TFC JEE - Gestor documental

Autor: **Jordi Blasco Planesas**

Enginyeria Tècnica en Informàtica de Sistemes

Consultor: **Jose Juan Rodríguez**

14 de Gener de 2008

Dedicat a

Eva,

Moltes gràcies per la teva paciència i comprensió al llarg de tots aquests anys d'estudis a distància, i en especial per aquests darrers semestres; gràcies pel suport incondicional rebut en tot moment, i per saber que davant qualsevol eventualitat sempre estaves al costat.

Agraïments

A tota la família de la UOC, començant pels companys, seguint pels consultors, i acabant per un tutor que en cada trobada té paraules de motivació i que ajuden a fer menys feixuga la tasca de continuar estudiant. Al meu germà David, per utilitzar un problema habitual en el seu món de les cerques històriques i donar com a resultat el present treball. A la família, amics i companys de feina, per la vostra comprensió mentre realitzava aquests estudis. A Jose Juan Rodríguez, per l'empenta i la motivació que m'ha anat donant al llarg del curs, per la seva enorme paciència i per saber que en tot moment al llarg de la realització d'aquest TFC he tingut el seu suport inestimable.

1 Resum

El present treball final de carrera es basa en la creació d'una aplicació web que permeti tant a entitats locals com a particulars, la gestió completa d'un arxiu de documents. Conseqüentment, l'estudi queda centrat en les fases d'especificació, anàlisi, disseny, implementació, prova i manteniment del present projecte, i més en concret de cadascun dels subsistemes que el formen: identificació, usuaris, catàleg i préstecs.

La particularitat del TFC radica en l'aspecte del disseny i de la implementació, ja que si bé al llarg de la carrera s'ha tingut l'ocasió d'estudiar la creació de programari, en el present cas s'introdueix un nou aspecte: l'ús de la tecnologia JEE pròpia de Java. Així, a part de mostrar pas a pas tot el procés de creació d'un producte, s'ha hagut d'analitzar com l'arquitectura JEE ajuda a que el programari resultant sigui més robust, escalable, reutilitzable i senzill d'implementar i mantenir, característiques del tot imprescindibles en qualsevol projecte d'avui en dia.

Per tal d'aprofitar les característiques de l'arquitectura JEE s'utilitza Spring MVC, un subconjunt del framework Spring. Malgrat no es tracta d'una solució tan consolidada com pot ser Struts, sí que presenta un enorme potencial de recorregut, i més tenint en compte les funcionalitats que actualment ja ofereix. Així, Spring MVC esdevé l'element de cohesió de tot el TFC, sent l'esquelet que sosté una nova forma de concebre i distribuir les aplicacions, i que explota completament la programació orientada a objectes.

2 Índex de continguts

3 Índex de figures

4 Cos de la memòria

4.1 Capítol 1: Introducció

4.1.1 Justificació i context

El TFC se centra en la creació d'un aplicatiu web que permet la gestió completa d'un arxiu de documents, tant si es vol utilitzar en arxius particulars com si es vol fer per entitats locals petites o mitjanes; l'elecció ha estat condicionat, en gran part, degut al buit trobat en aquest sector en els darrers anys.

Servint-me de l'experiència del meu germà en l'estudi de l'arbre de família i altres temes de caire històric, s'ha pogut observar com gran quantitat d'arxius, tant municipals com privats, o bé no disposen d'un catàleg, o bé en cas d'haver indexat tota la documentació en alguna aplicació informàtica quasi sempre és accessible únicament de forma presencial. Ambdós escenaris acaben causant la mateixa situació no desitjada: la pèrdua d'una enorme quantitat de temps als investigadors en qualsevol tasca que vulguin fer de recerca d'informació en els arxius.

El que pretén aquest TFC és desenvolupar una eina que permeti, per una banda, la gestió completa d'un arxiu, sigui de la naturalesa que sigui, i d'altra banda permetre'n la seva consulta a través de d'Internet, de manera que tot el temps que es poden estalviar en recerca molts cops inútils es pugui invertir en tasques més productives.

4.1.2 Objectius

El principal objectiu del TFC en quan a funcionalitat, és la creació d'una aplicació per mitjà de la tecnologia JEE que, a través d'una interfície web, pugui gestionar tots aquells aspectes propis d'un arxiu.

Per d'aconseguir aquest producte final, serà necessari assolir també els següents objectius:

- Detectar totes les necessitats en quant a funcionalitats i requeriments que una aplicació d'aquest tipus necessita.
- Saber planificar la temporització del TFC, de manera que es pugui adequar el temps a les necessitats reals del mateix.
- Conèixer la filosofia JEE i tots aquells elements que en formen part. Aquest punt serà un dels més importants, ja que requerirà un continu aprenentatge al llarg de tota la realització del TFC i no només de la part d'implementació.
- Programar de forma clara, utilitzant sempre que es pugui plantejaments que permetin la reutilització, i facilitant al màxim futures possibles tasques d'actualització de codi.
- Escollir i conèixer mínimament els productes necessaris per l'execució i posada en marxa del TFC, basant-se amb les seves possibilitats i, sobretot, en la seva gratuïtat (productes Open Source). Aquest segon aspecte és important, ja que es vol aconseguir una eina el més econòmica possible, tant en el seu desenvolupament com en la seva posada en marxa,

i això passa irremediablement per la no necessitat de pagament de llicències de software.

- Establir un diàleg fluid i dinàmic amb el consultor, ja que la realització del TFC és quasi impossible de concebre sense la seva experiència, comentaris, consells i directrius a seguir.

4.1.3 Enfocament i mètode seguit

L'enfocament seguit ha estat completament condicionat pels nuls coneixements que es tenen sobre Spring MVC com a punt de partida i, en general, sobre tot el que es refereix a la tecnologia JEE de Java. Per aquest motiu, al llarg del curs s'ha hagut de treballar contínuament en paral·lel: d'una banda implementant seqüencialment cadascuna de les fases que componen la creació d'una aplicació informàtica, i d'altra banda estudiant i destriant les possibilitats que ofereix Spring i escollint aquelles més interessants per la realització del TFC.

Es vol insistir molt en aquest darrer aspecte, ja que si bé la manca de coneixements de JEE i de qualsevol solució optimitzada (framework) ha estat el factor realment motivador del projecte, les hores que s'hi han hagut d'invertir han condicionat completament el resultat obtingut.

Per tal d'encarar aquest escenari, s'ha utilitzat segurament el model iteratiu més conegut, Rational Unified Process, amb l'objectiu d'implementar les millors pràctiques de l'enginyeria de software. La metodologia RUP divideix en quatre fases el desenvolupament d'una aplicació:

Figura 1. Esquema de la metodologia RUP.

Fase	*Objectius*	*Punt de control*
Inici.	Definir l'abast del projecte i comprendre què es va a construir.	Objectiu del projecte.
Elaboració.	Construir una versió executable de l'arquitectura de l'aplicació, i comprendre com es va a construir.	Arquitectura de l'aplicació.
Construcció.	Completar l'esquelet de l'aplicació amb funcionalitat i construir una versió inicial (beta).	Versió operativa inicial de l'aplicació.
Transició.	Posar a disposició dels usuaris finals l'aplicació i construir la versió definitiva (final).	Versió operativa final de l'aplicació

Cadascuna d'aquestes fases està desenvolupada per mitjà del cicle d'iteracions, el qual consisteix en reproduir a menor escala el cicle de vida en cascada. En el cas present, consisteix en utilitzar seqüencialment el cicle de vida en cascada a cadascun dels quatre subsistemes (identificació, usuaris, catàleg i préstecs), de manera que primer de tot s'aplica el cicle de vida a un d'ells, es detecten i corregeixen errors, s'implementa un codi que pot ser reutilitzat en els següents subsistemes... en definitiva, s'obté un subsistema finalitzat i s'agafa una experiència que a continuació s'aplica al desenvolupament del següent subsistema. Així, s'obté que el treball (esforç) a realitzar a l'inici és molt elevat, però aquest es va reduint de forma inversament exponencial a mesura que creix l'experiència.

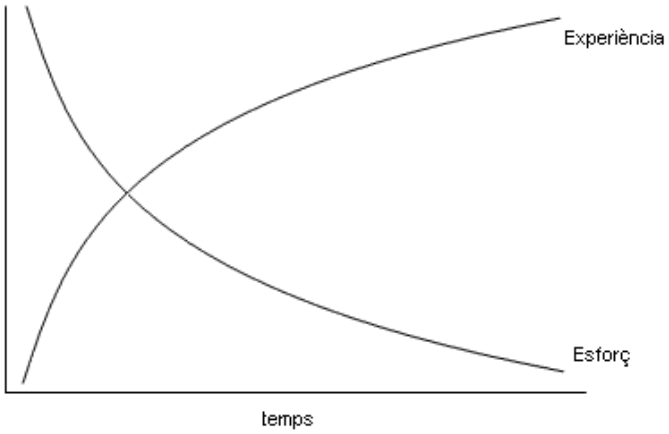

Figura 2. Relació esforç – experiència amb RUP.

4.1.4 Planificació

El projecte s'ha plantejat en base a les fases habituals del cicle de vida del programari, i en els períodes de temps delimitats per les entregues de les PAC realitzades durant el curs:

- Planificació: primera fase del TFC, que inclou la temporalització del propi TFC i la ubicació del context sobre el qual es realitzarà el treball. Correspon a la fase d'especificació i finalitza amb l'entrega de la PAC1 (28.09.07)

- Anàlisi i Disseny: segona i tercera fase de desenvolupament del programari, que contenen una descripció textual detallada de totes les funcionalitats, tota la relació de diagrames necessaris per mostrar el disseny de l'aplicació, el model ER, disseny de la base de dades, elecció justificada de patrons JEE, creació de l'estil global de tota la visió web, juntament amb la interfície de l'aplicació. Finalitza amb l'entrega de la PAC2 (29.10.07).

- Implementació i testeig: quarta i cinquena fase, on es realitza pròpiament la programació de l'aplicació. A més, també s'hi duu a terme el testeig de totes les opcions de l'aplicació i el corresponent control dels resultats obtinguts. Finalitza amb l'entrega de la PAC3 (17.12.07).

- Publicació: posada en producció de l'aplicació, creació de la documentació interna del codi de l'aplicació (JavaDoc), preparació de la memòria del TFC basada en l'entrega de les PAC i creació del document resum de presentació del TFC. Finalitza amb el lliurament final del TFC (14.01.08).

A més a més, hi haurà una etapa extra donada la singularitat del TFC, ja que es desconeix inicialment les possibilitats que ofereix la tecnologia JEE:

- Formació i aprofundiment de la tecnologia JEE. En aquesta fase s'escull la relació de programari que s'utilitzarà al llarg del curs, tant pel desenvolupament com de la posada en marxa de l'aplicació. Es realitza en paral·lel a les fases d'anàlisi i de disseny, i finalitza en començar la fase d'implementació; es marca aquest objectiu per tal d'assegurar que s'arriba a la fase d'implementació amb uns coneixements JEE bàsics i suficients per començar a treballar.

4.1.4.1 Fase de planificació

Conté totes les tasques inicials del projecte: elecció de la temàtica del TFC, i descripció genèrica del TFC (funcionalitats, objectius i la planificació). La fase finalitza amb el lliurament de la PAC1.

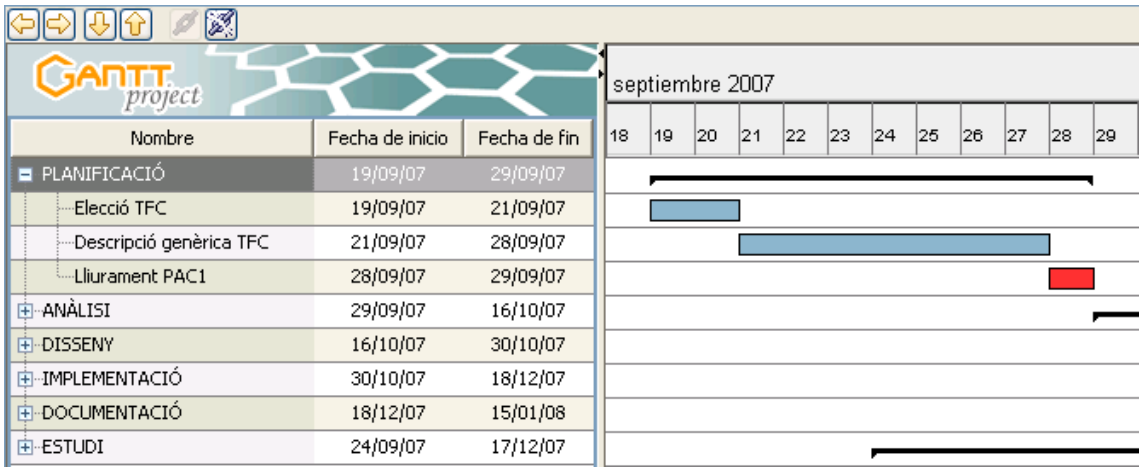

Figura 3. Diagrama de Gantt : fase de planificació.

4.1.4.2 Fase d'anàlisi

Conté la definició de tots els diagrames característics de l'anàlisi d'una aplicació informàtica: diagrames de casos d'ús, diagrames d'estats, diagrames de seqüència i diagrames de col·laboració, els quals ajuden a aprofundir un grau més en el projecte, clarificant llur funcionament intern. La fase no finalitza amb el lliurament de cap PAC, sinó que enllaça amb la següent de disseny de l'aplicació.

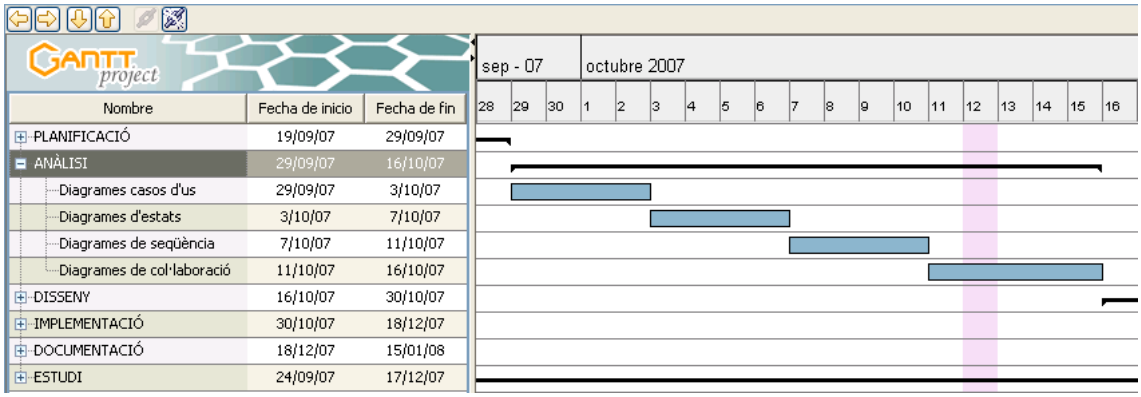

Figura 4. Diagrama de Gantt : fase d'anàlisi.

4.1.4.3 Fase de disseny

Conté els diagrames propis del disseny de l'aplicació (diagrama de classes, diagrama ER), el disseny de l'estructura de bases de dades que s'utilitza i l'elecció justificada de les tecnologies JEE emprades a l'aplicació. La fase conclou amb el lliurament de la PAC2.

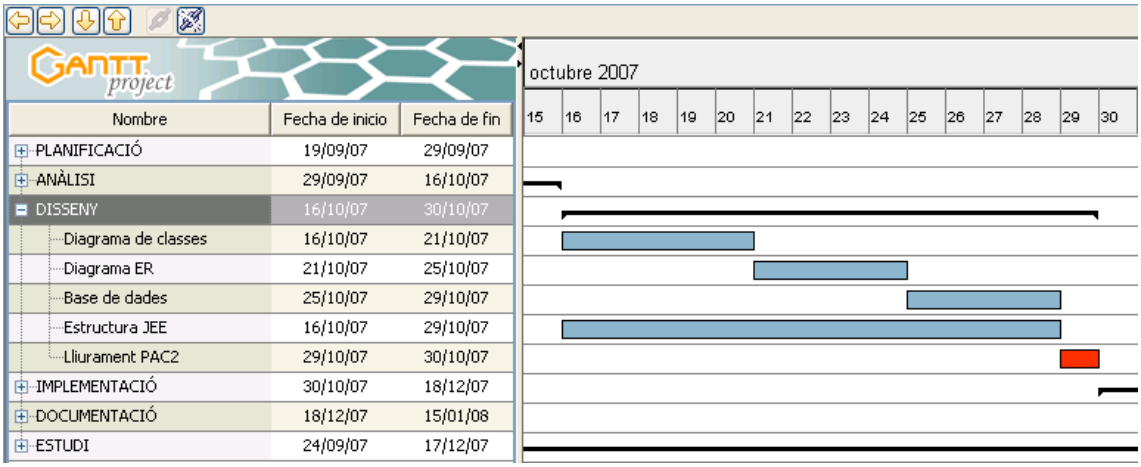

Figura 5. Diagrama de Gantt : fase de disseny.

4.1.4.4 Fase d'implementació

Durant aquesta fase es realitza pròpiament la programació a partir de les solucions que ofereix Spring. Es deixa per aquesta fase la creació de la interfície gràfica de l'aplicació, ja que si bé es comença a concebre durant la fase disseny, no és fins aquest punt que es decideix i es treballa l'aspecte visual definitiu. La fase finalitza amb el lliurament de la PAC3.

