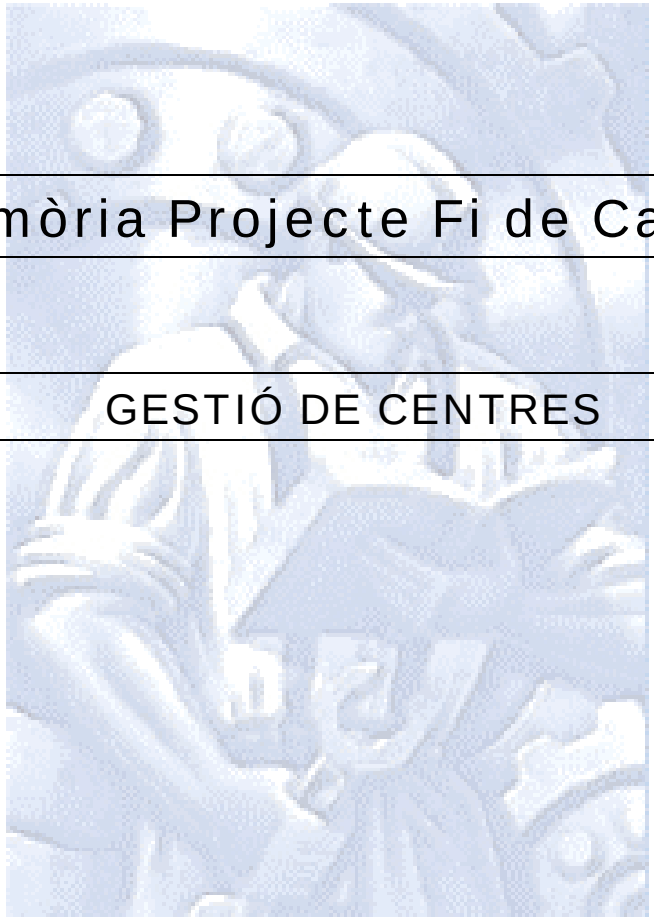
Memòria Projecte Fi de Carrera

GESTIÓ DE CENTRES



Emilio Valdivia Matarín

Enginyeria en Informàtica

Consultor

Javier Ferró Garcia

Data lliurament
14 de gener de 2008

Agraïments

"A Ángela, por ayudarme a seguir hasta el final,
por estar a mi lado y no dejarme abandonar,
por su paciencia y por llenarme de felicidad."

" A mi familia, por su cariño
y su comprensión."

" A mis amigos, por apoyarme
en todo momento."

Títol

Gestió de Centres en J2EE

Resum

Amb aquest projecte es pretén donar una primera visió sobre la solució òptima al problema actual que tenen algunes comunitats autònomes l'hora de coordinar els seus centre educatius, que gestionen tota la informació del centre de forma local i sense compartir les dades comuns, com són les del professorat o alumnat. A més gestionen aquesta informació amb eines desenvolupades en programari privatiu.

En aquest projecte es vol unificar les dades en una única base de dades relacional però amb la potència d'una aplicació desenvolupada amb programació basat en components, independent de la plataforma i totalment escalable, ja que el sistema educatiu va canviant la seva normativa contínuament i les eines utilitzades al centre per a la seva gestió han de ser revisades i ampliades per complir aquesta normativa.

Veurem el desenvolupament del projecte pas a pas, des de l'estudi dels marcs de treballs més importants que es poden incorporar en projectes J2EE, passant per un anàlisi i disseny acurat, fins arribar a la implementació dels mòduls bàsics que inclouria el sistema de gestió dels centres intentant aprofitar totes les avantatges que ens ofereixen els marcs de treball més adients i tecnologies de darrera generació con AJAX per a fer un sistema flexible i robust capaç d'assolir totes les necessitats de gestió de la informació dels centres. També veurem com apliquen diversos patrons en aquesta arquitectura client-servidor de tres capes aconseguint, entre altres aspectes, que cada component s'assigni a una capa a un cert nivell d'abstracció.

TAULA DE CONTINGUTS

ÍNDEX DE FIGURES

1 Introducció

1.1 Descripció del projecte

Com a descripció breu del projecte proposat, el que es vol realitzar és una eina que ajudi als centres en la seva gestió diària: expedient dels alumnes, horari de les matèries, fitxes del professorat...

Aquesta eina, bàsicament, tractaria de gestionar les dades emmagatzemades a una base de dades centralitzada mitjançant una aplicació WEB desenvolupada amb tecnologia JEE on el servidor d'aplicacions donaria suport als usuaris dels diferents centres.

Per entendre millor el plantejament d'aquest projecte, es necessari parlar sobre el context actual, on els centres gestionen les dades de forma local, sense tenir pràcticament cap connexió entre ells, simplement un document en paper amb el certificat acadèmic amb la relació dels cursos i notes de les matèries realitzats per l'alumne. A més, estan lligats a un entorn privatiu que fa impossible la potabilitat a altres configuracions diferents de les proposades a cada centre.

Això fa que les dades es dupliquin, i moltes vegades que es produeixin incoherències, ja que el mateix alumne pot tenir dates de naixement diferents en dos centres distints. Per altra banda, per intercanviar dades amb altres comunitat autònomes, o simplement quan els alumnes han d'anar a una altra institució, com per exemple una universitat, les seves dades s'han de enviar mitjançant l'emplenament d'un fitxer de text, feina que realment s'hauria d'automatitzar i estandarditzar, com per exemple amb la generació d'un arxiu XML ben estructurat amb les dades demanades d'una altra institució autoritzada.

És veu clarament la necessitat dels centre de compartir només aquelles dades que siguin necessàries, com poden ser les dades de l'alumne, el seu historial acadèmic, o les dades del professorat, ja que existeix una certa mobilitat tant per part d'alumnes com dels professors.

Però haurà unes altres que només correspondran a la gestió pròpia del centre. Concretament es poden distingir :

- Els departaments que té definit el centre.
- Horari d'impartició de cada matèria i associats a cada professor
- Ensenyaments i configuració d'optatives que ofereix el centre.
- Número de cursos i grups de cada ensenyament

Per altra banda, es tracta de dur a terme el desenvolupament d'una aplicació que gestioni les dades de tots el centres tenint en compte els diferents usuaris amb diferents perfils (professors, pares, alumnes, administradors, personal no docent...), la qual cosa implica la gestió d'accés a l'aplicació des de qualsevol ordinador a través de la xarxa amb connexió segura. D'aquesta manera, un usuari, segons el perfil, veurà i tindrà accés a una sèrie de funcionalitats diferents.

Algunes característiques d'aquest sistema que ja es poden distingir són:

- Flexibilitat: adaptatiu als canvis continus de les normatives en educació.
- Robustesa: sistema robust que suporti peticions de molt d'usuaris .
- Accessibilitat: ha de tenir en compte les normes d'accessibilitat per a varis perfils d'usuari.

1.2 Objectius

Es poden distingir dos objectius principal en aquest projecte:

Un és cobrir la necessitat existent de proporcionar una eina potent per a la gestió i centralització de totes les dades del centres com a una única aplicació amb la qual els centres puguin interactuar, a més de aconseguir separar l'estructura actual, en una arquitectura de tres capes, diferenciant la capa de persistència, la de lògica i la de presentació.

Altre, és investigar els bastidors de codi obert existents i darreres tecnologies aparegudes, que per les seves característiques, són més adients a l'hora de desenvolupar aquest projecte, i una vegada seleccionats, estudiar-los en profunditat per treure el màxim profit i així implementar una aplicació de qualitat, bona de mantenir i escalable. D'aquesta manera, s'adquireix un ampli coneixement sobre les tecnologies actuals i d'utilitat, que dotaran a l'alumne de més criteri a l'hora de desenvolupar altres projectes i veure altres possibilitats a més de les utilitzades normalment al món laboral.

Els beneficis de tenir una aplicació amb arquitectura client-servidor amb tecnologia JEE té múltiples avantatges. A continuació es presenta un breu resum de les característiques que pot aportar al projecte i que ens donarà una visió dels objectius específics a assolir:

- **Seguretat**: Les dades personals que gestiona el centre són de suma importància, i considerades de tercer nivell si parlem del registrament de les dades de necessitat educatius especials o de les fotografies dels alumnes. Aquesta arquitectura ens permetrà definir una política detallada per a cada un dels perfils dels usuaris de l'aplicació.
- **Integritat de les dades**: Hi ha conjunts de dades que s'ha de tractar com una unitat atòmica, evitant així, errors com per exemple que un professor estigui destinat a dos centres distints si no es itinerant. A més, les dades personals d'una persona no poden estar duplicades de tal manera que hi hagin diferències si el registren com a alumne o com a professor. També és de suma importància la persistència de les dades i els sistemes de recuperació .
- **Portabilitat**: La independència del entorn proporcionada per aquesta tecnologia, ofereix una gran avantatge per que fa a la instal·lació en qualsevol servidor, essent multiplataforma i no dependre d'un sistema operatiu concret. També s'ha d'aconseguir, amb l'ajuda dels bastidors, ser independent de la base de dades utilitzada. D'aquesta manera s'aconsegueix que l'aplicatiu programari tingui una llarga vida útil.
- **Comunicació**: Si utilitzem un format estàndard per a estructurar les dades a l'hora de fer viatjar les dades per la xarxa, aconseguirem que la comunicació e integració amb altres sistemes d'informació sigui molt més fàcil i que és pugui compartir la informació i coneixement de forma més àgil.
- **Manteniment i escalabilitat**: es pot esbrinar clarament, que el manteniment és la etapa cicle de vida del projecte que segurament ocuparà més esforços perquè els continus canvis en el sistema educatiu faran que s'hagin d'efectuar un manteniment adaptatiu i s'haurà d'afegir noves funcionalitats. Ha de ser un sistema intel·ligible, es a dir, que es pugui entendre a partir del codi i la documentació. Depenen d'això, es podrà mesurar el seu grau en termes de cost de manteniment. També la independència de capes beneficia el manteniment perfectiu, de tal manera

que es podran introduir micromillores de manera gradual a cadascuna de les capes sense problemes d'inconsistència.

