Memòria del TFC:

“Autoritat de certificació PKI amb serveis en línia”

Alumne: **Andrés Lara Moya**
ETIS

Consultor: **Carlos Ares Angulo**

Data: 20/01/2012

Índex de contingut

Capítol 1: Introducció

1.1. Justificació i motius per l'elecció

No tinc experiència professional en el camp de la Seguretat Informàtica, els meus coneixements de Seguretat Informàtica son els que he obtingut amb les assignatures dels estudis de ETIS i amb la realització del TFC els dos semestres anteriors, però sempre m'ha interessat aquesta especialitat, per això crec que escollir aquesta rama pel TFC es una bona oportunitat de comprendre aquest mon i assolir nous coneixement.

He decidit agafar la proposta principal de treball del Pla Docent de l'ultim semestre: “Autoritat de certificació PKI amb servei en línia”. Crec que es un treball molt interessant i bàsic per conèixer les infraestructures de seguretat més habituals avui en dia a Internet.

Com aprofito part del treball fet al darrer semestre, no m'he limitat a fer la proposta bàsica sinó que he afegit algunes millores o extensions.

1.2. Objectius

L'objectiu principal d'aquest TFC és:

- Crear una Autoritat de Certificació (CA), component principal d'una infraestructura de clau publica (PKI), que emeti i gestioni els certificats digitals necessaris per crear tot un sistema de seguretat en les comunicacions a Internet.

  Aquest sistema ens permet una comunicació segura en un entorn insegur com es Internet, fent possible assolir els següents objectius:

  a) **Autenticació**: només els usuaris autoritzats tenen accés a les comunicacions. La identitat dels usuaris a una PKI s'assegura mitjançant la utilització de certificats digitals.

  b) **Confidencialitat**: la comunicació entre dues persones no pot ser llegida (interpretada) per un tercer que la intercepta. La privacitat de la comunicació en una PKI s'assoleix mitjançant el xifratge.

  c) **Integritat**: les dades que s'envien son iguals a las que es reben, no han sigut alterades durant la seva transmissió. La PKI utilitza algorismes *hash* per assegurar la integritat de les dades.

  d) **No repudi**: els generadors d'una informació no poden negar la seva autoria. Per assegurar l'autoria d'un missatge la PKI utilitza la signatura digital.

Per arribar a aquest objectiu principal s'hauran d'assolir els objectius específics següents:

- Creació d'una autoritat de certificació (CA) amb tots els mecanismes necessaris per emetre certificats i porta un control de certificats anul·lats (llistes de revocació de certificats o CRL).
- Creació d'un sistema d'autenticació d'usuaris amb certificat digital.
- Verificació de certificats emesos per la mateixa CA.
- Comprovar que el certificat no esta anul·lat mitjançant l'accés a la CRL.

- Xifrat SSL de la connexió client-servidor.

Com a objectius secundaris tenim:

- Instal·lació i configuració d'un servidor web per allotjar l'aplicació.
- Instal·lació i configuració d'un servidor de bases de dades per emmagatzemar dades del certificats.
- Creació d'una interfície web per la CA.

I finalment com a objectius personals tinc:

- Assolir un millors coneixements dels sistemes de seguretat que ens envolten quan fem servir Internet. Més específicament tota la infraestructura i el funcionament que hi ha darrere els certificats digitals (autenticació, signatura, ...).
- Millorar el meu nivell de coneixement del llenguatge Java.

1.3. Descripció del producte que es vol obtenir

Aquest Treball de Fi de Carrera té com ha meta bàsica fer una aplicació que actuï com a autoritat de certificació (CA) en una infraestructura de clau publica (PKI).

Aquesta CA estarà instal·lada a un servidor web i disposarà d'una interfície web per interactuar amb ella. La aplicació de la CA serà totalment independent de la interfície web, això ens permetrà integrar l'aplicació de la CA amb qualsevol altre tipus d'interfície.

La proposta bàsica de la CA admet les següents operacions:

- **Operacions administratives**.

  Aquestes operacions només les pot fer un usuari amb drets d'administrador, aquests drets es poden verificar en dues maneres:

  1) Un usuari remot que s'autentica mitjançant un certificat digital emes per la pròpia CA i que tingui assignats drets d'administrador.
  2) Un usuari local (al mateix servidor) que s'autentica mitjançant un nom d'usuari i una contrasenya.

  Aquests usuaris poden fer les següents operacions:

  a) Inicialització de la CA: esborrat de totes les dades de la CA, això inclou claus, certificat, repositori de certificats, repositori de CRL, ...
  b) Posta en marxa de la CA: creació de les claus privada i publica de la CA, i emissió del certificat autosignat per la mateixa CA. Aquest certificat després s'haurà de posar a un lloc d'accés public de la web.
  c) Consulta de l'estat dels certificats: visualització de certificats emesos per la CA a partir de cerques basades en els camps identificatius.
  d) Revocació de certificats: possibilitat d'anular certificats abans de la seva data de caducitat.
  e) Publicació de la CRL: posar la CRL actualitzada a un lloc public de la web.

- **Operacions públiques**.

  Un usuari qualsevol es podrà connectar al servidor segur de la CA per fer una petició de certificació. Aquesta petició es podran formalitzar de tres maneres:

  - Enviant peticions de certificació en fitxers PKCS#10.
  - Enviant certificats autosignats pel propi usuari.
  - Omplint un formulari per que la mateixa CA generi el parell de claus i el certificat de l'usuari. El certificat i la clau privada després s'encapsulen en un fitxer PKCS#12 i es retornen a l'usuari.

Extensions sobre la proposta bàsica:

a) Renovació de certificats: tornar a activar certificats a punt de caducar.

b) Reactivació de certificats revocats: l'usuari autenticat podrà demanar la reactivació d'un certificat revocat, però sempre depenent de l'aprovació d'un administrador de la CA que verifiqui que es compleixen tots els requisits necessaris.

c) Donar d'alta o baixa usuaris com administradors de la CA. Hi ha dos nivells administrador:

   - Usuari administrador normal: només pot fer operacions de gestió de la CA (posta en marxa, consulta i gestió de certificats, publicació de la CRL).
   - Usuari administrador avançat (superadministrador): pot fer totes les operacions d'administrador normal, però també pot inicialitzar la CA i gestionar els usuaris administradors.

d) Donat que donem un servei on es transfereixen dades privades (claus, certificats, ...) a través d'un entorn public (Internet), farem servir el protocol SSL per crear una connexió segura entre servidor i client. Opcionalment el client es podrà autenticar mitjançant un certificat d'una altre CA reconeguda i tenir la possibilitat d'aprofitar la informació d'aquest certificat per crear el nou.

1.4. Planificació del treball

1.4.1.Planificació inicial

La planificació del treball es va basar en el Pla Docent de l'assignatura segons les dates de lliurament marcades i el treball proposat en cada un dels lliuraments.

El TFC inicialment es va desglossar en les següents tasques i subtasques:

PAC 1

Pla de treball: Descriu el TFC i els objectius que es volen assolir. S'enumeren les diferents tasques a realitzar i la seva planificació. També es dona una visió general de l'estat actual de les tecnologies implicades en la realització del TFC.

Aquest Pla de treball serveix com a guia per la resta del TFC.

PAC 2

Posada en marxa de la CA: Consisteix en la realització d'una aplicació que pugui

generar el parell de claus, publica i privada, de la CA i el seu propi certificat autosignat. A més hauria de tenir una interfície web bàsica que permeti a un usuari amb drets d'administrador la configuració inicial de la CA. Aquesta tasca la podem subdividir en:

2.1. Servidor web J2EE: Com primera tasca haurem de instal·lar i configurar un servidor web per allotjar l'aplicació.

2.2. Aplicació Java al servidor: Desenvolupament de la part de l'aplicació encarregada de la posta en marxa de la CA. Té els següents passos:

2.2.1. Creació d'un magatzem de claus: Contindrà les claus secretes, claus privades i certificats de la CA.

2.2.2. Generació de claus de la CA: Aplicació per generar el parell de claus de la CA.

2.2.3. Generació del certificat autosignat de la CA: La CA emet el seu propi certificat autosignat amb les claus generades a l'apartat anterior.

2.2.4. Guardar les claus i certificats de la CA al magatzem de claus.

2.3. Autenticació local: Possibilitat d'autenticar a un usuari amb drets d'administrador al mateix servidor mitjançant un nom i una contrasenya.

2.4. Interfície web: Creació de les planes web i l'aplicació bàsica per administrar la CA amb la opció per poder inicialitzar-la.

PAC 3

Creació de certificats dels clients: Consisteix en la generació de certificats d'usuari a partir d'un fitxer PKCS#10 o un certificat autosignat. També existeix la possibilitat de que l'aplicació sigui la que generi el parell de claus i el certificat de l'usuari, i retorni un fitxer PKCS#12. Aquesta funcionalitat requereix afegir a la interfície web unes planes d'accés public on qualsevol usuari pugui demanar la emissió d'un certificat. Aquesta tasca es pot subdividir en:

3.1. Aplicació Java al servidor: Desenvolupament de la part de l'aplicació encarregada de la creació del certificats de clients. Té el següents passos:

3.1.1. Generació d'un parell de claus d'usuari: Si es necessari, a partir de les dades introduïdes per l'usuari es genera un parell de claus.

3.1.2. Creació de certificat: A partir de la clau publica d'usuari, generar el certificat. La clau publica pot ser la generada a l'apartat anterior, la enviada en el fitxer PKCS#10 o la del certificat autosignat de l'usuari.

3.1.3. Creació de fitxer PKCS#12: Si es necessari, encapsular en un fitxer PKCS#12 la clau privada i el certificat.

3.1.4. Base de dades de certificats: Creació d'una base de dades on puguem guardar els camps identificatius de cada un dels certificats emesos.

3.1.5. Cerques de certificats: Aplicació que fa una cerca a la base de dades de certificats en funció dels camps identificatius.

3.1.6. Renovació de certificats: Crear l'aplicació per poder renovar un certificat que esta a punt de caducar. Bàsicament es tornar a generar el mateix

certificat amb un nou període de validesa.

3.2. Interfície web: Ampliar la interfície web amb les planes necessàries per demanar l'emissió o renovació d'un certificat.

PAC 4

Entorn d'administració de la CA: Completem les funcionalitats d'administració de la CA afegint la possibilitat de consultar els certificats emesos, revocar certificats i publicació de la llista de certificats revocats (CRL). També xifrarem les comunicacions servidor-client fent servir el protocol SSL. Aquest proces consta de les següents subtasques:

4.1. Aplicació Java al servidor: Desenvolupament de la part de l'aplicació encarregada de la gestió dels certificats dels clients i dels usuaris administradors. Té el següents passos:

4.1.1. Consulta de certificats emesos: L'aplicació, després d'una cerca a la base de dades, presenta el llistat de certificats que compleixen amb un requeriments.

4.1.2. Revocació de certificats: Fica un certificat a la CRL.

4.1.3. Publicació de CRL: Posa la CRL a un lloc public on tothom pot descarregar-la o visualitzar-la.

4.1.4. Autenticació remota d'administradors: Possibilitat d'autenticar a un usuari remot amb drets d'administrador, mitjançant un certificat digital de la pròpia CA.

4.1.5. Alta i baixa de usuaris amb drets d'administrador: Un administrador principal té drets per poder donar d'alta o de baixa a administradors de la CA.

4.1.6. Reactivació de certificats revocats: Un usuari autenticat pot demanar a l'administrador la reactivació d'un certificat revocat. Si l'administrador no veu cap problema el reactiva donant-li un nou període de validesa i el treu de la CRL.

4.2. Interfície web: Modificació de la interfície web d'administració per poder accedir a les noves funcionalitats.

4.3. Protocol SSL: Fer servir el protocol SSL per xifrar les connexions remotes. Com a opció existeix la possibilitat de que en l'establiment de la comunicació xifrada a més del servidor, s'autentiqui el client mitjançant el certificat d'una CA reconeguda (FNMT, IDCat o DNIe), això permetria fer servir les dades del certificat del client per generar el nou certificat.

La temporització inicial de les tasques estarà marcada per les fites del Pla Docent segons la planificació que s'adjunta a l'Annex I.

1.4.1.Planificació final

La planificació es va tenir que anar adaptant al treball real que s'anava fent i les duracions reals de cada una de les tasques, a més, hi ha hagut endarreriment que també van afectar a la planificació.

El TFC finalment va quedar amb la següent distribució de tasques i subtasques:

PAC 1

Pla de treball: Descriu el TFC i els objectius que es volen assolir. S'enumeren les diferents tasques a realitzar i la seva planificació. També es dona una visió general de l'estat actual de les tecnologies implicades en la realització del TFC.

PAC 2

Posada en marxa de la CA: Consisteix en la realització d'una aplicació que pugui generar el parell de claus, publica i privada, de la CA i el seu propi certificat autosignat. A més hauria de tenir una interfície web bàsica que permeti a un usuari amb drets d'administrador la configuració inicial de la CA. Aquesta tasca la podem subdividir en:

2.1. Preparació de l'entorn de desenvolupament.

2.1.1. Instal·lació d'un entorn integrat de desenvolupament de programari.

2.1.2. Instal·lació d'un servidor web per allotjar l'aplicació.

2.2. Aplicació Java al servidor: Desenvolupament de la part de l'aplicació encarregada de la posta en marxa de la CA.

2.2.1. Creació de la Autoritat de Certificació: Conté el magatzem de claus, claus, certificats i administradors de la CA.

2.2.2. Creació d'un magatzem de claus: Contindrà les claus secretes, claus privades i certificats de la CA.

2.2.3. Generació de claus de la CA: Aplicació per generar el parell de claus de la CA.

2.2.4. Generació del certificat autosignat de la CA: La CA emet el seu propi certificat autosignat amb les claus generades a l'apartat anterior.

2.2.5. Guardar les claus i certificats de la CA al magatzem de claus.

2.3. Autenticació local: Possibilitat d'autenticar a un usuari amb drets d'administrador al mateix servidor mitjançant un nom i una contrasenya.

2.4. Interfície web: Creació de les planes web i l'aplicació bàsica per administrar la CA amb la opció per poder inicialitzar-la.

2.4.1. Plana inicial de la CA.

2.4.2. Plana per autenticació d'administradors.

2.4.3. Plana per l'administració de la CA.

PAC 3

Creació de certificats dels clients: Consisteix en la generació de certificats d'usuari a partir d'un fitxer PKCS#10 o un certificat autosignat. També existeix la possibilitat de que l'aplicació sigui la que generi el parell de claus i el certificat de l'usuari, i retorni un fitxer PKCS#12. Aquesta funcionalitat requereix afegir a la interfície web unes planes d'accés public on qualsevol usuari pugui demanar la emissió d'un certificat. Aquesta tasca es pot subdividir en:

3.1. Preparació de l'entorn de desenvolupament.

3.1.1. Instal·lació d'un servidor de bases de dades per guardar la informació de certificats generats i usuaris administradors.

3.1.2. Instal·lació de suport per JPA.

3.2. Aplicació Java al servidor: Desenvolupament de la part de l'aplicació encarregada de la creació del certificats de clients. Té el següents passos:

3.2.1. Generació d'un parell de claus d'usuari: Si es necessari, a partir de les dades introduïdes per l'usuari es genera un parell de claus.

3.2.2. Extreure la petició de certificació PKCS#10: crear la petició de certificació llegint les dades del fitxer enviat per l'usuari.

3.2.3. Extreure el certificat autosignat: crear un certificat autosignat llegint les dades del fitxer enviat per l'usuari.

3.2.4. Creació de certificat: A partir de la clau publica d'usuari, generar el certificat. La clau publica pot ser la generada a l'apartat anterior, la enviada en el fitxer PKCS#10 o la del certificat autosignat de l'usuari.

3.2.5. Creació de fitxer PKCS#12: Si es necessari, encapsular en un fitxer PKCS#12 la clau privada i el certificat.

3.2.6. Mètodes per la generació del certificats: Creació dels mètodes necessaris per generar i guardar a la base de dades els certificats emesos.

3.3. Interfície web: Ampliar la interfície web amb les planes necessàries per demanar l'emissió o renovació d'un certificat.

3.3.1. Plana de sol·licitud de certificació.

3.3.2. Modificació de la plana per l'autenticació d'administradors.

PAC 4

Entorn d'administració de la CA: Completem les funcionalitats d'administració de la CA afegint la possibilitat de consultar els certificats emesos, revocar certificats i publicació de la llista de certificats revocats (CRL). Aquest proces consta de les següents subtasques:

4.1. Aplicació Java al servidor: Desenvolupament de la part de l'aplicació encarregada de la gestió dels certificats dels clients i dels usuaris administradors. Té el següents passos:

4.1.1. Autenticació remota d'administradors: Possibilitat d'autenticar a un usuari remot amb drets d'administrador, mitjançant un certificat digital de la pròpia CA.

4.1.2. Alta i baixa de usuaris amb drets d'administrador: Un administrador principal té drets per poder donar d'alta o de baixa a administradors de la CA.

4.1.3. Revocació de certificats: Fica un certificat a la CRL.

4.1.4. Consulta de certificats emesos: L'aplicació, després d'una cerca a la base de dades, presenta el llistat de certificats que compleixen amb un requeriments.

4.1.5. Reactivació de certificats revocats: Un usuari autenticat pot demanar a l'administrador la reactivació d'un certificat revocat. Si l'administrador no veu cap problema el reactiva donant-li un nou període de validesa i el treu de la CRL.