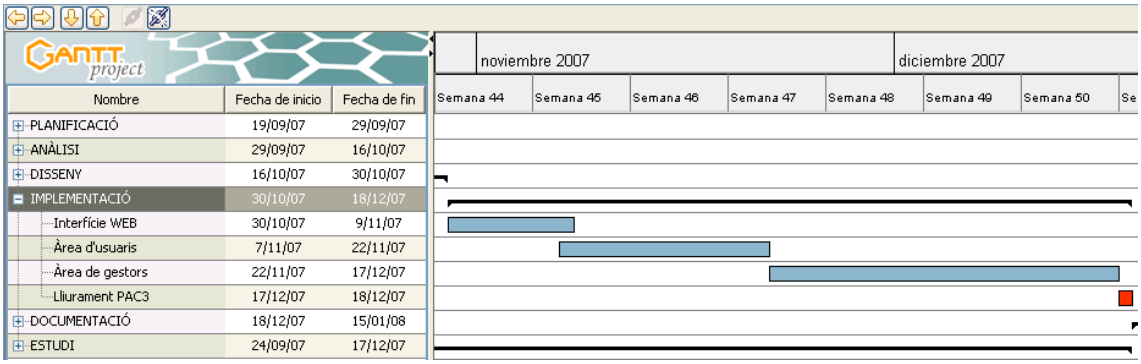

Figura 6. Diagrama de Gantt : fase d'implementació.

4.1.4.5 Fase de documentació

Conté els preparatius finals de documentació del TFC: d'una banda la creació de la documentació del codi (JavaDoc), i d'altra banda unir en una única Memòria la documentació presentada en les PAC i a més crear-ne un document de resum. La fase conclou amb el lliurament final del projecte.

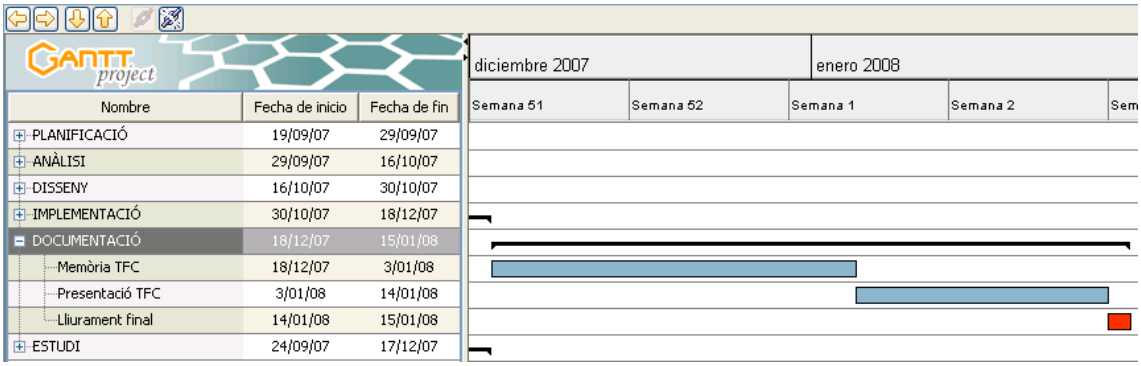

Figura 7. Diagrama de Gantt : fase de documentació.

4.1.4.6 Fase d'estudi.

Conté totes les tasques relacionades amb l'obtenció del coneixement necessari per l'elaboració del projecte. Donada la singularitat el mateix (es desconeixen inicialment les característiques que ofereix la tecnologia JEE, així com els seus avantatges i punts no tan forts), és necessari invertir una quantitat de temps important per poder afrontar el TFC amb un mínim de garanties. Comença a principis de curs i finalitza amb l'entrega de la PAC3.

A més, és durant aquesta fase del TFC que s'estudia, es configura i es testeja el programari gratuït per dur-lo a la pràctica, començant per les eines de desenvolupament (IDE, JDK, etc.) i finalitzant per les pròpies de producció (Tomcat, SGBD, patrons, etc.).

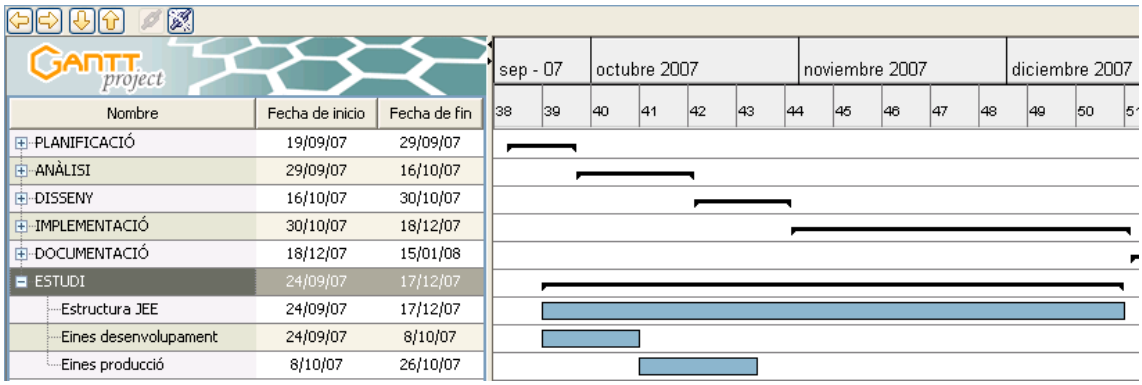

Figura 8. Diagrama de Gantt : fase d'estudi.

4.1.4.7 Conjunt de fases

A continuació es mostra el diagrama de Gantt, en setmanes, corresponent al conjunt de fases que composen el projecte.

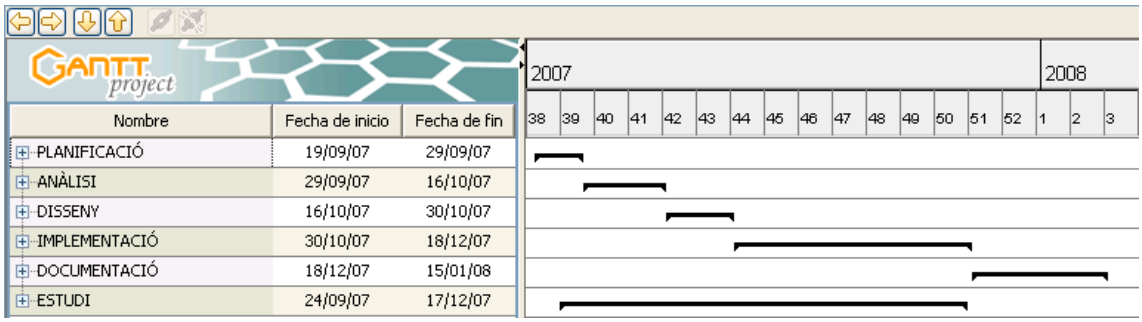

Figura 9. Diagrama de Gantt : conjunt de fases.

4.1.5 Productes obtinguts

El producte final obtingut és la implementació d'una aplicació web realitzada amb arquitectura JEE, i més concretament amb la visió definida que dóna el framework Spring.

La relació de productes obtinguts, a part de la Memòria i la Presentació, són:

- **gestorDocumental.war :** conté els binaris de les classes Java de l'aplicació, les llibreries necessàries per la seva execució, la interfície de l'aplicació i els fitxers de configuració.

- **script_sql.txt :** inicialitza la BDD per tal de començar a treballar amb l'aplicació amb un mínim de dades accessibles.

- **gestorDocumental-classes_java.zip :** conté el codi font de les classes Java incloses dins de l'arxiu gestorDocumental.war, per tal de mostrar la seva implementació i les

interrelacions que s'estableixen entre les diferents capes que conformen l'aplicació.

- **gestorDocumental-doc.zip :** la documentació (JavaDoc) corresponent al codi Java de l'aplicació.

4.1.6 Breu descripció dels altres capítols de la memòria

Els capítols que es descriuran a continuació són:

- **Anàlisi:** capítol dedicat a analitzar les funcionalitats que ofereix l'aplicació. S'hi inclou unes puntualitzacions inicials on es dóna un primer esbós de la lògica de negoci de l'aplicació (subsistemes), una relació de diagrames de casos d'ús, de diagrames d'estats i de diagrames de seqüència, que ajuden a comprendre en funcionament intern.

- **Disseny:** capítol encarregat de mostrar el disseny de l'aplicació. Hi tenen cabuda el diagrama de classes de l'aplicació, el diagrama ER, la persistència de dades i el disseny de la interfície (vista) de l'aplicació. També conté una descripció detallada sobre l'arquitectura JEE, es justifica l'elecció del framework Spring MVC com a esquelet del TFC i finalment es dóna un exemple detallat de com funciona internament l'aplicació.

- **Implementació:** capítol dedicat a la codificació de l'aplicació. S'hi explica amb detall les eines utilitzades per realitzar la implementació del projecte, l'estructura que s'ha seguit en el procés de creació, els passos necessaris pel seu desplegament, així com el testeig de certes funcionalitats.

4.2 Capítol 2: Anàlisi

4.2.1 Consideracions prèvies

Per tal de facilitar al màxim l'anàlisi del projecte, es divideix en quatre grans unitats lògiques de negoci: el subsistema d'identificació, el subsistema de catàleg, el subsistema d'usuaris i el subsistema de préstecs.

A nivell de diagrames, s'inclouen només aquells que són pròpiament representatius per la seva complexitat i importància dins del TFC, tret dels diagrames de casos d'ús, els quals s'han realitzat tots ja que permeten obtenir una visió global de tota l'aplicació –a part de no ser excessius en quant a nombre–.

4.2.2 Diagrames de casos d'ús

L'esquema general de casos d'ús del projecte és el següent:

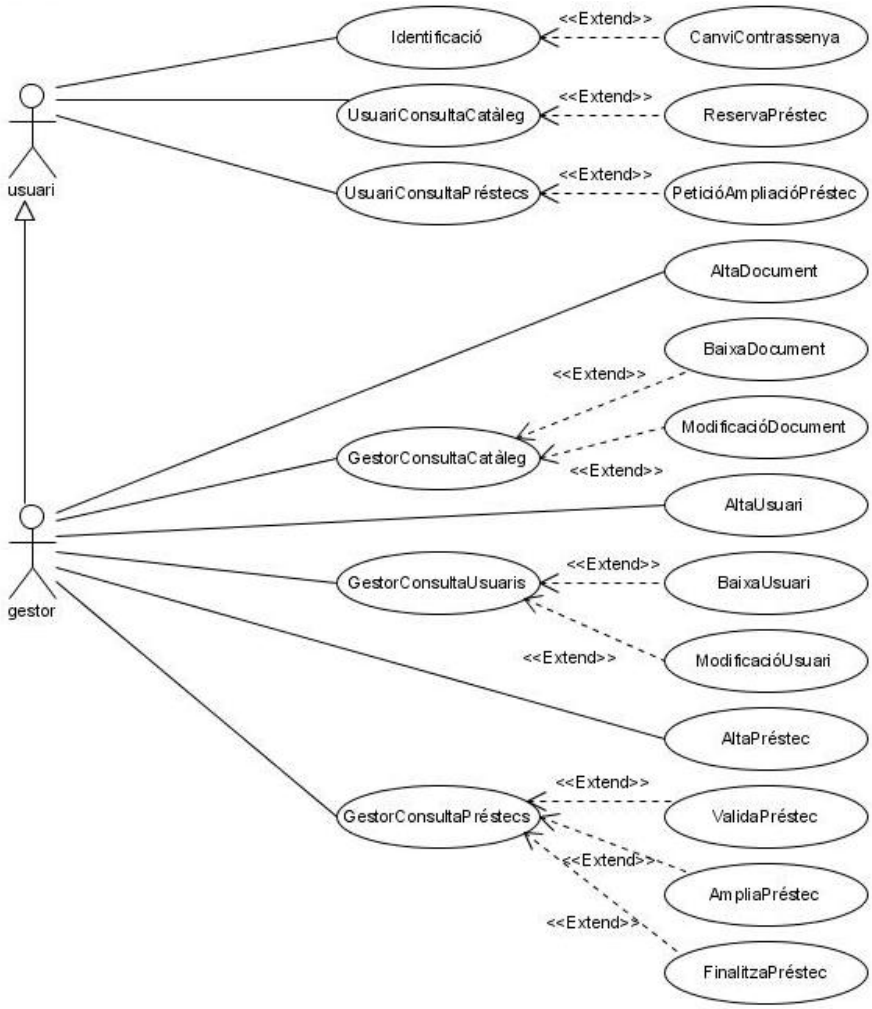

Figura 10. Esquema de casos d'ús: escenari general.

Puntualitzar que el que es busca és que la identificació inicial es faci únicament una vegada a l'inici de sessió, moment a partir del qual ja es permet l'accés a la resta de *cdu*[1]. Les relacions entre els diferents *cdu* són d'extensió, ja que els *cdu* de segon nivell de profunditat són part opcional en l'execució del corresponent *cdu* referenciat: a mode d'exemple, quan un usuari amb perfil gestor vulgui donar de baixa un usuari, inicialment haurà de realitzar una cerca entre tots els usuaris per tal de seleccionar-lo, i tot seguit ja estarà en disposició de donar-lo de baixa.

4.2.2.1 Subsistema d'identificació

Realitza la validació dels usuaris prèvia a l'entrada a l'aplicació, permetent a més canviar la contrasenya d'accés. Els *cdu* corresponents a aquest subsistema són:

1 A partir d'aquest punt, cada referència que es faci a cas d'ús estarà representada per les sigles cdu.

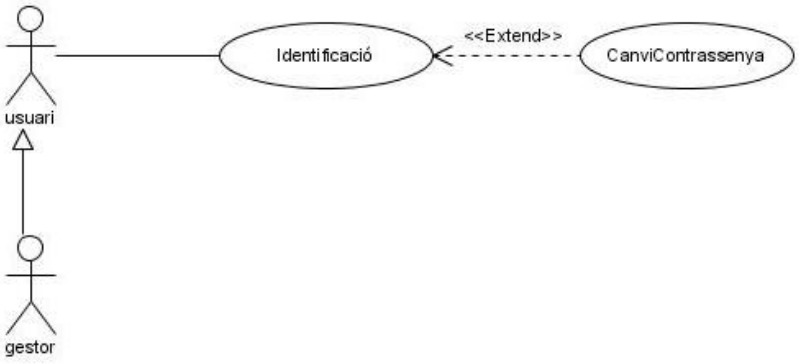

Figura 11. Esquema de casos d'ús: subsistema d'identificació.

Cas d'ús	**Identificació.**
Actors	Usuari i Gestor.
Funcionalitat	Validar usuari.
Resum	Identificar els actors que tenen accés a l'aplicació i, només en cas que hi puguin accedir, mostrar les funcionalitats corresponents a l'àrea a la que pertanyi l'actor.
Precondició	L'actor ha d'estar donat d'alta a la BDD de l'aplicació per un gestor, altrament no serà possible accedir-hi.
Postcondició	L'actor està validat i pot accedir a totes les funcionalitat que li permet el seu perfil (usuari o gestor).

Cas d'ús	**CanviContrasenya.**
Actors	Usuari i Gestor.
Funcionalitat	Canviar la paraula clau d'identificació de l'usuari.
Resum	Permetre canviar la contrasenya personal d'accés a l'arxiu. Els gestors podran canviar la de qualsevol usuari, però dins del subsistema d'Usuaris i concretament als cdu AltaUsuari i ModificacióUsuari.
Precondició	L'actor ha d'estar donat d'alta a la BDD de l'aplicació i ha de conèixer la contrasenya actual.
Postcondició	L'actor ha canviat la contrasenya que tenia fins aquest moment per una de diferent.

4.2.2.2 Subsistema de catàleg

Si s'hi accedeix amb un usuari, es permet realitzar totes les tasques de consulta dels documents de l'arxiu i veure'n la seva disponibilitat. En cas de tractar-se d'un gestor, es permet realitzar totes les tasques corresponents al manteniment del catàleg: alta, baixa i modificació de documents.

Com es pot observar, s'ha separat en dos *cdu* diferents les consultes sobre el catàleg depenent de l'actor que hi accedeixi, amb la única finalitat de separar perfectament les funcionalitats de cada actor: mentre que en la consulta del catàleg d'un usuari aquest només tindrà a més a més la opció de sol·licitar una reserva de préstec (*cdu* que pertany al subsistema de préstecs), en la consulta

d'un gestor podrà realitzar baixes i modificacions de documents.

Els *cdu* corresponents a aquest subsistema són[2]:

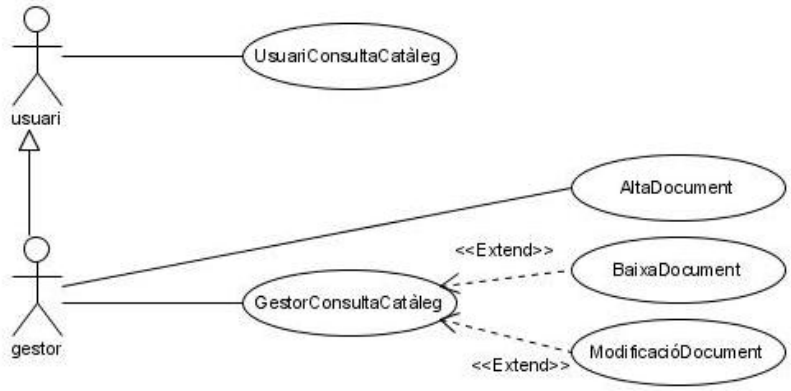

Figura 12. Esquema de casos d'ús: subsistema de catàleg.