- **Eficàcia i eficiència**: Per una banda eficàcia del programari per a cobrir les necessitats dels usuaris en fer les seves accions i funcions. Per altra banda eficiència de l'aplicatiu per a fer les seves funcions sense carregar excessivament els recursos del maquinari i donar servei permanent als usuaris sense talls innecessaris.
- **Explotació**: Les dades emmagatzemades també han de servir generar coneixement útil, com per exemple estadístiques comparatives entre distintes etapes que poden servir per a prendre decisions als responsables sobre decidir. Aquesta informació és més fiable quan es disposa d'un sistema robust i estructurat capaç de respondre a consultes pesades i complicades.

1.2.1 Tecnologia

Avui en dia existeixen a nostre abast tot una sèrie de bastidors que ens podrien ajudar en el desenvolupament d'aquest projecte, bastidors que treballen en totes les capes i amb diferents interfícies.

Cadascun del bastidors tenen una sèrie de característiques que el fan més apropiats per una funcionalitat que per a una altra. Per aquest motiu, abans de decidir quins bastidors utilitzar, s'han d'avaluar les seves avantatges i desavantatges segons les característiques del nostre projecte.

Es justifica la utilització de bastidors, a part d'evitar no reinventar la roda, per les dues raons principals següents:

- Estalviant temps i esforç
- Ens donen una implementació parcial del disseny que podem reaprofitar.

Però per altra banda, el principal inconvenient de l'ús de bastiments és la necessitat d'un procés inicial d'aprenentatge per treure el màxim profit.

1.2.2 Entorn de desenvolupament

Tampoc s'ha d'oblidar la importància de la metodologia de treball a seguir i la selecció de eines per a l'entorn de desenvolupament. A continuació es reflexa alguns aspectes a tenir en compte:

- **Control de versions**: Tots els elements de configuració del projecte (documentació, diagrames, codi...) han de ser gestionats en cada una de les seves versions, les quals s'han de poder recuperar en qualsevol moment. Una eina molt útil per aquesta tasca és el *Subversion*, que permet inclús crear branques diferents.
- **Editor de codi**: És necessari un entorn de programació adient per a realitzar un desenvolupament còmode que faciliti la integració d'eines que ens ajudi a generar codi òptim i de qualitat, aconseguint d'aquesta manera minimitzar les errades i produir un bon sistema d'informació. Una eina estesa com editor amb aquestes característiques és *Eclipse*.
- **Tasques programades**: Hi ha moltes tasques automàtiques i repetitives que sovint es fan manualment, com pot ser la fase compilació i construcció del projecte a nivell de comandes o traslladar el codi compilat al servidor. Una eina que ens pot ajudar a programar aquestes tasques de forma automàtica és *Apache Ant,* amb la qual les poden programar a través de configuracions d'arxius XML, i a més és independent del sistema de comandes del sistema operatiu, és multiplataforma.
- **Generador de codi**: A vegades perdem molt de temps realitzant codi de forma mecànica, però hi ha eines que ens podem ajudar a generar aquest codi automàticament. Com exemple tenim *xDoclet* pel llenguatge Java, que està associat a la programació orientada als atributs, es a dir, es pot afegir funcionalitat *metadades* al codi mitjançant etiquetes *JavaDoc*.
- **Servidor d'aplicacions**: Ens necessita un servidor d'aplicacions robust que permeti incorporar les llibreries necessàries per funcionar amb els bastidors seleccionats i que estigui preparat per atendre múltiples peticions, a més de poder integrar-se bé amb l'entorn de treball. En aquest sentit, una bona opció és el *Jboss,* un servidor d'aplicacions J2EE (implementa tot el paquet de serveis J2EE), i que a més és de codi obert, implementat en Java pur, amb l'avantatge que pot ser utilitzat en qualsevol sistema operatiu que el suporti.

1.3 Planificació

1.3.1 Definició de mòduls

Per afrontar la resolució del projecte, es convenient distingir els diferents àmbits en la gestió d'un centre. Amb aquest objectiu podem dividir el projecte en les següents mòduls:

- **Gestió d'alumnat**: En aquest mòdul es desenvoluparà les dades pròpies associades a l'alumne: dades personals, dades dels tutors (pares o tutor legitimis), historial acadèmic (estudis ja cursats i matèries superades en el mateix centre o altres institucions), càrrecs de l'alumne (per exemple si pertany al consell escolar) i dades extraescolars (usuari de menjador, ajuda bus.... , assegurança escolar...). Tot aquest conjunt de dades, formen la fitxa de l'alumne, independentment de la seva matriculació en un ensenyament de l'alumne.
- **Gestió del professorat**: Els mestres i professors estan destinats en un centre durant un període determinat, i poden anar canviant de centre des de un curs acadèmic a un altre. El centre ha de disposar de les seves dades i poder assignar-li un horari amb les classes del cursos que impartirà, així com els càrrecs com tutor, cap d'estudis, director... Aquesta assignació d'horari esta pensant per construir-lo manualment, sense pretendre convertir-se en un programa de generació d'horaris.
- **Gestió de matrícula**: La matrícula té certa complexitat perquè primer l'alumne ha de passar per una prescripció, i una vegada acceptada, s'ha de matricular en el centre d'un ensenyament en concret i en un grup amb plaça vacant. En aquesta matrícula s'ha de tenir en compte les matèries en estat de convalidació, renuncia o exempció (per exemple d'un idioma propi de la comunitat autònoma en cas d'alumne estranger o altra comunitat).
- **Gestió ensenyaments**: Un altre mòdul molt important, és la gestió de totes les etapes, cursos i matèries que es poden cursar en cada curs, ja que s'ha de definir tot un conjunt de regles per les qual un alumne pot accedir a uns estudis i a més de ser una informació bastant extensa i jerarquitzada. Es pot entendre com un arbre d'ensenyaments general, a partir del qual cada centre defineix el seu propi arbre amb els ensenyaments que oferta.
- **Gestió del centre**: hi ha tota una sèrie d'informació sobre el centre que només ha de ser gestionat per ell, com exemple, el seu codi, nom, tipus de centre, els departaments dels que disposa o inclús informació més rellevant com si disposa de menjador, l'horari d'aquest o quina es la ruta del transport escolar per arribar al centre. Quan parlem de centres, més concretament, ens referim a centre de primari, instituts o concertats que tenen una oferta educativa pròpia.
- **Gestió d'avaluació**: L'avaluació de l'alumnat també pot arribar a ser bastant complexa. Ha de ser àgil i tenir en compte les regles de puntuació de cada ensenyament, tipus de nota (numèrica o domini), període d'avaluacions durant el curs, mitjanes segons exempcions, etc. Finalment, s'ha de definir els conceptes adients pels quals l'alumne supera un curs o promociona a un altre i per acabar, quan és proposat per a títol. No només parlem de notes, sinó objectius avaluables que pot tenir una matèria segons el curs tractat.

- **Gestió d'usuaris**: Els perfils dels usuaris d'aquest aplicatiu poden ser molt variats i s'ha de poder assignar el permisos de tal manera que no puguin accedir a aquells mòduls que li siguin restringits. Aquesta protecció s'ha de fer tant a la part de persistència, com a la presentació, definint un menú adient per a cada perfil. Així doncs, per exemple, poden trobar-nos amb el perfil de tutor de curs que podrà accedir als cursos dels que és tutor, el de professor que només podrà accedir a l'avaluació del seus alumnes de les matèries impartides, el perfil de tutor d'alumne que podrà accedir a la fitxa de l'alumne i les seves notes, etc.
- **Explotació de dades**: Apart dels informes necessaris per a la gestió del centre (horari, certificat acadèmic, certificat de matriculació, llista de tutors...), hi ha tota una mena d'informes estadístics d'avaluació que es podrien construir tant a nivell d'un centre com a nivell global sobre tots els centres, com podria ser la mitjana d'aprovats d'un curs o d'una matèria.

Aquest mòduls intenten agrupar funcionalitats relacionades dins del propòsit gestió d'un centre. Aquestes funcionalitats s'hauran de desenvolupar en l'anàlisi de requeriments i etapes posteriors.

1.3.2 Fites principals

DATA FI	TASCA
28/09/2007	Pla de treball
04/10/2007	Investigació frameworks actuals per a treballar en JEE
07/11/2007	Anàlisi i disseny
17/12/2007	Implementació del sistema d'informació
14/01/2008	Lliurament memòria, aplicatiu i presentació PFC

1.3.3 Riscos a minimitzar

Hi ha alguns riscos generals que es poden anar produint i fer variar la planificació inicial. Per això s'ha de controlar i intentar que el seu impacte sigui mínim.