4.1.6. Publicació de CRL: Posa la CRL a un lloc public on tothom pot descarregar-la o visualitzar-la.

4.2. Interfície web: Modificació de la interfície web d'administració i usuari per accedir a noves funcionalitats i millores de les existents.

4.3. Protocol SSL: Requeriment no assolit.

La temporització final de les tasques esta reflectida a la planificació que s'adjunta a l'Annex II.

1.5. Descripció i estructura de la memòria

Aquesta memòria consta de:

- Descripció detallada del TFC dividida en cinc capítols:
  - Capítol 1: Introducció del TFC, on s'expliquen els motius per escollir-lo, els objectius que es volen assolir, una descripció del producte que es vol fer i com s'ha plantejat el treball.
  - Capítol 2: Anàlisi inicial del treball. En aquest apartat es descriuen les tecnologies existents relacionades amb el producte que volem fer, quines d'aquestes tecnologies hem escollit i per què, i com s'ha preparat l'entorn de desenvolupament.
  - Capítol 3: Descripció del producte. Es descriu en detall el producte fet, estudiant per separat l'aplicació de la CA que es un producte independent, i la interfície que es una aplicació web que facilita l'accés a les funcions de la CA.
  - Capítol 4: Manual d'usuari del producte. S'explica la instal·lació i funcionament del producte en un entorn real.
  - Capítol 5: Millores del productes. Es proposen millores del producte.
- Conclusions: Valoració del treball fet i els resultats obtinguts.
- Glossari: Descripció del principals termes especialitzats que s'han fet servir a la memòria.
- Recursos: Relació dels principals recursos bibliogràfics i electrònics consultats per la realització del treball.
- Annexos.

Capítol 2: Anàlisi inicial.

2.1. Estat de l'art de les tecnologies implicades

En aquest apartat veurem els conceptes bàsics i una visió generals de les tecnologies implicades en una PKI, després les tecnologies necessàries pel nostre treball de desenvolupament d'una CA i per últim alternatives existents al mercat amb la mateixa finalitat que l'aplicació que vull desenvolupar.

2.1.1 Infraestructura de Clau Pública (PKI)

La Autoritat de Certificació (CA) es el component principal de una Infraestructura de Clau Pública (PKI), per això tenim que saber que es una PKI i quins elements la composen a part de la CA.

PKI (Public Key Infrastructure): és el conjunt de maquinari, programari, persones, polítiques i procediments necessaris per a crear i gestionar certificats digitals basats en la criptografia de clau pública.

En aquesta definició apareixen dos termes que requereixen una explicació més detallada:

a) **Criptografia de clau pública**. Existeixen dues vessants en la criptografia:

- Criptografia simètrica o de clau compartida. S'utilitza la mateixa clau pel xifratge i desxifratge, aleshores aquesta clau l'han de tenir els dos usuaris que es volen transmetre un missatge, l'un per xifrar el missatge i l'altre per desxifrar-lo. Els grans problemes d'aquest sistema son:
  1) La distribució de les claus: Els dos usuaris ha de fer servir un canal segur per comunicar-se la clau.
  2) La gestió de les claus: si tenim un grup d'usuaris, cada parell d'usuaris necessita una clau diferent per comunicar-se entre ells, per tant hi han un total de $n \cdot (n-1)/2$ claus per tot el grup, si el grup creix el nombre de claus creix exponencialment. Cada usuari necessita emmagatzemar de forma segura un gran nombre de claus.
  3) No hi ha signatura digital.
- **Criptografia asimètrica o de clau pública**. Cada usuari te dues claus, una pública a l'abast de tothom que la necessiti i una altre privada que només ell coneix. Quan un usuari vol enviar un missatge el xifra amb la clau pública de l'usuari a qui va dirigit el missatge, només aquest usuari final podrà desxifrar el missatge amb la seva clau privada. Aquest sistema soluciona els problemes de la criptografia de clau compartida:
  1) No hi han claus compartides que s'hagin de comunicar per un canal segur.
  2) En un grup, cada usuari te dues claus, si augmenta el nombre d'usuaris del grup el nombre de claus només creix linealment. Cada usuari només s'ha de preocupar d'emmagatzemar de forma segura la seva clau privada.
  3) Hi ha possibilitat de signatura digital: un usuari que vulgui signar un missatge farà servir la seva clau privada per crear la signatura que s'afegeix al

missatge. Qualsevol persona que rebi el missatge pot confirmar qui l'ha signat verificant la signatura amb la clau publica del signatari. A més té la seguretat de que el missatge no a sigut alterat.

b) **Certificat digital**. Es una estructura de dades que conté informació del propietari de les claus criptogràfiques, la clau pública i una signatura digital dels dos camps anterior per donar-li validesa.

   Els certificats solucionen el gran problema de la criptografia de clau pública: com poden estar segurs de que una determinada clau pública pertany a qui diu que es el seu propietari. El certificat associa la clau pública amb les dades de la identitat del propietari i esta signat per una entitat reconeguda de confiança que assegura que es correcte i que no ha sigut modificat.

La CA es la entitat de confiança que després de verificar la identitat del propietari d'una clau pública, emet un certificat que legitima aquesta identitat i el signa per evita qualsevol modificació posterior.

A més de la CA en una PKI poden trobar altres components:

- Autoritat de registre (RA): es l'entitat encarregada de verificar el lligam entre la clau pública i la identitat del propietari, descarregant d'aquesta tasca a la CA. Es un element opcional de la PKI.
- Entitats finals i subscriptors: són les entitats o usuaris que tenen un parell de claus, pública i privada, i el corresponent certificat d'una CA.
- Usuaris: son els agents que fan servir les claus públiques de entitats finals o subscriptors.
- Repositoris: son les estructures on s'emmagatzemen els certificats emesos per la CA o les llistes de revocació de certificats (CRL). Les CRL son llistes on apareixen tots els certificats que per algun motiu són invàlids abans de la data de caducitat.
- Autoritat de validació (VA): son entitats encarregades de verificar la validesa dels certificats digitals. Es un element opcional de la PKI.
- Autoritat de segellat de temps (TSA): es una entitat encarregada de signat un missatge posant una marca de temps que permeti provar que el missatge existia fins a un determinat instant. Es un element opcional de la PKI.

Les PKI es troben en un moment on estan fent la transició d'un model propietari a un estàndard obert. Hem deixat enrere un mon on les solucions eren propietàries (IBM, Entrust, ChosenSecurity, Microsoft, ...), però encara no hem arribat a un estàndard universalment acceptat.

Els organismes i empreses que treballant per assolir aquest estàndard son:

- Internet Engineering Task Force (IETF).
- International Standards Organisation (ISO).
- European Telecommunications Standards Institute (ETSI).
- European Committee for Standardisation (CEN).
- RSA Laboratories.

Actualment el model PKI més àmpliament acceptat es l'anomenat PKIX, desenvolupat pel mateix grup PKIX dintre del IETF. Aquest model fa servir els certificats X.509 que avui en dia es troben en la seva versió 3. Jo em basaré en aquest model pel meu treball.

Encara que la PKI agafi com a model general les especificacions de PKIX, hi han detalls particulars que utilitzen les normes PKCS (Public-key cryptography standards). Aquestes normes son un estàndard de fet desenvolupat per l'empresa RSA Laboratories amb la col·laboració de altres fabricants (Apple, Microsoft, Sun, DEC, ...).

2.1.2 Tecnologies implicades en el desenvolupament de la CA

A continuació farem un repas de totes les tecnologies que es poden fer servir pel desenvolupament d'aquest TFC.

- Llenguatges de programació: son llenguatges artificials que permeten expressar totes les operacions que pot fer un ordinador.

  Hi ha un ventall immens de llenguatges de programació, des de els més simples i propers a la màquina (codi màquina o emsamblador), fins als més complexos i propers al llenguatge humà (llenguatges d'alt nivell com Pascal, C, Java i Basic).

  Actualment predominen els llenguatges orientats a objectes (OO), els més popular son Java, C++ o C#:

  - Java va ser desenvolupat per Sun Microsystems als anys 90. Es un llenguatge interpretat que s'executa a sobre d'una màquina virtual estàndard. Actualment es el llenguatge més utilitzat a Internet.
  - C++ es una adaptació del llenguatge C per poder programar amb objectes, per tant es hibrid. Va ser desenvolupat a mitjans dels 80.
  - C# es un llenguatge OO basat en la sintaxis de C/C++. Va ser desenvolupat per Microsoft al 2001 com a part de la seva plataforma .NET.

- Entorn de desenvolupament de programari (IDE): es un programari amb totes les eines necessàries per desenvolupar aplicacions. Hi han que son específics per un llenguatge i altres que admeten molts.

  Els dos més coneguts de codi obert son Eclipse i NetBeans.

  - Eclipse va ser inicialment desenvolupat per IBM a finals dels 90, però des de 2003 està gestionat per la Fundació Eclipse que es una institució sense ànim de lucre. Encara que va ser desenvolupat per Java actualment permet qualsevol llenguatge.
  - NetBeans va ser desenvolupat per un grup d'estudiants a Praga a finals del 90. Aquest projecte després va rebre el suport de Sun MicroSystems. També va néixer com un entorn per desenvolupament amb Java, però mitjançant mòduls pot admetre qualsevol llenguatge.

- Planes web amb contingut dinàmic: són planes web on el contingut presentat pot variar segons les seleccions que fa el client. Podem dir que hi ha una part estàtica (plana HTML) invariable i incrustades a la plana HTML unes dades que son actualitzades pel programari que s'executa al servidor web.

Les tecnologies més esteses fan servir Java donat que es el llenguatge més popular a Internet. Tenim disponible dues tècniques per fer això, els Servlets o el JSP (Java Server Page):

  - Els Servlets son programes Java que estan al servidor i que quan reben una petició del client retornen una plana en codi HTML amb les dades incrustades.
  - El JSP permet incorporar codi Java dintre de planes HTML, quan la plana es cridada el codi Java incrustat es compila en un Servlet que genera la part de codi HTML dinàmic.

- Servidor web: Es un programari encarregat de servir planes web als clients que ho demanen.

  Els més estesos són: Apache Web Server, IIS de Microsoft, Oracle iPlanet Web Server o GWS de Google. Tots ells son servidors molt potents amb moltes funcionalitats, però hi han alternatives més lleugeres com Apache Tomcat o Sun Glassfish.

- Bases de dades: són magatzem d'informació estructurada i indexada per facilitar l'accés. Normalment van associades a un SGBD (Sistema de gestió de base de dades) que gestiona l'estructura i l'accés a les dades.

  Hi han multitud de bases de dades basades en diferents models i estructures. Actualment les més comuns són les bases de dades relacionals que estructuren la informació en taules amb diferents camps i relacions entre aquestes taules.

  El servidors de bases de dades més populars són: Microsoft SQL Server, Oracle, MySQL o PostgreSQL.

  Tots ells tenen suport pel llenguatge SQL que actualment es un estàndard per fer consultes a bases de dades.

- Creació de CA, emissió de certificats i creació de CRL: Per fer totes les operacions de la CA necessitarem biblioteques de funcions criptogràfiques. Donat que no venen per defecte incorporades als llenguatges de programació haurem de incorporar unes biblioteques externes.

  El més popular es l'API de Bouncy Castle que té biblioteques tan per Java, com per a C#.

- Connexió segura: Pel cas de la connexió remota, hem d'incorporar un sistema segur que permeti l'autenticació mitjançant certificats digitals a través d'una connexió segura.

  El protocol més adequat i més àmpliament acceptat per aquesta comunicació servidor-client es el SSL.

2.1.3. Aplicacions existents per la creació i gestió d'una CA

Els productes existents al mercat similars a la meva aplicació i que siguin de codi obert son:

- EJBCA (http://www.ejbca.org/)
- OpenCA (http://www.openca.org/)

- OpenXPKI (http://www.openxpki.org/)
- GnoMint (http://sourceforge.net/projects/gnomint/)
- DogTag (http://pki.fedoraproject.org/)
- XCA (http://xca.sourceforge.net/)
- TinyCA (http://tinyca.sm-zone.net/)
- pyCA (http://www.pyca.de/)

Mentre uns son aplicacions completes per la creació i gestió de CA's (EJBCA, OpenCA, ...), altres son només interfícies pel paquet OpenSSL (TinyCA i pyCA).

El paquet OpenSSL encara que permet fer totes les operacions necessàries per crear i gestionar una CA, no esta desenvolupat amb aquesta finalitat, a més la seva utilització no es molt intuïtiva, per això no l'he considerat una alternativa a la meva aplicació.

2.2. Tecnologies seleccionades

La Infraestructura de Clau Pública estarà basada en el model PKI més àmpliament acceptat, l'anomenat PKIX, desenvolupat pel mateix grup PKIX dintre del IETF. Aquest model fa servir els certificats X.509 que avui en dia es troben en la seva versió 3.

Al meu treball també faré servir algunes de les normes PKCS, com la PKCS#10, pel format de les peticions de certificats, o la PKCS#12, per la transferència de claus privades i certificats dels usuaris.

Pel desenvolupament de l'aplicació tractaré de fer servir les tecnologies de codi obert amb una implantació més amplia en el mon informàtic per diverses raons:

- Aquestes tecnologies han sigut les més utilitzades al llarg de tots els estudis d'ETIS i son les que millor conec.
- Normalment tenen molta mes documentació accessible lliurament, això facilita la recerca i aprenentatge de nous coneixements.
- Permet fer una aplicació oberta que es pot adaptar fàcilment a la majoria de sistemes existents.

Tecnologies seleccionades per l'entorn de desenvolupament:

- Llenguatge de programació: La primera decisió a prendre es quin llenguatge de programació farem servir per desenvolupar l'aplicació. Aquesta decisió ens marcarà la resta de tecnologies o eines que farem servir donat que s'hauran d'adaptar a aquest llenguatge.

  La meva selecció està entre Pascal, C o Java que son el que he fet servir en algun moment durant els meus estudis. C i Java son clarament els mes populars, i aleshores més estesos. Finalment me decidit per agafar Java per tres raons:

  - Java es un llenguatge de programació clarament OO.
  - Es el més popular i amb més difusió en Internet, qualsevol aplicació web porta codi Java en part o en la seva totalitat.
  - Es el llenguatge que més he fet servir durant els estudis d'ETIS, aleshores es

el llenguatge que millor conec.

Com tindrem que treballar en un entorn client-servidor havia pensat fer servir la plataforma Java J2EE de Oracle, però donat el canvi en les llicencies de Oracle per fer servir el SDK de Java EE, que cada vegada es tornen més restrictives, he decidit fer servir una versió de codi obert “OpenJDK” en la seva versió 6 que es compatible amb la “Java EE 6 SDK” de Oracle.

- Entorn de desenvolupament de programari: tindrà que ser un amb suport per llenguatge Java.

  Entre tots els que son de codi obert, els dos més coneguts son Eclipse i NetBeans. He triat Eclipse per dues raons:

  - Es l'IDE que més he utilitzat als meus estudis de ETIS.
  - Es un dels IDE més complerts i flexibles per a Java:
    - Permet la integració al mateix IDE de servidors d'aplicacions, bases de dades, etc. No es necessari sortir de l'IDE per fer-los servir.
    - Es modular. Això permet afegir múltiples extensions per ampliar les seves funcionalitats.
    - Permet adaptar la seva interfície molt fàcilment als diferents entorns de treball (bases de dades, web, programació, ...).

  Concretament faré servir “Eclipse IDE for Java EE Developers” en la seva darrera versió que es la 3.7 anomenada “Indigo”.

- Planes web amb contingut dinàmic: Si el codi Java es relativament petit es mes intuïtiu fer servir JSP, creem una pàgina HTML i després incrustem petits blocs de codi Java per generar la part dinàmica. Fer servir Servlets directament es recomanable quan hi ha una gran quantitat de codi Java, així la pàgina HTML no queda “saturada” de codi Java.

  Descarto l'opció de fer servir un *Framework* per desenvolupar l'aplicació web donat que no tinc experiència en cap d'ells, entenc que son per projectes grans que en aquest cas no queden justificats i requeririen d'un aprenentatge molt més llarg.

  Faré servir HTML i Servlets separats donat que crec que el codi Java de la interfície pot ser voluminós.

  Me basat en les especificacions HTML4 i l'API Servlet 3.0.

- Servidor web: Donat que farem servir llenguatge Java, HTML i Servlets, haurem de triar un servidor que admeti aquestes tecnologies, concretament un servidor web J2EE.

  Tomcat i Jetty son els més lleugers, els altres incorporen gran quantitat de funcionalitats que per al nostre projecte no són necessàries i carregarien l'ordinador.

  M'he decidit per Tomcat donat que sembla ser el més estes, inclús com a part de servidors d'aplicacions més potents, hi ha molta informació sobre la seva configuració i s'integra perfectament a l'IDE Eclipse.

Concretament faré servir Apache Tomcat en la seva versió 7.0.22.

- Base de dades: pel meu treball he cercat bases de dades relacionals amb suport per SQL, donat que son les que tenen una major implantació al mercat i son les que he fet servir més sovint durant els meus estudis. Les dues més conegudes de codi obert són Apache Derby i MySQL.

  He decidit fer servir Derby per que permet una integració molt senzilla amb l'entorn de desenvolupament Eclipse.