Cas d'ús	**UsuariConsultaCatàleg.**
Actors	Usuari
Funcionalitat	Consultar documents del catàleg.
Resum	Permetre a l'usuari fer múltiples cerques al catàleg de l'arxiu. A més, dins del resultat de la consulta podrà realitzar reserves de préstecs si el document ho permet (cdu ReservaPréstec).
Precondició	L'usuari ha d'estar validat a l'aplicació.
Postcondició	S'obté el conjunt de documents que compleixen les condicions imposades per l'usuari.

Cas d'ús	**GestorConsultaCatàleg.**
Actors	Gestor
Funcionalitat	Consultar documents del catàleg.
Resum	Permetre a l'usuari fer múltiples cerques al catàleg de l'arxiu. A més, dins del resultat de la consulta podrà realitzar baixes i modificacions de documents del catàleg (cdu BaixaDocument i ModificacióDocument respectivament).
Precondició	El gestor ha d'estar validat a l'aplicació.
Postcondició	S'obté el conjunt de documents que compleixen les condicions imposades pel gestor.

2 Tant en aquest subsistema com en els següents, només es dibuixaran aquells cdu que en formin part; la resta, o bé no es dibuixaran o bé quedaran sombrejats en gris (en funció de la seva posició dins del diagrama de cdu).

Cas d'ús	**AltaDocument.**
Actors	Gestor
Funcionalitat	Donar d'alta un document.
Resum	Permetre al gestor donar d'alta un document al catàleg de l'arxiu.
Precondició	El gestor ha d'estar validat a l'aplicació i el document a donar d'alta no ha d'existir prèviament.
Postcondició	Un nou document passa a formar part del catàleg de l'arxiu.

Cas d'ús	**BaixaDocument.**
Actors	Gestor
Funcionalitat	Donar de baixa un document.
Resum	Permetre al gestor donar de baixa un document al catàleg de l'arxiu.
Precondició	El gestor ha d'estar validat a l'aplicació, i el document a donar de baixa ha d'existir prèviament.
Postcondició	El catàleg de l'arxiu conté un document menys.

Cas d'ús	**ModificacióDocument.**
Actors	Gestor
Funcionalitat	Modificar un document.
Resum	Permetre al gestor modificar un document al catàleg de l'arxiu.
Precondició	El gestor ha d'estar validat a l'aplicació, i el document a modificar ha d'existir prèviament.
Postcondició	El catàleg de l'arxiu conté el document modificat.

4.2.2.3 Subsistema de préstecs

Conté totes les funcionalitats relacionades amb els préstecs i és, juntament amb les consultes al catàleg de l'arxiu, el nucli de l'aplicació. Si es tracta d'un usuari, es permet realitzar una reserva de préstec, consultar tots els que té (actius i finalitzats), i realitzar una petició d'ampliació de préstec; altrament, en cas de tractar-se d'un gestor, es podrà donar d'alta un préstec, validar-ne una reserva, ampliar-ne la data de devolució i finalitzar-lo. Igual que en la resta de subsistemes, és necessari haver-se identificat prèviament a l'aplicació.

Els *cdu* corresponents a aquest subsistema són:

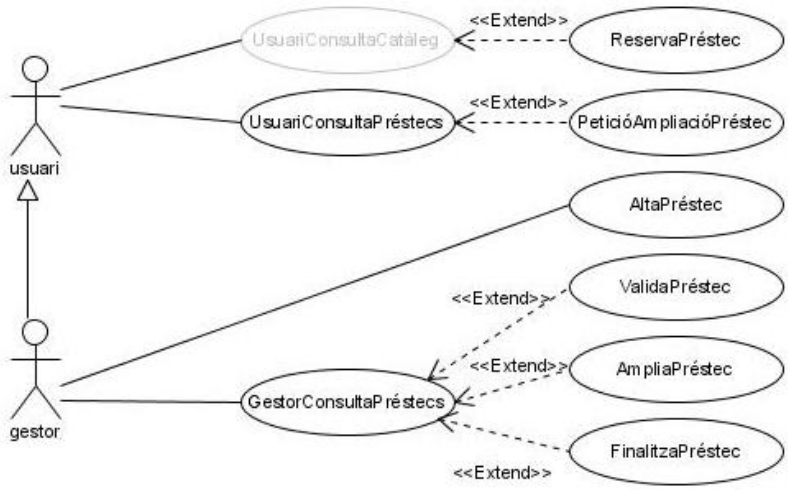

Figura 13. Esquema de casos d'ús: subsistema de préstecs.

Cas d'ús	**ReservaPréstec.**
Actors	Usuari
Funcionalitat	Realitzar una sol·licitud de préstec.
Resum	Permetre a l'usuari realitzar una petició de préstec de document, la qual haurà de ser posteriorment validada per un gestor al cdu ValidaPréstec. Després de ser validat per un gestor, si no s'ha anat a recollir en els següents 3 dies, la reserva caduca i el préstec finalitza.
Precondició	L'usuari ha d'estar validat a l'aplicació.
Postcondició	S'obté una reserva de préstec sobre un document pendent de validar per un gestor.

Cas d'ús	**UsuariConsultaPréstecs.**
Actors	Usuari
Funcionalitat	Consultar préstecs realitzats per l'usuari.
Resum	Permetre a l'usuari que pugui consultar els seus préstecs actius, pendents de validar (reserva préstec i/o ampliació de data de devolució) i finalitzats.
Precondició	L'usuari ha d'estar validat a l'aplicació.
Postcondició	S'obté un llistat dels préstecs realitzats per l'usuari.

Cas d'ús	**Petició Ampliació Préstec.**
Actors	Usuari
Funcionalitat	Ampliar en temps un préstec d'un document.
Resum	Permetre a l'usuari fer una petició d'ampliació del temps que es tindrà el document en préstec. Es tracta d'una sol·licitud, pel que l'ampliació de data d'entrega no quedarà modificada fins que un gestor validi aquest canvi en el cdu AmpliaPréstec.
Precondició	L'usuari ha d'estar validat a l'aplicació i ha de tenir actiu el préstec del qual es demana l'ampliació.
Postcondició	S'obté un llistat dels préstecs realitzats.

Cas d'ús	**GestorConsultaPréstecs.**
Actors	Gestor
Funcionalitat	Consultar tots els préstecs de l'usuari.
Resum	Permetre al gestor que pugui consultar tots préstecs actius, pendents de validar (reserva préstec i/o ampliació de data de devolució) i finalitzats.
Precondició	El gestor ha d'estar validat a l'aplicació.
Postcondició	S'obté un llistat de préstecs segons la consulta realitzada.

Cas d'ús	**AltaPréstecs.**
Actors	Gestor.
Funcionalitat	Donar d'alta un préstec.
Resum	Permetre al gestor donar d'alta un préstec.
Precondició	El gestor ha d'estar validat a l'aplicació, l'usuari que sol·licita el préstec ha d'estar donat d'alta a l'aplicació, i el document del que se sol·licita el préstec ha d'existir i s'ha de poder demanar en préstec.
Postcondició	S'ha creat un préstec per un document a un usuari, i el document passa a estar en préstec.

Cas d'ús	**ValidaPréstec.**
Actors	Gestor.
Funcionalitat	Validar qualsevol modificació feta a un préstec per part d'un usuari.
Resum	Permet al gestor validar tant les sol·licituds de préstecs realitzades pels usuaris com també les seves peticions d'ampliació en la data de devolució del préstec.
Precondició	El gestor ha d'estar validat a l'aplicació, l'usuari prèviament haurà fet una petició de reserva o de renovació del préstec. Mentre que en cas de tractar-se del primer cas, el document del que se sol·licita el préstec ha d'existir i s'ha de poder demanar en préstec, en cas de ser una renovació de préstec, aquesta ha de ser possible.
Postcondició	S'ha actualitzat un préstec segons la sol·licitud o la renovació de préstec realitzada per un usuari.

Cas d'ús	**AmpliaPréstec.**
Actors	Gestor.
Funcionalitat	Ampliar .
Resum	Permetre al gestor ampliar la data de devolució d'un préstec actiu.
Precondició	El gestor ha d'estar validat a l'aplicació i l'usuari demana in situ una ampliació del temps d'un préstec que té actiu.
Postcondició	S'ha modificat la data de devolució d'un préstec.

Cas d'ús	**FinalitzaPréstec.**
Actors	Gestor.
Funcionalitat	Donar per finalitzat un préstec.
Resum	Permetre al gestor donar per finalitzat un préstec prèviament actiu.
Precondició	El gestor ha d'estar validat a l'aplicació i finalitza el préstec quan l'usuari retorna in situ el document que fins ara tenia en préstec.
Postcondició	El préstec deixa d'estar actiu i passa a estar a un estat de finalitzat.

4.2.2.4 Subsistema d'usuaris

Conté totes les funcionalitats relacionades amb la gestió d'usuaris, amb el que aquest sistema únicament serà accessible des d'un actor amb rol de gestor. Permet donar d'alta, de baixa i modificar qualsevol usuari de l'aplicació, així com indicar-li si a més a més del rol d'usuari té també funcions de gestor, segons indica la relació d'herència entre usuari i gestor expressada en diagrames anteriors.

Els *cdu* corresponents a aquest subsistema són:

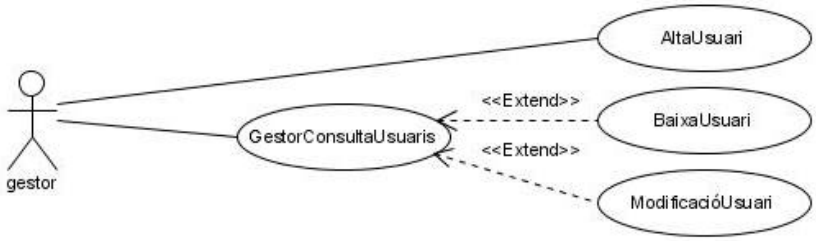

Figura 14. Esquema de casos d'ús: subsistema d'usuaris.

Cas d'ús	**GestorConsultaUsuaris.**
Actors	Gestor.
Funcionalitat	Consultar els usuaris de l'aplicació.
Resum	Permetre al gestor consultar tots els usuaris donats d'alta a l'arxiu, inclosos els gestors.
Precondició	El gestor ha d'estar validat a l'aplicació.
Postcondició	S'obté un llistat dels usuaris de l'arxiu.

Cas d'ús	**AltaUsuari.**
Actors	Gestor.
Funcionalitat	Donar d'alta un usuari.
Resum	Permetre al gestor donar d'alta un usuari a l'arxiu, per tal que pugui demanar documents en préstec. En el procés d'alta, es pot indicar si l'usuari a més a més tindrà rol de gestor.
Precondició	El gestor ha d'estar validat a l'aplicació i l'usuari a donar d'alta no ha d'existir prèviament.
Postcondició	Un nou usuari passa a formar part de l'arxiu.

Cas d'ús	**BaixaUsuari.**
Actors	Gestor.
Funcionalitat	Donar de baixa un usuari.
Resum	Permetre al gestor donar de baixa un usuari de l'arxiu.
Precondició	El gestor ha d'estar validat a l'aplicació, i l'usuari a donar de baixa ha d'existir prèviament sense tenir cap préstec actiu.
Postcondició	L'arxiu conté un usuari menys.

Cas d'ús	**ModificacióUsuari.**
Actors	Gestor.
Funcionalitat	Modificar un usuari.
Resum	Permetre al gestor modificar un usuari de l'arxiu.
Precondició	El gestor ha d'estar validat a l'aplicació, i l'usuari a modificar ha d'existir prèviament.
Postcondició	El catàleg de l'arxiu conté el document modificat.

4.2.3 Diagrames d'estats

4.2.3.1 Préstec iniciat i renovat per un gestor

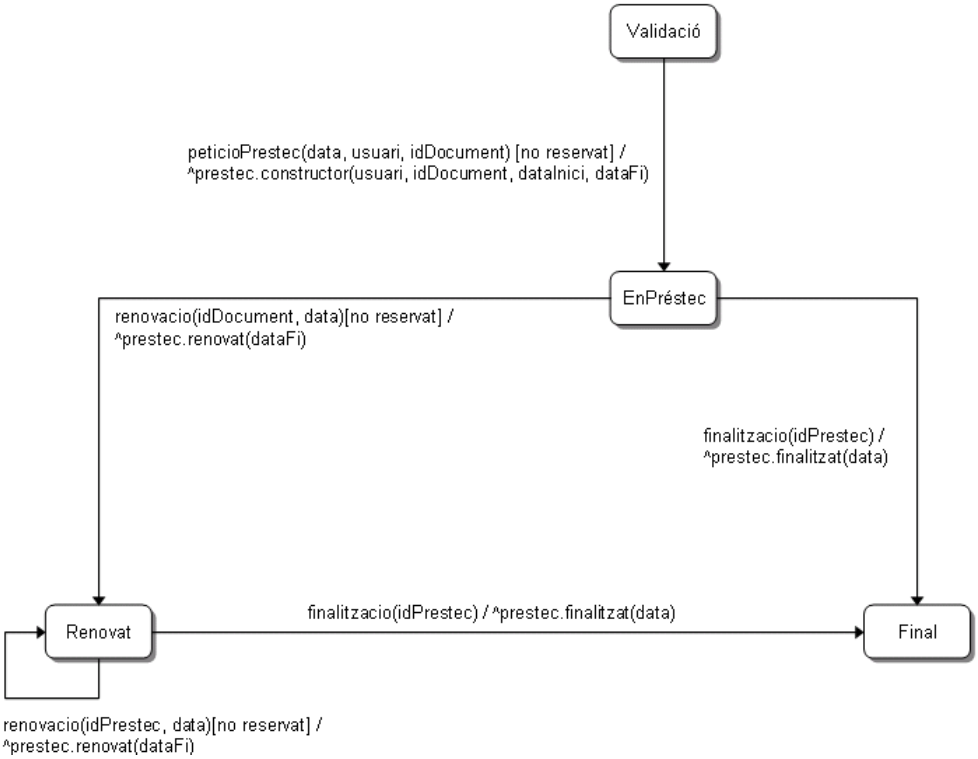

Figura 15. Diagrama d'estats: préstec iniciat i renovat per un gestor.

El diagrama d'estats representa tots els estats pels quals passa un préstec realitzat[3] per un gestor. Com es pot observar, el gestor comença en un estat de validació, una vegada fet el login a l'aplicació, i a continuació realitza un préstec, representat per l'esdeveniment *peticioPrestec*; aquest és possible de realitzar, ja que compleix la condició [no reservat] o, el que és el mateix, no està en préstec ni té cap reserva realitzada.

Tot seguit existeixen dues opcions: o bé finalitzar el préstec una vegada l'usuari peticionari del mateix retorna el document, representat per l'esdeveniment *finalització*, i s'arriba a l'estat *Final*, o bé pot renovar el préstec de l'usuari mitjançant l'esdeveniment renovació. A l'estat *Renovat* s'hi accedirà tantes vegades com renovacions es realitzin del préstec actiu, fins que l'usuari faci efectiva la devolució del document, moment en què s'arribarà finalment a l'estat final.

3 La realització del préstec comprèn dues fases: la creació inicial del préstec i la posterior i optativa renovació del mateix.

4.2.3.2 Préstec iniciat i renovat per un usuari

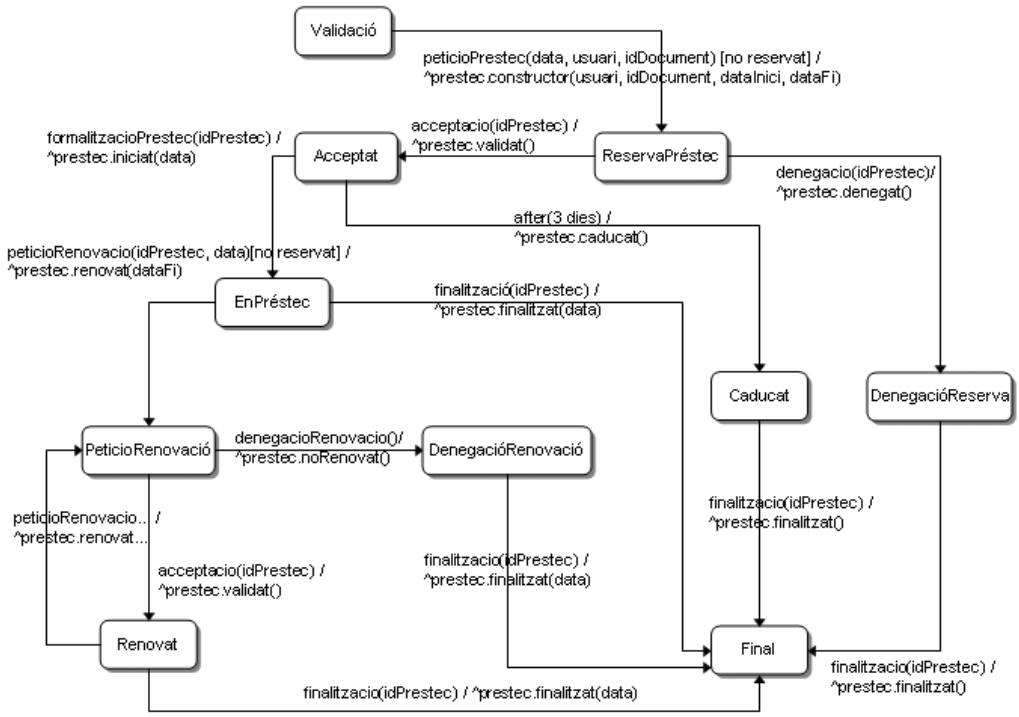

Figura 16. Diagrama d'estats: préstec iniciat i renovat per un usuari

Aquest segon diagrama d'estats representa tots els estats pels quals passa un préstec realitzat[4] per un usuari. A diferència de l'anterior, qualsevol actuació[5] que realitza l'usuari ha de passar pel garbell de la validació d'un gestor; així, per passar de l'estat *ReservaPréstec* a l'estat de préstec *Acceptat*, un gestor ha de validar el préstec, i d'igual manera per passar de l'estat de *Petició Renovació* al de *Renovat*; en cas contrari, s'arribarà als estats *DenegacióReserva* i *DenegacióRenovació* respectivament.