Aquest riscos són:

- **Tecnologia nova**: el desconeixem d'algunes tecnologies que es volen emprar fa que la línia d'aprenentatge d'aquestes no pugui ser avaluada a priori. Per això és necessari l'estudi previ d'aquestes tecnologies i una delimitació de les tasques a desenvolupar del projecte proposat.
- **Poca experiència**: tant en l'aplicació de metodologies de gestió de projectes com en el desenvolupament de sistemes d'aquesta mena, influeix directament en la planificació del projecte. Per això s'intentarà contar amb la col·laboració i supervisió del consultor, i l'ajuda del fòrum.
- **Temps reduït**: el temps assignat per a la realització de totes les tasques és molt ajustat, sobre tot si tenim en compte els riscos esmentats anteriorment, per la qual cosa és necessari ajustar-se a cada una de les fites per a no produir retards acumulatius.

1.3.4 Temporalització

Feta aquesta primera aproximació a la visió global del projecte, es poden distingir les següents tasques a realitzar en els períodes determinats per a cadascuna:

	Nombre de tarea	Duración	Comienzo	Fin
1	**- PDF**	**93 días?**	**mié 19/09/07**	**lun 14/01/08**
2	**- Avantprojecte i Pla de treball**	**9 días?**	**mié 19/09/07**	**vie 28/09/07**
3	Definició del projecte i objectius	4 días?	mié 19/09/07	dom 23/09/07
4	Anàlisi de riscos	2 días?	lun 24/09/07	mar 25/09/07
5	Planificació i temporalització	2 días?	mié 26/09/07	jue 27/09/07
6	Pla de treball	1 día?	vie 28/09/07	vie 28/09/07
7	**- Investigació frameworks actuals**	**5 días?**	**sáb 29/09/07**	**jue 04/10/07**
8	Estudi i exposició dels bastidors actuals	3 días?	sáb 29/09/07	mar 02/10/07
9	Comparativa de bastidors	2 días?	mié 03/10/07	jue 04/10/07
10	Selecció de bastidors	1 día?	jue 04/10/07	jue 04/10/07
11	**- Anàlisi i disseny**	**27 días?**	**jue 04/10/07**	**mié 07/11/07**
12	Glossari	2 días?	jue 04/10/07	vie 05/10/07
13	Definició casos d'us	2 días?	sáb 06/10/07	dom 07/10/07
14	Anàlisi de requeriments (funcionals i no funcior	6 días?	lun 08/10/07	lun 15/10/07
15	Disseny conceptual	5 días?	mar 16/10/07	lun 22/10/07
16	Disseny lògic	3 días?	mar 23/10/07	jue 25/10/07
17	Diagrames de fluxes	8 días?	vie 26/10/07	mar 06/11/07
18	Lliurament Anàlisi i disseny	1 día?	mié 07/11/07	mié 07/11/07
19	**- Implementació del sistema de informació**	**31 días?**	**jue 08/11/07**	**lun 17/12/07**
20	Configuració de base de dades	3 días?	jue 08/11/07	sáb 10/11/07
21	Programació	23 días?	lun 12/11/07	mié 12/12/07
22	Proves unitàries i d'integració	3 días?	jue 13/12/07	sáb 15/12/07
23	Revisió documentació i implementació	2 días?	dom 16/12/07	lun 17/12/07
24	Lliurament implementació	1 día?	lun 17/12/07	lun 17/12/07
25	**- Preparació lliurament projecte**	**22 días?**	**mar 18/12/07**	**lun 14/01/08**
26	Elaboració Memòria	14 días?	mar 18/12/07	vie 04/01/08
27	Preparació presentació PFC	6 días?	sáb 05/01/08	vie 11/01/08
28	Revisió documentació i elements de conf. a lliur	2 días?	vie 11/01/08	dom 13/01/08
29	Lliurament memòria, aplicatiu i presentació PFC	1 día?	lun 14/01/08	lun 14/01/08

Imatge 1 – Planificació del projecte

2 Productes a obtinguts

Els obtinguts i lliurats a l'entrega final són:

- Prototipus de l'aplicatiu (gescen/prototipus): prototipus de l'aplicatiu en HTML i AJAX on es pot veure la navegació i les peticions asíncrones a arxius XML estàtics.
- Implementació de la solució GESCEN: on es veu la utilització dels marcs de treball Spring i Hibernate, i l'aprofitament de la tecnologia AJAX.
- Documentació memòria final: amb tots els aspectes rellevants del projecte que inclou tant de l'anàlisi i disseny com de la implementació.
- Manual d'instal·lació: amb els passos a seguir per a la instal·lació el programari

3 Anàlisi de requeriments

3.1 Context

En el context actual, els centres gestionen les dades de forma local, sense tenir pràcticament cap connexió entre ells, simplement un document en paper anomenat certificat acadèmic amb la relació dels cursos i notes de les matèries realitzats per l'alumne. A més, estan lligats a un entorn privatiu que fa impossible la potabilitat a altres configuracions diferents de les proposades a cada centre i a més tampoc són portables les dades a altes sistemes.

Això fa que les dades es dupliquin, i moltes vegades que es produeixin incoherències, ja que el mateix alumne pot tenir dates de naixement diferents en dos centres distints o inclús diferent nom no normalitzat.

És veu clarament la necessitat dels centre de compartir només aquelles dades que siguin necessàries, com poden ser les dades de l'alumne, el seu historial acadèmic, o les dades del professorat, ja que existeix una certa mobilitat tant per part d'alumnes com dels professors.

Per aquests motius es vol dur a terme el desenvolupament d'una aplicació que gestioni les dades de tots el centres tenint en compte els diferents usuaris amb diferents perfils (professors, pares, alumnes, administradors, personal no docent...), la qual cosa implica la gestió d'accés a l'aplicació des de qualsevol ordinador a través de la xarxa Internet amb connexió segura. D'aquesta manera, un usuari, segons el perfil, veurà i tindrà accés a una sèrie de funcionalitats diferents.

3.2 Glossari

- **Curs acadèmic**: període en el qual es dona un ensenyament complert, amb una duració normalment de 9 mesos (setembre a juny de l'any següent).
- **Any acadèmic**: relacionat amb el curs acadèmic, representa el any inicial d'aquest curs, es a dir, de curs acadèmic 2006-2007, l'any acadèmic és el 2007.
- **Ensenyament**: ensenyament reglat que pot cursar un alumne: primària, infantil, ESO... que pot estar subdividit en altres ensenyaments (bacxillerat per exemple en batxillerat de ciències de humanitat... o ESO en primer, segon, tercer i quart de ESO,)
- **Arbre d'ensenyament**: cada ensenyament es pot dividir en altre ensenyaments així successivament formant d'aquesta manera un arbre de ensenyaments. Amb els de grau superior es pot obtenir un títol qual s'hagin superat tots els ensenyaments de grau inferior.
- **Centre**: institució dedicada a la educació i impartició d'ensenyaments reglats que estarà dirigida per un equipo directiu, part del professorat.
- **Tipus centre:** un centre és d'un tipus determinat segons els ensenyaments que imparteix i quin sigui el règim del centre. Així doncs ens podem trobar amb centres d'educació d'infantil i primària (CEIP), centres concertats (CC) i instituts d'educació (IES).
- **Etapa**: ensenyament superior del qual es pot treure un títol (Infantil, primària, ES0...)

- **Curs**: número del curs d'un ensenyament concret que té un ordre concret. Per exemple de l'ensenyament ESO tenim primer de ESO, segon d'ESO...
- **Matèria**: ensenyament inferior de l'arbre d'ensenyament (node fulla) que correspon a la identificació de les per un professor.
- **Alumne**: persona que es matricula d'un ensenyament i que rep classes de les matèries matriculades.
- **Tutor**: persona responsable de la tutela d'un alumne, amb una certa relació amb el alumne que no té perquè ser el pare o la mare, com aquells casos que els pares perden la tutela i la té una persona d'un centre d'acollida.
- **Professor**: persona associada a un centre encarregada de donar classe de certes matèries als alumnes matriculats d'aquestes
- **Destinació**: lloc de feina en un centre al que pot se destinat un professor, ja que el mateix professor por donar classe en dos centres distint en dos cursos diferents.
- **Càrrec professor**: funció especifica que realitza un professor dins del centre, tutor de grup, director, cap d'estudis, etc
- **Departament**: departaments del centre corresponents a la organització i coordinació de cada tipus de matèria.
- **Relació persona**: relació que s'estableix entre dues persones: pare, mare, tutor, germà...
- **Horari lectiu**: horari en el qual s'imparteix les matèries. Cada matèria té un horari i en el seu conjunt fan l'horari lectiu,
- **Nota**: puntuació que es posa al a una matèria a una avaluació de tipus numèrica o alfanumèrica.
- **Avaluació**: període en el qual es puntua una matèria: primera, segona, tercera, quarta avaluació ordinària i avaluació extraordinària.
- **Matrícula:** concepte que representa la realització, no la superació, d'un ensenyament (un curs d'una etapa) per un alumne en un centre i un any acadèmic en concret. Una matrícula pot estar en tres estats: anul·lada, activa o pendent.
- **Cos del professor**: un professor realment pertany a un cos determinat: cos de mestres, cos de professors de secundària, cos de formació professional....
- **Especialitat:** un professor és d'una o varies especialitats d'ensenyament (matemàtiques, biologia , física , llengua castellana...) amb les quals esta habilitat per a donar classe de les matèries corresponents
- **Falta d'assistència:** es produeix quan un alumne o professor no assisteix al horari assignat a una matèria. Pot haver varis motius de la baixa i a més pot estar justificada i no justificada.
- **Grup:** agrupació en un curs a la que pertanyen els alumnes i al qual se l'imparteix una sèrie de matèries per un o més professors. Es denominen
- **Aula:** lloc físic d'un centre on s'imparteix una matèria, que es troba a un pis i pot ser compartida per més d'un grup,

3.3 Anàlisi funcional

Els requeriments ho podem separar segons el mòdul que estiguem analitzant. Així dons podem realitzar el següent desglossament:

3.3.1 Gestió centre

En aquest mòdul es gestionaran totes aquelles dades pròpies del centre com són els serveis del centre, les seves dades com telèfon correu electrònic, etc.