  Instal·laré Apache Derby versió 10.8.1.2, més concretament faré servir el plugin existent per Eclipse que permet un control total de la base de dades sense sortir de l'entorn de desenvolupament.

  Per poder tractar objectes de Java com a si fossin dades relacionals he triat les llibreries JPA. JPA només és una especificació però hi han diverses implementacions: TopLink, Hibernate, EclipseLink, OpenJPA i altres. La versió d'Eclipse que faig servir porta integrat suport per la implementació de JPA "EclipseLink". He descarregat i instal·lat la versió 2.3.0

- Creació de CA, emissió de certificats i creació de CRL: Donat que Java no té un suport complert de totes les operacions que s'hauran de fer amb els certificats X.509, faré servir les llibreries Bouncy Castle que sí tenen implementades totes les operacions necessàries.

- Connexió segura: Faré servir el protocol SSL.

2.3. Preparació de l'entorn de desenvolupament

2.3.1. Descripció de la plataforma de treball

El meu entorn de desenvolupament estarà instal·lat a un ordinador de sobretaula amb les següents característiques:

- Processador: Intel Core 2 Quad CPU Q9300 a 2,50 Ghz amb 4 nuclis.
- Memòria: 2,0 GB.
- Targeta gràfica: GeForce 9500 GT.
- Disc Dur: 2 unitats PATA, la primaria amb 500 GB i la secundaria amb 250 GB.
- Lector/gravador de DVD.
- Monitor amb una resolució màxima de 1920 x 1080 píxels.

A l'ordinador tinc instal·lat el sistema operatiu Linux Ubuntu 11.10 i faig servir l'entorn gràfic Gnome 3.

2.3.2 Instal·lació dels diferents components de l'entorn de desenvolupament

El procés per preparar l'entorn de desenvolupament consta dels següents passos:

1) El primer que hem de fer es instal·lar la maquina virtual i les llibreries de la distribució Java.

- Fent servir l'entorn gràfic de Ubuntu només hem de cercar el paquet “openjdk-6-jdk” al *Centro de Software de Ubuntu.*
- També es pot fer obrint un terminal i escrivint la següent comanda:

  $ `sudo apt-get install openjdk-6-jdk`

Encara que no es imprescindible, també he instal·lat els paquets amb la documentació i les fonts que poden ser útils durant el desenvolupament: “openjdk-6-doc” i “openjdk-6-source”.

2) Després instal·lem el IDE Eclipse.
   - Primer descarreguem el paquet “eclipse-jee-indigo-SR1-linux-gtk.tar.gz” de la web del projecte Eclipse.
   - Aquest paquet el descomprimim i crea un directori anomenat “eclipse”. Dintre d'aquest directori trobem el fitxer executable “eclipse” que engega el IDE.
3) Instal·lació del servidor web i configuració per fer-lo servir des de Eclipse.
   - Descarreguem el paquet “apache-tomcat-7.0.22.zip” de la web del projecte Apache Tomcat.
   - Descomprimim el paquet.
   - Engeguem Eclipse i accedim a les opcions: “File” --> “New” --> “Other”. A la finestra que s'obre cerquem el grup “Server” i dintre agafem l'opció “Server”. S'obre una nova finestra per la definició del servidor, seleccionem el grup “Apache” i el servidor “Tomcat v7.0 Server”. Els camps de nom del host i del servidor es poden deixar per defecte o canviar-los. Al camp “Server Runtime Enviroment” tenim que seleccionar el directori on està el nostre servidor Tomcat.
   - A la part baixa d'Eclipse ara tindrem una solapa anomenada “Servers” on tindrem el nostre servidor. Des d'aquí podrem engegar o aturar el servidor, i carregar al servidor els projectes web que vulguem executar.
4) Instal·lació del servidor de bases de dades i configuració per fer-lo servir des de Eclipse.
   - Descarreguem els paquets “derby_core_plugin_10.8.1.zip” i “derby_ui_doc_plugin_1.1.3.zip” de la web del projecte Apache Derby.
   - Descomprimim els dos paquets i fiquem les 3 carpetes que hi ha a dins (“org.apache.derby.core_10.8.1”, “org.apache.derby.ui_1.1.3” i “org.apache.derby.plugin.doc_1.1.3”) a el directori “plugins” de Eclipse.
   - Per afegir la base de dades a un projecte de Eclipse només hem de pitjat el botó dret del ratolí a sobre del projecte que volem i seleccionar les opcions: “Apache Derby” --> “Add Apache Derby nature”. Això afegeix les llibreries necessàries al projecte i també afegeix a la opció “Apache Derby” noves funcions per engegar o aturar el servidor de la base de dades i accedir a les eines de la base de dades.
5) Instal·lació de les llibreries necessàries: aquesta instal·lació es pot fer amb l'opció: “Build Path” --> “Configure Build Path” --> solapa “Libraries”, del menú contextual del nostre projecte a Eclipse.

- EclipseLink (JPA): Per fer la persistència d'objectes. Aquestes llibreries es poden descarregar directament des de Eclipse amb l'opció: “Help” --> “Install new software”. També es poden descarregar de la web del projecte EclipseLink i incloure-les al nostre projecte.
- Bouncy Castle: per treballar amb certificats. Hem de incloure al nostre projecte les llibreries “bcprov-jdk16-146.jar” i “bcmail-jdk16-146.jar” que es poden descarregar de la web del projecte Bouncy Castle.
- Servlets: Per poder treballar amb servlets. Aquesta llibreria es el fitxer “servlet-api.jar” que està dintre del directori “lib” de la nostre instal·lació del servidor Tomcat.
- FileUpload: Per poder pujar fitxers al servidor i treballar amb ells. També es una de les llibreries del servidor Tomcat, el fitxer “tomcat-coyote.jar”.

Capítol 3: Descripció del producte

3.1. Nucli de la CA

Aquesta es la part de la aplicació encarregada de fer totes les funcions de la CA. Consta de les classes: *AutoritatCertificacio*, *MagatzemClaus*, *ParellClaus*, *CertificatCa*, *Administrador*, *Certificat*, *FitxerPkcs12*, *PeticioCertificacio* i *LlistatCrl*.

Es una aplicació totalment independent de la interfície que s'utilitzi per accedir a les seves funcions.

He decidit treballa sempre amb fitxer amb format PEM (codificat en Base64) que són el format per defecte de *openssl*.

3.1.1. Classe AutoritatCertificacio

Es la entitat principal de la CA i té totes les funcions necessàries pel seu funcionament.

Hi han diverses constants per els valors per defecte de noms, contrasenyes o alies a fer servir en la creació del magatzem, administradors i certificats de la CA.

Com a atributs té un magatzem per les claus de la CA, l'administrador actiu, la clau privada i pública, el certificat i la CRL.

En la creació de la CA només es genera un usuari administrador amb el nom i contrasenya per defecte, la resta d'atributs de la classe tenen valor nul. Aquest administrador per defecte es l'encarregat de posar en marxa la CA.

Té mètodes implementats per:

a) Inicialització de la CA (*inicialitzacio*):

   Esborra els atributs de la CA, el fitxer del magatzem, el fitxer de la CRL i les bases de dades de certificats i administradors, després crea un usuari administrador per defecte.

b) Posta en marxa de la CA (*postaMarxa*):

   Com a paràmetres podem passar el nom del fitxer del magatzem, la contrasenya de protecció i la *url* per la CRL. Si existeix un magatzem de claus llegeix el certificat i la clau privada de la CA, si no existeix un magatzem, el crea, crea un parell de claus noves i el certificat corresponent, i guarda tot al magatzem. Després crea la CRL.

c) Llistat de certificats(*llistatCertificats*):

   Com a paràmetres passem els valors de diversos camps dels certificats per fer una cerca a la base de dades i tornar un llistat amb tots els que compleixen. Si el paràmetre té valor *null*, *aquest camp no influirà a la cerca* i es presentaran tots el valors possibles.

d) Revocació d'un certificat (*revocacioCertificat*):

   Cerca el certificat a la base de dades pel nombre de serie i, si no està ja revocat, el revoca. Com a data de revocació s'agafa l'actual, a més és guarden les raons per

la revocació i la data d'invalidesa en la que es creu va quedar compromès el certificat. Torna un valor booleà indicant si s'ha pogut fer la revocació.

e) Reactivació d'un certificat (*reactivacioCertificat*):

Cerca el certificat a la base de dades pel nombre de serie i si està revocat el reactiva, però les dades de revocació no s'esborren i no es treu el certificat de la CRL. La practica habitual es que un certificat revocat no es pot treure de la llista de revocació, el que es fa es canviar les raons de la revocació a *removeFromCRL* que indica a tot el que consulti la llista que el certificat torna a ser vàlid fins la seva data de caducitat. Torna un valor booleà indicant si s'ha pogut fer la reactivació.

f) Publicar la CRL (*publicarCrl*):

Torna a crear la llista de certificat revocats fent una cerca a la base de dades i la publica creant un fitxer a la *url* de la web que passem com a paràmetre.

g) Llistat d'administradors (*llistatAdministrador*):

Torna un llistat de tots els administradors de la CA existents a la base de dades.

h) Afegir un administrador (*afegirAdministrador*):

Afegeix un usuari administrador nou a la base de dades amb les dades passades com a paràmetre. El nom d'usuari i contrasenya sempre son necessaris, el certificat associat a un administrador es opcional. Torna un nombre sencer positiu indicant si hi ha hagut cap error o zero si tot a sigut correcte.

i) Esborrar un administrador (*esborrarAdministrador*):

Cerca i esborra un usuari administrador de la base de dades. Com a paràmetres agafa el nom d'usuari i contrasenya, ambdós obligatoris. Torna un nombre sencer positiu indicant si hi ha hagut cap error o zero si tot a sigut correcte.

j) Crear un certificat amb les dades de l'usuari (*certificatUsuari*):

Crea un parell de claus aleatòries, després un nou certificat amb la clau pública i les dades passades com a paràmetres, guarda el certificat a la base de dades, crea un magatzem PKCS#12 per la clau privada i el certificat protegit amb la contrasenya facilitada per l'usuari. Torna un *OutputStream* que permet la descarrega directa del fitxer PKCS#12.

k) Crear un certificat amb una petició de certificació (*certificatUsuari*):

Genera un certificat a partir de les dades de la petició passada com a paràmetre i es guarda a la base de dades. Després es genera un fitxer amb format PEM i es torna un InputSream que llegeix les dades del fitxer PEM i permet la descarrega directa. Si hi ha hagut cap problema es torna un valor null.

l) Crear un certificat amb un certificat autosignat (*certificatUsuari*):

Genera un certificat a partir de les dades del certificat autosignat passat com a paràmetre i es guarda a la base de dades. Després es genera un fitxer amb format PEM i es torna un InputSream que llegeix les dades del fitxer PEM i permet la descarrega directa. Si hi ha hagut cap problema es torna un valor null.

m) Renovació d'un certificat (*renovacioCertificat*):

S'actualitzen les dates d'inici i final de validesa del certificat passat com a

paràmetre i s'actualitza a la base de dades. Després es genera un fitxer amb format PEM i es torna un InputSream que llegeix les dades del fitxer PEM i permet la descarrega directa. Si hi ha hagut cap problema es torna un valor *null*.

n) Sol·licitar la revocació d'un certificat (*sollicitarRevocacio*):

   Agafa el certificat passat com a paràmetre, el marca com a pendent de revocació i guarda els comentaris de l'usuari de la raó per la revocació. Després actualitza la informació del certificat a la base de dades. Torna un valor booleà indicant si s'ha pogut fer la sol·licitud de revocació.

o) Sol·licitar la reactivació d'un certificat (*sollicitarReactivacio*):

   Agafa el certificat passat com a paràmetre, el marca com a pendent de reactivació i guarda els comentaris de l'usuari de la raó per la reactivació. Després actualitza la informació del certificat a la base de dades. Torna un valor booleà indicant si s'ha pogut fer la sol·licitud de reactivació.

p) Cercar un administrador (*cercarAdministrador*):

   Cerca un administrador a la base de dades pel nom d'usuari o, si aquest nom es *null*, pel nombre de serie del certificat. Torna l'administrador trobat o *null* si no ha trobat cap.

q) Cercar un certificat (*cercarCertificat*):

   Cerca un certificat a la base de dades pel nombre de serie del certificat passat com a paràmetre. Torna el certificat trobat o null si no ha trobat cap.

r) Verificar si la CA es aturada (*esAturada*):

   Torna un valor booleà indicant si la CA es aturada (pendent de la posta en marxa), per això comprova si la CA té un magatzem de claus associat.

s) Mètodes *valueBound* i *valueUnbound*:

   Són mètodes heretats de la classe *HttpSessionBindingListener*. Només fem servir el mètode *valueUnbound* per desactivar el administrador actiu quan es tanca la sessió de l'usuari administrador o quan a passat un temps sense activitat.

t) Mètodes “get” i “set” pels atributs necessaris.

3.1.2. Classe MagatzemClaus

Crea el magatzem de tipus *KeyStore* i el fitxer al disc on es guardarà. Si quan es crea el magatzem ja existeix un fitxer al disc amb el mateix nom, no crea un de nou sinó que recupera la informació del fitxer existent. La ruta on es crearà el fitxer està fixada a la constant *directoriMagatzem*.

Té mètodes implementats per:

a) Esborrar el magatzem (*esborrar*):

   Esborra el fitxer del magatzem a disc i després dona valor *null* als atributs del magatzem.

b) Llegir la clau del magatzem (*getClau*):

   Llegeix del magatzem i torna la clau privada identificada amb el alies passat com a

paràmetre.

c) Guardar la clau al magatzem (*setClau*):

   Guarda al magatzem la clau i la cadena de certificats associada, i els identifica amb el alies donat.

d) Llegir el certificat del magatzem (*getCertificat*):

   Llegeix del magatzem i torna el certificat identificat amb el alies passat com a paràmetre.

e) Guardar el certificat al magatzem (*setCertificat*):

   Guarda al magatzem el certificat i l'identifica amb el alies donat.

3.1.3. Classe ParellClaus

Per la generació de les claus he decidit fer servir l'algorisme RSA amb una longitud de 1024 bits, però es pot canviar l'algorisme i els paràmetres per generar les claus modificant les constants *algorismeClaus* i *parametresAlgorisme* de la classe.

Crea un parell de claus aleatori de tipus *KeyPair* que conté una clau privada i una pública.

Té mètodes implementats per:

a) Llegir la clau pública (*getPublica*):

   Torna la clau pública de *KeyPair*.

a) Llegir la clau privada (*getPrivada*):

   Torna la clau privada de *KeyPair*.

3.1.4. Classe CertificatCa

Crea un certificat X.509 versió 1. La CA emet el seu propi certificat autosignat amb les claus generades a l'apartat anterior.

Per crear el certificat faig servir l'algorisme de signatura SHA256 amb xifratge RSA, aquest algorisme es pot canviar modificant el valor de la la constant *algorismeSignatura*. El període de validesa del certificat està fixada per la constant *duracioCertificatCa* en 10 anys.

Té mètodes implementats per:

a) Llegir el certificat de la CA (*getCertificat*):

   Torna el certificat.

b) Canviar el certificat de la CA (*setCertificat*):

   Canvia el certificat.

c) Enviar el certificat (*enviar*):

   Genera un fitxer amb format PEM amb el certificat de la CA. Torna un *InputSream* que llegeix les dades del fitxer PEM i permet la descarrega directa. Si hi ha hagut cap problema es torna un valor *null*.

3.1.5. Classe Administrador

Emmagatzema el nom d'usuari, la contrasenya, un indicador de si es administrador avançat i el certificat de cada un dels administradors de la CA.

Aquesta classe es persisteix a una taula de la base de dades, això s'especifica mitjançant anotacions en la classe i en els atributs. La classe té l'anotació *@Entity* indicant que se ha de persistir, com no indiquem res més, el nom de la taula a la base de dades serà el nom de la classe. Al menys un dels atribut a de tenir l'anotació *@Id* per indicar que serà la clau primaria de la taula, en el nostre cas es el nom d'usuari. També he fet servir *@Column(nullable = false)* per indicar quin atributs son obligatoris (no poden ser *null* a la taula). Com el certificat de l'administrador es un del que hi han a la base de dades de certificats tenim l'anotació *@OneToOne* per indicar que cada administrador pot estar relacionat amb un certificat.

Hi han mètodes per llegir cada un dels quatre atributs.

3.1.6. Classe Certificat

A partir de la clau publica d'usuari genera un certificat X.509 versió 3. La clau publica pot ser la generada amb la classe *ParellClaus*, la enviada en el fitxer PKCS#10 o la del certificat autosignat de l'usuari.

Té tres mètodes constructors:

- Un crea un certificat nou a partir de la clau pública generada aleatoriament i el nom distingit del propietari. El nom distingit s'obté amb les dades del usuari introduïdes a un formulari. S'incorpora l'adreça de correu electrònic com una extensió de tipus *subjectAltName*. Altres extensions que s'incorporen al certificat son: *authorityKeyIdentifier*, *subjectKeyIdentifier*, *basicConstraints*, *keyUsage*, *extendedKeyUsage* i *cRLDistributionPoints*.
- Un altre crea el certificat a partir d'una petició de certificació. Les dades necessàries pel nou certificat es llegeixen de la petició de certificació: nom distingit de l'usuari, clau pública i algorisme de signatura. Les extensions que afegim son les mateixes del constructor anterior.
- El tercer crea el certificat a partir d'un certificat autosignat. Les dades necessàries pel nou certificat es llegeixen també del certificat autosignat i afegim les mateixes extensions.