Es limita el temps d'espera de l'usuari per passar a recollir el document una vegada acceptada la seva reserva, sent el límit màxim de 3 dies, moment en el qual s'arribarà a l'estat *Caducat*. Per poder tenir aquest control, és necessari tenir un estat *EnPréstec*, que ens informa quan físicament el document ha estat prestat.

Mentre que en el diagrama del punt 3.1 tota la operativa que realitza l'usuari la fa davant d'un gestor (físicament), en aquest punt l'usuari realitza totes les operacions que té disponibles (petició de reserva i petició de renovació) a distància.

4 La realització del préstec comprèn dues fases: la creació inicial del préstec i la posterior i optativa renovació del mateix.

5 Petició de préstec i petició de renovació.

4.2.4 Diagrames de seqüència

4.2.4.1 Préstec iniciat i renovat per un usuari.

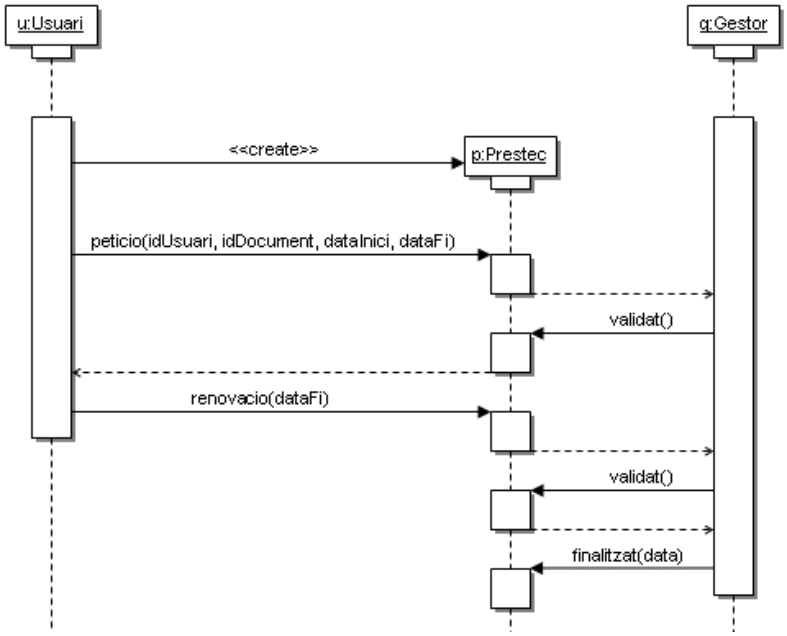

Figura 17. Diagrama de seqüència: préstec iniciat i renovat per un usuari.

El diagrama de seqüències mostra el cas exposat en el punt 3.2, que és a la vegada la seqüència més complexa de l'aplicació: un usuari realitza de forma no presencial una petició de reserva d'un document (crea un préstec), el qual és validat pel gestor en el moment de recollida del document; a continuació realitza a distància una petició de renovació que torna a validar un gestor, i finalment retorna el document in situ, moment en el qual el gestor indica que s'ha finalitzat el préstec.

4.2.4.2 Préstec iniciat per un usuari i caducat

El següent diagrama mostra la situació en què un usuari sol·licita a distància una reserva de préstec, el gestor la valida, però en canvi l'usuari no el passa a recollir en els següents 3 dies, motiu pel qual el gestor finalitza el préstec; com es pot observar, la finalització és automàtica per tal de descarregar de feina el gestor.

Es podria contemplar la possibilitat de fer que el gestor eliminés el préstec, però es vol tenir constància de totes aquestes actuacions per tenir controlat si un usuari infringeix molt sovint el període limitat de temps.

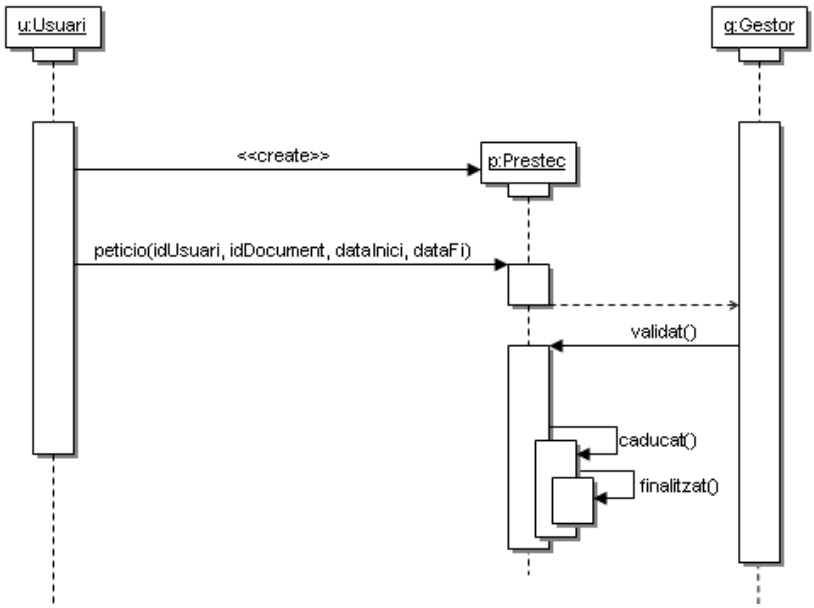

Figura 18. Diagrama de seqüència: préstec iniciat per un usuari i caducat.

4.3 Capítol 3: Disseny

4.3.1 Diagrama de classes

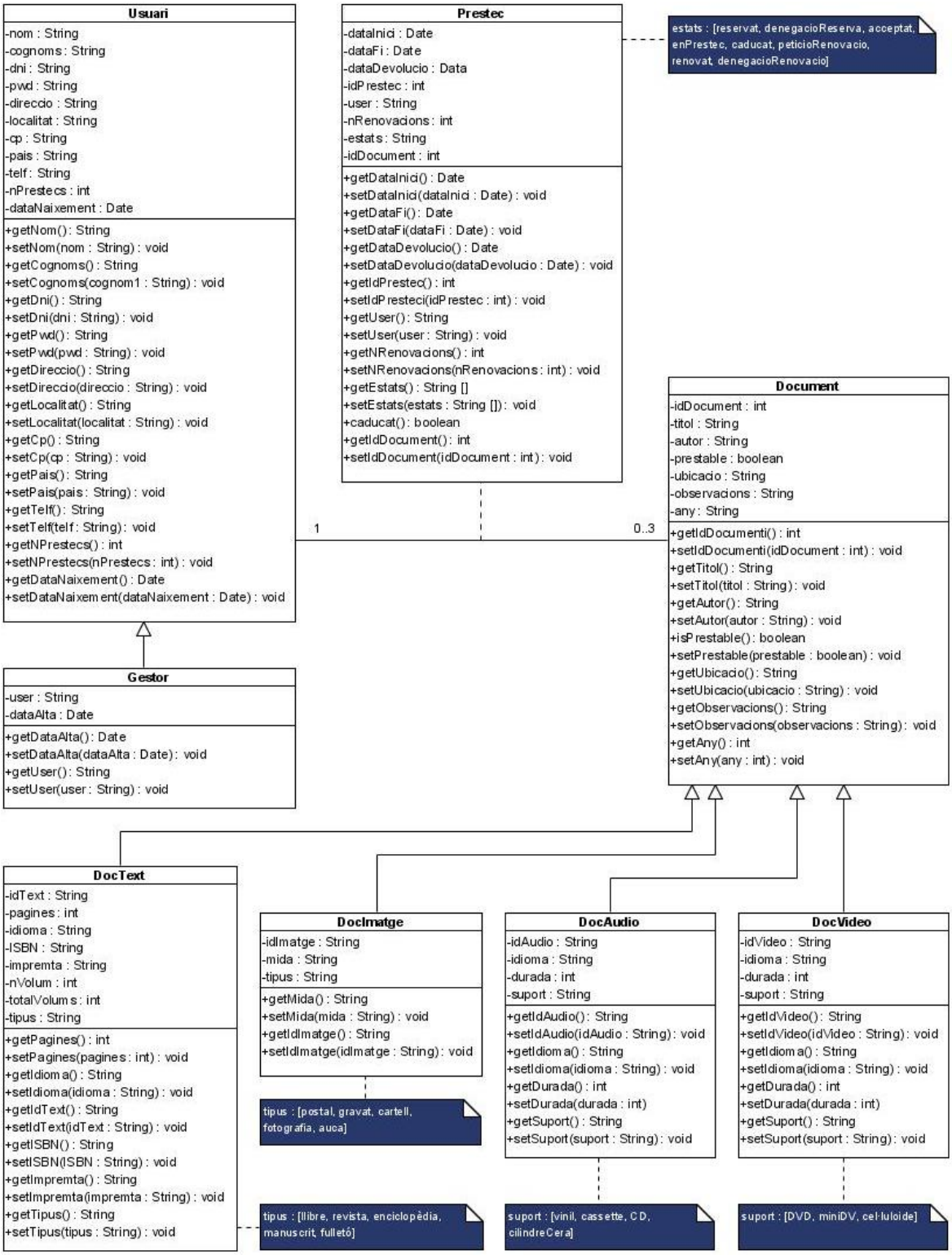

Figura 19. Diagrama de classes: escenari general.

4.3.2 Diagrama ER

L'esquema de classes mostrat al punt anterior permet arribar al següent diagrama ER:

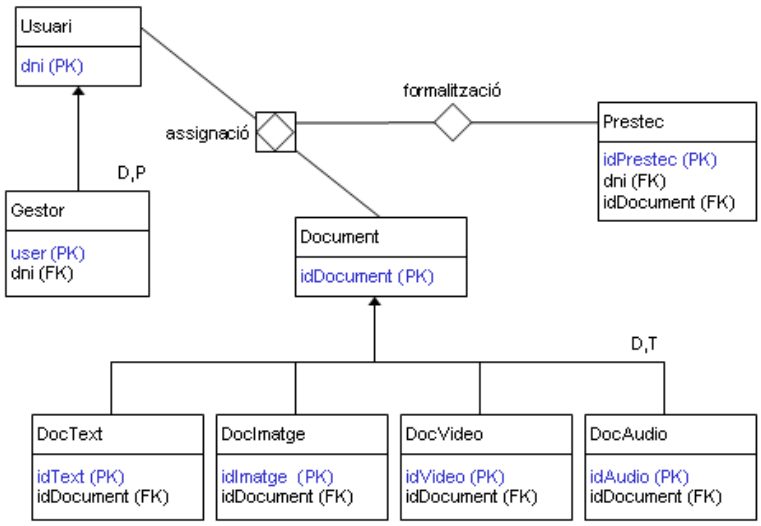

Figura 20. Diagrama ER: escenari general.

Les entitats filles de DocText, DocImatge, DocVideo i DocAudio estan incloses en els seus pares, ja que no hi ha cap necessitat de guardar-les en taules pròpies. A més, com que es vol portar un control tant global de documents, com separat per categories (text, imatge, vídeo, àudio), l'entitat pare i les seves filles se separaran en taules diferents i relacionades.

Com ja s'ha indicat al diagrama de classes del punt 5, un usuari podrà tenir en actiu fins a tres préstecs a la vegada.

4.3.3 Persistència de dades

A continuació es mostra com queda definida la persistència de dades. Els valors que pot prendre la columna clau fan referència a clau primària (PK), a clau forana (FK) i a clau alternativa (AK).

4.3.3.1 Taula d'usuaris

Camp	Tipus	Clau	Descripció
nom	VARCHAR(15) NOT NULL		Nom.
cognoms	VARCHAR(100) NOT NULL		Cognoms.
dni	VARCHAR(8) NOT NULL	PK	DNI.
pwd	VARCHAR(8) NOT NULL		Contrasenya.
direccio	VARCHAR(80) NOT NULL		Direcció física.
localitat	VARCHAR(30) NOT NULL		Localitat.
cp	VARCHAR(6) NOT NULL		Codi postal
pais	VARCHAR(15) NOT NULL		País.
telf	VARCHAR(20) NULL		Telèfon de contacte
nPrestecs	INT(1) NOT NULL		Número de préstecs actius.
dataNaixement	DATETIME NULL		Data de naixement.

4.3.3.2 Taula de gestors

Camp	Tipus	Clau	Descripció
user	VARCHAR(8) NOT NULL	PK	Identificador del gestor.
dataAlta	DATETIME NOT NULL		Data d'alta com a gestor.

4.3.3.3 Taula de préstecs

Camp	Tipus	Clau	Descripció
idPrestec	INT(7) NOT NULL AUTO_INCREMENT	PK	Identificador
dataInici	DATETIME NOT NULL		Data d'inici del préstec.
dataFi	DATETIME NULL		Data de finalització del préstec.
dataDevolucio	DATETIME NULL		Data de devolució del préstec.
dni	VARCHAR(10) NOT NULL	FK	Usuari que l'ha sol·licitat.
idDocument	INT(7) NOT NULL	FK	Document en préstec.
nRenovacions	INT(1) NULL		Número de renovacions fetes.
estats	VARCHAR(50) NOT NULL		Històric d'estats del préstec

4.3.3.4 Taula de documents

Camp	Tipus	Clau	Descripció
idDocument	INT(7) NOT NULL AUTO_INCREMENT	PK	Identificador.
titol	VARCHAR(20) NOT NULL		Títol.
autor	VARCHAR(50) NOT NULL		Autor.
prestable	BIT NOT NULL		Indica si és o no prestable.
ubicacio	VARCHAR(10) NOT NULL		Codi per localitzar-lo físicament.
observacions	TEXT NULL		Observacions
any	VARCHAR(10)		Any

4.3.3.5 Taula de document de text

Camp	Tipus	Clau	Descripció
idText	INT(6) NOT NULL AUTO_INCREMENT	PK	Identificador de document de text.
idDocument	INT(7) NOT NULL	FK	Identificador global de document.
tipus	VARCHAR(15) NOT NULL		Tipus de document de text: llibre, revista, enciclopèdia, manuscrit, fulletó.
pagines	INT(5) NOT NULL		Número de pàgines.
idioma	VARCHAR(10) NOT NULL		Idioma.
ISBN	VARCHAR(10) NULL		Codi ISBN.
impremta	VARCHAR(100) NULL		Impremta.
nVolum	INT(3) NULL		Número de volum.
totalVolums	INT(4) NULL		Número total de volums.

4.3.3.6 Taula de documents d'imatges

Camp	Tipus	Clau	Descripció
idImatge	INT(6) NOT NULL AUTO_INCREMENT	PK	Identificador de document d'imatge.
idDocument	INT(7) NOT NULL	FK	Identificador global de document.
mida	VARCHAR(10) NULL		Tamany alçada x amplada.
tipus	VARCHAR(15) NULL		Tipus de document d'imatges: postal, gravat, cartell, fotografia, auca.

4.3.3.7 Taula de documents de vídeo

Camp	Tipus	Clau	Descripció
idVideo	INT(6) NOT NULL AUTO_INCREMENT	PK	Identificador de document de vídeo.
idDocument	INT(7) NOT NULL	FK	Identificador global de document.
idioma	VARCHAR(10) NOT NULL		Idioma.
durada	INT(1) NOT NULL		Durada en minuts.
format	VARCHAR(15) NULL		Classe de document de vídeo: DVD, miniDV, cel·luloide.

4.3.3.8 Taula de documents d'àudio

Camp	Tipus	Clau	Descripció
idAudio	INT(6) NOT NULL AUTO_INCREMENT	PK	Identificador de document d'àudio.
durada	INT(1) NULL		Durada en minuts.
idDocument	INT(7) NOT NULL	FK	Identificador global de document.
format	VARCHAR(15) NULL		Format : CD, disc pedra, disc vinil, caset, cilindre cera, altres.

4.3.4 Interfície

A nivell d'implementació s'han unificat dins d'un mateix jsp les vistes corresponents a usuaris i a gestors. Amb aquesta decisió s'aconsegueix un menor nombre d'arxius a mantenir, i una capa de vista amb menys granularitat.

