a CRL.

# issuerAltName=issuer:copy
authorityKeyIdentifier=keyid:always,issuer:always
```

Annex C: La classe LogManager.

A mesura que el projecte anava creixent seguir el rastre de comentaris mostrats per consola es feia cada cop més complicat. Es va optar per crear una classe específica per escriure tots aquests comentaris a un fitxer físic en el disc.

Les dades que s'emmagatzemen a cada escriptura són:

- El mètode que escriu al log.
- El moment en el temps.
- El missatge a reportar.

Només s'utilitza un mètode per escriure. A més hi ha dos constructors, un dels quals esborra el fitxer de log. Normalment es va veure interessant que a cada nova execució del servidor es borrés l'històric del log.

Si haguéssim d'afegir al diagrama de classes veuríem que en general, totes les classes accedeixen als serveis que LogManager ofereix. Per evitar crear un diagrama confús hem optat per no incloure-la.

Un exemple d'una part de les dades contingues en el fitxer de Log podria ser el següent. Conté el log de l'execució de la primera fase de l'esquema criptogràfic.

```
Sat Dec 25 21:00:54 CET 2004: SWTProfessor: Fase 1: Redaccio de l'examen.
Sat Dec 25 21:00:54 CET 2004: ProtocolSeguretat: Servidor trobat.
Sat Dec 25 21:00:54 CET 2004: AuthManager: La preparat.
Sat Dec 25 21:00:54 CET 2004: AuthManager: Na preparat.
Sat Dec 25 21:00:54 CET 2004: AuthManager: Ea preparat.
Sat Dec 25 21:00:54 CET 2004: ProtocolSeguretat: init cridat.
Sat Dec 25 21:00:54 CET 2004: DBManager: Connexio a la BD correcta.
Sat Dec 25 21:00:54 CET 2004: DBManager: Persona Existeix a la BD.
Sat Dec 25 21:00:54 CET 2004: DBManager: Certificat de Persona obtingut
correctament de la BD.
Sat Dec 25 21:00:55 CET 2004: IMetodesImpl: Lb preparat.
Sat Dec 25 21:00:55 CET 2004: IMetodesImpl: Nb preparat.
Sat Dec 25 21:00:55 CET 2004: IMetodesImpl: Eb preparat.
Sat Dec 25 21:00:55 CET 2004: DBManager: Connexio a la BD correcta.
Sat Dec 25 21:00:55 CET 2004: DBManager: Certificat del Gestor obtingut
correctament de la BD.
Sat Dec 25 21:00:55 CET 2004: AuthManager: NB Encriptat.
Sat Dec 25 21:00:55 CET 2004: AuthManager: Tot Correcte, protocol finalitzat.
Sat Dec 25 21:00:55 CET 2004: DBManager: Connexio a la BD correcta.
Sat Dec 25 21:00:55 CET 2004: DBManager: Client esta autenticat.
Sat Dec 25 21:00:55 CET 2004: ProtocolProfessor: Examen signat.
Sat Dec 25 21:00:55 CET 2004: ProtocolProfessor: Dades (Id, E, Es) enviades al
servidor.
Sat Dec 25 21:00:55 CET 2004: ProtocolGestor: El gestor procedeix a fer la
verificacio de Es.
Sat Dec 25 21:00:55 CET 2004: ProtocolGestor: Es verificat pel gestor
d'examens.
Sat Dec 25 21:00:55 CET 2004: DBManager: Connexio a la BD correcta.
Sat Dec 25 21:00:55 CET 2004: DBManager: Dades (Id, E, Es) introduides
correctament a la BD.
Sat Dec 25 21:00:55 CET 2004: IMetodesImpl: Dades (Id, E, Es) guardades a la
BD.
Sat Dec 25 21:00:55 CET 2004: DBManager: Connexio a la BD correcta.
Sat Dec 25 21:00:56 CET 2004: SWTProfessor: Fase 1: Acabada Correctament.
```

Annex D: Fitxer de configuració de la base de dades.