A més de l'atribut del propi certificat generat, hi han atributs per cada un dels camps del certificat que pot interessar guardar a la base de dades per fer cerques o per reconstruir el certificat.

Mitjançant anotacions indiquem com es guardaran les dades a la base de dades. Tenim que la classe te l'anotació *@Entity* per indicar que serà una entitat de la base de dades (una taula a la base de dades relacional). Els atributs que no fa falta que es guardin tenen *@Transient.* L'atribut clau que identificarà cada un dels registres de la taula està marcat amb l'anotació *@Id*, en el nostre cas serà el nombre de serie del certificat. També tenim l'anotació *@Column* per indicar una característica de la columna de la taula, aquí ens indica si pot tenir valor *null* o no. Finalment l'anotació *@Temporal(DATE)* indica els camps que son una data.

Per crear el certificat faig servir per defecte l'algorisme de signatura SHA256 amb xifratge RSA, aquest algorisme es pot canviar modificant el valor de la la constant *algorismeSignatura*, però quan es crea el certificat a partir d'una petició de certificació o un certificat autosignat es fa servir l'algorisme que ve a la petició o al certificat. El període de validesa del certificat està fixada per la constant *duracioCertificatUsuari* en 1 anys. També hi han les constants *usClauUsuari* i *usEstesClauUsuari* per fixar els usos permesos de las claus de l'usuari.

Té mètodes implementats per:

a) Actualitzar les dates del certificat que es vol renovar (*actualitzar*):

   Canvia les dates d'inici i final de validesa com si s'emetes de nou i conserva la resta de camps.

b) Revocar el certificat (*revocar*):

   Omple el atribut de data de revocació, amb la data actual, i els atributs de les raons de revocació i data de invalidesa amb els valors que es passen com a paràmetre. Les raons es codifiquem com un nombre sencer entre 0 i 10 que indica el motiu de la revocació. També desactiva la indicació de si està pendent de revocació.

c) Reactivar el certificat (*reactivar*):

   Canvia la raó de la revocació al valor 8 que correspon a *removeFromCrl*, la resta d'atributs es deixen igual. També desactiva la indicació de si està pendent de reactivació.

d) Extreure un certificat d'un fitxer PEM (*deFitxerPEM*):

   Llegeix un fitxer amb un certificat a través d'un *InputStream*. A continuació extreu el certificat del fitxer i genera un nou certificat amb aquestes dades. Torna el certificat generat.

e) Crear un fitxer PEM del certificat (*aFitxerPEM*):

   Crear un fitxer PEM amb el certificat. Crea un *InputStream* que llegeix les dades del fitxer PEM i permet la seva descarrega directa.

f) Verificar el certificat:

   Verifica el certificat amb la clau pública que es passa com a paràmetre. Comprova que esta dintre del període de validesa i que la signatura es correcta. Trona un nombre sencer indicant si ha hagut cap error durant la verificació.

g) Mètodes “get” i “set” pels atributs necessaris.

3.1.7. Classe FitxerPkcs12

Crea un *KeyStore* de tipus PKCS#12 per encapsular a dintre la clau privada i el certificat d'un usuari.

El magatzem es protegeix amb la contrasenya introduïda per l'usuari. L'entrada de la clau privada es torna a protegir amb la mateixa clau.

Té només un mètode implementat:

a) Enviar el magatzem (*enviar*):

   Genera un fitxer amb format PKCS#12 amb el magatzem i torna un *InputSream* que llegeix les dades del fitxer i permet la descarrega directa.

3.1.7. Classe PeticioCertificacio

Per defecte crea una petició de certificació buida.

Té dos mètodes implementats per:

a) Extreure una petició de certificació d'un fitxer PEM (*deFitxerPEM*):

   Llegeix un fitxer amb una petició de certificació a través d'un *InputStream*. A continuació extreu la petició del fitxer i genera una nova petició amb aquestes dades. Torna la petició de certificació generada.

b) Mètode “get” per llegir la petició de certificació.

3.1.7. Classe LlistatCrl

Conté les dades de la CRL.

En el procés de creació primer mira el nombre de serie de la última CRL publicada llegint-la del fitxer al servidor, incrementa en 1 el nombre i crea la nova CRL. A continuació afegeix tots els certificats revocats fent una cerca a la base de dades.

Té mètodes implementats per:

a) Llegir l'atribut de la URL de publicació (*getCrlUrl*).

b) Publicar la CRL (*publicar*):

   Fica la CRL a un fitxer PEM que s'escriu a la web en la URL passada com a paràmetre.

c) Esborrar la CRL (*esborrar*):

   Esborra el fitxer de la CRL a la web, posa *null* a l'atribut de la CRL i inicialitza el nombre de serie de la CRL a 1.

a) Enviar la CRL (*enviar*):

   Si existeix el fitxer de la CRL (si ha sigut publicada), torna un *InputSream* que llegeix les dades del fitxer i permet la descarrega directa.

3.2. Interfície web

La interfície web esta formada per diferents planes estàtiques i dinàmiques que donen accés a totes les funcions de la CA. Les planes dinàmiques estan gestionades per *servlets* que s'encarreguen de accedir a la CA per obtenir les dades que presentaran a la plana.

Els *servlets* son aplicacions independents de l'aplicació de la CA. La solució per poder transferir informació entre els servlets i l'aplicació, o entre els mateixos servlets, es basa en que tots el *servlets* d'una mateixa aplicació web executant-se en un servidor tenen accés al mateix *ServletContext.* Aquesta classe actua com a un repositori d'informació accessible per tots els *servlets*, aleshores només tenim que posar la referencia dels

objectes que volem que siguin accessibles a la classe *ServletContext*.

Un problema comú en molt dels *servlets*, es que per pujar al servidor i tractar els fitxers amb les peticions de certificació o els certificats no hi ha un mètode estàndard. Com no es una tasca trivial, m'he estimat més fer servir llibreries *java* preparades per facilitar aquesta tasca. He fet servir el paquet *Commons FileUpload* de la Fundació Apache que ve per defecte amb el servidor Tomcat. Tots els fitxer que es pugen al servidor els he tractat amb aquestes llibreries.

3.2.1. Servlet d'inici.

La CA s'instancia mitjançant el *servlet ServletIniciCa* en el seu mètode *init*.

Aquest *servlet* s'executa automàticament quan s'engega el servidor Tomcat, només hem de indicar el paràmetres *loadOnStartup* en la capçalera.

Després en el seu mètode *init()* s'encarrega de crear la CA, cridant al constructor de la classe *AutoritatCertificacio*, i guardar la referencia a la CA en el *ServletContext* de l'aplicació web per que la resta de *servlets* puguin accedir-la.

3.2.2. Plana inicial de la CA.

La plana *index.html* es una plana HTML estàtica on només tenim dos àrees, una per administració i l'altre per d'usuaris.

A l'àrea d'usuari tenim enllaços per:

- Accés a la plana de sol·licitud d'emissió de certificats. Ens dirigeix a una plana dinàmica gestionada pel *servlet ServletFormulariCertificat* que a la web té la URL “/formularicertificat”.
- Accés a la plana de gestió de certificats. Ens dirigeix a una plana dinàmica gestionada pel *servlet ServletGestioCertificat* que a la web té la URL “/gestiocertificats”.
- Descarrega del fitxer de la CRL. La descarrega del fitxer la gestiona el *servlet ServletLlistatCrl* amb URL “/llistatcrl”. Aquest *servlet* verifica que la CA no està aturada i que la CRL està publicada, si això es correcte crida al mètode *enviar()* de la CRL de la CA, aquest mètode retorna un *InputStream* que es redirigeix cap a un *ServletOutputStream* que ens serveix el fitxer com a una descarrega.
- Descarrega del certificat de la CA. La descarrega del fitxer la gestiona el *servlet ServletCertificatCa* amb URL “/certificatca”. Aquest *servlet* verifica que la CA no està aturada, si això es correcte crida al mètode *enviar()* del certificat de la CA, aquest mètode retorna un *InputStream* que es redirigeix cap a un *ServletOutputStream* que ens serveix el fitxer com a una descarrega.

A l'àrea d'administració només tenim un enllaç que ens dirigeix a una plana per l'autenticació de l'administrador. Aquesta plana està servida pel *servlet ServletAutenticacioAdm* amb URL “/autenticacioadministrador”

3.2.2. Plana per autenticació d'administradors.

Es una plana dinàmica servida pel servlet ServletAutenticacioAdm. Aquesta plana es un

formulari que demana un nom d'usuari i una contrasenya, o un certificat per autenticar a l'usuari administrador. La informació del formulari es tractada al mètode *doPost* del servlet.

Com a mètode de seguretat es verifica si la sessió d'usuari es nova, si es nova presenta la plana d'autenticació, si no es nova verifica si l'usuari està autenticat, en aquest cas l'hi donaria accés a la plana d'administració directament, però si no està autenticat llegeix els camps del formulari i els tracta. La sessió de l'administrador autenticat no es tanca fins que:

- No surt de la plana d'administració amb l'opció “Sortir”.
- Passa un temps de 120 segons sense activitat.

El *servlet* detecta si la connexió es local o remota comparant les IP del servidor i del client. Si la connexió es local només estan actius els camps “usuari” i “contrasenya” del formulari, si es remota només esta actiu el camp per enviar un certificat.

Quan es tracten els camps del formulari es fan les següents comprovacions:

- Que no hi ha cap administrador actiu. Només es permet un administrador actiu, fins que no surti el administrador actiu no podrà entrar un altre administrador.
- En accés local verifica si l'usuari i contrasenya coincideixen amb algun dels administradors a la base dades. Si l'usuari no existeix o la contrasenya no es correcta es dona un missatge d'error.
- En accés remot verifica el certificat, si no es correcte dona un error que pot ser:
  - No s'ha seleccionat un fitxer.
  - L'arxiu no conté un certificat.
  - El certificat no pertany a la CA.
  - El certificat no és vàlid: no està dintre del seu període de validesa o la signatura no es correcta.
  - El certificat no està a la base de dades de la CA.
  - El certificat està revocat.
  - El certificat no pertany a un usuari administrador.

3.2.3. Plana per l'administració de la CA.

Es una plana dinàmica servida pel *servlet ServletAdministracio* amb URL “/administracio”. Aquesta plana es un formulari on tenim totes les funcions d'administració de la CA. La informació del formulari es tractada al mètode *doPost* del *servlet*.

Com a mètode de seguretat es verifica si la sessió d'usuari es nova, si es nova ens redirigeix cap a la plana d'inici (“index.html”).

El servlet activa o desactiva opcions segons l'estat de la CA i el tipus d'administrador (normal o avançat) que accedeix. Si la CA està aturada només estan actives les opcions d'inicialització, posta en marxa i sortir, quan s'engega totes les opcions estan actives menys la de posta en marxa. A més segons el tipus d'administrador també es poden desactivar opcions, mentre un administrador avançat pot accedir a totes les opcions, un

administrador normal només pot fer la posta en marxa de la CA, consultar certificats, gestionar certificats o publicar la CRL, però no pot ni inicialitzar la CA, ni gestionar usuaris administradors.

Quan es tracten els camps del formulari, primer es comprova l'opció escollida:

- Inicialització de la CA. Es crida al mètode *inicialitzacio()* de la CA.
- Posta en marxa de la CA. Es crida al mètode *postaMarxa()* de la CA. No és posa cap nom, ni contrasenya pel magatzem de claus de la CA (valors a *null*) i es passa l'adreça URL per la publicació de la CRL.
- Consulta certificats. Es llegeixen els camps de cerca de certificats i es fa una crida al mètode *llistatCertificats()* de la CA amb els valors del camps de cerca. Això ens torna un llistat de certificats que compleixen els requisits. En una nova plana es presenten els certificats o un missatge de error indican que no hi han certificats que compleixin les condicions.
- Gestió certificats. Comprova la següent opció seleccionada:
  - Revocació: Llegeix el nombre de serie del certificat que es vol revocar i les raons per la revocació, i crida al mètode *revocacioCertificat*() passant com a paràmetres el nombre, les raons i la data de invalidesa (que per defecte agafo l'actual). Pot donar error si el certificat no es pot revocar (nombre de serie equivocat, certificat ja revocat, ...) o si el nombre de serie no es un enter.
  - Reactivació: Llegeix el nombre de serie del certificat que es vol reactivar i crida al mètode *reactivacioCertificat*(). Pot donar error si el certificat no es pot reactivar (nombre de serie equivocat, certificat sense revocar, ...) o si el nombre de serie no es un enter.
- Publicació de la CRL. Es crida al mètode *publicarCrl()* de la CA passant com a paràmetre la URL de publicació.
- Gestió administradors. Comprova la següent opció seleccionada:
  - Consulta: Crida al mètode *llistatAdministradors()* que torna un llistat de administradors. En una nova plana es presenta el llistat d'administradors o un missatge de error indican que no hi han administradors.
  - Alta: Llegeix el nombre d'usuari, la contrasenya, el nombre de serie del certificat de l'administrador i si es un administrador avançat. Dona un error si falta el usuari o la contrasenya que son obligatoris. Crida al mètode *afegirAdministrador*() i pot donar error si l'usuari ja existeix o el certificat no existeix.
  - Baixa: Llegeix el nombre d'usuari i la contrasenya. Dona un error si falta el usuari o la contrasenya que son obligatoris. Crida al mètode *esborrarAdministrador*() i pot donar error si l'usuari no existeix, si la contrasenya es incorrecta o si el usuari administrador que es vol esborrar el el que esta actiu.
- Sortir de la plana d'administració. Totes les altres opcions acaben tornant a la plana d'administració, però amb aquesta s'invalida la sessió d'administració i es torna a la plana d'inici.

3.2.4. Plana de sol·licitud de certificació.

Es una plana dinàmica servida pel *servlet ServletFormulariCertificat*. Aquesta plana es un formulari on tenim les tres opcions per demanar l'emissió d'un certificat a la CA. La informació del formulari es tractada al mètode *doPost* del *servlet*.

El *servlet* comprova si la CA es aturada per donar un missatge de que no es poden demanar certificats fins que la CA no estigui en servei.

La plana té tres opcions:

- Petició de certificació. Permet al client seleccionar un fitxer amb una petició de certificació PKCS#10 amb codificació PEM (la que normalment genera la comanda *openssl*) del seu ordinador i fer l'enviament al servidor. El servlet agafa el *InputStream* amb el fitxer i el redirigeix al mètode *deFitxerPem()* de la classe *PeticioCertificacio* per obtenir una petició de certificació. Aquesta petició es verifica i després es passa al mètode *certificatUsuari()* de la CA. Si tot es correcte obtenim un *InputStream* que redirigim cap al *ServletOutputStream* per descarregar el fitxer del nou certificat. Es dona error si:
  - No s'ha seleccionat un fitxer.
  - El fitxer no conté una petició de certificació.
  - La clau de la petició de certificació no es correcta.
  - Si no tenim l'algorisme o el proveïdor necessaris per llegir la petició.
  - La signatura no es correcta.
- Certificat autosignat. Permet al client seleccionar un fitxer amb un certificat autosignat amb codificació PEM del seu ordinador i fer l'enviament al servidor. El servlet agafa el *InputStream* amb el fitxer i el redirigeix al mètode *deFitxerPEM()* de la classe *Certificat* per obtenir un certificat. Aquest es verifica i després es passa al mètode *certificatUsuari()* de la CA. Si tot es correcte obtenim un *InputStream* que redirigim cap al *ServletOutputStream* per descarregar el fitxer del nou certificat. Es dona error si:
  - No s'ha seleccionat un fitxer.
  - El fitxer no conté un certificat.
  - La clau del certificat autosignat no es correcta.
  - Si no tenim l'algorisme o el proveïdor necessaris per llegir el certificat.
  - La signatura no es correcta.
  - El certificat no es dintre del període de validesa.
  - Hi ha hagut algun problema llegint-lo.
- Formulari de petició. Llegeix els valors dels camps del formulari. Si falta algun del camps obligatoris dona un missatge d'error indicant quins falten. Quan hi son tots els camps obligatoris crida al mètode *certificatUsuari()* de la CA passant com a paràmetres els valors dels camps. Si tot va be rep un *OutputStream* que es redirigeix cap el *ServeltOutputStream* que ens servira la descarrega del fitxer. Si per alguna raó no s'ha pogut generar el fitxer PKCS#12 amb la clau i el certificat

d'usuari dona un error.

3.2.5. Plana de gestió de certificats.

Es una plana dinàmica servida pel *servlet ServletGestioCertificat*. Aquesta plana es un formulari on tenim les tres opcions que podem demanar amb certificats emesos per la CA. La informació del formulari es tractada al mètode *doPost* del *servlet*.

El *servlet* comprova si la CA es aturada per donar un missatge de que no es poden fer gestions amb els certificats fins que la CA no estigui en servei.