Les vistes creades són:

Nom arxiu	*Descripció*
login.jsp	Vista inicial de validació de l'usuari. Una vegada identificat correctament l'usuari, s'accedeix a la vista menu.jsp.
menu.jsp	Vista del menú principal de la pantalla. En cas que l'usuari que hi accedeix tingui perfil de gestor, es mostraran totes les opcions possibles per cadascun dels subsistemes de documents (catàleg), usuaris i préstecs, amb visió sobre tot el domini d'usuaris donats d'alta a l'aplicació; en canvi, si es tracta d'un usuari sense perfil de gestor, únicament es mostrarà una opció de consulta pel catàleg de documents, una de modificació de les seves dades personals, i una de visualització dels seus préstecs.
document_alta.jsp	Vista on es dóna d'alta un document al catàleg de l'aplicació. Només accessible pels gestors.
document_consulta.jsp	Vista on es poden realitzar cerques sobre tot el catàleg de documents de l'aplicació. Accessible tant per usuaris com per gestors. Enllaça amb la vista document_detall.jsp.
document_detall.jsp	Vista on es mostra el detall (tots els atributs) d'un determinat document i, en cas que es tracti d'un gestor, permet opcions de modificació i eliminació.
prestec_alta.jsp	Vista on es permet donar d'alta (formalitzar) un préstec. Únicament activa pels gestors.
prestec_consulta.jsp	Vista on es poden realitzar cerques els préstecs de l'aplicació. Accessible tant per usuaris com per gestors, amb la diferència que mentre pels usuaris el domini de consulta es limita als préstecs propis, en els gestors el domini és el de tots els usuaris de l'aplicació, tinguin o no perfil d'usuari. Enllaça amb la vista prestec_detall.jsp.
prestec_detall.jsp	Vista on es mostra el detall del préstec, tot facilitant totes les dades del préstec, del llibre sobre el qual es fa, i de l'usuari que el demana.
usuari_alta.jsp	Vista on es dóna d'alta un usuari a l'aplicació. Només accessible pels gestors.
usuari_consulta.jsp	Vista on es poden realitzar cerques sobre tots els usuaris de l'aplicació. Només accessible pels gestors. Enllaça amb la vista usuari_detall.jsp.
usuari_detall.jsp	Vista on es mostra el detall (tots els atributs) d'un determinat usuari i, en cas que es tracti d'un gestor, permet opcions de modificació i eliminació; d'altra banda, si es tracta d'un usuari sense perfil de gestor, es permet l'accés a aquesta vista però únicament amb la possibilitat de modificar atributs del seu propi usuari.
include.jsp	Arxiu auxiliar que s'inclou en totes les vistes de l'aplicació, i que conté: la configuració de la codificació de la pàgina per tal que no hi hagi problemes en la visualització a través del servidor d'aplicacions (charset=ISO-8859-1" pageEncoding="ISO-8859-1"), la definició dels tags JSTL que s'utilitzen en les vistes (java.sun.com/jstl/core i java.sun.com/jstl/fmt), i la definició de la fulla d'estils CSS que s'utilitza en totes les pantalles.

L'ús dels tags JSTL, juntament amb una fulla d'estils CSS, permet obtenir un codi més clar i fàcil de mantenir. A més, la definició de missatges a través de JSTL, deixa una porta oberta a l'aplicació de cara a una futura activació de diferents idiomes canviant únicament el fitxer de configuració messages.properties.

El principal objectiu que té la capa de vista de l'aplicació és que sigui agradable, intuïtiva i fàcil d'utilitzar per qualsevol tipus d'usuari, sense descuidar llur disseny. Per aquest motiu s'ha utilitzat una fulla d'estils CSS

- **Login:** pantalla inicial d'identificació, és la primera vista de l'aplicació que es mostra:

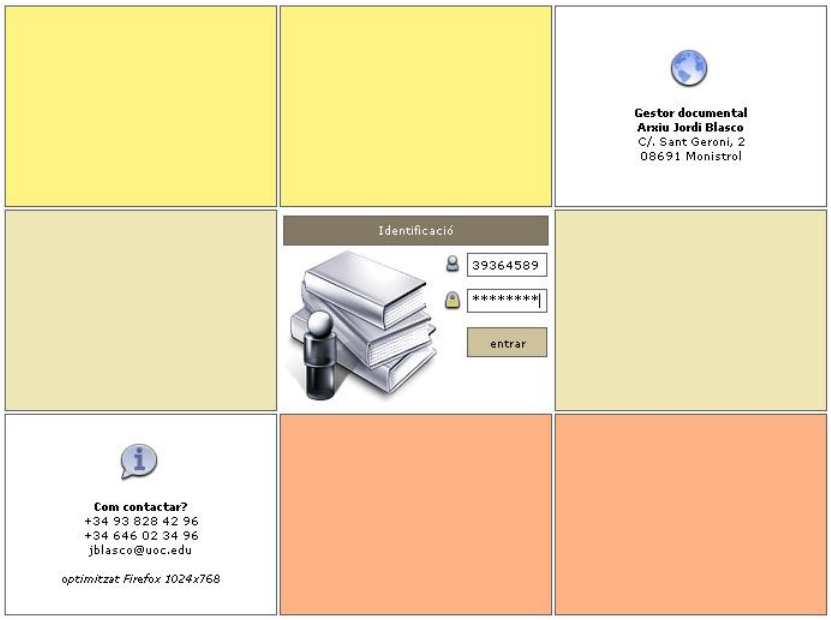

Figura 21. Diagrama de classes: escenari general.

En cas que es produeixi algun error en la identificació, retorna un missatge informatiu:

Figura 22. Diagrama de classes: escenari general.

- **Menú :** pantalla principal de l'aplicació. Si l'usuari identificat té perfil de gestor, es mostraran totes les opcions habilitades per cadascuna de les àrees de catàleg, usuaris i préstecs.

Figura 23. Vista: menú principal dels gestors.

D'altra banda, en cas que únicament tingui perfil d'usuari, les opcions actives seran de consulta sobre el catàleg de tots els documents, i sobre els préstecs propis (tant actius com finalitzats), i de modificació sobre les seves dades d'usuari.

Figura 24. Vista: menú principal dels usuaris.

Sempre es pot accedir a aquesta pantalla inicial de menú fent un clic sobre el logotip de l'arxiu, situat a la part superior dreta de (la Terra, en aquest cas). Per desconnectar l'usuari i poder-se connectar amb un altre, únicament s'ha de fer clic sobre el candau de la part superior dreta.

- **Gestió de catàleg :** tant si es vol fer una modificació com una baixa de document, la pantalla inicial és la de la consulta, on es pot filtrar per camp i paraula clau. Tot seguit, fent clic sobre l'identificador del document, s'accedeix a la pantalla que mostra el detall del document i en permet la baixa (si les condicions ho permeten) o la modificació.

Figura 25. Vista: consulta del catàleg de documents.

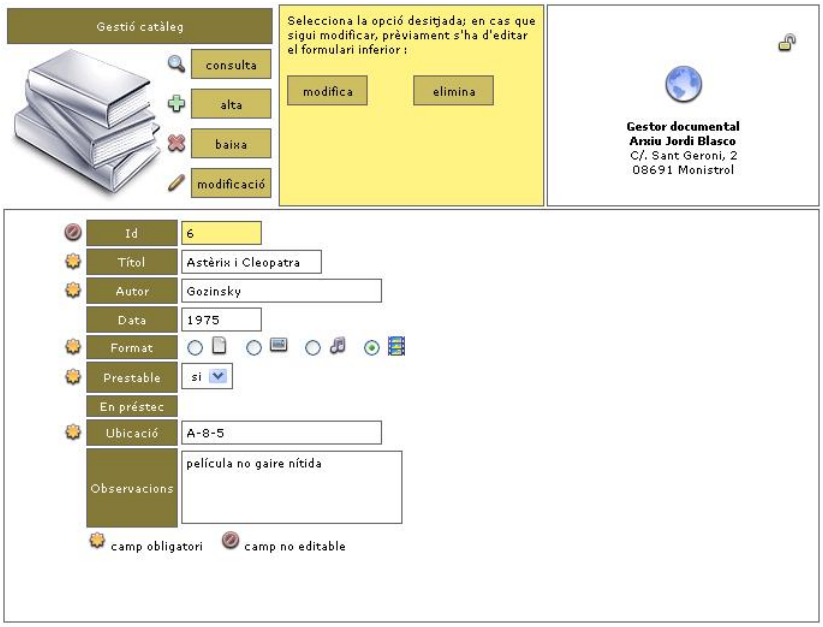

Figura 26. Vista: detall del document.

Si es vol fer una alta de document (gestor), es mostra directament una pantalla com la de detall de document.

- **Gestió d'usuaris :** el funcionament segueix la mateixa pauta oberta per l'àrea anterior: si es vol donar de baixa o modificar un usuari, prèviament s'ha de fer una consulta per tal de seleccionar l'usuari en qüestió; al detall s'hi accedeix fent clic sobre l'identificador de l'usuari.

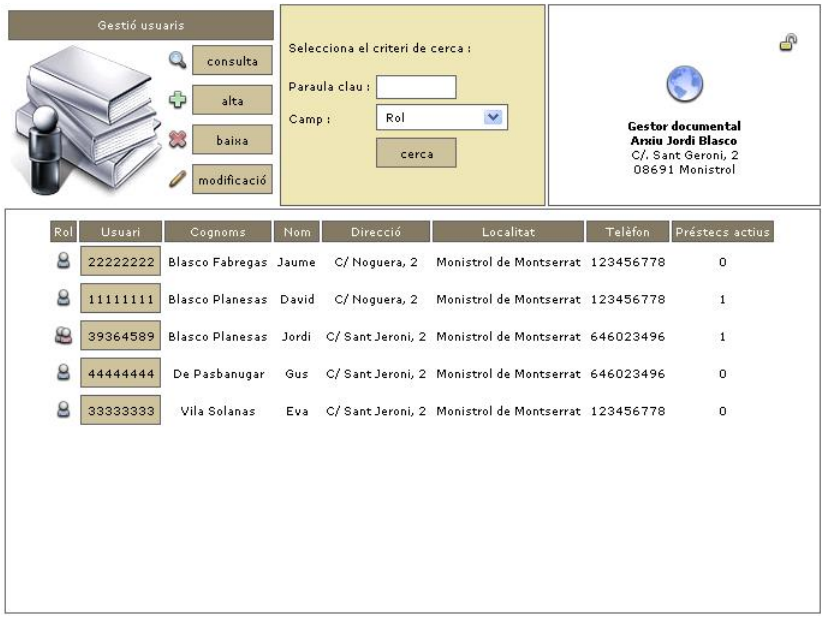

Figura 27. Vista: consulta d'usuaris.

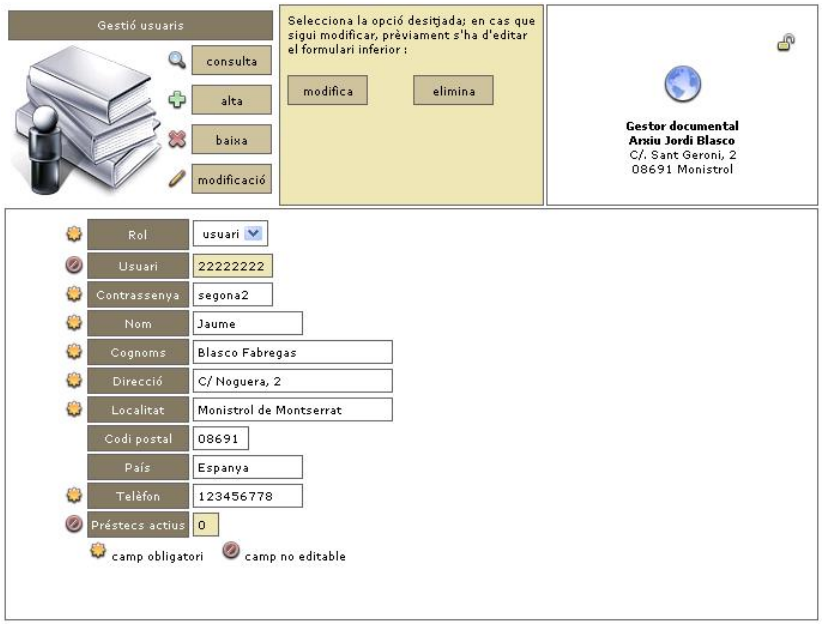

Figura 28. Vista: detall de l'usuari.

En aquest cas en concret, l'usuari identificat a l'aplicació té perfil de gestor, i accedeix a la fitxa d'un usuari que no té cap préstec actiu, amb el que es permet, a part de modificar-lo, eliminar-lo.

- **Gestió de préstecs :** segueix el mateix funcionament que els dos subsistemes anteriors, ja que si es vol fer una devolució del préstec, prèviament s'ha d'haver seleccionat a través de la consulta genèrica de préstecs. Malgrat les opcions de consulta són les mateixes pels usuaris amb perfil gestor i pels que no el tenen, mentre que pels primers els retorna consultes sobre tot el conjunt d'usuaris, pels segons únicament es visualitzen llurs préstecs, finalitzats i actius.

Figura 29. Vista: consulta de préstecs.

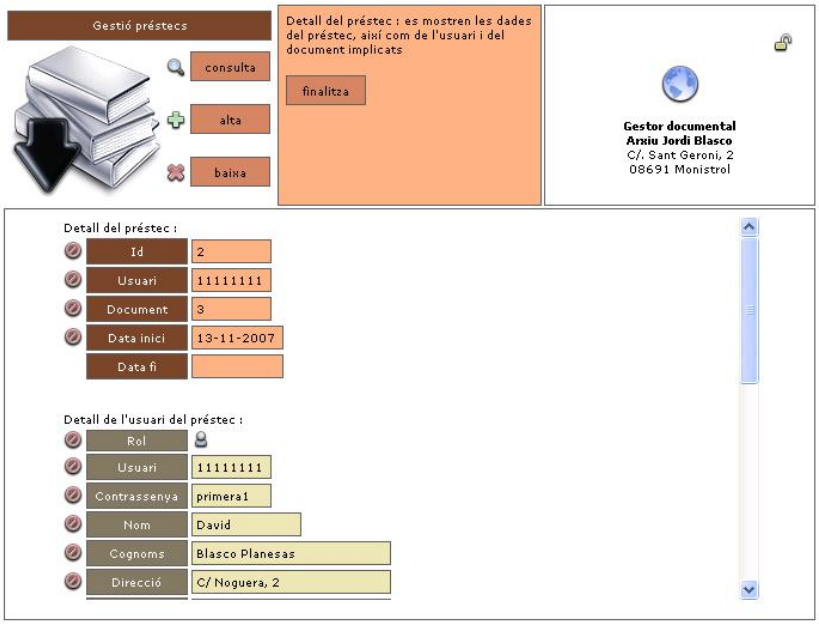

Figura 30. Vista: detall del préstec.

La darrera captura mostra el cas en concret on es permet finalitzar el préstec, ja que el que s'està visualitzant encara no té data de fi assignada. Tant la data d'inici com la data de fi no s'inserten manualment, sinó que a l'iniciar o finalitzar el préstec provoca l'automàtica inicialització amb la data actual.

4.3.5 Estructura JEE

4.3.5.1 Introducció

JEE, conegut també com Java Platform Enterprise Edition o Java EE, és una plataforma de programació que té com a objectiu desenvolupar i executar aplicacions Java amb una arquitectura distribuïda i de diversos nivells. Java EE inclou vàries especificacions com JDBC, RMI, e-mail, JMS, Serveis Web, XML, etc. i defineix de quina forma coordinar-los. A més, Java EE també configura algunes tecnologies pròpies com Enterprise JavaBeans, servlets, JavaServer Pages, etc. i frameworks com Spring o Struts.

L'agrupació sota un mateix paraigües de tots aquests recursos permet al desenvolupador de software crear una aplicació portable entre plataformes, escalable i integrable amb les tecnologies anteriorment mencionades. Altres beneficis són la possibilitat que el servidor d'aplicacions pugui manejar transaccions, la seguretat, l'escabilitat, la concurrència i la gestió dels components desplegats i escalable, de manera que els desenvolupadors es puguin concentrar més en la lògica de negoci que no pas en les tasques més rutinàries de manteniment a baix nivell.

Aprofitant la potència que ofereix JEE, l'arquitectura de l'aplicació quedarà separada en tres capes; aquesta arquitectura s'ha utilitzat àmpliament al mercat gràcies, entre altres aspectes, a la capacitat de creixement que ofereix, a la seva escalabilitat i a la facilitat de manteniment.

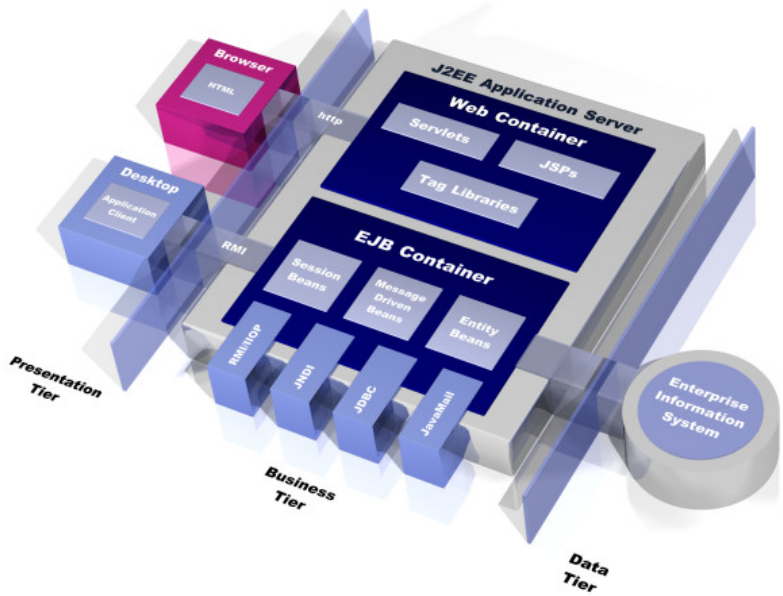

Figura 31. Esquema estructuració de la tecnologia JEE.

4.3.5.2 Elecció de tecnologia JEE.

L'actual projecte està condicionat al fet que el coneixement previ de què es disposa sobre les tecnologies JEE és nul. Aquest escenari condiciona completament l'enfocament que se li dóna al TFC on, a més a més, s'ha de destriar entre la gran quantitat de solucions que es poden implementar dins de l'arquitectura JEE. Des d'un bon inici s'han plantejat la solució al voltant d'un gran framework que pugui oferir una solució completa, en detriment de l'ús de multitud de patrons arquitectònics, malgrat alguna combinació dels mateixos puguin arribar a oferir un millor rendiment.