- Manteniment de departaments dels centres: associem al centre els departaments que el siguin pertinents segons els ensenyaments
- Manteniment serveis del centre: els centre ofereixen alguns serveis als seus alumnes com menjador, escoleta, etc.
- Manteniment dades centre: dades del centre com telèfon, correu electrònic, etc...
- Creació de les aules d'un centre: un centre tindrà vàries aules repartides pels pisos de l'edifici .
- Establir les avaluacions del centre: cada centre és lliure d'establir quantes les avaluacions entre 3 i 5 a més de d'extraordinària i les seves dates de inici i final, es a dir, el període de l'avaluació.
- Establir l'arbre d'ensenyament centre: cada curs acadèmic el centre ofereix certs ensenyaments i matèries optatives diferents. Per tant aquest ensenyaments es poden ampliar o reduir per a cada any acadèmic.
- Creació de grups: per a cada curs s'han de crear els grups necessaris en un any acadèmic.

3.3.2 Gestió Alumnat

En aquest mòdul es gestionaran totes aquelles dades relacionades amb les dades pròpies de l'alumne independentment de la seva matrícula actual en un centre.

- Manteniment dades personals: dades de naixement, adreça, telèfon de contacte....
- Manteniment de tutors relacionats: s'ha de introduir al sistema els pares, els mares o tutors dels fills que tindran permisos per a veure la fitxes dels seus tutorats.
- Historial acadèmic: est. El concepte d'expedient acadèmic desapareix, per una bada perquè així ho diu la nova legislatura sobre educació (es canvia per historial acadèmic), i per altra banda perquè parlem d'alumnes compartits per tots els centres.
- Cerca d'alumnes propis del centre: el centre ha de poder accedir a tos els alumnes que han passat pel centre.
- Manteniments càrrecs de l'alumne dins del centre: l'alumne tindrà assignats alguns càrrecs dins l'organització del centre com membre de l'associació escolar.
- Manteniment de necessitats educatives especials: un alumne pot tenir una sèrie de necessitats especials que els professors han de tenir en compte a

l'hora d'impartir les classes, com dificultats per la llengua o discapacitats motores o visuals. Estan dividides en dos grups ANCE o ANEE.

3.3.3 Gestió professorat (personal del centre)

També s'ha de incloure el personal d'un centre en el sistema, més concretament el professorat (encara que també hi ha personal no docent però que no tractarem en aquest sistema).

- Manteniment dades personals: telèfons, correu, adreça, etc.
- Especificar el cos al que pertany (cos de mestres, cos de secundària o cos de formació professional).
- Especificar les seves especialitats de les quals pot impartir classes: especialitat de matemàtiques, de biologia, d'anglès... que a més estan associades a un cos de professorat en concret.
- Donar la destinació d'un professor en un centre: cada any acadèmic un professor pot estar destinat a un centre distint i per un període en concret, encara que en principi és per tot l'any.
- Incloure el professor en un departament del centre: cada professor pertany a un departament concret en el centre.
- Assignar al professor càrrecs com director, secretari, cap d'estudis...
- Assignar la tutoria d'un grup d'un curs al professor si és el cas
- Determinar les matèries a impartir per professor en l'horari concret.

3.3.4 Gestió matrícula

La gestió de la matrícula s'esdevé prou complicada per a englobar les funcionalitats relacionades en un nou mòdul, ja que representa l'entrada d'un alumne en un centre i en l'aplicació. Aquestes funcionalitats són:

- Donar d'alta l'alumne abans de la preinscripció en el cas de què encara no existeixi en el sistema.
- Realització de la prescripció de l'alumne en un centre concret d'un curs per a un any en concret.
- Formalitzar la matrícula d'un alumne en una etapa i un curs. L'alumne pot ser que tingui o no una preinscripció anterior, la qual passar a l'estat de acceptada quan es realitza la matrícula.
- Assignació de les matèries seleccionades a la matrícula d'un curs , ja que hi ha matèries comuns i obligatòries, i després també hi ha matèries pendents de cursos anteriors.
- Assignació d'un grup del curs matriculat a l'alumne una vegada realitzada la matrícula.

3.3.5 Gestió Horari i faltes

La configuració de l'horari pot ser sigui uns dels aspectes més complexes d'establir a principi de curs, ja que intervenen varies variables amb distintes disponibilitats. A qui tractarem els funcionaments mínims per a dur a terme la gestió dels horaris de cada matèria i les faltes, que possiblement una aplicació especifica (un generador d'horaris) ajudarà a cada centre per a concretar l'horari del professorat.

- Designar l'horari d'una matèria en una aula del centre i d'un grup en concret. S'ha d'indicar l'hora d'inici i la duració de la classe.
- Assignar un professor en aquesta hora per a impartir la matèria.
- Introduir falta de l'alumne d'una matèria en un dia en concret. S'ha d'indicar també el motiu de la falta (problemes familiars, problemes mèdics, etc..) i si està justifica o no.
- Introduir falta del professorat d'una matèria en un dia concret. Igualment s'ha d'indicar el motiu i si s'ha justificat.
- Es poden generar varies consultes com: consultar l'horari d'un professor, l'horari d'un grup, horari d'un matèria.., la qual cosa queda pendent com a primera fase

3.3.6 Gestió Avaluació

En aquest mòdul es reflecteixen les funcionalitat per les quals es puntuen als alumnes de les seves matèries i finalment és designa si promociona al següent curs o opten una titulació de l'etapa acabada.

- Avalua alumnes d'una matèria: el professor ha de poder
- Assignar una nota a una matèria cursada per un alumne a una avaluació. La qualificació tindrà la seva equivalència en nota numèrica i no numèrica.
- Indicar si un alumne promociona de curs a l'avaluació ordinària o extraordinària (setembre). És decisió del professorat que un alumne promocioni amb més d'una suspesa.
- Comprovar permís per avaluar: només és pot avaluar d'una matèria de la qual el professor és tutor o que imparteix aquella matèria als alumnes.
- Indicar si una matèria queda pendent o aprovada a les avaluacions finals ordinaria i.

3.3.7 Explotació de dades

Tot els sistema ens serveix per a gestionar les dades en cada un dels centres, però també el sistema ens dóna la possibilitat de realitzar una explotació mitjançant estadístiques diverses sobre les variables que més interessin, que finalment es podran utilitzar per a millorar el sistema educatiu sent analitzades per inspectors i l'equip directiu de cada centre. Els informes estadístics poden ser molt variats i complexos segons les necessitats, per això a continuació es nombren tres possibles informes, però sense dubte es podrien generar molt més:

- Generar informe mitjana d'aprovats per curs (aquells que han promocionat)
- Generar informe mitjana d'aprovats per matèries per curs
- Generar informe de faltes d'assistència dels alumnes d'un grup

3.4 Anàlisi no funcional

Pel tipus de sistema que volem definir s'han de especificar i assegurar bastants aspectes no funcionals, ja que es tracta d'un projecte molt ampli i amb un volum de dades considerable que a més necessita d'una política de seguretat estricta. Per tant, no només haurem d'identificar quins són els aspectes de qualitat que s'han de considerar, sinó també prioritzar-los. A continuació s'enumera aquests aspectes i es descriuen breument:

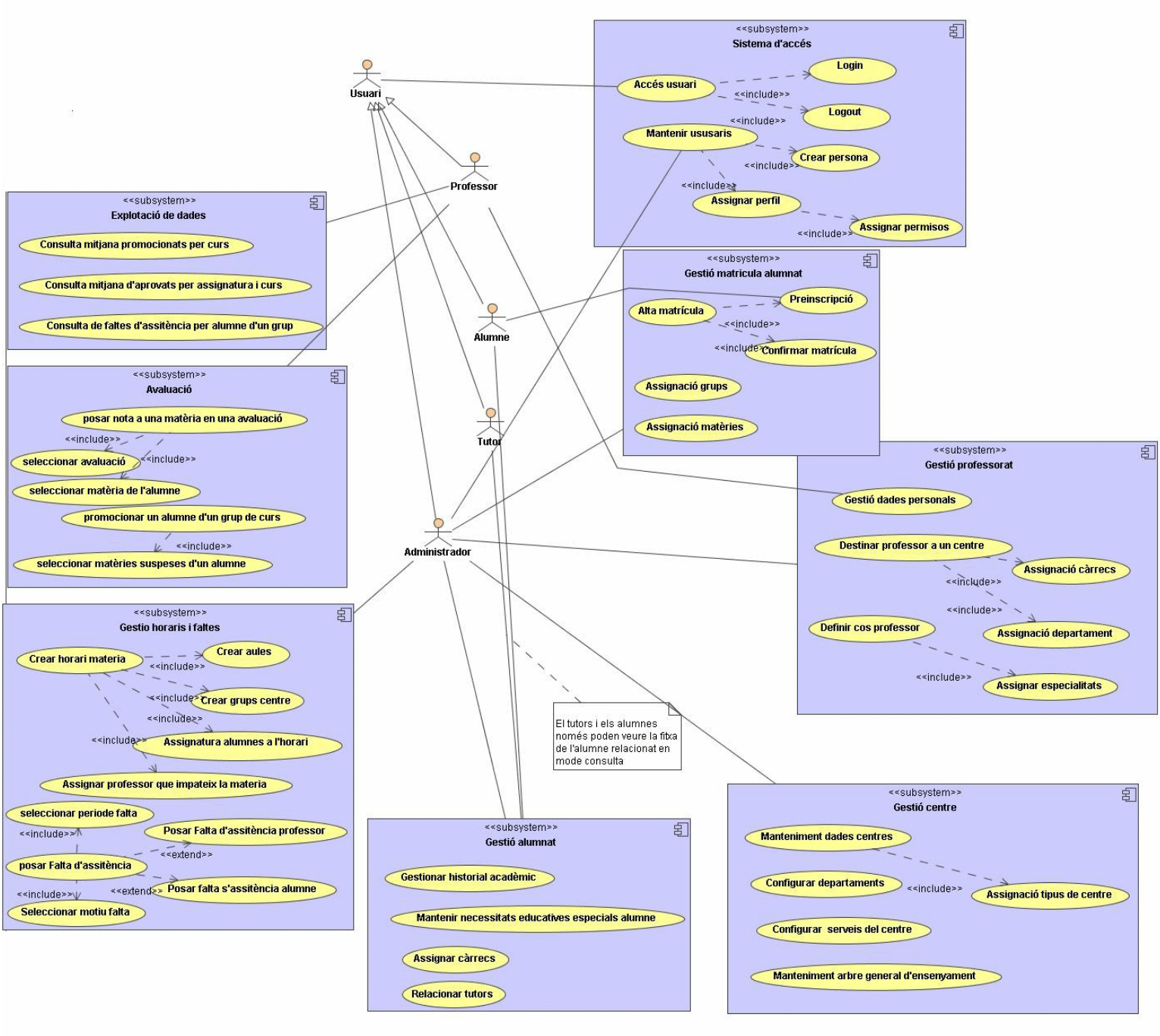
- Extensibilitat: s'ha de poder afegir noves característiques fàcilment al sistema ja que les noves normes d'educació van canviant contínuament i tenen un temps determinat per a entrar en vigor. Això farà que el disseny del sistema sigui complex i que es necessiti un major grau d'abstracció, de tal manera que s'hagi de preveure els punts dels sistema on hagi més variabilitat i pugin ser aplicables durant la vida del sistema. Per exemple, es pot voler també gestionar el personal no docent del centre (personal de neteja, conserge...) o afegir un mòdul per gestionar el transport que du els alumnes
- Modificabilitat: els canvi de requisits ha de ser fàcil de dur a terme al sistema ja que com s'ha dit abans, els constants canvis en la normativa farà que, a més dels nous requeriments, s'hagin d'adaptar components canviant algunes de les seves funcionalitats i comportaments. Aquesta propietat, encara que diferent de l'anterior, necessitarà de les mateixes tècniques de detecció dels punts susceptibles al canvi. En el nostre cas l'avaluació de les matèries, la selecció de matèries optatives i l'organització arbre d'ensenyament, són possibles aspectes dels quals saben que les seves funcionalitats canviaran cada any acadèmic.
- Simplicitat: el sistema ha de ser fàcil d'entendre i d'implementar l'estil, encara que aquesta característica es difícil es complicada de compaginar amb les esmentades anteriors, però amb d'utilització de marcs de treball que abstrauen parts complexes del sistema es pot arribar a un grau adient de simplicitat encara que es prioritzaran l'extensibilitat i la modificabilitat del sistema.
- Eficiència i robustesa: el sistema ha de donar una servei ràpid i no fer esperar a l'usuari cada vegada que vulgui fer una operació. Amb les característiques anteriors combinar també aquesta es torna complicat, però fent una interfície d'usuari rica amb components que interactuïn en el client amb l'usuari es pot aconseguir un grau entremig on la resposta sigui viable i no s'hagi de sacrificar l'arquitectura plantejada. També la part de persistència ha de poder gestionar el gran volum de dades amb eficiència i ha de tenir una bona gestió de recuperació de fallades i un sistema de copies de seguretat.
- Seguretat: s'ha de prioritzar aquesta característica ja que en el sistema interactuen diferents perfils i les dades que manipula els sistema són de tercer nivell, com les dades de l'alumne i les seves necessitats educatives. S'han de protegir tant les dades com les funcionalitats que poden executar cada un dels perfils del sistema. Per exemple, un pare només pot accedir a les dades del seu fill i un professor només pot posar notes de les matèries impartides.
- Usabilitat / Accessibilitat: els usuaris que utilitzaran els sistema seran molt diversos i poden tenir diferents graus de necessitats i de coneixements

sobre el maneig de noves tecnologies. Per això, els sistema ha de ser fàcil d'utilitzar i seguir unes certes normes d'accessibilitat (la norma doble A establerta per W3C), que permetin als usuaris adaptar la lletra, els colors, etc i evitar accions possiblement complicades en quant a la coordinació com arrossegar elements o obrir finestres i que es quedin ocultes sense haver finalitzat l'acció.

3.5 Casos d'ús

En relació a l'anàlisi fet anteriorment obtenim el següent diagrama d'alt nivell

Al diagrama es veu com es defineix els casos d'ús:

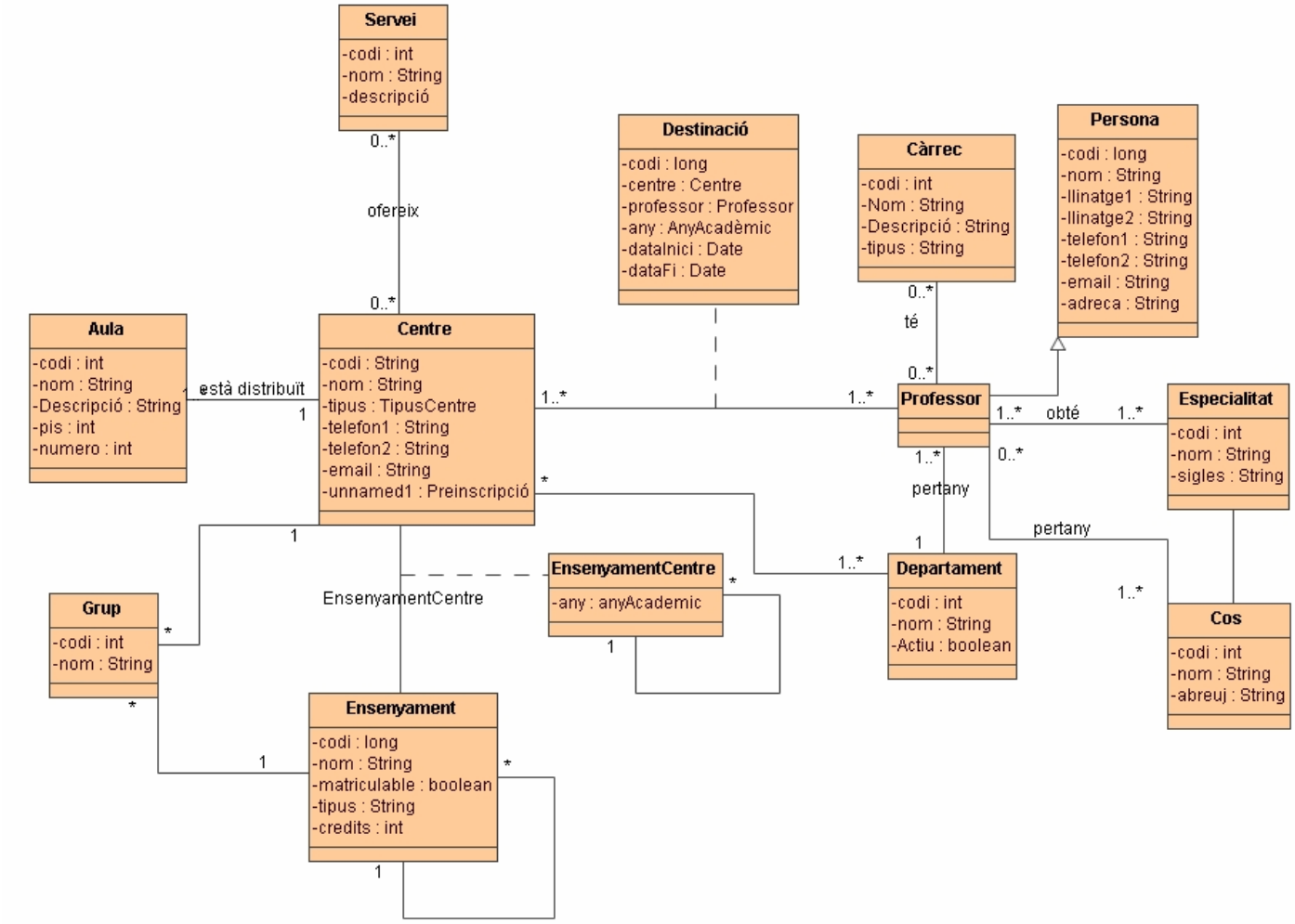


Imatge 2 – Casos d'us

3.6 Model conceptual – diagrama estàtic de classes

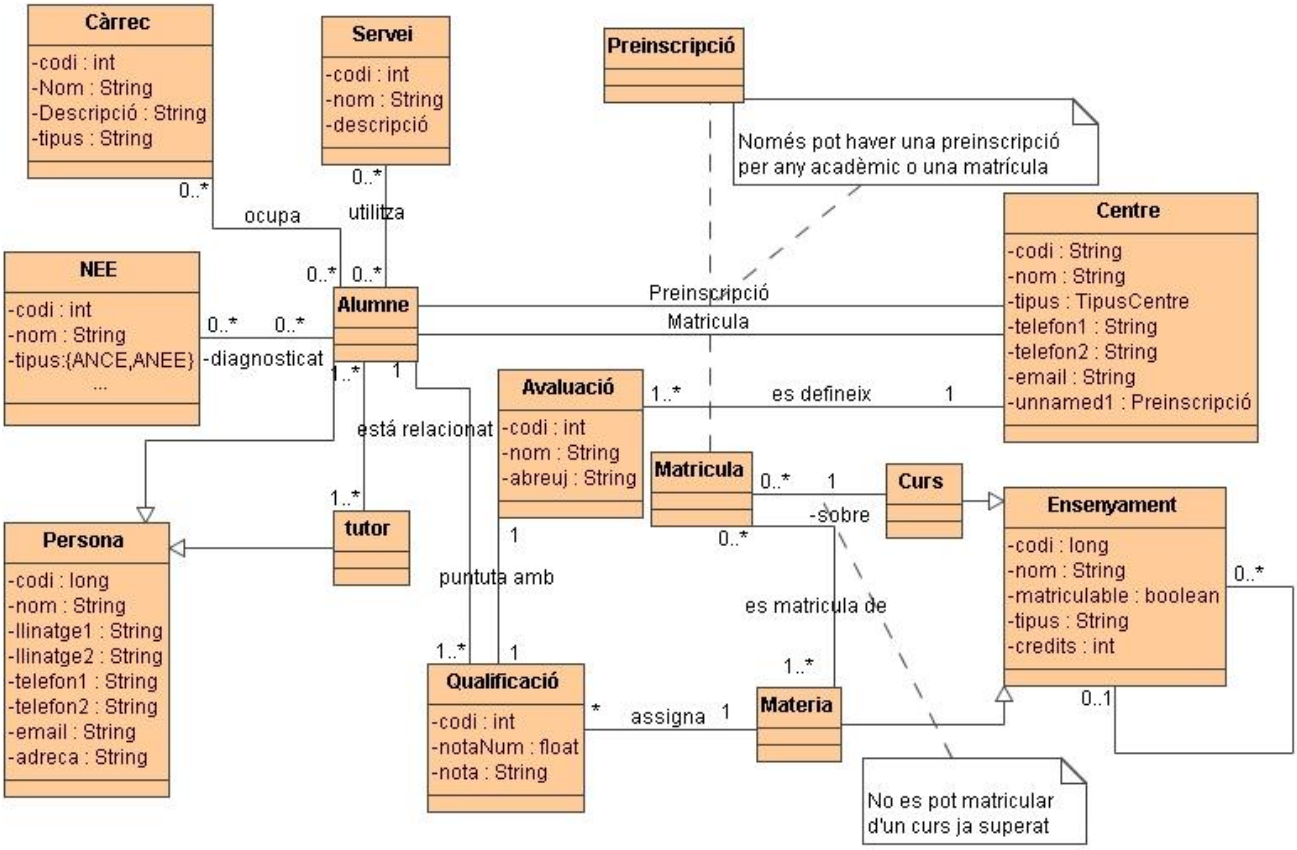
Per a entendre millor el model conceptual, ho he subdividit en tres parts.

Classes relacionades amb centre i professorat:



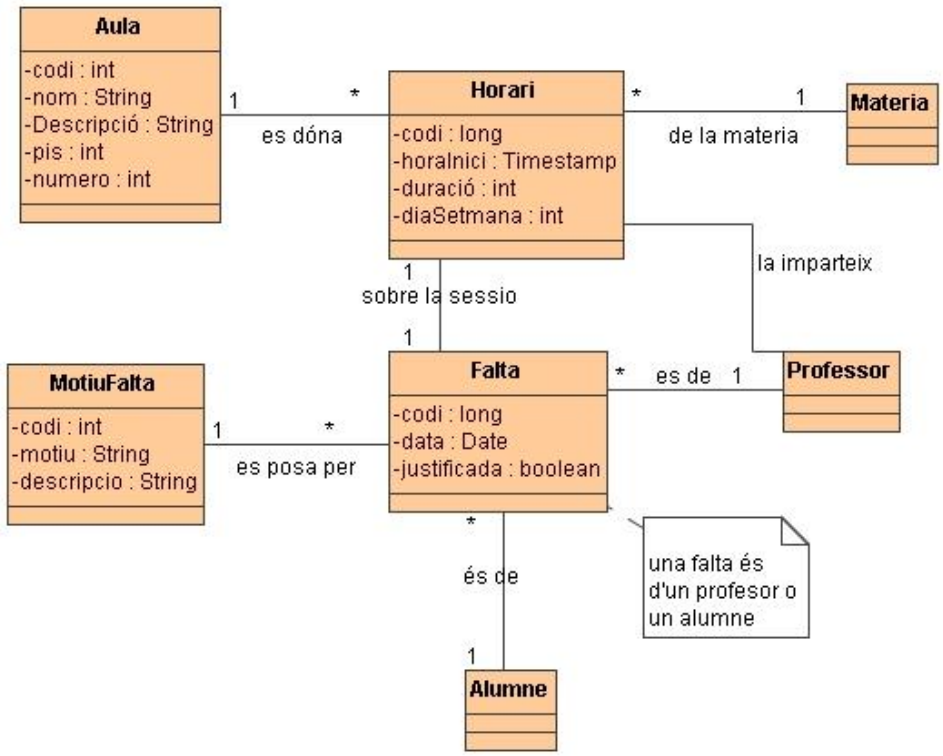
Imatge 3 – Diagrama de classes. Relacions entre centre i professorat

Classes relacionades amb alumne i avaluació



Imatge 4 – Diagrama de classes. Relacions entre alumne i avaluació

Classes relacionades amb horari i faltes



Imatge 5 – Diagrama de classes. Relacions entre horari i faltes

Vegem com s'han utilitzar certs patrons d'anàlisi com la associació històrica entre centre i professor, de la qual sorgeix la destinació, ja que un professor estarà destinat a un centre per a un any acadèmic concret i durant un període determinat.

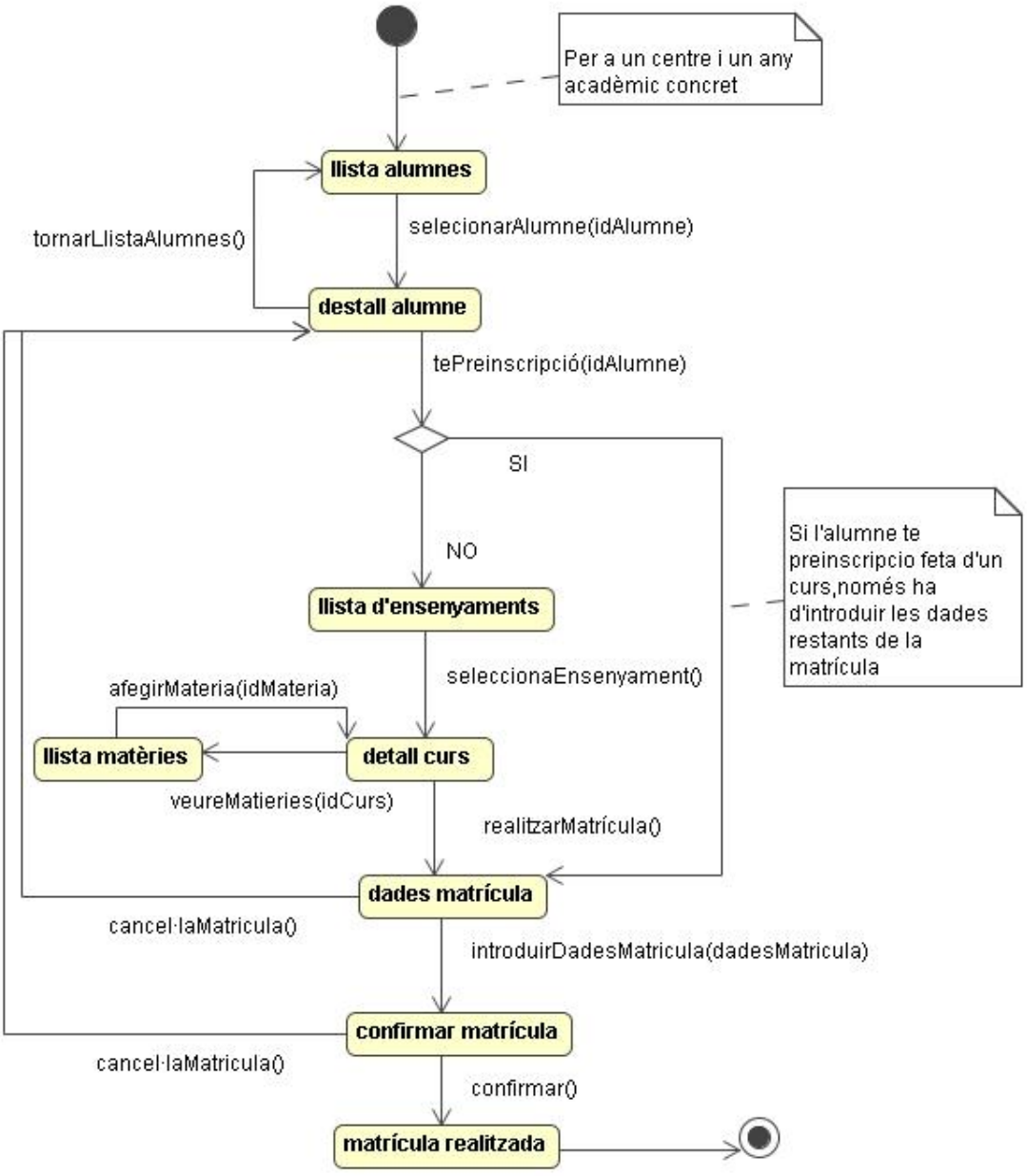
També ho apliquem a l'arbre d'ensenyament, ja que cada centre tindrà un arbre d'ensenyament diferent cada any acadèmic.