La plana té tres opcions:

- Renovació. Permet al client fer la renovació d'un certificat abans de la seva caducitat. S'ha de seleccionar un fitxer amb el certificat que es vol renovar amb codificació PEM i fer l'enviament al servidor. El servlet agafa el *InputStream* amb el fitxer i el redirigeix al mètode *deFitxerPEM()* de la classe *Certificat* per obtenir un certificat. Aquest es verifica i després es passa al mètode renovacioCertificat*()* de la CA. Si tot es correcte obtenim un *InputStream* que redirigim cap al *ServletOutputStream* per descarregar el fitxer del nou certificat amb les dates de validesa actualitzades. Es dona error si:
  - No s'ha seleccionat un fitxer.
  - L'arxiu no conté un certificat.
  - El certificat no pertany a la CA.
  - El certificat no és vàlid: no està dintre del seu període de validesa o la signatura no es correcta.
  - El certificat no està a la base de dades de la CA.
  - El certificat està revocat.
- Revocació. Permet al client fer la sol·licitud de revocació d'un certificat. La revocació del certificat només la pot fer un administrador. S'ha de seleccionar un fitxer amb el certificat que es vol revocar amb codificació PEM i fer l'enviament al servidor. El servlet agafa el *InputStream* amb el fitxer i el redirigeix al mètode *deFitxerPEM()* de la classe *Certificat* per obtenir un certificat. Aquest es verifica i si tot es correcte es llegeix el camp amb comentaris de l'usuari. El certificat i els comentaris es passen al mètode *sollicitarRevocacio()* de la CA que marca el certificat com a pendent de revocació i guarda els comentaris. Es dona error si:
  - No s'ha seleccionat un fitxer.
  - L'arxiu no conté un certificat.
  - El certificat no pertany a la CA.
  - El certificat no és vàlid: no està dintre del seu període de validesa o la signatura no es correcta.
  - El certificat no està a la base de dades de la CA.
  - El certificat ja està revocat.

- Reactivació. Permet al client fer la sol·licitud de reactivació d'un certificat. La reactivació del certificat només la pot fer un administrador. S'ha de seleccionar un fitxer amb el certificat que es vol reactivar amb codificació PEM i fer l'enviament al servidor. El servlet agafa el *InputStream* amb el fitxer i el redirigeix al mètode *deFitxerPEM()* de la classe *Certificat* per obtenir un certificat. Aquest es verifica i si tot es correcte es llegeix el camp amb comentaris de l'usuari. El certificat i els comentaris es passen al mètode *sollicitarReactivacio()* de la CA que marca el certificat com a pendent de reactivació i guarda els comentaris. Es dona error si:
  - No s'ha seleccionat un fitxer.
  - L'arxiu no conté un certificat.
  - El certificat no pertany a la CA.
  - El certificat no és vàlid: no està dintre del seu període de validesa o la signatura no es correcta.
  - El certificat no està a la base de dades de la CA.
  - El certificat no està revocat.

Capítol 4: Manual d'usuari del producte

4.1. Instal·lació del producte i el programari necessari

El producte be empaquetat en un fitxer WAR preparat per la seva instal·lació a un servidor Tomcat.

Com a requisits previs necessitem:

- Una versió de Java 6.
- El servidor web Tomcat en la seva versió 7.
- El servidor de bases de dades Derby versió 10.8.1.

Jo he fet la instal·lació sobre una màquina virtual amb Ubuntu 11.10.

Per instal·lar Java he fet servir el “Centre de Software d'Ubuntu” i he descarregat el paquet “openjdk-6-jre”. Java no necessita més configuració.

Per la instal·lació de Tomcat també he fet servir el “Centre de Software d'Ubuntu” i he cercat el paquet “tomcat7-user”. Aquest paquet permet la creació de múltiples instancies del servidor que poden funcionar independentment. Per crear la nostra instancia del servidor hem d'obrir un terminal i executar la comanda:

```
$ tomcat7-instance-create /home/andres/Tomcat
```

Això crea una carpeta “/Tomcat” a la meva carpeta personal on estan tots els fitxers del servidor.

Per la base de dades Derby he descarregat el paquet “db-derby-10.8.1.2-bin.zip” de la web del projecte Apache Derby i l'he descomprimit al directori “/Derby” de la meva carpeta personal. També hem de introduir algunes modificacions a les variables d'entorn per que apuntin al directori de Derby, jo he fet les modificacions al fitxer “/etc/environment”:

- S'ha d'afegir la ruta als binaris de Derby a la variable PATH, en el meu cas he afegit “/home/andres/Derby/bin:” al final de la variable PATH.
- S'ha de crear una nova variable d'entorn anomenada DERBY_HOME que apunta al directori base de Derby, he creat una nova línia al fitxer amb “DERBY_HOME = /home/andres/Derby”.

Amb això ja tenim instal·lats i configurats tots els requeriments previs, només queda agafar el nostre fitxer WAR amb el projecte i posar-lo a dintre la carpeta “/webapps” del nostre servidor Tomcat.

4.2. Funcionament del producte

Per fer funcionar el producte hem de:

1) Engegar el servidor de bases de dades Derby. Obrim un terminal i executem la comanda:

   ```
   java -jar $DERBY_HOME/lib/derbyrun.jar server start
   ```

   Veurem un missatge que indica que el servidor s'ha engegat i que esta utilitzant el

port per defecte que es el 1527.

2) Engegar el servidor web Tomcat. Anem a la carpeta “/bin” del directori d'instal·lació de Tomcat i executem l'script “startup.sh”.

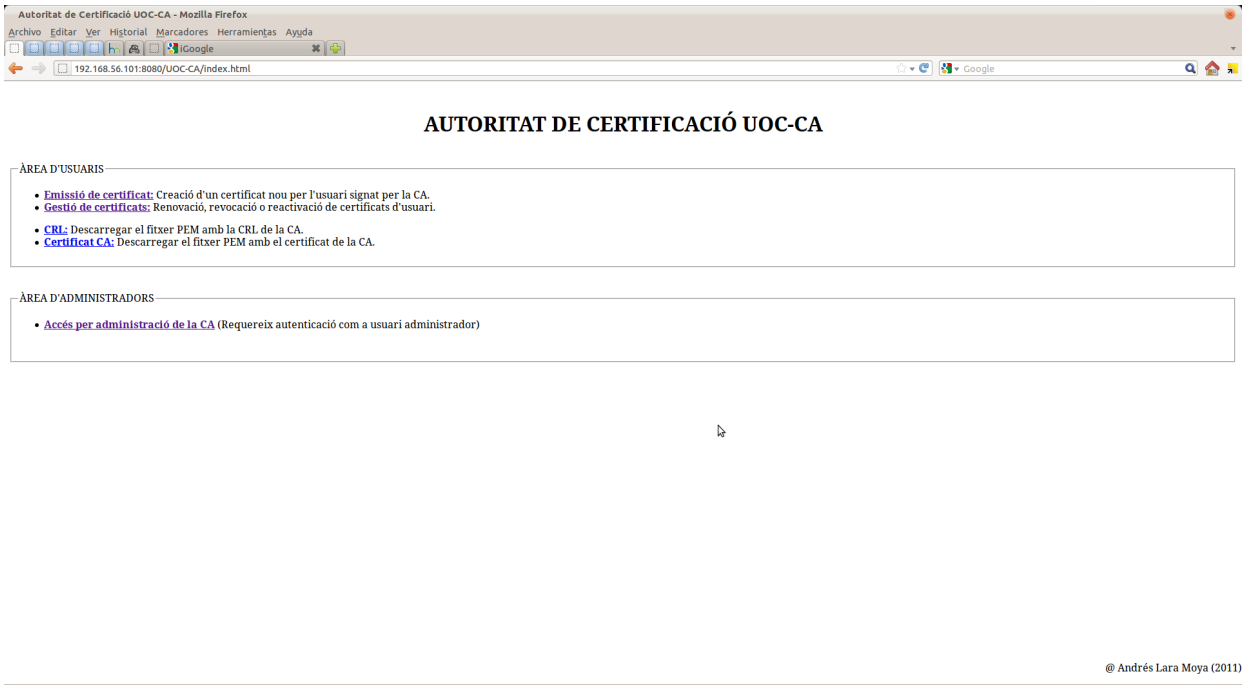
3) Ja podem accedir a l'aplicació amb el navegador web. Si ens connectem des de la mateixa màquina (accés local) la URL seria:

   `http://localhost:8080/UOC-CA/`

   Peró si la connexió es remota tindríem que substituir “localhost” per la identificació del servidor (una adreça IP o un nom de servidor).

Aara amb el navegador web podem accedir a les diferents planes de la interfície de la aplicació que son:

a) Plana d'inici (“/index.html”):



Es una plana HTML estàtica d'inici que ens presenta dos seccions:

- Àrea Usuaris: permet accedir a les planes per sol·licitar l'emissió d'un nou certificat o gestionar certificats emesos per la CA, i també permet la descarrega dels fitxers amb la CRL o el certificat de la CA.

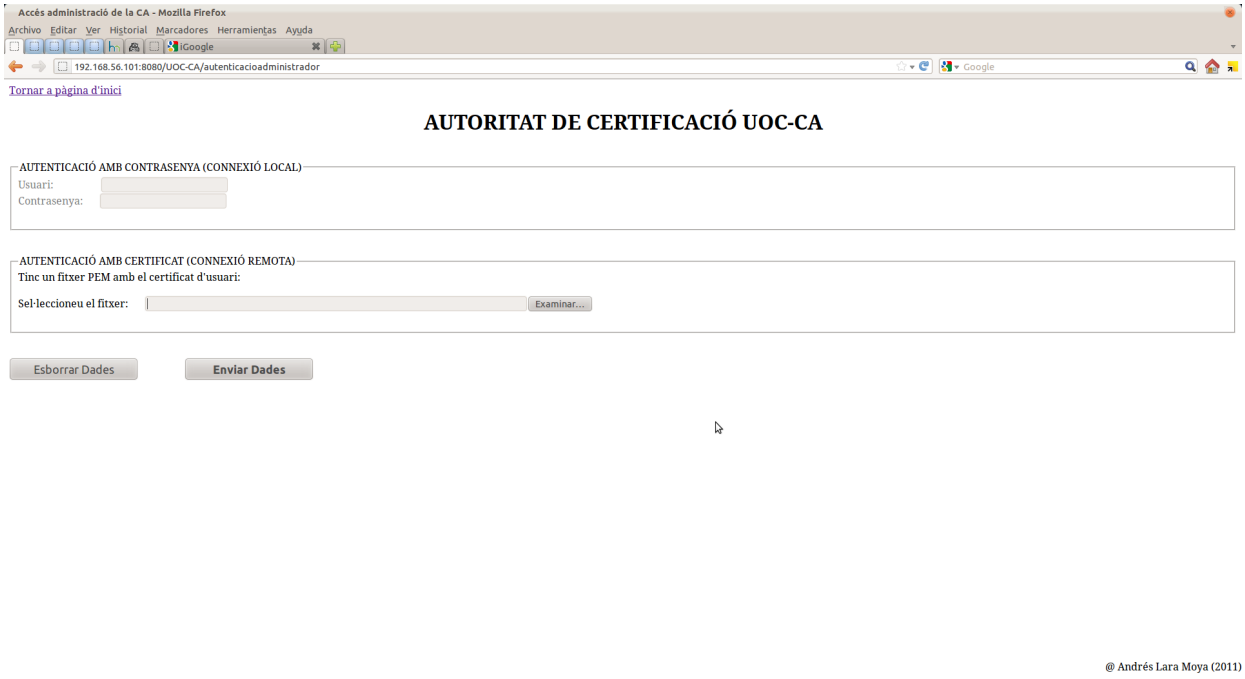
  Si la CA no s'ha posat en marxa els enllaços de descarrega no fan res i si provem d'accedir a les planes d'emissió o gestió de certificat ens donarà un missatge que no es poden fer operacions donat que la CA es aturada.

  El fitxer de la CRL es pot veure fent servir l'eina *openssl* amb la comanda:

  openssl crl -in *nomfitxer_crl.pem* -text -noout

- Àrea Administradors: només tenim un enllaç per anar a la plana d'autenticació.

b) Autenticació d'administradors (“/autenticacioadministrador”):



Es un servlet que genera una plana dinàmica on tenim les dos opcions d'autenticació mitjançant nom d'usuari i contrasenya, o amb un certificat digital de la CA.

Detecta automàticament si l'accés es local o remot i només deixar actius els camps que corresponen al tipus de connexió.

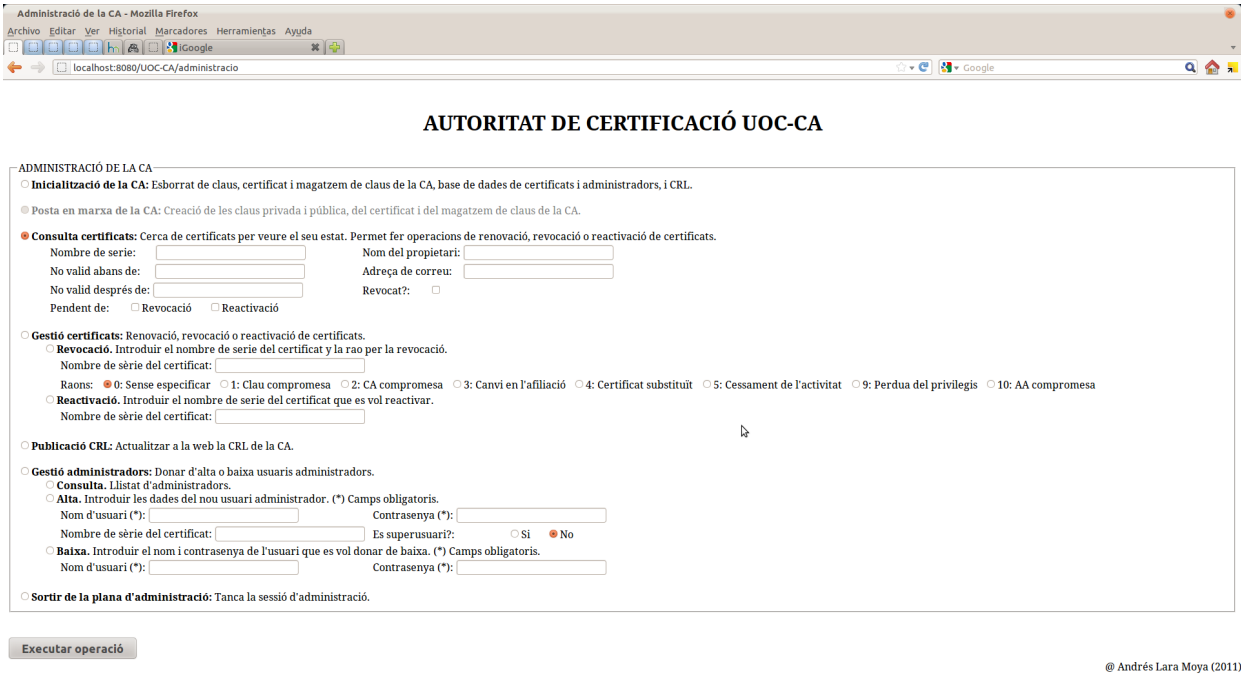
Quan s'inicia l'aplicació per primera vegada o després d'una inicialització, només hi ha un usuari administrador per defecte per accedir localment que té com a nom “administrador” i com a contrasenya “contrasenya”.

Mentre no es posa en marxa la CA no es possible l'accés remot mitjançant certificat donat que no hi ha accés a la base de dades de la CA.

Per seguretat només es permet que hi hagi un administrador actiu treballant. Si s'intenta accedir mentre hi ha un administrador actiu dona un error. Per evitar que l'accés a l'administració de la CA quedi bloquejat si un administrador deixa oberta una sessió, les sessions tenen un temps límit de 120 segons si no hi ha activitat.

Si les dades introduïdes als camps no son correctes (nom inexistent, contrasenya errònia, ...) dona un missatge a la part baixa indicant el tipus d'error.

c) Administració de la CA (“/administracio”):



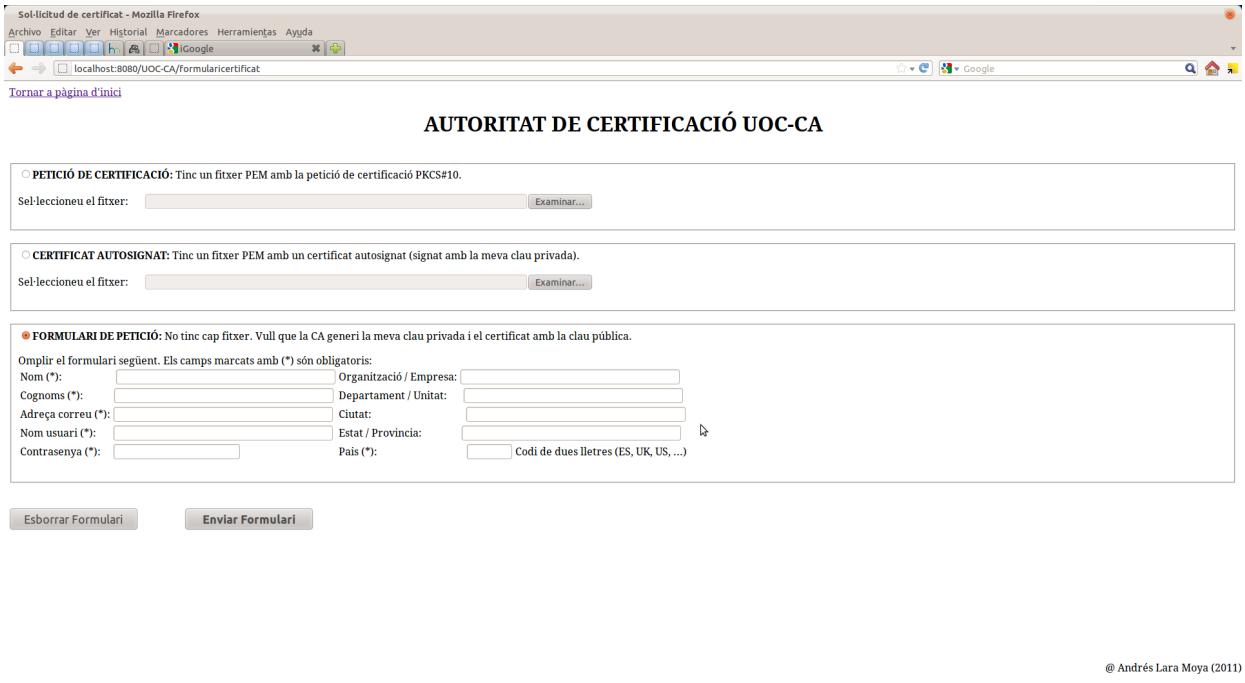
Es un servlet que genera una plana dinàmica on tenim totes les opcions d'administració de la CA:

- Inicialització de la CA.
- Posta en marxa de la CA.
- Consulta de certificats. Tenim uns camps accessoris per si es vol limitar la cerca a uns valors determinats.
- Gestió de certificats. Podem fer la revocació o reactivació d'un certificat introduint el nombre de serie del mateix.
- Publicació de la CRL.
- Gestió d'usuaris administradors. On podem consultar la llista d'administradors o donar d'alta o baixa un usuari administrador.
- Sortir de la plana d'administració.