```
/*
SQLyog v4.0
Host - localhost : Database - pfc
*********************************************************
Server version 4.1.7-nt
*/

create database if not exists `pfc`;

use `pfc`;

/*
Table structure for autenticats
*/

drop table if exists `autenticats`;
CREATE TABLE `autenticats` (
  `la` text NOT NULL,
  `na` text NOT NULL,
  `nb` text NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*
Table structure for enunciat
*/

drop table if exists `enunciat`;
CREATE TABLE `enunciat` (
  `id` int(11) NOT NULL auto_increment,
  `idProfessor` int(11) NOT NULL default '0',
  `idExamen` text NOT NULL,
  `enunciat` text NOT NULL,
  `Es` text NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*
Table structure for gestor
*/

drop table if exists `gestor`;
CREATE TABLE `gestor` (
  `id` int(11) NOT NULL auto_increment,
  `FingerPrint` text NOT NULL,
  `Certificate` text NOT NULL,
  `SubjectDN` text NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*
Table structure for persona
*/

drop table if exists `persona`;
CREATE TABLE `persona` (
  `id` int(11) NOT NULL auto_increment,
  `FingerPrint` text NOT NULL,
  `Certificate` text NOT NULL,
  `SubjectDN` text NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*
Table structure for resposta
*/
```

```
drop table if exists `resposta`;
CREATE TABLE `resposta` (
  `id` int(11) NOT NULL auto_increment,
  `idAlumne` text NOT NULL,
  `idExamen` text NOT NULL,
  `Resposta` text NOT NULL,
  `Nota` text,
  `Rs` text,
  `T` text,
  `Ts` text,
  `X` text,
  `Xs` text,
  `Ns` text,
  `Rv` text,
  PRIMARY KEY  (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Annex E: Contingut dels arxius adjunts a la memòria.

Juntament amb aquesta memòria s'adjunten uns arxius que contenen tant el codi de l'aplicació, com el codi compilat, etc. Anem a detallar-ho:

Carpeta	Contingut	Descripció
/pki	alumne.p12 alumne2.p12 professor.p12 gestor.p12	Arxius PKCS12[20] amb els Certificats dels actors del sistema. Conté a mes l'estructura de la PKI per a crear més usuaris.
/src	Tots els arxius .java que utilitza l'aplicació.	Codi font del sistema.
/bin	Tots els arxius .class log.txt hosts.txt portada.jpg BDPFC.sql	Codi compilat i arxius necessaris per a la configuració i execució.
/lib	/jar /dll	Dues carpetes que contenen les llibreries, a instal·lar segons s'explica en el capítol de Joc de Proves.
/doc	pfc.mpp pfc.doc pfc.pdf	Arxiu MSProject amb la planificació del projecte i aquest document.

Annex F: Codi del joc de proves de l'esquema criptogràfic.

A continuació s'adjunten alguns fragments del codi de la classe que implementa el joc de proves de la part de l'esquema criptogràfic. S'ha ficat èmfasi en ressaltar cadascuna de les fases per veure com es segueix punt a punt cadascuna de les parts que s'han descrit en el capítol 3.

Fase 1: Redacció de l'examen.

```
IdExamen Id = new IdExamen();

Id.setId("PFC", "10013", "0405", "280904", "12345" );
System.out.println("Id preparat.");

Examen E = new Examen("Aquest podria ser el text de l'examen.");

byte[] Eb64 = b64Manager.toB64(E.getExamen().getBytes());
System.out.println("Text en base64: " + b64Manager.b64toString(Eb64));

byte[] idb64 = b64Manager.toB64(Id.toString().getBytes());
System.out.println("Id en base64: " + b64Manager.b64toString(idb64));

String EASignar = concatena(b64Manager.b64toString(idb64),
b64Manager.b64toString(Eb64));
byte[] Es = SVProfe.signData(EASignar);
System.out.println("Examen signat.");

SVGestor.verifySignature(Es, EASignar.getBytes());
System.out.println("Signatura verificada pel gestor d'examens.");
```

Fase 2: Resposta de l'examen.

```
SVAlumne.verifySignature(Es, EASignar.getBytes());
System.out.println("Signatura de l'examen verificada per l'alumne.");

E.setResposta("Aquesta podria ser la resposta de l'examen.");

byte[] seedIe = new byte[8];
SecureRandom Ieb = SecRandom.getDefault();

int Ie = Ieb.nextInt();

byte[] Ieb64 = b64Manager.toB64((Ie+"").getBytes());
byte[] Respostab64 = b64Manager.toB64(E.getResposta().getBytes());

byte[] RASignar = concatena(concatena(Es, Ieb64), Respostab64);
byte[] Rs = SVAlumne.signData(RASignar);
System.out.println("Resposta signada.");

SVGestor.verifySignature(Rs, RASignar);
System.out.println("Signatura de la resposta verificada pel gestor.");

Date Tm = new Date();
String T = Id.toString() + Ie + Tm.toString();
byte[] RebutExamen = concatena(concatena(Es,
b64Manager.toB64(E.getResposta().getBytes())),
b64Manager.toB64(T.getBytes()));

byte[] Ts = SVGestor.signData(RebutExamen);
System.out.println("Rebut de l'examen signat.");

//En aquest punt hem d'enviar T i Ts a l'aplicatiu de l'alumne.
```

```
Xifrador cipher = new Xifrador("Gestor.p12", "uoc2004");
cipher.encrypt(Rs);

byte[] X = cipher.getEncryptedData();

byte[] XsASignar = concatena(concatena(X, Es),
b64Manager.toB64(E.getResposta()).getBytes());
byte[] Xs = SVGestor.signData(XsASignar);
System.out.println("Xs generat.");

System.out.println("Finalment l'alumne comproba les dades rebudes (Ts i T).");

SVAlumne.verifySignature(Ts, RebutExamen);
System.out.println("Signatura del rebut verificada per l'alumne.");
```