També apliquem el patró quantitat per a representar la nota d'una matèria a una avaluació. En lloc de posar-la dins de l'avaluació, la posant a la classe qualificació on representem la nota numèrica i no numèrica.

3.7 Diagrames d'estats

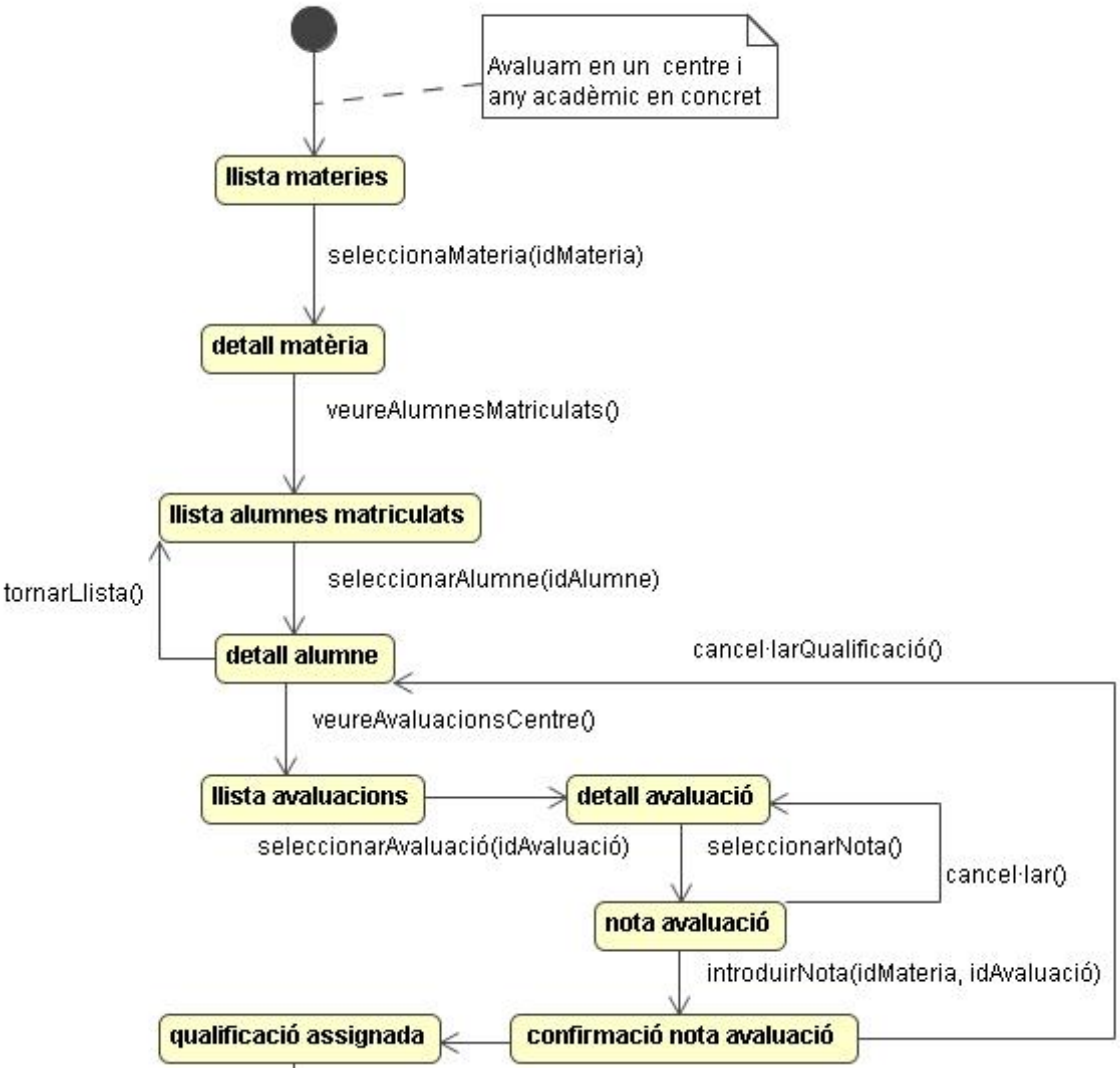
A continuació es presenten alguns diagrames d'estats d'algunes funcionalitats crítiques:

3.7.1 Matrícula d'un alumne:



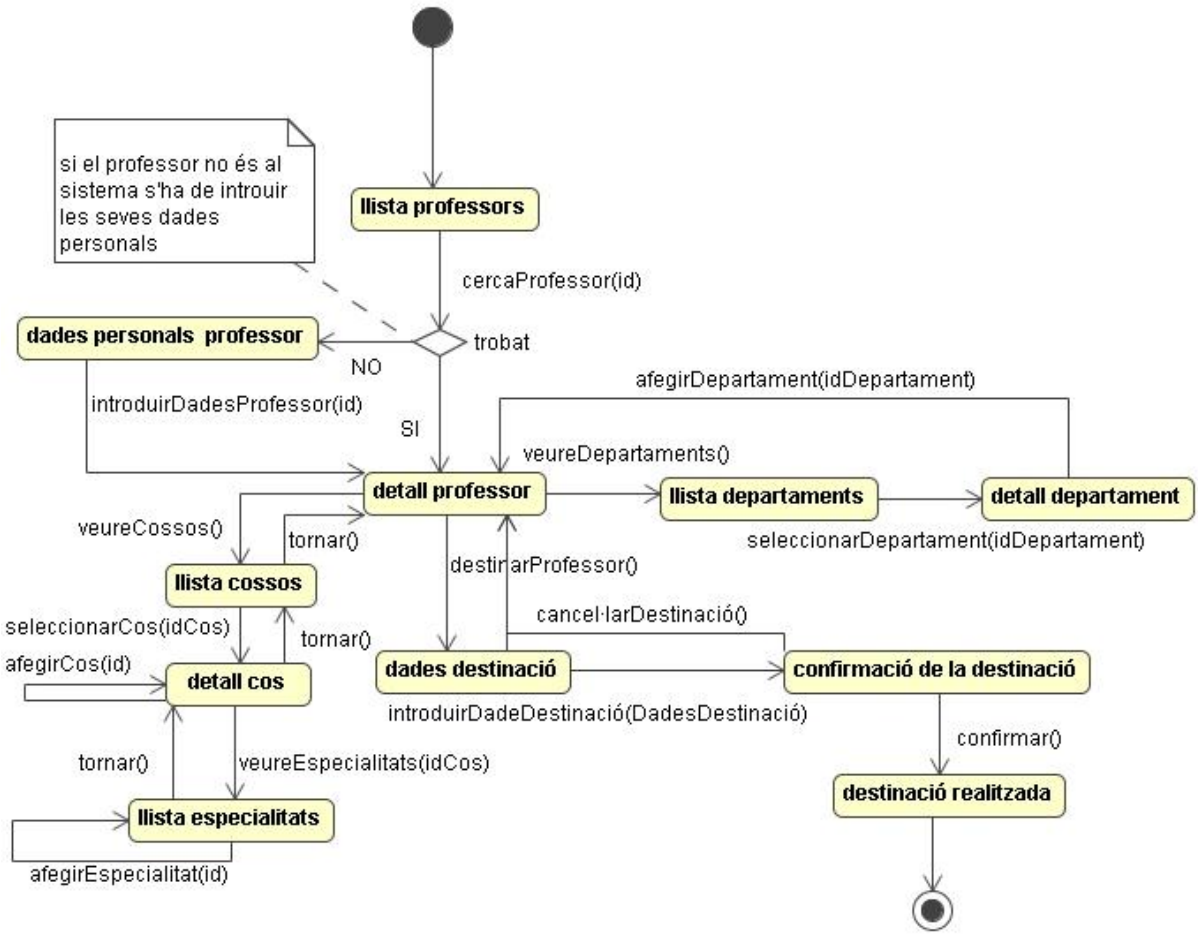
Imatge 6 – Diagrama d'estats. Matrícula d'un alumne

3.7.2 Avaluació d'una matèria



Imatge 7 – Diagrama d'estats. Avaluació d'una matèria

3.7.3 Destinació d'un professor a un centre



Imatge 8 – Diagrama d'estats. Destinació d'un professor en un centre

4 Disseny

4.1 Estudi previ de marcs de treball

A continuació es presenta l'estudi realitzat sobre els diferents marcs de treballs del mercat per a incorporar al projecte.