El servlet activa o desactiva opcions segons l'estat de la CA i el tipus d'administrador (normal o avançat) que accedeix. Si la CA està aturada només estan actives les opcions d'inicialització, posta en marxa i sortir, quan s'engega totes les opcions estan actives menys la de posta en marxa. A més segons el tipus d'administrador també es poden desactivar opcions, mentre un administrador avançat pot accedir a totes les opcions, un administrador normal només pot fer la posta en marxa de la CA, consultar certificats, gestionar certificats o publicar la CRL, però no pot ni inicialitzar la CA, ni gestionar usuaris administradors.

Si les dades introduïdes als camps no son correctes dona un missatge a la part baixa indicant el tipus d'error.

d) Sol·licitud de certificats (“/formularicertificat”):



Es un servlet que genera una plana dinàmica on tenim les tres possibilitats per demanar l'emissió d'un certificat:

- Amb una petició de certificació.
- Amb un certificat autosignat.
- Amb les dades introduïdes en un formulari.

Els fitxers amb la petició de certificació o el certificat autosignat han de tenir el format PEM. Per generar els fitxers de petició de certificació i certificat autosignat podem fer servir l'eina *openssl* amb les següents comandes:

- Generació d'una clau privada:

  openssl genrsa -out *nomfitxer_clau.pem* 1024

- Generació de la petició de certificació:

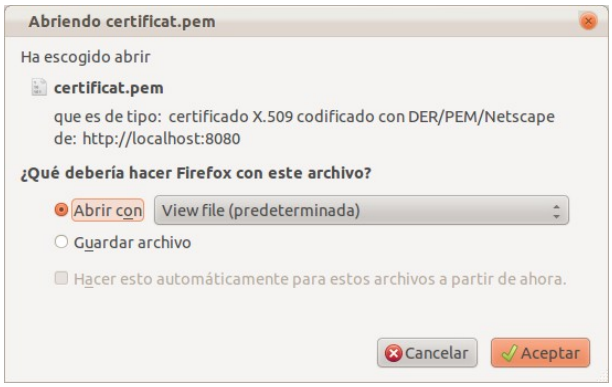
  openssl req -new -key *nomfitxer_clau.pem* -out *nomfitxer_pkcs10.pem* -outform PEM

- Generació d'un certificat autosignat:

  openssl req -x509 -new -key *nomfitxer_clau.pem* -out *nomfitxer_certificat.pem* -outform PEM

Un detall important a tenir en compte quan es generen els fitxers de petició de certificació o de certificat autosignat amb l'eina *openssl*, es que quan demana l'adreça de correu electrònic s'ha de deixar buida, si no crea un atribut *“E= adreça_correu”* a dintre del nom distingit que no es reconegut per la classe *X500Principal*, donat que aquest atribut no està contemplat a cap dels dos estàndards que reconeix, el RFC 2253 i RFC 1779.

Els certificats, si no hi ha cap problema, es generen immediatament, sortint una finestra que es demana que volem fer amb l'arxiu.



Si el certificat s'ha generat amb una petició de certificació o un certificat autosignat el fitxer descarregat es directament un certificat, però si s'ha utilitzat el formulari el fitxer que descarreguem es un magatzem PKCS#12 amb la clau privada i el certificat. Podem tornar a fer servir l'eina openssl per extreure el certificat i la clau:

- Extracció del certificat:

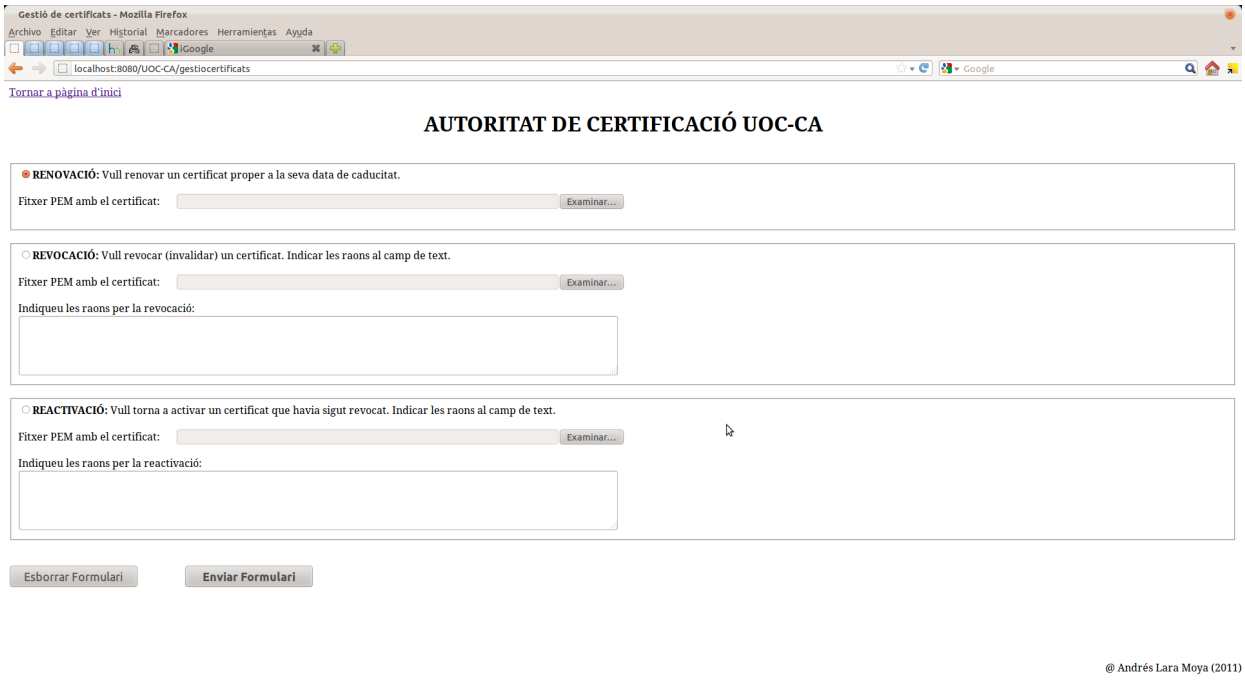
  openssl pkcs12 -in *nomfitxer_descarregat.p12* -nokeys -clcerts -out *nomfitxer_certificat.pem*

- Extracció de la clau privada:

  openssl pkcs12 -in *nomfitxer_descarregat.p12* -nocerts -out *nomfitxer_certificat.pem*

Si els fitxers o les dades introduïdes als camps no son correctes, o si falta alguna dada obligatòria, dona un missatge a la part baixa indicant el tipus d'error o el camp que falta.

e) Gestións amb certificats de la CA (“/gestiocertificats”):



Es un servlet que genera una plana dinàmica on tenim tres possibilitats per gestionar els nostres certificats emesos per la CA:

- Fer la renovació.

- Demanar la revocació.
- Demanar la reactivació d'un certificat revocat.

Per qualsevol de les tres opcions s'ha de enviar un fitxer amb el certificat amb format PEM.

En el cas de la renovació, si no hi ha cap problema, el certificat renovat es genera immediatament i apareix una finestra per demanar que volem fer amb el fitxer del nou certificat.

Les opcions de revocació o reactivació marquen els certificats a la base de dades i només un administrador pot fer la operació final de revocació o reactivació. Els camps de comentaris estan per indicar al administrador de la CA la raó per la que es demana el canvi al certificat.

Al final si volem aturar els servidors faríem el següent:

1) Aturar el servidor web Tomcat. Anem a la carpeta “/bin” del directori d'instal·lació de Tomcat i executem l'script “shutdown.sh”.
2) Aturar el servidor de bases de dades Derby. Obrim un nou terminal i executem la comanda:

```
java -jar $DERBY_HOME/lib/derbyrun.jar server shutdown
```

Capítol 5: Millores del producte

Després del treball fet es veuen moltes possibles millores, unes per requeriments no assolits i altres per que a mesura que coneixes les tecnologies implicades veus millors solucions per les funcionalitats de l'aplicació.

Algunes de les millores que he vist son:

- Xifrat de les connexions remotes amb el protocol SSL: Aquest era un del requeriments inicials que no he pogut assolir. Consisteix a generar un certificat a la CA per aquesta funció. Després s'ha de configurar el servidor Tomcat per activar les comunicacions fent servir el protocol SSL i dir-li al servidor on es troba el certificat de la CA que ha de utilitzar per aquest protocol. Tot ve explicat al document “ssl-howto.html” de la documentació de Tomcat.
- Autenticació de client mitjançant certificats digitals: es una millora que està lligada a la anterior del protocol SSL. Si tenim una connexió SSL el client pot fer servir un certificat digital propi en lloc de fer servir el que genera el servidor quan s'estableix la comunicació.
- Utilització d'un certificat intermedi per la CA: Per evitar que la CA utilitzi el seu certificat arrel per signar els certificats d'usuari, es una practica habitual que la CA generi un o varis certificats X.509 versio 3 signats amb el seu certificat arrel. Aquest o aquests certificats intermedis seran els certificats de “treball” que farà servir per signar certificats d'usuaris, mentre el certificat arrel queda emmagatzemat en un lloc segur per si fos necessari en alguna situació excepcional.
- Utilització de fitxers de certificats en codificació DER: la meva aplicació només treballa amb fitxers en format PEM, un bon complement seria que també pugues llegir i generar fitxers DER.
- Doble contrasenya pel fitxer PKCS#12: a la meva aplicació faig servir la mateixa contrasenya per protegir el magatzem i les entrades amb les claus privades. Es pot millorar la seguretat demanant i fent servir claus diferents per cada una.
- La interfície es podria millorar en moltes coses, la primera el disseny, però hi han dos elements que evitarien possibles errors a l'hora d'interactuar amb ella:
    - Demanar confirmació mitjançant finestres emergents de les operacions “crítiques”, com esborrat d'administradors o inicialització de la CA.
    - Per fer la baixa d'administradors o les revocacions i reactivacions de certificats en lloc de introduir les dades a través d'un formulari fer la selecció directament des de un llistat.

Conclusions

Els treball final crec que assoleix a nivell tècnic la majoria del objectius plantejats en un inici, només ha quedat pendent el xifrat de la connexió amb SSL per manca de temps.

El producte fa el que es vol i crec que es suficientment solid. Tot i això es pot millorar molt:

- Depuració de codi: crec que si el pogués repassar amb tranquil·litat podria eliminar moltes línies de codi innecesaries.
- Estandardització dels procediments: a vegades per arribar a un mateix resultat ho faig per dos camins diferents.
- Millorar l'aspecte visual: el disseny de les planes es molt simple, encara que la interfície web no era el requeriment principal del producte, si que es la part que es veu i la que dona la primera impressió de si un producte està be o no.

En quant als objectius personals crec els he assolit perfectament, més tenint en compte que inicialment la majoria de les tecnologies que he fet servir eren totalment desconegudes per mi.

Cada dia aprenia alguna cosa nova que hem feia tornar-me a replantejar i millorar el que havia fet fins el moment. Degut això la aplicació final no s'assembla en res a com va començar.

Crec que en general el meu gran problema a sigut el temps, em faltava temps. No tens molt temps per fer un projecte important, però jo, a més, me confiat massa i he deixat molt treball per el final. Si hem tornes a plantejar el projecte deixaria un llarg període al final del projecte per fer proves i correccions.

En general es pot dir que estic molt satisfet del treball fet.

Glossari

- Application Programming Interface (API): es el conjunt de funcions i procediments que ofereix una llibreria de programari.
- Certificat digital: és una estructura de dades que conté una clau pública i informació del propietari, tot signat digitalment per una entitat de confiança, normalment una CA, que legitima les dades anterior.
- Certification Authority (CA): entitat de confiança que després de verificar la identitat del propietari d'una clau pública, emet un certificat digital que legitima aquesta identitat.
- Certificate Revocation List (CRL): son llistats de certificats que per alguna raó han sigut revocats (inhabilitats).
- Criptografia de clau pública: criptografia basada en dos claus, una pública a l'abast de tothom i altre privada només en poder de l'usuari. El missatge que es vol mantindre secret es xifra amb la clau pública i només el posseïdor de la clau privada associada a la clau pública podrà desxifrar el missatge.
- Entorn de Desenvolupament de Programari (IDE – Integrated Development Environment): Son plataformes compostes de eines de programació que faciliten el desenvolupament de aplicacions.
- Java Persistence API (JPA): es una llibreria que permet la persistència d'objectes, com si fossin dades relacionals, dintre de la plataforma Java EE.
- Orientat a Objectes (OO – Object Oriented): són llenguatges de programació que basen la seva estructura en objectes i les seves interrelacions.
- Privacy Enhanced Mail (PEM): format de fitxer que codifica en Base64 un certificat posant-li unes marques d'inici i final ("-----BEGIN CERTIFICATE-----" i "-----END CERTIFICATE-----"). Es el format per defecte quan treballem amb OpenSSL.
- Public Key Infrastructure (PKI): conjunt de maquinari, programari, persones, polítiques i procediments necessaris per a crear i gestionar certificats digitals basats en la criptografia de clau pública.
- Public-key Cryptography Standards (PKCS): es un conjunt de normes desenvolupades per la empresa RSA Laboratories i que s'han convertit en un estàndard de fet.
- PKCS#10: Norma PKCS que descriu l'estructura de dades d'una petició de certificació.
- PKCS#12: Norma PKCS que descriu el format d'un magatzem per les claus privades i els certificats d'usuari, que permet una transmissió segura.
- Structured Query Language (SQL): es un llenguatge d'accés a bases de dades relacionals.
- Secure Sockets Layer (SSL): es un protocol criptogràfic que permet comunicacions segures a través d'una xarxa informàtica.

- Web (World Wide Web): es un sistema de distribució d'informació a través d'Internet que fa servir hiperenllaços.

Recursos

Criptografia:

- *Apunts assignatura "Criptografia" de la UOC*.
  Coneixements generals de Criptografia.
- *Llibre "Handbook of Applied Cryptography".*
  Coneixements avançats de criptografia.
  http://www.cacr.math.uwaterloo.ca/hac/
- *Llibre "Beginning Cryptography with Java".*
  Implementació de sistemes criptogràfics amb Java.
  http://www.wrox.com/WileyCDA/WroxTitle/Beginning-Cryptography-with-Java.productCd-0764596330.html

PKI:

- *Llibre "The Open-source PKI Book"*.
  Implementació de una PKI.
  http://ospkibook.sourceforge.net/
- *OpenSSL command line tool.*
  Manual d'us de l'eina *openssl*.
  http://www.openssl.org/docs/apps/openssl.html

Tomcat i Eclipse:

- *Eclipse IDE for Java EE Developers.*
  Descarrega i instal·lació de Eclipse en la versió Indigo.
  http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/indigor
- *Apache Tomcat 7 Installation*.
  Instal·lació del servidor Tomcat en la versió 7.
  http://tomcat.apache.org/tomcat-7.0-doc/appdev/installation.html
- *Tutorial: Installing Tomcat 7 and Using it with Eclipse.*
  Instal·lació de Eclipse i Tomcat 7, integrant el servidor dintre de Eclipse.
  http://www.coreservlets.com/Apache-Tomcat-Tutorial/tomcat-7-with-eclipse.html

Base de dades i persistència:

- *Using de 10 Core and 1.1 UI Derby plug-ins.*
  Manual de instal·lació i treball amb el plug-in de Derby per Eclipse.
  http://db.apache.org/derby/integrate/plugin_howto.html
- *Installing and Configuring EclipseLink.*
  Instal·lació i configuració de EclipseLink.
  http://wiki.eclipse.org/EclipseLink/Installing_and_Configuring_EclipseLink
- *Article "Using the Java Persistence API in Desktop Applications".*
  http://java.sun.com/developer/technicalArticles/J2SE/Desktop/persistenceapi/
- *Manual de JPA 2.0*
  http://www.davidmarco.es/blog/entrada.php?id=144

Java:

- *Apunts assignatura "Programació en Java" de la UOC.*
  Coneixements generals de programació en Java.