L'aplicació es construeix en base a una solució que implementa el patró MVC o, el que és el mateix, un patró d'arquitectura de software que separa les dades de l'aplicació (model), la interfície de l'usuari (vista) i la lògica de control (controlador) en tres capes, facilitant-ne així el control, el manteniment i la implementació de noves millores a les aplicacions amb un impacte mínim sobre la resta de capes. A més, tot i que en el present escenari té una importància més aviat nul·la, una aplicació basada amb un model MVC permet crear aplicacions de forma més ràpida i efectiva ja que diferents grups de desenvolupadors poden estar treballant en paral·lel en les tres capes del model sense afectar-se mútuament.

Per tal de seguir el model MVC, s'han considerat inicialment dos candidats per formar el nucli de l'aplicació: els frameworks Struts i Spring MVC (un subconjunt del framework Spring). Tot i que ambdós són més que suficients per realitzar el present TFC, finalment s'ha decidit crear una solució basada amb Spring MVC. L'elecció ha estat en gran part condicionada precisament per la falta de coneixements indicada anteriorment: posats a adquirir uns nous coneixements i a treballar amb una estructura determinada, s'ha preferit enfocar-ho cap a una tecnologia com Spring que en els darrers ha sofert un gran creixement, en comptes d'enfocar-ho cap a una solució clàssica basada en el més que consolidat Struts.

S'és conscient (i així s'ha pogut constatar fins ara) que la poca edat del patró Spring davant d'Struts també té una repercussió negativa en l'apartat documental, ja que si bé a nivell de conceptes teòrics aquesta és més que acceptable, a nivell pràctic de codi font és molt limitada; a més, també s'ha pogut observar com les solucions per desenvolupar aplicacions amb Struts està més avançada que no pas amb Spring; no obstant això, es prefereix continuar amb Spring MVC per ser un patró amb molt més recorregut en els propers anys.

4.3.5.3 Spring MVC.

Les aplicacions basades amb Spring MVC s'estructuren en 5 capes:

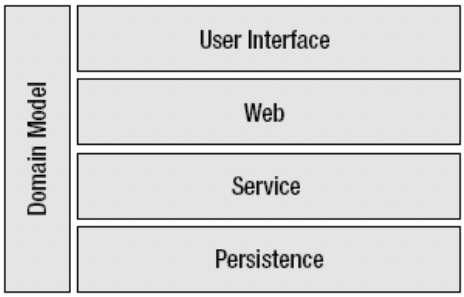

Figura 32. Capes estructurals d'Spring MVC.

- **User Interface Layer :** la capa d'interfície d'usuari és la responsable de presentar l'aplicació a l'usuari final i, per tant, qualsevol petició que prèviament hagi fet el client a l'aplicació, li serà tornada la resposta amb la vista que se li hagi especificat a través d'aquesta capa. Els dos principals components d'Spring MVC per aquesta capa són org.springframework.web.servlet.View, que representa una vista determinada, i org.springframework.web.servlet.ViewResolver, que permet establir una relació directa (mapeig) entre noms lògics i vistes, facilitant així tota la gestió de crides de vistes.

- **Web Layer :** la capa web gestiona la navegació a través del site. El nucli d'aquesta capa és org.springframework.web.servlet.mvc.Controller, el controlador a través del qual es permet acceptar les crides HttpServletRequest i gestionar les corresponents HttpServletResponse; a més, també s'ocupa de passar el control a la vista (View) determinada, utilitzant per aquesta funció ModelAndView, la peça dins de l'engranatge d'Spring MVC que conté el model de resposta i la referència de la vista que s'utilitzarà.

- **Service Layer :** la capa de servei és on són implementats els *cdu*, deixant al client interactuar amb aquesta capa superior i impedint-li l'accés al les classes de més baix nivell, de manera que el que s'aconsegueix és un nombre menor d'accessos entre client i aplicació i, conseqüentment, un estalvi de temps en la seva execució. La unitat bàsica amb què es treballa en aquesta capa són els POJO[6], enfocats cap als processos de negoci. Una peça clau de l'arquitectura en aquest nivell és ApplicationContext, que permet la injecció d'instàncies del servei al elements Controller de l'aplicació.

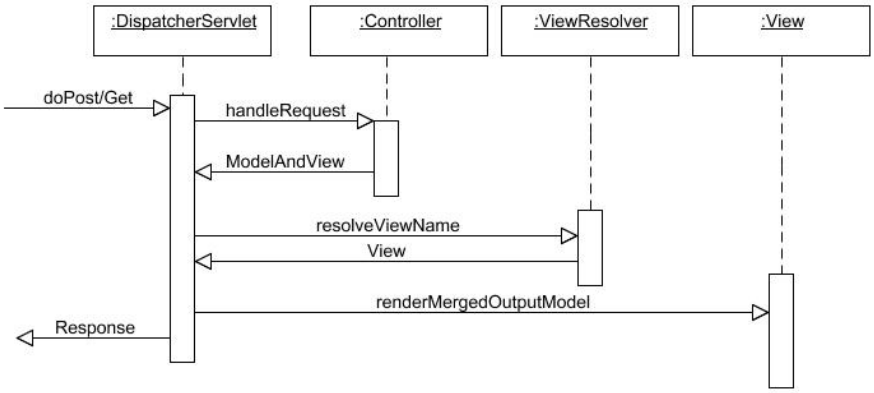

Figura 33. Diagrama de seqüència: exemple de comunicació entre capes amb Spring MVC

- **Domain Model Layer :** és la capa més important ja que conté tot el domini lògic de l'aplicació. La unitat bàsica d'aquest nivell són els anteriorment referits POJO, Plain Old Java Object, un concepte que precisament té el seu origen en framework "no intrusius" com Spring i Hibernate. Amb aquest plantejament el que es busca és que la jerarquia de classes del domini lògic de l'aplicació no sigui conscient que està envoltada per un framework, la qual cosa permet que aquest nivell s'alleugereixi, sigui més escalable i mantenible, i perdi la rigidesa característica de solucions com els EJB d'Struts.

6 En la següent capa Domain Model Layer s'explica més aquest concepte.

- **Data Access Layer :** la capa d'accés a dades d'Spring no disposa d'una única interfície d'accés a la persistència de l'aplicació degut a les múltiples opcions de què es disposa: Hibernate (HibernateTemplate), JDBC (JdbcTemplate), iBATIS (SqlMapTemplate), i altres opcions disponibles dins dels paquets org.springframework.jdbc i org.springframework.orm d'Spring. A més, també disposa de suport pel patró DAO, patró que també s'utilitzarà en el present TFC juntament amb Spring Jdbc.

  Pel que fa a l'accés a la persistència de dades, la implementació de la interfície DAO quedarà establerta per mitjà del connector JDBC d'Spring. Amb aquesta elecció s'aconsegueix per una banda construir tota l'aplicació únicament amb les eines que ofereix el framework Spring, i d'altra banda aprofitar l'experiència amb l'accés a BDD a través de JDBC (similar en quan a funcionament que el connector JDBC d'Spring) que s'ha anat treballant en diferents assignatures d'ETIS de la UOC.

A més a més dels avantatges que s'han enumerat, l'elecció d'Spring també ve condicionada per la possibilitat que aquest framework ofereix per testejar les classes creades abans de tenir finalitzada l'aplicació: gràcies a la gran llibertat que ofereixen els objectes *POJO*, a la implementació dins del framework del concepte de *dependency injection* i a l'aparició dels *mock object*, Spring permet deslocalitzar momentàniament qualsevol *POJO* de l'aplicació i testejar-lo sense necessitat de tenir tota l'aplicació muntada. Aquest aspecte, a diferència d'altres frameworks com Struts, permet tenir la lògica de negoci completament testejada abans i tot d'insertar-la a la posició que li correspon dins l'aplicació.

4.3.5.4 Exemple d'implementació Spring MVC.

A continuació s'explica pas a pas com s'han implementat tots els conceptes teòrics exposats al punt anterior en un cas concret del TFC: la consulta de tots els usuaris de l'aplicació realitzada per un usuari amb perfil gestor.

L'esquema corresponent a aquest escenari particular del *cdu* GestorConsultaUsuaris és:

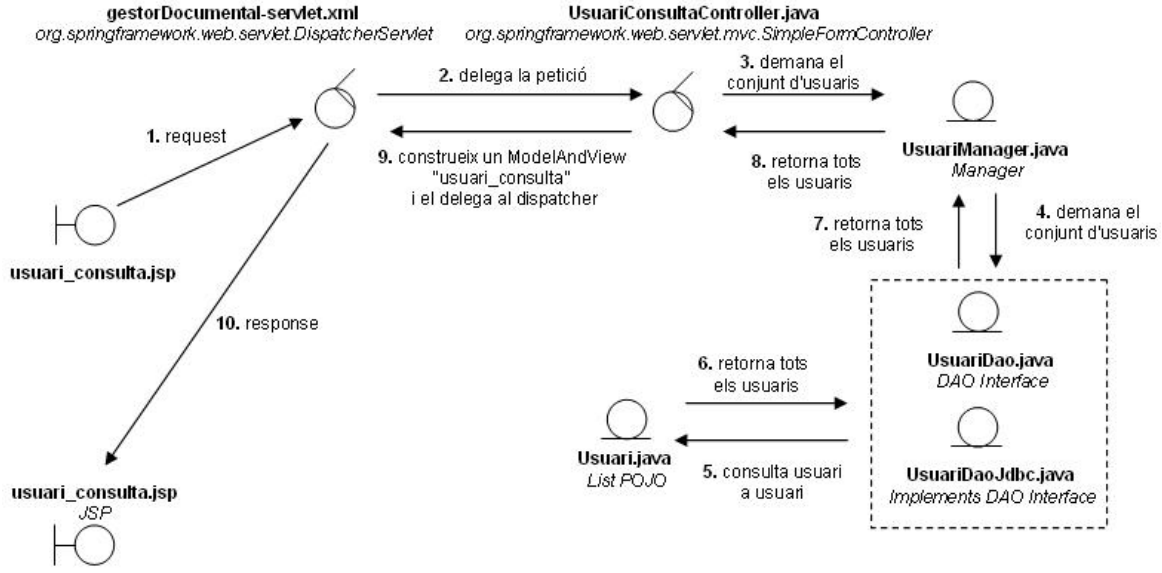

Figura 34. Diagrama de col·laboració: consulta d'usuaris realitzada per un gestor.

El punt de partida és la interfície amb la que interactua el gestor (recordar que aquest *cdu* és exclusiu dels gestors). A la pantalla inicial es permet al gestor realitzar una cerca entre tots els usuaris donats d'alta a l'aplicació en funció d'una determinada condició.

La petició que s'inicia a través de la pàgina .jsp és interceptada pel *DispatcherServlet*, el *Front Controller* de l'aplicació que permet recollir totes les peticions entrants (*requests*) i coordinar els diferents subsistemes d'Spring MVC. El *dispatcher* reconeix que la petició fa referència a *usuaris*, i la delega al controlador d'usuaris *UsuariController* per tal que la gestioni, passant així de la *User Interface Layer* a la *Web Layer*.

El controlador té la funció de recuperar les dades demanades dins del domini dels usuaris de l'aplicació. En aquest cas concret, transmet la petició al manager *UsuariManager*, el qual pertany a la lògica de negoci de l'aplicació. Malgrat les dimensions relativament reduïdes del present projecte, s'ha volgut implementar aquesta capa de manager per practicar com funciona la comunicació entre tots els estrats del negoci.

El manager s'encarrega de transferir la petició de consulta a la implementació de la interfície DAO. Un dels objectius dels objectes *DAO* és el d'embolcallar els *POJO* per tal que la seva interacció amb els *SGBD* sigui el més òptima possible, i que futurs canvis amb els *SGBD* tinguin un impacte nul amb la resta de l'aplicació (afecta només a la definició d'aquest revestiment). Així, si es volgués canviar el connector d'accés a la BDD Spring Jdbc per qualsevol altre, només seria necessari implementar la interfície DAO amb el nou connector, i modificar un apuntador del dispatcher per tal que enllacés al nou DAO.

Una vegada realitzada la consulta al SBGD, el DAO recupera tots els POJO Usuari que compleixen la condició de consulta, i retorna el conjunt al *manager*, que el retorna a la seva vegada al *controller*. El controlador *UsuariController* construeix i retorna un objecte *ModelAndView* que conté d'una banda el nom de la vista que s'utilitzarà per mostrar el resultat, i d'altra banda el resultat de la consulta a la BDD (conjunt d'usuaris).

Per finalitzar, l'objecte *ModelAndView* arriba al *DispatcherServlet*, que recupera la vista a què fa referència i les dades que transporta, i per acabar mostra la resposta (*response*) en la vista recuperada amb les dades retornades.

4.4 Capítol 4: Implementació

4.4.1 Eines utilitzades

Al llarg de les proves realitzades durant l'elaboració del TFC, s'ha pogut observar com hi ha força entrebancs a l'hora de seleccionar un conjunt d'eines que permetin treballar amb les tecnologies que hom escull. A continuació es detalla l'elecció de cada element de desenvolupament utilitzat i justificat:

- **Java jdk1.6.0_02 :** un dels punts que no ha suposat cap problema, ja que la versió escollida és més que suficient per implementar totes les funcions java i permetre'n el seu acoblament amb el framework Spring MVC.

  Descàrrega del producte: http://java.sun.com/javase/downloads/index.jsp

- **Eclipse SDK 3.3.1 WTP all-in-one sdk R 2.0.1 :** s'escull la darrera versió estable d'Eclipse per la seva facilitat d'integració amb Tomcat 6.0, i pels plugins existents i fàcil

connectivitat amb les llibreries d'Spring; a més, per facilitar la creació de projectes s'utilitza concretament el paquet *WTP all-in-one sdk R 2.0.1 Europe*, amb el plugin Spring IDE.

Prèviament a aquesta elecció es testeja l'IDE[7] Netbeans 5.5, i es comprova que tot i facilitar enormement la gestió de les BDD respecte a Eclipse, és del tot impossible integrar-hi les llibreries d'Spring, a més de disposar d'un suport per aquest framework més que mínim; amb seguretat aquest incident pot ser resolt, però es decideix no invertir-hi més temps de l'estrictament necessari per fer funcionar un IDE.

També s'ha revisat la solució que es proposa amb APPFUSE, un esquelet d'aplicació predeterminat amb totes les funcionalitats que pot necessitar, però finalment es descarta per la quantitat important de temps i recursos (programari addicional) que s'hi han de dedicar per tal d'obtenir algun tipus de resultat.

Descàrrega del producte: http://download.eclipse.org/webtools/downloads/drops/R2.0/R-2.0.1-20070926042742/

- **Apache Tomcat 6.0 :** malgrat inicialment es planteja l'ús del servidor d'aplicacions JBoss, per si finalment es decidia plantejar la capa de persistència amb CMP, la seva integració dins l'entorn de desenvolupament Eclipse passa per ser una utopia, a no ser que s'utilitzin les versions concretes d'Eclipse 3.1.2 i de JBoss 1.4.05.GA. Finalment s'opta per la darrera versió de Tomcat, que funciona a la perfecció dins de l'IDE Eclipse.

  Descàrrega del producte: http://tomcat.apache.org/download-60.cgi

- **MySQL Server 5.0.45 Community Edition (GPL):** una de les premisses que es busca al TFC és la d'utilitzar eines de desenvolupament completament gratuïtes, amb la qual cosa l'elecció d'aquest SGBD és quasi obligat degut a les grans possibilitats que ofereix i el cost nul que té. La gestió es realitza per *CLI*[8], tot i que hi ha múltiples eines de gestió gràfica.

  Descàrrega del producte: http://dev.mysql.com/downloads/mysql/5.0.html

- **Apache Ant 1.7.0 :** tot i que inicialment no es plantejava el seu ús, una vegada comprovada la seva potència a l'hora de desplegar aplicacions als servidors d'aplicacions i de testejar-ne funcionalitats ha estat quasi obligatori la seva elecció. A més, a nivell de la implementació de l'aplicació, s'utilitza el corresponent *build.xml* associat a *Ant* per permetre la creació i inicialització automàtica de les taules de la BDD, i facilitar així el desplegament de l'aplicació.

  Descàrrega del producte: http://ant.apache.org/bindownload.cgi

- **Mozilla Firefox 2.0.0.11 :** navegador gratuït utilitzat per validar tota la interfície gràfica de l'aplicació, utilitzant una resolució de pantalla de 1024x768 píxels.

  Descàrrega del producte: http://www.mozilla-europe.org/es/products/firefox/

7 Integrated development environment, o entorn integral de desenvolupament.

8 Command line Interface, o línia de comandes.

- **Llibreries :** a hores d'ara, les llibreries utilitzades en les proves inicials d'implementació de *cdu* amb Spring MVC són: *commons-logging.jar*, (necessàries per aplicacions JEE), jstl.jar (definició de tags propis d'Spring), *junit.jar* (testeig *POJO*), *log4j-1.2.9.jar* (creació de logs d'aplicació per facilitar la feina de trobar errors), spring.jar (llibreries pròpies d'Spring), standard.jar, javax.servlet.jar (contenen classes HttpRequest i HttpResponse necessàries per l'aplicació), mysql-connector-java-5.0.7-bin.jar (per accedir a la BDD relacional MySQL).

4.4.2 Estructura

L'aplicació de gestió documental s'ha estructurat seguint les especificacions de les aplicacions J2EE, sempre sota el prisma del framework Spring. Així, l'estructura de carpetes en la seva implementació ha estat la següent

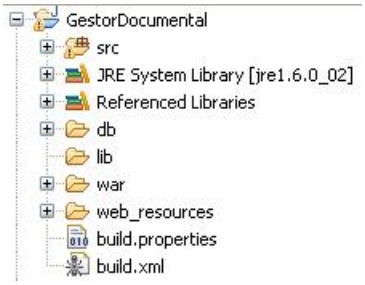

Figura 35. Estructura de carpetes : GestorDocumental.