Fase 3: Correcció de l'examen:

```
SVProfe.verifySignature(Es, EASignar.getBytes());
System.out.println("Signatura de Es verificada pel professor.");

SVProfe.verifySignature(Xs, XsASignar);
System.out.println("Signatura de Xs verificada pel professor.");

String N = "A";
System.out.println("Nota N: " + N);

byte[] NsASignar = concatena(concatena(Es, Ts),
b64Manager.toB64(N.getBytes()));

byte[] Ns = SVProfe.signData(NsASignar);
System.out.println("Correcció de l'examen signada.");

SVGestor.verifySignature(Ns, NsASignar);
System.out.println("Signatura de la correcció verificada pel Gestor.");
```

Fase 4: Obtenció de la nota:

```
SVAlumne.verifySignature(Ns, NsASignar);
System.out.println("Signatura de la correcció verificada per l'alumne.");

System.out.println("Nota N: " + N);
```

Fase 5: Revisió de l'examen:

```
byte[] seedIr = new byte[8];
SecureRandom Irb = SecRandom.getDefault();

int Ir = Irb.nextInt();

byte[] Irb64 = b64Manager.toB64((Ir + "").getBytes());
System.out.println("Ir en base64: " + b64Manager.b64toString(Irb64));

byte[] RvASignar = xml.concatena(Irb64, Ts);
byte[] Rv = SVAlumne.signData(RvASignar);
System.out.println("Id de Revisio de l'examen signada.");

SVGestor.verifySignature(Rv, RvASignar);
```

Annex G: Codi del joc de proves de la representació de les dades en XML.

S'inclou a continuació una part del codi per a exemplificar per a cada fase l'ús de la classe XMLManager.

Fase 1: Redacció de l'examen.

```
IdExamen Id = new IdExamen();
Id.setId("PFC", "10013", "0405", "280904", "12345" );

xml.setIdExamen(Id.getAs().getBytes(),Id.getCd().getBytes(),
Id.getQu().getBytes(), Id.getDa().getBytes(), Id.getNs().getBytes());
xml.setEnunciat("Text de l'examen".getBytes());

String EASignar = xml.getEASignar();

byte[] Es = SVProfe.signData(EASignar);
System.out.println("Examen signat.");

xml.setEs(Es);

SVGestor.verifySignature(xml.getEs(), xml.getEASignar().getBytes());
System.out.println("Signatura verificada pel gestor d'examens.");
```

Fase 2: Resposta de l'examen.

```
SVAlumne.verifySignature(xml.getEs(), xml.getEASignar().getBytes());
System.out.println("Signatura de l'examen verificada per l'alumne.");

xml.setResposta("Aquesta podria ser la resposta de l'examen.".getBytes());

byte[] seedIe = new byte[8];
SecureRandom Ie = SecRandom.getDefault();

xml.setIe((Ie.nextInt()+"").getBytes());

String RASignar = xml.getRASignar();

byte[] Rs = SVAlumne.signData(RASignar);
System.out.println("Resposta signada.");

xml.setRs(Rs);

SVGestor.verifySignature(xml.getRs(), xml.getRASignar().getBytes());
System.out.println("Signatura de la resposta verificada pel gestor.");

Date Tm = new Date();
xml.setTm(Tm.toString().getBytes());
byte[] T = xml.getT();

xml.setT(T);
String RebutASignar = xml.getRebutASignar();

byte[] Ts = SVGestor.signData(RebutASignar);
System.out.println("Rebut de l'examen signat.");

xml.setTs(Ts);

Xifrador cipher = new Xifrador("Gestor.p12", "uoc2004");
cipher.encrypt(xml.getRs());
byte[] X = cipher.getEncryptedData();

xml.setX(X);
```

```
String XsASignar = xml.getXASignar();
byte[] Xs = SVGestor.signData(XsASignar);
System.out.println("Xs generat.");

xml.setXs(Xs);

SVAlumne.verifySignature(xml.getTs(), xml.getRebutASignar().getBytes());
System.out.println("Signatura del rebut verificada per l'alumne.");
```