- *OpenJDK*
  http://openjdk.java.net/
- *API Java SE 6*.
  http://download.oracle.com/javase/6/docs/api/
- *API Java EE 6*.
  http://download.oracle.com/javaee/6/api/
- The Java EE 6 Tutorial.
  http://docs.oracle.com/javaee/6/tutorial/doc/
- *API Bouncy Castle*.
  http://www.bouncycastle.org/docs/docs1.6/index.html
  http://www.bouncycastle.org/docs/mdocs1.6/index.html

HTML:

- *Manual de HTML.*
  Tutorial de llenguatge HTML.
  http://www.desarrolloweb.com/manuales/21/
- *Referencia de llenguatge HTML 4.*
  http://www.htmlhelp.org/reference/html40/

Servlets:

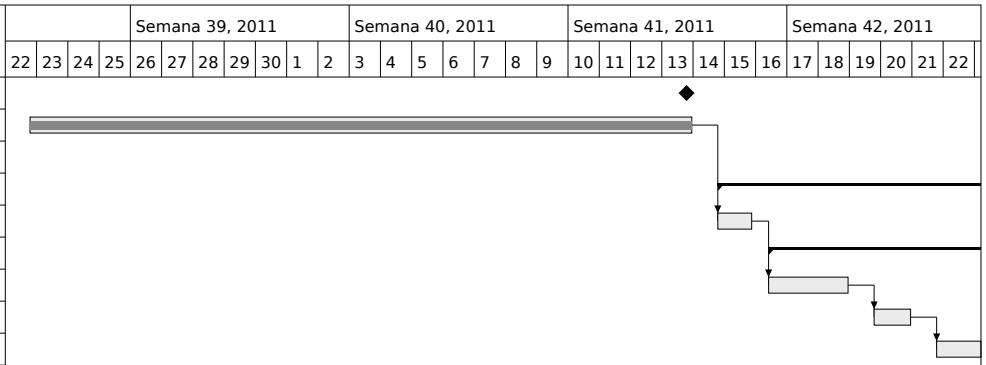
- *Building Web Apps in Java: Beginning & Intermediate Servlet & JSP Tutorials.*
  Tutorial de servlets i JSP.
  http://courses.coreservlets.com/Course-Materials/csajsp2.html
- *Llibre "Java servlet programming".*
  Programació de servlets.
  Accés amb un enllaç a la Biblioteca de la UOC.
- *Apache Commons FileUpload.*
  Llibreries en Java per pujar fitxer a un servidor.
  http://commons.apache.org/fileupload/index.html

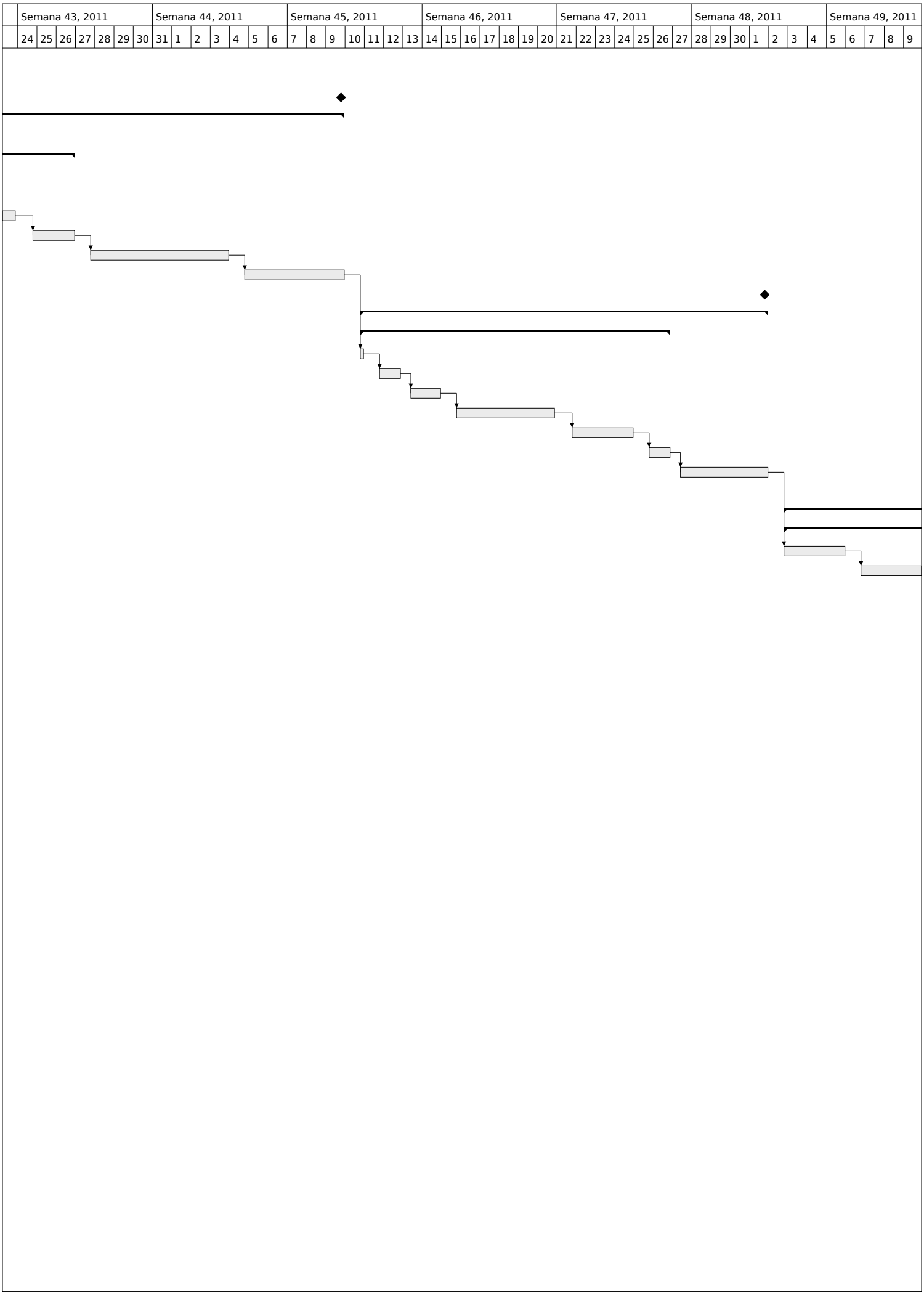
General:

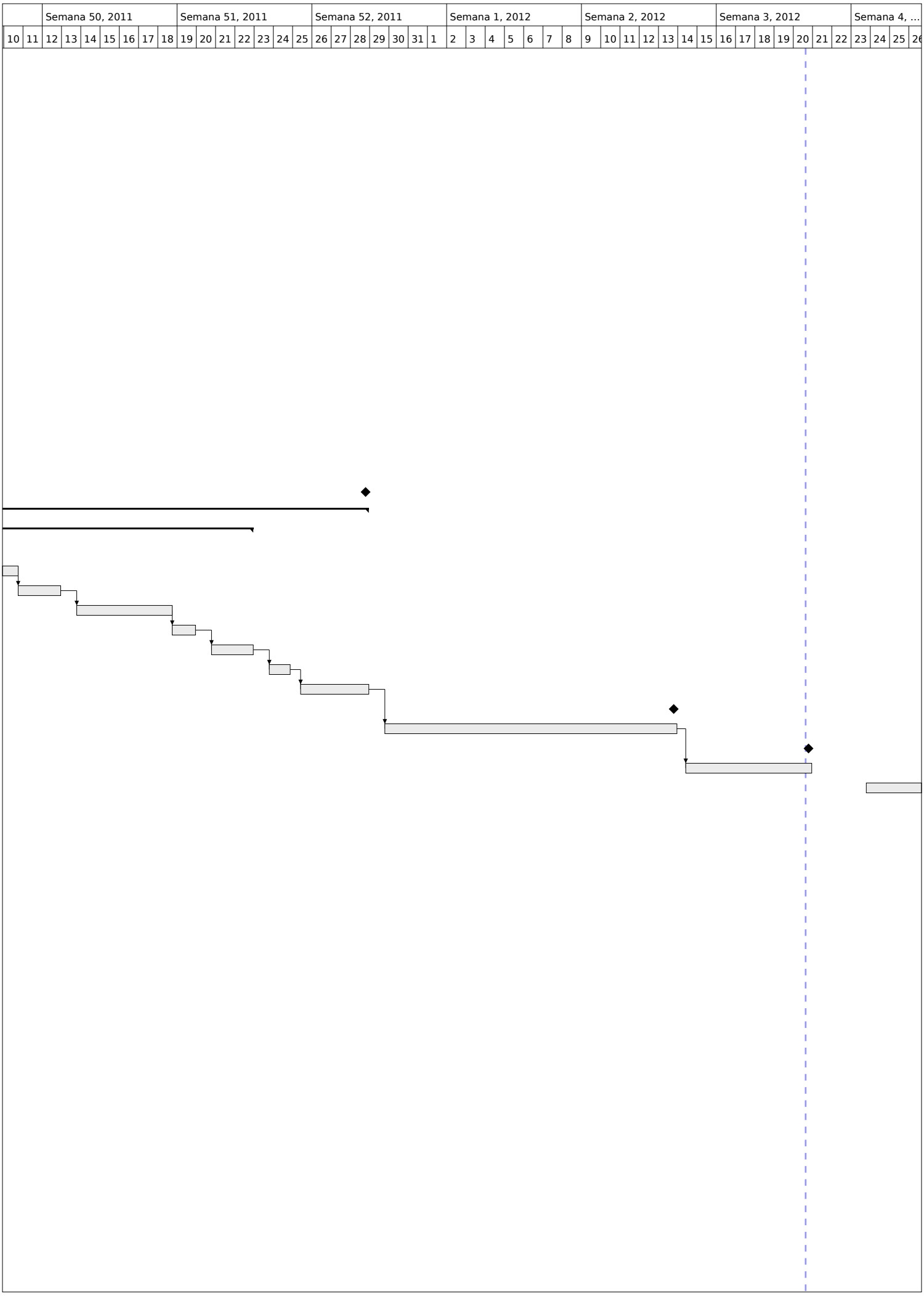
- *Wikipedia*:
  Tant en la seva versió anglesa (http://en.wikipedia.org/) com la espanyola (http://es.wikipedia.org/), per tenir una visió general de determinades tecnologies i com a font de nous recursos més especialitzats a la xarxa.

Annex I: Planificació de tasques inicial

Nombre	Trab...
Lliurament PAC1	
Pla de treball i estat de l'art	30d
Lliurament PAC2	
Posada en marxa de la CA	**37d 2h**
Servidor web J2EE	3d 1h
Aplicació Java al servidor	**15d**
Creació d'un magatzem de claus	4d 1h
Generació d'un parell de claus	2d
Generació de certificat autosignat de la CA	5d 2h
Guardar la clau privada i el certificat	3d
Interfície web	10d 2h
Autenticació local	8d 2h
Lliurament PAC3	
Creació de certificats dels clients	**30d**
Aplicació Java al servidor	**23d 2h**
Generació de parell de claus d'usuari	1d
Creació de certificat d'usuari	3d 1h
Creació de fitxer PKCS#12	3d 1h
Base de dades de certificats	8d 2h
Cerques de certificats	4d
Renovació de certificats	3d 1h
Interfície web	6d 1h
Lliurament PAC4	
Entorn d'administració de la CA	**37d 2h**
Aplicació Java al servidor	**29d**
Consulta de certificats emesos	6d 2h
Revocació de certificats	5d 1h
Publicació de CRL	4d 1h
Autenticació remota d'administradors	7d 2h
Alta i baixa d'usuaris administradors	2d
Reactivació de certificats revocats	3d
Interfície web	3d 1h
Protocol SSL	5d 1h
Lliurament memòria	
Memòria i producte	21d 1h
Lliurament presentació	
Presentació	9d 2h
Torn de preguntes	8d




Semana 43, 2011
Semana 44, 2011
Semana 45, 2011
Semana 46, 2011
Semana 47, 2011
Semana 48, 2011
Semana 49, 2011
24 25 26 27 28 29 30 31 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 1 2 3 4 5 6 7 8 9

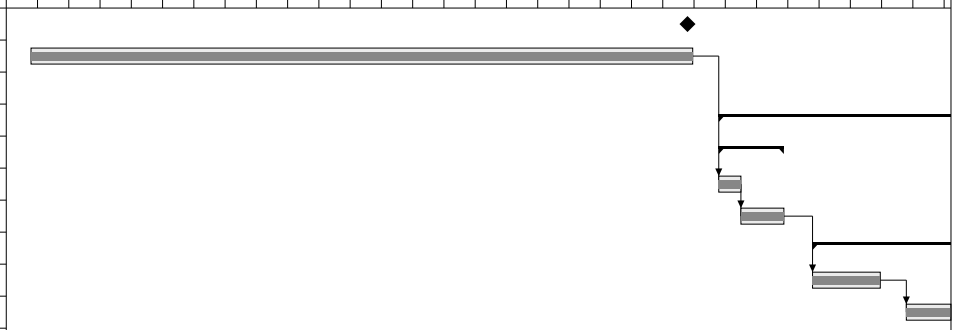

Semana 50, 2011
Semana 51, 2011
Semana 52, 2011
Semana 1, 2012
Semana 2, 2012
Semana 3, 2012
Semana 4, ...
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

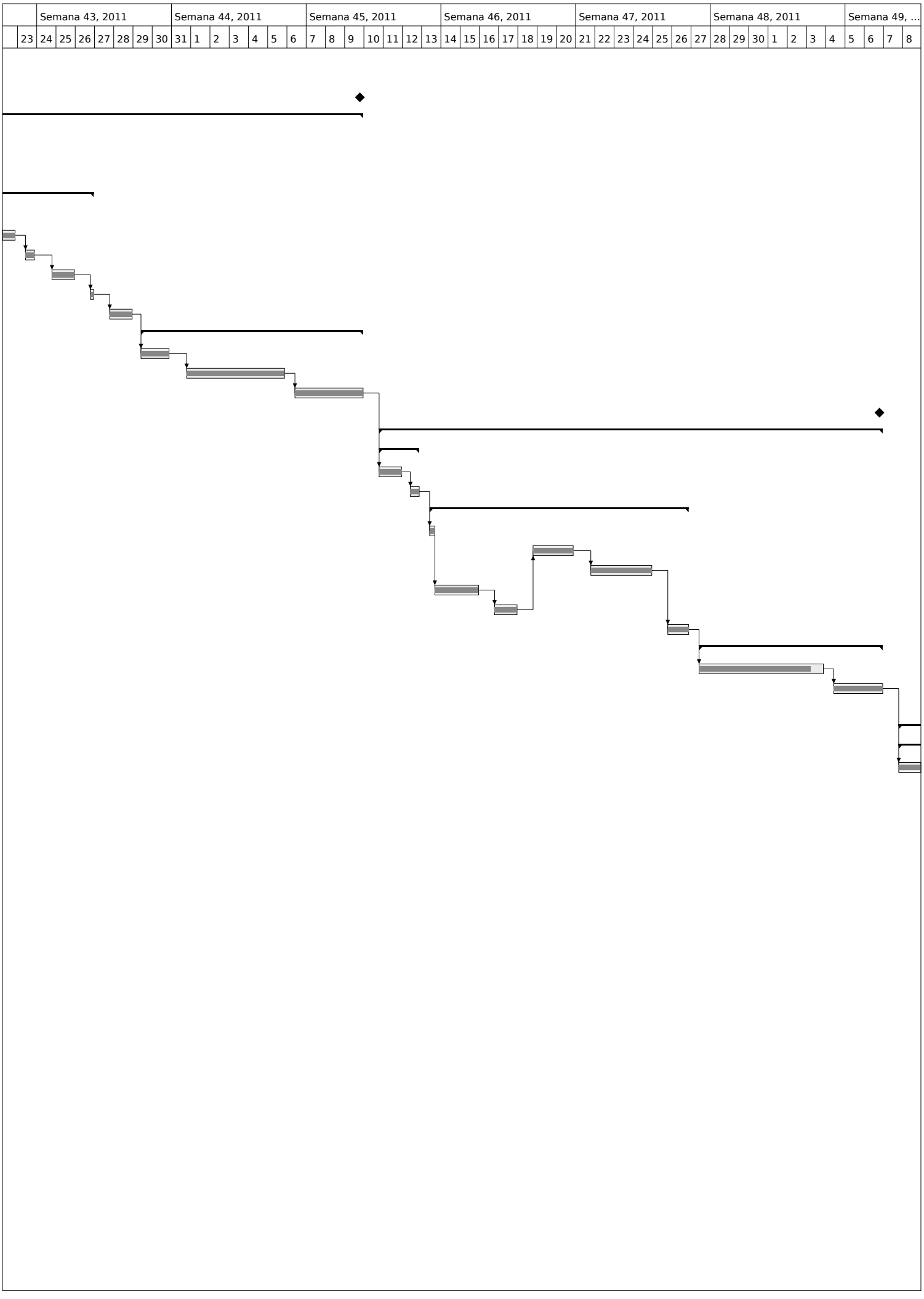
	Semana 5, 2012							Semana 6, 2012							Semana 7, 2012							Semana 8, 2012							Semana 9, 2012							Semana 10, 2012							Seman..
27	28	29	30	31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	1	2	3	4	5	6	7	8	9	10	11	12	13	.