Dins de la carpeta *src* s'hi han contingut totes les classes Java pròpies de la aplicació. Per tal de millorar la seva gestió i localització d'objectes pertanyents a una mateixa família lògica, s'han agrupat en quatre grans packages:

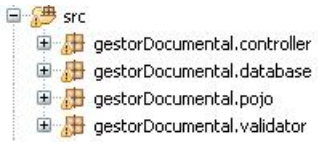

Figura 36. Estructura de carpetes : src.

El package *gestorDocumental.controller* conté totes les classes *Controller* d'Spring de l'aplicació, ja siguin de tipus *SimpleFormController* o *AbstractController*. Mentre que la primera s'ha utilitzat per tots aquells controladors que necessiten recuperar dades a través del corresponent *Command*, rellançar vistes utilitzant el mecanisme que ofereix *BindException*, etc, la segona s'ha utilitzat en aquells casos en els quals no es requereixen tantes funcionalitats.

Tot i que al llarg del TFC s'ha tingut en compte la possibilitat d'unir diferents *Controller* d'un mateix tipus per tal de reduir-ne el seu nombre, finalment s'ha desestimat ja que així s'aconsegueix tenir una estructura més clara.

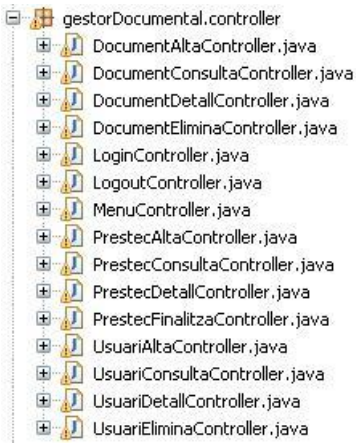

Figura 37. Estructura de carpetes : gestorDocumental.controller.

Dins del package gestorDocumental.database s'hi han inclòs totes aquelles classes que tenen relació directament amb la persistència de dades: interfícies *Dao* i la seva implementació, en aquest cas *DaoJdbc* ja que s'ha utilitzat la solució Jdbc que ofereix el framework Spring; aquesta estructura permet que, en cas que es vulgui canviar la persistència de dades, únicament substituint la classe *xxxDaoJdbc* per la corresponent de la nova tecnologia escollida (*CMP*, *BMP*, etc), l'aplicació seguirà funcionant.

A més de les interfícies *Dao* i llurs implementacions *DaoJdbc*, dins del package també s'hi han inclòs els *Manager* del TFC, els encarregats de gestionar tota la lògica del negoci. D'igual manera que en el cas anterior, s'ha volgut separar aquestes tres classes segons els tres grans Pojo que formen l'aplicació MVC, *Document*, *Prestec* i *Usuari*, per tal de facilitar la seva comprensió i estudi:

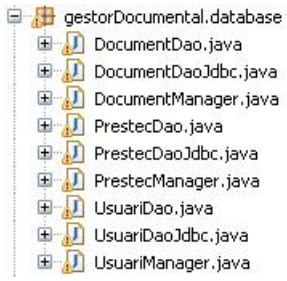

Figura 38. Estructura de carpetes : gestorDocumental.database.

El package *gestorDocumental.pojo* conté tots els *Pojo* de l'aplicació. Com que un *Pojo* no deixa de ser un objecte clàssic de Java (no aporta cap novetat), s'ha construït el TFC de manera que utilitzi el mínim nombre de *Pojo* possible: un per cada element bàsic del gestor documental (*Document*, *Prestec*, *Usuari*), i un de reforç que s'utilitzarà en les cerques que es podran realitzar sobre, precisament, aquests elements bàsics:

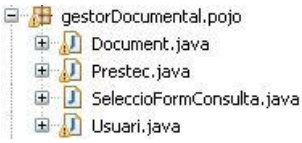

Figura 39. Estructura de carpetes : gestorDocumental.pojo.

El darrer package gestorDocumental.validator conté les classes que s'utilitzaran com a validadores de primer nivell dels formularis del gestor documental (bàsicament atributs dels formularis que no poden estar en blanc). A més d'elles, també es farà una posterior validació d'altres dades (per exemple duplicitat o no existència de PK), però ja en elements més propers a la capa de persistència (*Manger - Controller*)

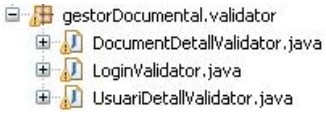

Figura 40. Estructura de carpetes : gestorDocumental.validator.

A més de les classes Java, també són necessàries llibreries suplementàries que permetin la construcció de la aplicació (llibreries d'Spring, d'accés a la BDD relacional, per l'ús d'etiquetes JSTL, per utilitzar servlets, etc) :

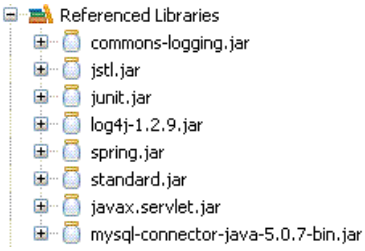

Figura 41. Estructura de carpetes : Referenced Libraries.

Separada de les dues àrees anteriors, hi ha tota la part corresponent al disseny WEB, juntament amb alguns elements propis dels frameworks com Spring.

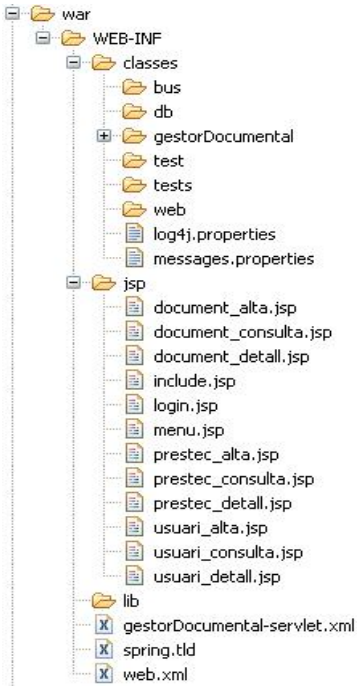

Figura 42. Estructura de carpetes : war.

Dins de *WEB-INF* hi ha dos elements importants: *log4j.properties*, que permet activar logs al servidor d'aplicacions i que facilita enormement la tasca de detecció d'errades de programació, i *messages.properties*, un arxiu també de text que conté una sèrie de variables per inicialitzar a posteriori tags JSTL, i actualitzar així de forma immediata múltiples elements HTML que mostren un mateix valor.

Dins de jsp hi tenim totes les pàgines JSP del TFC. Com que no aporten res de nou i ja es té més experiència en la seva implementació, s'han creat els jsp pensant que visualitzi continguts diferents si el perfil de l'usuari és gestor o no. Hi ha força excepcions també que es controlen des d'aquest primer nivell (per exemple, si un document està en préstec, no es mostra el botó per donar-lo de baixa, etc.).

Finalment, a part dels arxius *spring.tld* (propi del framework) i *web.xml* (propi de l'arquitectura J2EE), hi ha el dispatcher d'Spring *gestorDocumental-servlet.xml*. Es pot considerar el core de l'aplicació, ja que defineix totes les interrelacions que hi haurà entre els elements que composen tota l'aplicació MVC.

Separada de la carpeta *WEB-INF*, es defineix web-resources, una carpeta que conté tots aquells elements complementaris (imatges i fulls d'estil *CSS,* en aquest cas), i que degut a la seva naturalesa, a l'arquitectura J2EE i als servidors d'aplicacions (com l'utilitzat *Tomcat*), no poden estar dins del context de *WEB-INF*.

Figura 43. Estructura de carpetes : web+resources.

Per finalitzar, tenim *build.xml*, un arxiu per mitjà de l'ús d'ANT, no només permet fer *deploy*, *build*, *undeploy*, *clean*... de l'aplicació, sinó que també s'utilitza per crear les BDD implicades a MySQL i la seva inicialització. L'arxiu *build.properties* s'utilitza per definir-hi les constants de configuració utilitzades pel *build.xml*.

Figura 44. Estructura de carpetes : build.properties i build.xml.

4.4.3 Desplegament

L'aplicació s'entrega en format WAR, estàndard d'aplicacions: gestorDocumental.war. L'estructura interna de carpetes del .war és similar al mostrat en l'apartat 2, amb la diferència que les compilacions de classes de /src aquí queden ubicades dins de la carpeta classes.

Figura 45. Estructura de carpetes : desplegament gestorDocumental.

Per desplegar l'aplicació, s'han validat dues opcions:

- **Apache Tomcat 6.0:** descomprimint el .war dins de la carpeta deploy, arrancant l'aplicació i accedint a través del navegador a la url: http://localhost:8080/gestorDocumental/login.htm

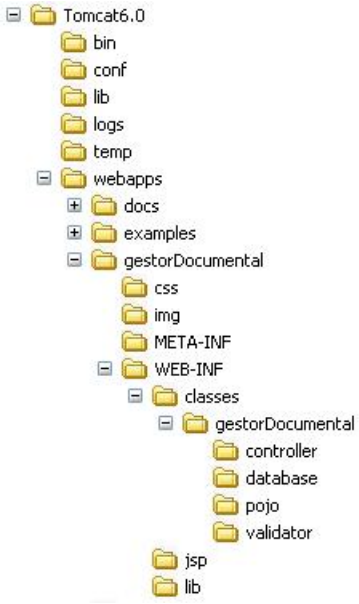

Figura 46. Estructura de carpetes : desplegament amb Tomcat 6.0.

- **JBoss 4.0.4-GA :** es copia l'arxiu .war dins de la carpeta deploy (exemple: C:\jboss-4.0.4.GA\server\default\deploy\gestorDocumental.war), s'executa C:\jboss-4.0.4.GA\bin\run.bat, que, entre d'altres coses, desplega tots els .war que hi ha dins de deploy, i s'accedeix d'igual manera a l'aplicació que en el cas anterior: http://localhost:8080/gestorDocumental/login.htm

En ambdós casos, la pantalla inicial que es mostrarà serà la mateixa que la indicada a la pàgina 30 del present document.

Com a pas previ a l'execució de l'aplicació, s'ha d'inicialitzar la BDD amb l'script adjunt a l'entrega: *script_sql.txt*. Dins de l'script es poden veure quins són els usuaris i contrasenyes per entrar a l'aplicació utilitzant el perfil d'usuari o bé el de gestor.

Pel que fa a la configuració de la connexió entre l'aplicació i MySQL, s'ha de verificar que els valors de connexió indicats al bean dataSource del dispatcher gestorDocumental-servlet.xml s'avenen a la realitat:

```
<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName"><value>com.mysql.jdbc.Driver</value></property>
        <property name="url">
                <value>jdbc:mysql://localhost:3306/gestordocumental</value>
        </property>
        <property name="username"><value>root</value></property>
        <property name="password"><value>root</value></property>
</bean>
```

Figura 47. Definició bean dataSource : configuració de connexió a MySQL.

4.4.4 Testeig

A més de validar que totes les condicions que s'han anat comentant en la fase d'anàlisi es compleixin, s'han realitzat les següents validacions addicionals:

Acció	*Validació realitzada*
Identificació	No es permet l'entrada si es deixa el camp usuari i/o password en blanc. Les dades introduïdes han de coincidir amb les que guarda la BDD
Alta i modificació de documents	Els camps obligatoris no es poden deixar en blanc.
Modificació de documents	Un document en préstec no es pot definir com a no prestable
Baixa de documents	Un document en préstec no es pot donar de baixa.
Alta i modificació d'usuaris	Els camps obligatoris no es poden deixar en blanc.
Modificació d'usuaris	Els usuaris sense perfil gestor únicament es poden modificar el seu usuari; els gestors poden modificar qualsevol usuari, excepte el rol del propi gestor que s'ha identificat a l'aplicació.
Baixa d'usuaris	Un usuari amb un préstec actiu no es pot donar de baixa.
Alta de préstecs	Existeix la referència al document que es vol demanar en préstec i a l'usuari que el demana. Aquests dos camps no es poden deixar en blanc. Un document no pot estar en préstec a la vegada a dos usuaris diferents. Un usuari no pot tenir més de tres préstecs actius a la vegada.
Consulta de préstecs	Per part d'un usuari sense perfil de gestor, només pot consultar tots els seus préstecs, tant els actius com d'històric. La consulta per part d'un gestor, en canvi, permet obtenir tots els préstecs de tots els usuaris (o filtrar-los).
Baixa de préstecs	No es permet retornar un préstec ja retornat anteriorment.

4.5 Capítol 5: Documentació

La Memòria i la Presentació del TFC formen gran part del nucli de documentació, no obstant, també s'ha creat una documentació extra corresponent al codi de l'aplicació amb JavaDoc, amb l'objectiu d'oferir una visió a més baix nivell que no les dues primeres.

JavaDoc és una utilitat que permet generar de forma automàtica i a partir d'un codi font .java, API en format HTML. L'únic requisit necessari per la producció de la documentació és la de comentar internament totes les classes de Java, utilitzant una nomenclatura i uns tags estàndards, segons les indicacions de la web http://java.sun.com/j2se/1.5.0/docs/tooldocs/windows/javadoc.html.

Com ja s'ha indicat a l'apartat 5.1.5, l'entrega del TFC conté la documentació Javadoc de la codificació de l'aplicació (gestorDocumental-doc.zip). Per tal de visualitzar-la, únicament s'ha de descomprimir en local i accedir a la pàgina inicial /doc/index.html:

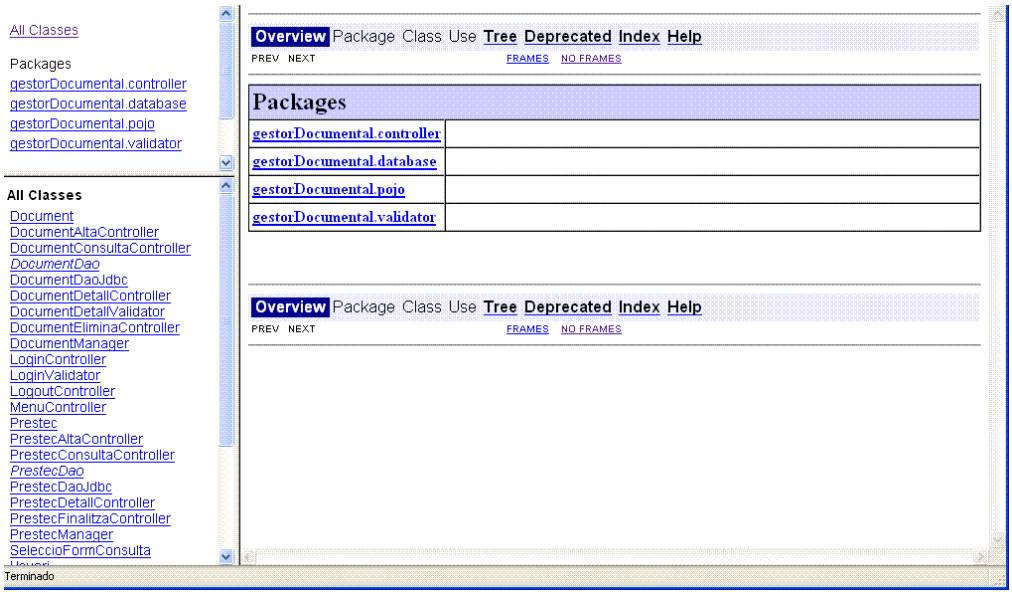

Figura 48. Javadoc : pàgina inicial de documentació del Gestor Documental

La informació que s'hi mostra està pensada per complementar i aprofundir la facilitada en el present document, i dóna més dades a baix nivell de tot l'engranatge muntat pel Gestor Documental. A mode d'exemple, s'adjunta part de la descripció de la interfície DocumentDao:

Overview Package Class Use Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

gestorDocumental.database

Interface DocumentDao

All Known Implementing Classes:
DocumentDaoJdbc

public interface **DocumentDao**

Document DAO interface gestorDocumental.database.DocumentDao. Interfície que conté els mètodes de gestió de la capa de persistència (CRUD = create, read, update, delete) de l'objecte Document. Per tal de facilitar al màxim la modularitat que pot arribar a oferir Spring, qualsevol gestor de la capa de persistència haurà d'implementar els mètodes d'aquesta interfície. La classe que implementa la interfície queda definida dins del dispatcher gestorDocumental-servlet.xml, bean documentDao, vincle definit per l'atribut class.

Version:
1.0
Author:
Jordi Blasco Planesas

Method Summary

Figura 49. Javadoc : documentació de la interfície DocumentDao.

4.6 Capítol 6: Conclusions

4.6.1 Visió general

Al llarg de tot el procés de creació del present TFC s'ha pogut comprovar com les estructures multicapa que segueixen el model MVC presenten avantatges molt importants. Amb seguretat, el tret més destacat és la separació del negoci entre model, vista i controlador, ja que facilita en gran mesura el desenvolupament del programari i el posterior manteniment.

El fet de tenir capes independents permet el desenvolupament en paral·lel de l'aplicació entre diferents departaments, augmentant així la producció, el poc condicionament de possibles retards en la implementació d'alguna capa vers la resta, i la millora de les tasques futures de manteniment, ja que qualsevol revisió que es faci d'un element d'una capa no ha de tenir afectació en la resta de capes.