Fase 3: Correcció de l'examen:

```
SVProfe.verifySignature(xml.getEs(), xml.getEASignar().getBytes());
System.out.println("Signatura de Es verificada pel professor.");

SVProfe.verifySignature(xml.getXs(), xml.getXASignar().getBytes());
System.out.println("Signatura de Xs verificada pel professor.");

String N = "A";
xml.setNota(N.getBytes());

String NsASignar = xml.getNASignar();
byte[] Ns = SVProfe.signData(NsASignar);
System.out.println("Correcció de l'examen signada.");

xml.setNs(Ns);

SVGestor.verifySignature(xml.getNs(), xml.getNASignar().getBytes());
System.out.println("Signatura de la correcció verificada pel Gestor.");
```

Fase 4: Obtenció de la nota:

```
SVAlumne.verifySignature(xml.getNs(), xml.getNASignar().getBytes());
System.out.println("Signatura de la correcció verificada per l'alumne.");
System.out.println("Nota N: " + xml.getNotaStr());
```

Fase 5: Revisió de l'examen:

```
byte[] seedIr = new byte[8];
SecureRandom Ir = SecRandom.getDefault();

xml.setIr((Ir.nextInt()+"").getBytes());

String RvASignar = xml.getRvASignar();
byte[] Rv = SVAlumne.signData(RvASignar);
System.out.println("Id de Revisio de l'examen signada.");

xml.setRv(Rv);

SVGestor.verifySignature(xml.getRv(), xml.getRvASignar().getBytes());
System.out.println("Signatura de l'Id de Revisio verificada pel Gestor.");
```

Es pot comprovar com ja no s'utilitza la classe Examen i que totes les dades ja s'emmagatzemen directament a la instància de la classe XMLManager anomenada xml. Cal fer notar que els documents XML[10] no són escrits físicament a disc, és a dir, l'estructura és sempre lògica, de manera que quan un mètode mor, les dades que estan contingudes en els documents XML[10] desapareixen. D'aquesta manera es força a que les dades acabin sempre a la base de dades si es necessita que aquestes dades siguin persistents.

Annex H: Missatges d'error dels aplicatius.

```
#1: "No has emplenat tots els camps de l'identificador. L'operació no es
realitzarà."

#2: "Impossible conectar amb el Servidor Gestor d'Exàmens. Aquest error pot
ser degut a que el servidor no està funcionant o bé que la base de dades no
respon. Si us plau, comprova la connexió."

#3: "Hi ha hagut un error en l'autenticació amb el Gestor d'Exàmens, no es pot
continuar."

#4: "Hi ha hagut un error amb les dades subministrades, no es pot continuar"

#5: "Hi ha hagut un error en el Servidor Gestor d'Exàmens, es possible que la
tasca no hagi acabat correctament."

#6: "L'usuari que intenta accedir al Servidor Gestor d'Exàmens no està
autenticat."

#7: "No s'ha pogut instanciar correctament la classe per a signar i verificar
Exàmens, comproba que el nom d'usuari i la contrasenya són correctes."

#8: "Hi ha hagut un error a l'intentar signar."

#9: "Hi ha hagut un error a l'intentar verificar una signatura."

#10: "El camp de l'enunciat de l'examen no pot estar buit. L'operació no es
realitzarà."

#11: "L'examen amb aquest identificador ja existeix a la base de dades, no es
pot crear de nou."

#12: "L'examen amb aquest identificador no existeix a la base de dades."

#13: "El camp de la resposta no pot estar buit. L'operació no es realitzarà."

#14: "La resposta que envies ja existeix a la base de dades, no es pot crear
de nou."

#15: "El camp de la nota no pot estar buit. L'operació no es realitzarà."

#16: "No existeix una resposta a la que ficar la nota. L'operació no es
realitzarà."

#17: "No s'ha pogut connectar amb la base de dades. Aquest error és crític."

#18: "La persona amb aquest identificador ja existeix a la base de dades, no
es pot tornar a crear."

#19: "El Gestor d'Exàmens amb aquest identificador ja existeix a la base de
dades, no es pot tornar a crear."

#20: "No hi ha més exàmens a mostrar, es torna a la primera posició."

#21: "No s'ha trobat cap resposta per aquest examen a la base de dades."

#22: "No s'ha trobat cap demanda de revisió per aquest examen a la base de
dades."

#23: "El protocol no està sincronitzat amb el Servidor. Si us plau intenta
repetir l'operació."

#24: "Abans de poder assignar una nota has de tenir un examen seleccionat."
```

```
#25: "Encara no s'ha assignat una nota a aquest examen, no es pot demanar una
revisió."

#26: "Ja s'ha demanat una revisió a aquest examen, no se'n pot demanar una
altra."
```