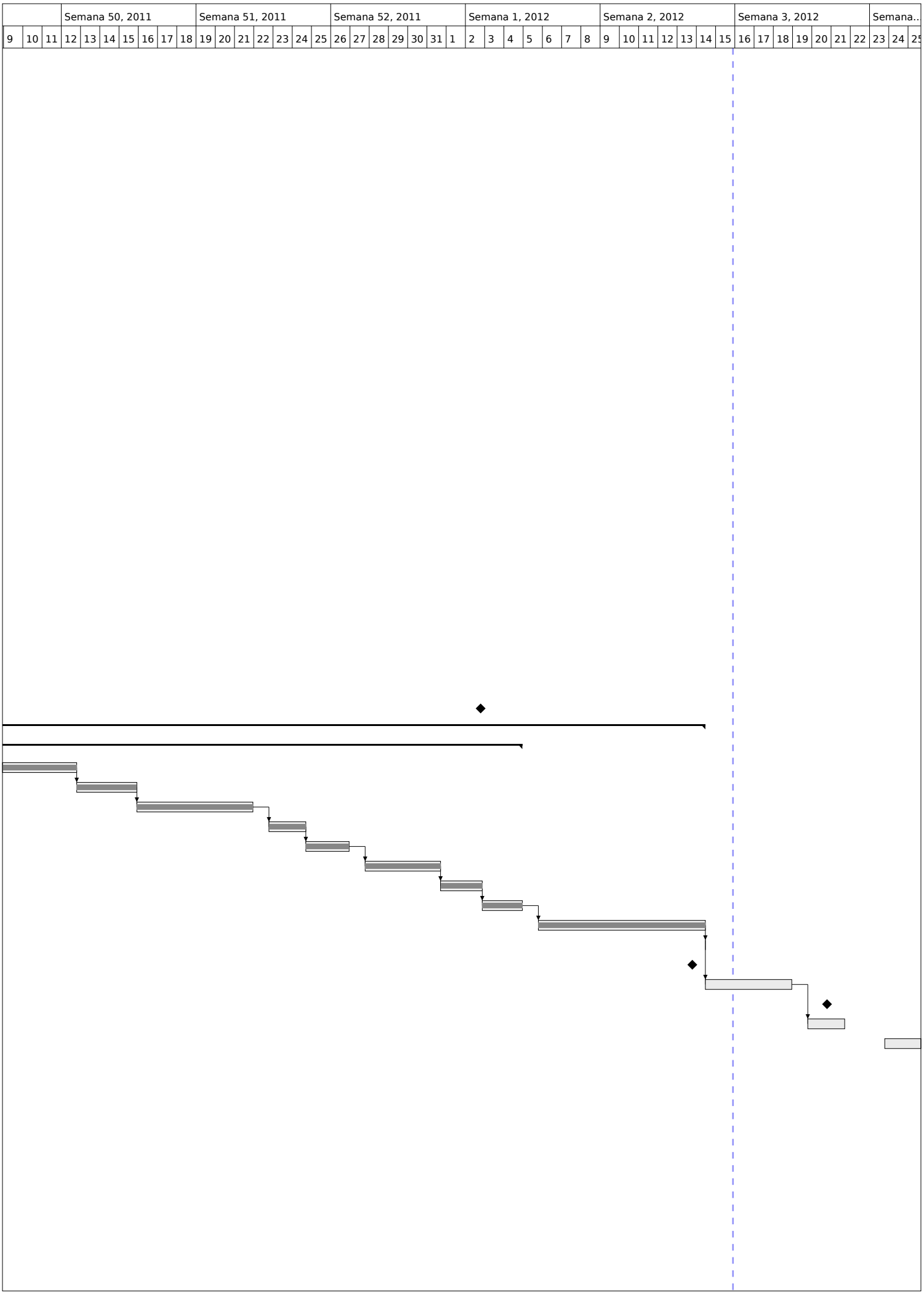
WBS	Nombre	Inicio	Fin	Trabajo	Duración	Desperdicio	Coste	Asignado a	% Completado
1	Lliurament PAC1	oct 13	oct 13	N/D	N/D	147d	0		0
2	Pla de treball i estat de l'art	sep 22	oct 13	30d	30d	9d 2h	0		100
3	Lliurament PAC2	nov 9	nov 9	N/D	N/D	109d 1h	0		0
4	**Posada en marxa de la CA**	**oct 14**	**nov 9**	**37d 2h**	**37d 2h**	**108d 1h**	**0**		0
4.1	Servidor web J2EE	oct 14	oct 15	3d 1h	3d 1h	9d 2h	0		0
4.2	**Aplicació Java al servidor**	**oct 16**	**oct 26**	**15d**	**15d**	**127d 2h**	**0**		0
4.2.1	Creació d'un magatzem de claus	oct 16	oct 18	4d 1h	4d 1h	9d 2h	0		0
4.2.2	Generació d'un parell de claus	oct 19	oct 20	2d	2d	9d 2h	0		0
4.2.3	Generació de certificat autosignat de la CA	oct 21	oct 23	5d 2h	5d 2h	9d 2h	0		0
4.2.4	Guardar la clau privada i el certificat	oct 24	oct 26	3d	3d	9d 2h	0		0
4.3	Interfície web	oct 27	nov 3	10d 2h	10d 2h	9d 2h	0		0
4.4	Autenticació local	nov 4	nov 9	8d 2h	8d 2h	9d 2h	0		0
5	Lliurament PAC3	dic 1	dic 1	N/D	N/D	79d 1h	0		0
6	**Creació de certificats dels clients**	**nov 10**	**dic 1**	**30d**	**30d**	**78d 1h**	**0**		0
6.1	**Aplicació Java al servidor**	**nov 10**	**nov 26**	**23d 2h**	**23d 2h**	**84d 2h**	**0**		0
6.1.1	Generació de parell de claus d'usuari	nov 10	nov 10	1d	1d	9d 2h	0		0
6.1.2	Creació de certificat d'usuari	nov 11	nov 12	3d 1h	3d 1h	9d 2h	0		0
6.1.3	Creació de fitxer PKCS#12	nov 13	nov 14	3d 1h	3d 1h	9d 2h	0		0
6.1.4	Base de dades de certificats	nov 15	nov 20	8d 2h	8d 2h	9d 2h	0		0
6.1.5	Cerques de certificats	nov 21	nov 24	4d	4d	9d 2h	0		0
6.1.6	Renovació de certificats	nov 25	nov 26	3d 1h	3d 1h	9d 2h	0		0
6.2	Interfície web	nov 27	dic 1	6d 1h	6d 1h	9d 2h	0		0
7	Lliurament PAC4	dic 28	dic 28	N/D	N/D	41d 2h	0		0
8	**Entorn d'administració de la CA**	**dic 2**	**dic 28**	**37d 2h**	**37d 2h**	**40d 2h**	**0**		0
8.1	**Aplicació Java al servidor**	**dic 2**	**dic 22**	**29d**	**29d**	**49d 1h**	**0**		0
8.1.1	Consulta de certificats emesos	dic 2	dic 5	6d 2h	6d 2h	9d 2h	0		0
8.1.2	Revocació de certificats	dic 6	dic 10	5d 1h	5d 1h	9d 2h	0		0
8.1.3	Publicació de CRL	dic 10	dic 12	4d 1h	4d 1h	9d 2h	0		0
8.1.4	Autenticació remota d'administradors	dic 13	dic 18	7d 2h	7d 2h	9d 2h	0		0
8.1.5	Alta i baixa d'usuaris administradors	dic 18	dic 19	2d	2d	9d 2h	0		0
8.1.6	Reactivació de certificats revocats	dic 20	dic 22	3d	3d	9d 2h	0		0
8.2	Interfície web	dic 23	dic 24	3d 1h	3d 1h	9d 2h	0		0
8.3	Protocol SSL	dic 25	dic 28	5d 1h	5d 1h	9d 2h	0		0
9	Lliurament memòria	ene 13	ene 13	N/D	N/D	20d 1h	0		0
10	Memòria i producte	dic 29	ene 13	21d 1h	21d 1h	9d 2h	0		0
11	Lliurament presentació	ene 20	ene 20	N/D	N/D	10d 2h	0		0
12	Presentació	ene 14	ene 20	9d 2h	9d 2h	9d 2h	0		0
13	Torn de preguntes	ene 23	ene 27	8d	5d		0		0

Annex II: Planificació de tasques final

Nombre	Trab...
Lliurament PAC1	
Pla de treball i estat de l'art	30d
Lliurament PAC2	
Posada en marxa de la CA	**37d 2h**
Entorn de desenvolupament	**5d 2h**
Instal.lació d'un IDE	1d 2h
Instal.lació d'un servidor web	4d
Aplicació Java al servidor	**12d 2h**
Creació de la Autoritat Certificació	3d
Creació d'un magatzem de claus	4d 1h
Generació de claus de la CA	2d 1h
Generació de certificat autosignat de la CA	2d
Guardar la clau privada i el certificat	1d
Autenticació local	2d
Interfície web	**17d 1h**
Plana inicial de la CA	4d 2h
Plana per autenticació d'administradors	7d 1h
Plana per administració de la CA	5d 1h
Lliurament PAC3	
Creació de certificats dels clients	**37d 2h**
Entorn de desenvolupament	**4d 1h**
Instal.lació d'un servidor de bases de dades	2d
Instal.lació d'un implementació de JPA	2d 1h
Aplicació Java al servidor	**19d 1h**
Generació de parell de claus d'usuari	2h 30...
Extreure la petició de certificació	5d 2h
Extreure el certificat autosignat	4d
Creació de certificat d'usuari	3d 1h
Creació de fitxer PKCS#12	2d
Mètodes per la generació de certificats	3d 1h
Interfície web	**14d**
Plana de sol.licitud de certificació	9d 2h
Modificació planes autenticació administradors	4d 1h
Lliurament PAC4	
Entorn d'administració de la CA	**51d 2h**
Aplicació Java al servidor	**39d 2h**
Cerques de certificats	7d 2h
Renovació de certificats	3d 2h
Consulta de certificats emesos	8d 2h
Revocació de certificats	3d
Publicació de CRL	4d 2h
Autenticació remota d'administradors	5d
Alta i baixa d'usuaris administradors	4d 1h
Reactivació de certificats revocats	2d 1h
Interfície web	12d
Protocol SSL	
Lliurament memòria	
Memòria i producte	7d
Lliurament presentació	
Presentació	3d
Torn de preguntes	8d




Semana 43, 2011
Semana 44, 2011
Semana 45, 2011
Semana 46, 2011
Semana 47, 2011
Semana 48, 2011
Semana 49, ...
23 24 25 26 27 28 29 30 31 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 1 2 3 4 5 6 7 8


Semana 50, 2011
Semana 51, 2011
Semana 52, 2011
Semana 1, 2012
Semana 2, 2012
Semana 3, 2012
Semana..
9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

	Semana 5, 2012							Semana 6, 2012							Semana 7, 2012							Semana 8, 2012							Semana 9, 2012							Semana 10, 2012							Se...
26	27	28	29	30	31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	1	2	3	4	5	6	7	8	9	10	11	12	.

WBS	Nombre	Inicio	Fin	Trabajo	Duración	Desperdicio	Coste	Asignado a	% Completado
1	Lliurament PAC1	oct 13	oct 13	N/D	N/D	147d	0		0
2	Pla de treball i estat de l'art	sep 22	oct 13	30d	30d	8d 2h	0		100
3	Lliurament PAC2	nov 9	nov 9	N/D	N/D	109d 1h	0		0
4	**Posada en marxa de la CA**	**oct 14**	**nov 9**	**37d 2h**	**37d 2h**	**108d 1h**	**0**		0
4.1	**Entorn de desenvolupament**	**oct 14**	**oct 16**	**5d 2h**	**5d 2h**	**140d 1h**	**0**		0
4.1.1	Instal.lació d'un IDE	oct 14	oct 15	1d 2h	1d 2h	8d	0		100
4.1.2	Instal.lació d'un servidor web	oct 15	oct 16	4d	4d	7d 1h	0		100
4.2	**Aplicació Java al servidor**	**oct 17**	**oct 26**	**12d 2h**	**12d 2h**	**127d 2h**	**0**		0
4.2.1	Creació de la Autoritat Certificació	oct 17	oct 19	3d	3d	8d 2h	0		100
4.2.2	Creació d'un magatzem de claus	oct 20	oct 22	4d 1h	4d 1h	7d 1h	0		100
4.2.3	Generació de claus de la CA	oct 23	oct 23	2d 1h	2d 1h	7d 1h	0		100
4.2.4	Generació de certificat autosignat de la CA	oct 24	oct 25	2d	2d	8d 2h	0		100
4.2.5	Guardar la clau privada i el certificat	oct 26	oct 26	1d	1d	8d 2h	0		100
4.3	Autenticació local	oct 27	oct 28	2d	2d	8d 2h	0		100
4.4	**Interfície web**	**oct 29**	**nov 9**	**17d 1h**	**17d 1h**	**108d 1h**	**0**		0
4.4.1	Plana inicial de la CA	oct 29	oct 30	4d 2h	4d 2h	7d 1h	0		100
4.4.2	Plana per autenticació d'administradors	oct 31	nov 5	7d 1h	7d 1h	7d 1h	0		100
4.4.3	Plana per administració de la CA	nov 6	nov 9	5d 1h	5d 1h	8d 2h	0		100
5	Lliurament PAC3	dic 6	dic 6	N/D	N/D	71d 2h	0		0
6	**Creació de certificats dels clients**	**nov 10**	**dic 6**	**37d 2h**	**37d 2h**	**70d 2h**	**0**		0
6.1	**Entorn de desenvolupament**	**nov 10**	**nov 12**	**4d 1h**	**4d 1h**	**104d**	**0**		0
6.1.1	Instal.lació d'un servidor de bases de dades	nov 10	nov 11	2d	2d	8d 2h	0		100
6.1.2	Instal.lació d'un implementació de JPA	nov 12	nov 12	2d 1h	2d 1h	7d 1h	0		100
6.2	**Aplicació Java al servidor**	**nov 13**	**nov 26**	**19d 1h**	**19d 1h**	**84d 2h**	**0**		0
6.2.1	Generació de parell de claus d'usuari	nov 13	nov 13	2h 30min	2h 30min	8d 2h	0		100
6.2.2	Extreure la petició de certificació	nov 18	nov 20	5d 2h	5d 2h	7d 1h	0		100
6.2.3	Extreure el certificat autosignat	nov 21	nov 24	4d	4d	8d 2h	0		100
6.2.4	Creació de certificat d'usuari	nov 13	nov 15	3d 1h	3d 1h	8d 2h	0		100
6.2.5	Creació de fitxer PKCS#12	nov 16	nov 17	2d	2d	8d 2h	0		100
6.2.6	Mètodes per la generació de certificats	nov 25	nov 26	3d 1h	3d 1h	7d 1h	0		100
6.3	**Interfície web**	**nov 27**	**dic 6**	**14d**	**14d**	**70d 2h**	**0**		0
6.3.1	Plana de sol.licitud de certificació	nov 27	dic 3	9d 2h	9d 2h	7d 1h	0		90
6.3.2	Modificació planes autenticació administradors	dic 4	dic 6	4d 1h	4d 1h	8d 2h	0		100
7	Lliurament PAC4	ene 2	ene 2	N/D	N/D	34d	0		0
8	**Entorn d'administració de la CA**	**dic 7**	**ene 14**	**51d 2h**	**51d 2h**	**18d 2h**	**0**		0
8.1	**Aplicació Java al servidor**	**dic 7**	**ene 4**	**39d 2h**	**39d 2h**	**31d**	**0**		0
8.1.1	Cerques de certificats	dic 7	dic 12	7d 2h	7d 2h	9d 1h	0		100
8.1.2	Renovació de certificats	dic 12	dic 15	3d 2h	3d 2h	8d 2h	0		100
8.1.3	Consulta de certificats emesos	dic 15	dic 21	8d 2h	8d 2h	8d 2h	0		100
8.1.4	Revocació de certificats	dic 22	dic 24	3d	3d	8d 2h	0		100
8.1.5	Publicació de CRL	dic 24	dic 26	4d 2h	4d 2h	8d 2h	0		100
8.1.6	Autenticació remota d'administradors	dic 27	dic 31	5d	5d	8d 2h	0		100
8.1.7	Alta i baixa d'usuaris administradors	dic 31	ene 2	4d 1h	4d 1h	9d	0		100
8.1.8	Reactivació de certificats revocats	ene 2	ene 4	2d 1h	2d 1h	8d 2h	0		100
8.2	Interfície web	ene 5	ene 14	12d	12d	8d	0		100
8.3	Protocol SSL	ene 14	ene 14			8d	0		0
9	Lliurament memòria	ene 13	ene 13	N/D	N/D	20d 1h	0		0
10	Memòria i producte	ene 14	ene 18	7d	7d	8d 2h	0		0
11	Lliurament presentació	ene 20	ene 20	N/D	N/D	10d 2h	0		0
12	Presentació	ene 19	ene 21	3d	3d	8d 2h	0		0
13	Torn de preguntes	ene 23	ene 27	8d	5d		0		0