Dins del conjunt de possibilitats existents que implementen el model MVC s'escull el framework Spring, i més concretament el seu subconjunt Spring MVC. A més de les millores pròpies del l'arquitectura MVC, Spring ofereix avantatges com l'ús de POJO en comptes d'EJB per tal d'obtenir aplicacions Lightweight[9] no intrusives, l'ús de *dependency injection* i *mock object* que permet deslocalitzar qualsevol *POJO* de l'aplicació i testejar-lo sense necessitat de tenir tota l'aplicació muntada, o l'ús de DAO per fer completament modulable l'accés a la BDD.

9 Aplicacions poc feixugues gràcies al retorn a un escenari en el qual les classes del domini lògic de l'aplicació (POJO) esdevenen el centre del negoci; a més, els POJO no necessiten ser conscients que formen part d'un framework, fet que alleugereix el propi framework.

4.6.2 Visió local

Malgrat l'elecció d'Spring MVC, en cap moment s'ha perdut consciència que l'actual projecte desenvolupat sota el model MVC de JEE, ja sigui utilitzant una visió optimitzada (framework) o una solució completament pròpia, presentaria els avantatges inicials anteriorment comentats. Spring és un framework amb moltes possibilitats, un gran recorregut, i amb seguretat la seva productivitat millora a mesura que la magnitud del projecte augmenta.

Precisament per aquest darrer motiu, es pot arribar a qüestionar si és necessari tanta estructura i control per una aplicació reduïda com aquesta i, després de treballar-hi durant tot el curs, s'ha arribat la conclusió que l'estructura escollida acaba sent un reflex de fins on es vol aprofundir en una tecnologia, tenint sobretot una visió de futur. Amb tota seguretat es pot construir la mateixa aplicació eliminant certs elements utilitzats[10], però s'ha intentat seguir una possible solució que podria ser perfectament vàlida per un projecte major, traient partit en tot moment els avantatges del paradigma MVC.

Com a punt menys positiu s'ha de destacar la quantitat enorme d'informació que hi ha disponible de totes les possibilitats implementades sota el paraigua de JEE, un punt que es veu agreujat si, com és el cas, és la primera vegada que s'ha entrat en el món de les aplicacions distribuïdes. A més, una vegada feta l'elecció d'Spring MVC, s'ha corroborat com el punt més incòmode pertany, precisament, en aquest mateix àmbit documental: hi ha molt recursos a nivell teòric, però relativament pocs a nivell pràctic, tret que dificulta enormement el primer contacte amb aquesta tecnologia inicialment desconeguda.

Finalment mencionar que la realització del projecte hagués estat impossible sense les directrius dictades per la metodologia RUP, i en particular sense el consell del consultor de fer coincidir cadascuna de les iteracions (cicle de vida en cascada) amb la codificació d'un subsistema complet del projecte: s'ha comprovat empíricament com l'experiència acumulada en preparar completament un subsistema ha servit per reduir per 10 el temps necessari per implementar completament el segon subsistema.

10 Figura 34. Diagrama de col•laboració: consulta d'usuaris realitzada per un gestor.

5 Glossari

- **Document :** per *document* s'entén tot aquell element que és susceptible de formar part d'un arxiu. A mode d'exemple, es consideren *documents* els daguerreotips, cianotips, ambrotips, fotografies, albúmines, estereoscòpiques, negatius (35mm, plaques vidre, cel·luloide), llibres (amb enciclopèdies com a extensió), manuscrits, microfilms, gravats a l'acer, gravats amb fusta, impresos tríptics díptics i fulls, cartells, litografies, revistes (periòdiques o no), discos pedra i vinil, cilindres de cera, CD, K7, videoteca...

- **Catàleg :** conjunt de documents que conformen l'arxiu.

- **Subsistema :** cadascuna de les grans àrees en què es divideix el projecte: subsistema d'usuaris, subsistema de catàleg (documents), subsistema de préstecs i subsistema d'identificació.

- **Perfil :** tret que distingeix un usuari que pot realitzar tasques administratives de l'arxiu d'un altre usuari que, a grans trets, només pot realitzar consultes sobre el catàleg. També es pot referir a perfil per mitjà del mot *rol*.

- **MVC :** patró arquitectònic que separa les dades d'una aplicació (model), la interfície que veu l'usuari (vista), i la lògica de control (controlador). El patró MVC es troba habitualment en aplicacions WEB, on la vista està formada per pàgines HTML o JSP, el model correspon al SGBD i els controladors per accedir-hi, i el controlador representa la lògica de negoci de l'aplicació en forma d'arxius JAVA, XML, etc.

- **SGBD :** els sistemes de gestió de base de dades són un tipus de programari específic que serveix d'interfície entre la base de dades, l'usuari, i les aplicacions que hi tenen accés. Entre els SGBD lliures (gratuïts) hi destaca MySQL, l'utilitzat en el present TFC.

- **CSS :** acrònim de Cascading Steel Sheet, són fulls d'estils que neixen per separar l'estructura d'un document de la seva presentació (forma). Els arxius CSS s'utilitzen per donar format a una pàgina web, i permet obtenir una codificació més neta i la reutilització d'estils entre els diferents elements que composen una pàgina web (HTML, HTM, JSP...).

- **JSTL :** la llibreria JavaServer Pages Standard Tag Library és un component de l'especificació JEE, composada per un conjunt de llibreries d'etiquetes amb utilitats que s'usen habitualment en el desenvolupament de pàgines dinàmiques. Les que s'utilitzen en aquest projecte són *core* (per iteracions, condicionals, manipulació d'URL, etc.) i *fmt* (inclou la internacionalització i el format de valors).

- **ER :** fa referència als models entitat-relació, diagrames que s'utilitzen pel modelat de la persistència de dades d'un sistema d'informació. En aquests models s'hi mostren les entitats rellevants i les seves interrelacions i propietats.

- **Open Source :** llicència que s'utilitza majoritàriament en programes, els quals segueixen els principis del moviment Open Source. Per aconseguir que una aplicació tingui llicència Open Source, és necessari, entre d'altres coses, que sigui de lliure distribució i que es pugui obtenir el seu codi font.

- **Framework :** representa una arquitectura de programari que modela les relacions generals entre totes les entitats que conformen el domini, i són dissenyats amb l'objectiu de facilitar el desenvolupament de les aplicacions.

6 Bibliografia

LADD Seth, DAVISON Darren, DEVIJVER Steven, and YATES Colin. Spring MVC and Web Flows. New York, U.S.A.: APress, 2006

HEMRAJANI Anil. Agile Java Development with Spring, Hibernate and Eclipse. U.S.A.: Sams Publishing, 2006

RAIBLE, Matt. Spring Live. U.S.A.: SourceBeat LLC, 2005

JOHNSON Rod, HOELLER Juergen, ARENDSEN Alef, RISBERG Thomas, SAMPALEANU Colin. Professional Java Development with the Spring Framework. U.S.A.: Wiley Publishing Inc, 2005.

RICHARDSON Chris. POJO's In Action. Developing enterprise applications with lightweight frameworks. Greenwitch CT: Manning Publications Co, 2006

CAMPDERRICH FALGUERAS, Benet. Enginyeria del programari. 1a edició, Barcelona: Fundació per a la Universitat Oberta de Catalunya, 2004

RISBERG, Thomas. Developing a Spring Framework MVC Application Step by Step. 2005.

http://www.springframework.org/

http://dev.mysql.com/doc/index.html

http://www.javabeat.net/articles/2007/09/introduction-to-spring-ide-2-0/6

http://es.wikipedia.org/wiki/Wiki

http://www.springhub.com/component/option,com_weblinks/catid,17/Itemid,23/

7 Annexos

7.1 Implementació

En tot projecte informàtic existeixen una sèrie de factors externs que poden arribar a causar algún edarreriment puntual sobre la planificació realitzada al principi del mateix. En el cas present, tal i com s'ha informat durant la fase d'implementació amb Spring, al llarg de tot aquest període han anat sorgint múltiples incidents, tant per errors de llibreries[11], com problemes que generen amb els tags JSTL[12], errors propis d'implementació motivats per la falta d'experiència, massa informació per consultar i a la vegada massa poc detallada per les solucions que un té pensat implementar, pocs exemples disponibles i que funcionin, dies perduts per malaltia, etc.

Tots aquests motius han dut a fer alguns retocs sobre la idea inicial del TFC, que si bé gran part ja s'han exposat al llarg del present document (com per exemple, tota la capa de la vista de l'aplicació, que ha sofert canvis respecte llur concepció inicial), hi ha una sèrie de canvis que es prefereixen exposar a mode d'annex, per tal que es pugui veure el procés d'evolució del TFC.

- s'uneixen usuaris i gestors en una sola taula a nivell de persistència, i afegint l'atribut rol que mostra si un usuari té perfil de gestor o bé únicament d'usuari.

- es treballa sobre una única taula de documents, ja que si bé es perd profunditat en l'especificació d'aquesta família d'objectes, no aporta res destacable pel que fa a l'aprenentatge de la creació d'una aplicació utilitzant el model MVC, i permet invertir el temps que s'estalvia en l'estudi d'Spring MVC.

- el subsistema d'identificació es modifica en el fet que el canvi de contrasenya es deixa de realitzar a partir de la validació de la mateixa (cdu CanviContrasenya), i es traspassa dins de la pròpia modificació de dades de l'usuari (cdu ModificacióUsuari). A més, es dota a qualsevol usuari de l'opció de modificar la resta dels seus atributs, funcionalitat inexistent en el plantejament inicial.

- es descarta el control de sol·licitud de reserva i posterior espera de 3 dies, ja que la seva implementació és més pròpia d'un trigger intern de la BDD que no de l'estructura MVC de l'aplicació

- s'implementa únicament la possibilitat de realitzar el préstec in situ: els usuaris amb perfil gestor són els únics que poden donar d'alta un préstec, i també validar-ne la devolució. El temps que s'estalvia en aquest punt, s'inverteix en l'estudi i testeig dels tags JSTL, i en l'ús de classes Validator d'Spring MVC per validar dades de formularis[13].

11 Veure incidents exposats a l'apartat 5.4.1.

12 JSTL té un bug no documentat que consisteix en no acceptar com a vàlids mètodes getters/setters que tinguin dins el seu nom dues majúscules seguides qualsevols; per exemple, un objecte que tingui el mètode getMMxxxx( ) causarà un error de tipus org.apache.jasper.JasperException.

13 Tota la informació referent a aquest punt està detallada a la documentació del codi Javadoc, ja que es considera que és un element de baix nivell i que ha de ser comentat més àmpliament de l'espai de què es disposa a la Memòria.

- es millora l'apartat de cerques sobre una BDD, habilitant noves opcions com la consulta múltiple (si no es posa cap cadena de caràcters a consultar, retorna tots els objectes de la BDD), la consulta per atributs (es pot realitzar sobre la majoria dels atributs dels objectes), etc.

- es millora l'apartat de redireccionaments pels POJO de l'aplicació (usuaris, documents i préstecs), permetent que la consulta prèvia necessària a la posterior acció de modificació/baixa/devolució enllaci directament a una pantalla de detall de l'objecte, sense necessitat d'introduir l'identificador únic de l'objecte.

A continuació es mostren els escenaris modificats:

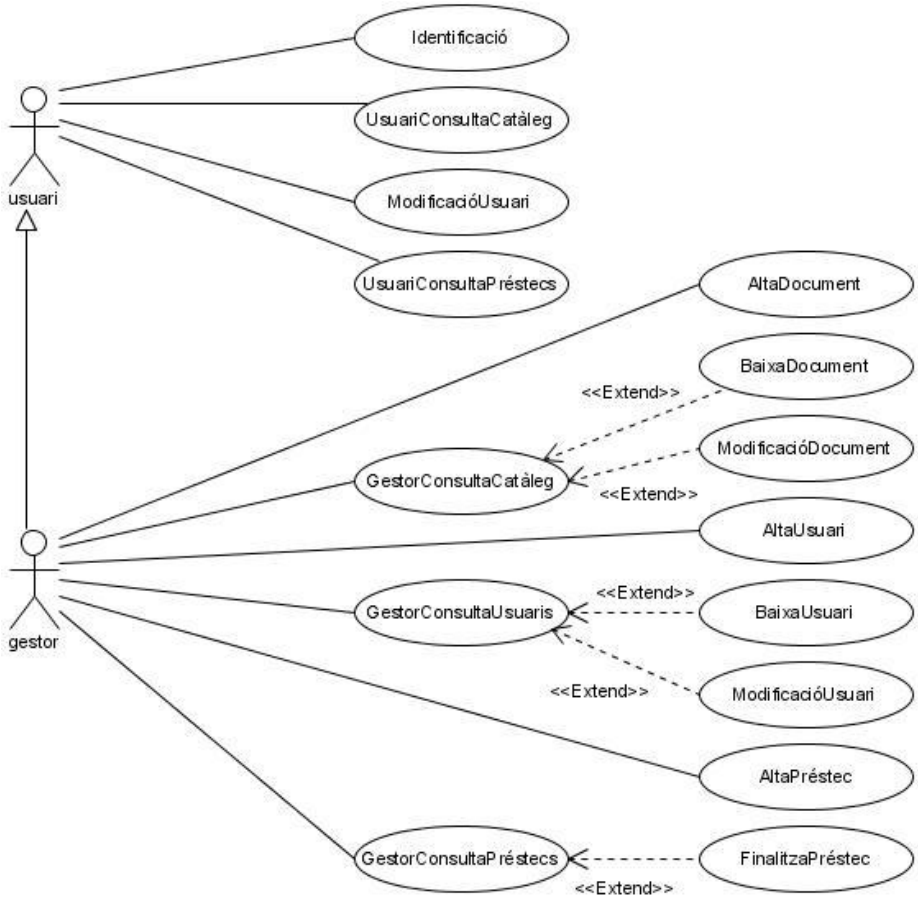

Figura 50. Esquema de casos d'ús: escenari general modificat.

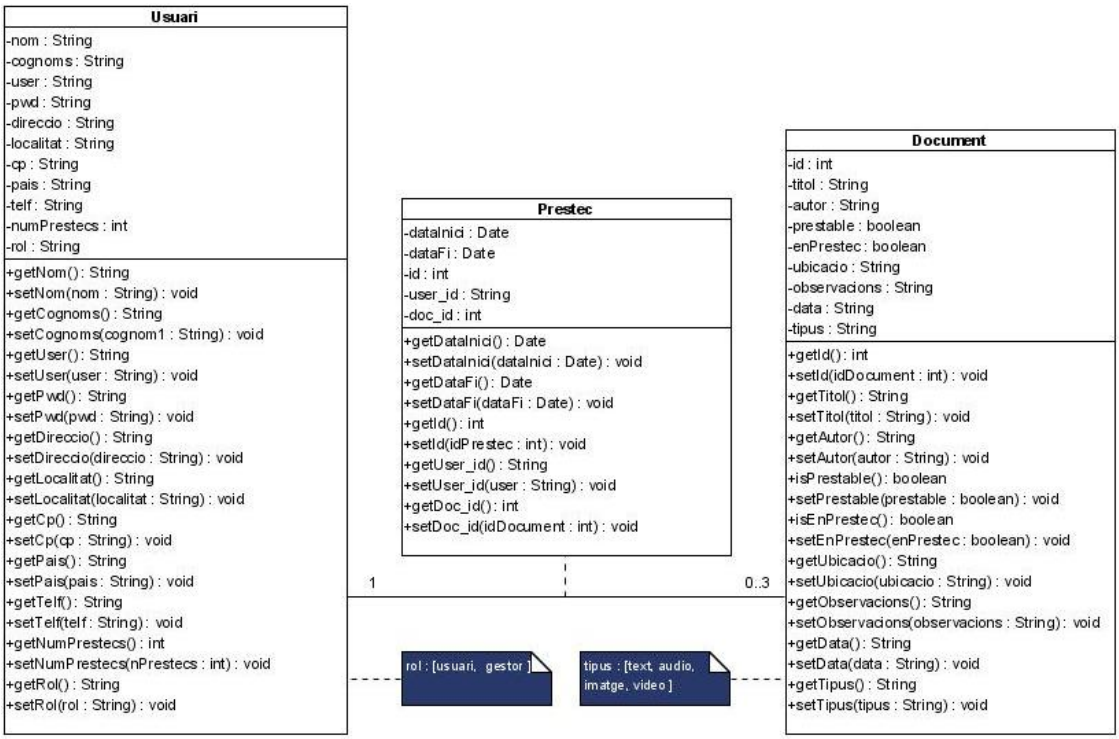

Figura 51. Diagrama de classes: escenari general modificat.

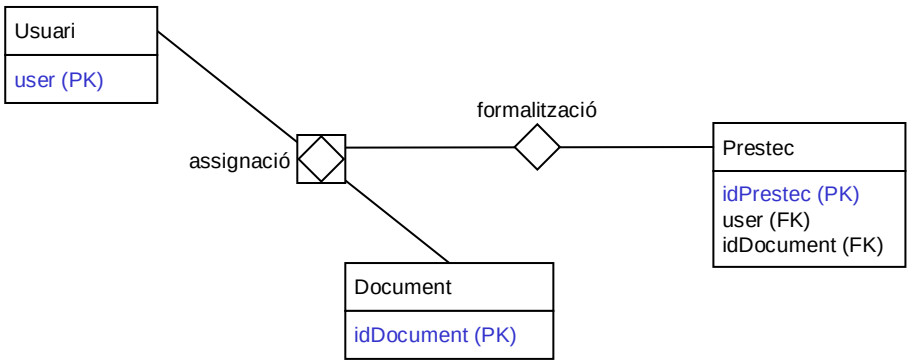

Figura 52. Diagrama ER: escenari general.