4.1.1 ZK FRAMEWORK

Podríem definir ZK com un marc de treball d'aplicacions web basat en un mecanisme d'events en AJAX, que amb poca programació permet desenvolupar interfícies d'usuari iteratives. A més, és de codi obert i proporciona molts de components en XUL i XHTML que faciliten el disseny d'interfícies d'usuari sense utilitzar directament JavaScript i amb poc esforços.

D'aquesta manera, pels programadors d'aplicacions web els codis AJAX són completament transparents, i es centren en programar la interfície amb el llenguatge de marcació XUL i ZK es centra en la sincronització en el servidor dels diferents components i dels clients i servidor, amb la qual cosa es disminueix la complexitat de desenvolupament, guanyant temps i eficiència.

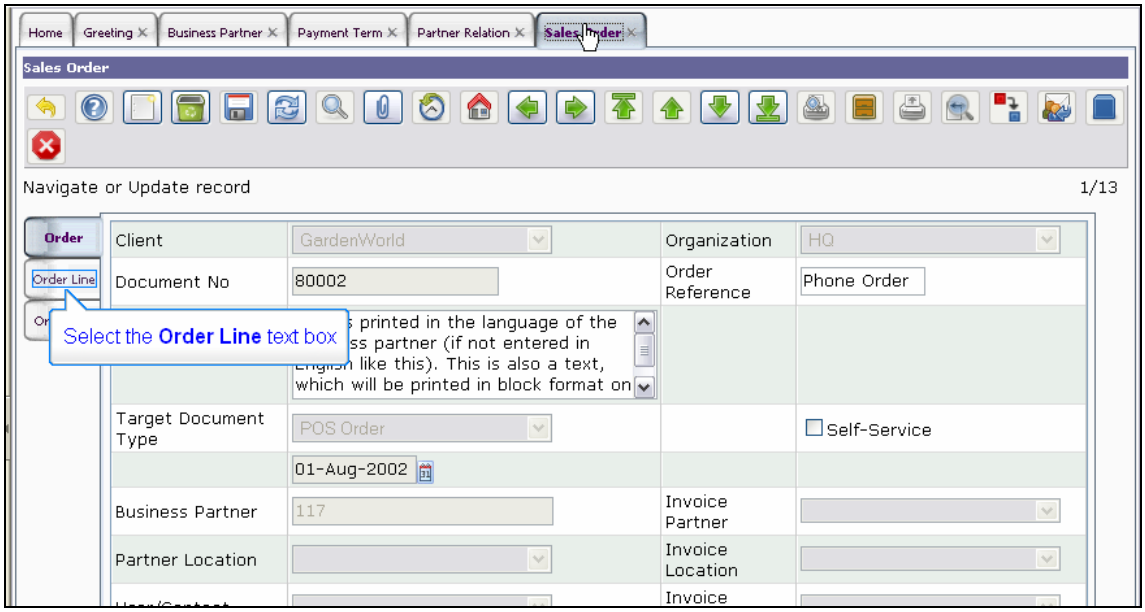
A més ZK proporciona un llenguatge de marcació anomenat ZUML que permet utilitzar diferents tipus de marcació com XUL de Mozilla i XHTML (encara que Zk no suporta tots els atributs de XUL e introdueix algunes extensions propietàries). A més es pot incloure scripts en llenguatge Java (per exemple per a fer comprovacions de dades) a la mateixa pàgina

Poden veure les següents avantatges amb aquest Marc de treball:

- Programació basada en components i model basant en events de forma intuïtiva.
- Amb ZUML els no experts poden dissenyar interfícies d'usuari prou eficients
- Abstrau el codi Ajax dels programadors
- És compatible per a diferents navegadors encara que no suportin XUL.

A continuació i després de realitzar algunes proves, es poden esmentar les següents desavantatges:

- Pot sobrecarregar molt el servidor i pot haver errors en la comunicació entre components difícils de controlar.
- Limitacions per a programadors experts que volen aplicar funcionalitats més específiques que obliguen a introduir codi espagueti.
- Renderitza els components amb etiquetes HTML que no compleixen bastants de recomanacions d'accessibilitat de W3C, per exemple, empra anidament de taules com a contenidors, amb la qual cosa el codi descarregat en el navegador és bastant caòtic.



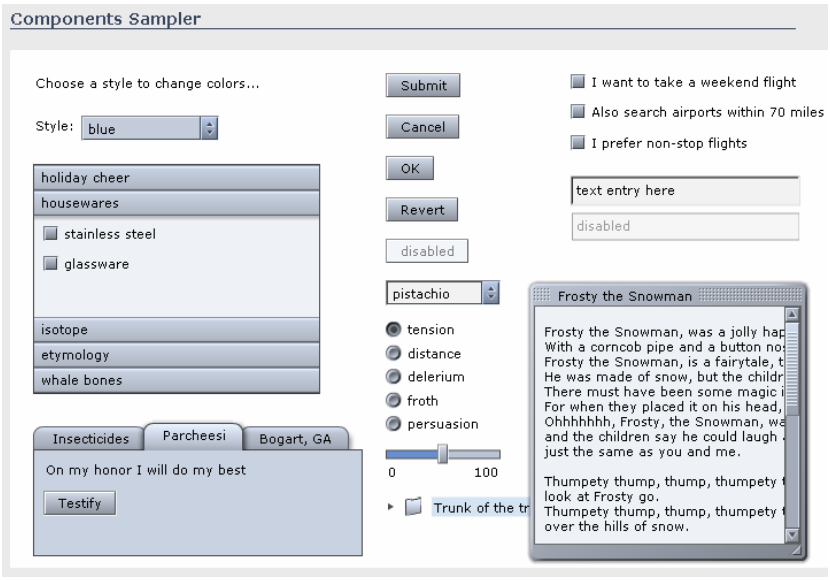
Imatge 9 – Mostra d'una pantalla amb components ZK 1

4.1.2 OpenLaszlo

OpenLaszlo és una plataforma de codi obert que permet crear unes sorprenents i riques interfícies d'usuari que consisteix en utilitzar un llenguatge de marcació LZX similar a ZUL sobre el servidor OpenLaszlo que s'encarrega mitjançant *servlets* Java de compilar les aplicacions LZX en binaris executable que es visualitzen en els navegadors que tenen instal·lats el plugin de flash. D'aquesta manera permet als desenvolupadors que tenen certa familiaritat amb HTML i JavaScript, crear ràpidament prototipus d'aplicacions webs.

Poden destacar dos aspectes important. Per una banda el servidor web ha de poder córrer amb el servidor OpenLaszlo i alternativament, les aplicacions Laszlo poden ser compilades per LZX en arxius binari SWF i carregar-los estàticament dins d'una pàgina ja existent. Aquest mètode és conegut com "*SOLO deployment*". Aquestes aplicacions tenen l'habilitat de consumir XML mitjançant cridades a procediments remots funcionant així amb serveis web entrant així en el marc d'una arquitectura SOAP.

Un pantalla d'exemple que mostra els components bàsic proporcionats per Laszlo:



Imatge 10 – Mostra d'una pantalla dels components disponibles ZK

4.1.3 Hibernate

Hibernate es defineix com un bastidor de codi obert que dóna una solució implementada pel mapeig d'objectes relacionals (ORM) per aplicacions Java, sobre una base de dades relacional, aconseguir així d'alliberar el programador d'un seguit de tasques pròpies de la persistència de dades relacionals i dotar les aplicacions de portabilitat entre *SGBDs* diferents, ja que realment al món comercial i no comercial segueixen dominant les bases de dades relacionals a causa del seu bon rendiment durant aquells anys.

Per exemple, amb aquesta abstracció oferta per Hibernate, ens permetria portar l'aplicació d'Oracle a MySQL sense canvia practicament codi de la capa de persistència, només amb mínima càrrega addicional, gràcies a la seva funcionalitat bàsica del mapeig de classes Java en taules de base de dades i tipus de dades Java sobre tipus de dades d'SQL. Hibernate també proveeix un llenguatge de consulta anomenat HQL i facilitats per la recuperació de dades. Hibernate genera les crides SQL i delega al desenvolupador la gestió manual del resultat de la consulta i la conversió als objectes pertinents. Això sí, per a ser possible aquesta portabilitat i que una aplicació basada en Hibernate pugui usar una base dades SQL, ha d'existir la peça de programari o *driver* encarregada de la traducció d'HQL al llenguatge SQL del SGBD sobre el que ha de córrer l'aplicació.

Hibernate pot ser usat tant en aplicacions *standalone* com en aplicacions JEE (mitjançant el component *Hibernate Annotations* que implementa l'estàndard JPA) i l'únic requeriment d'una classe persistent és oferir un constructor *public* i sense arguments. Els seu bon funcionament ha fet que la seva darrera versió 3.x hagi incorporat moltes característiques com els filtres definits per l'usuari i que a més es converteix en una peça central de a implementació d'EJB 3.0 sobre el servidor d'aplicacions JBoss.

4.1.4 Spring framework

Aquest marc de treball de codi obert per a la plataforma Java, no força cap model de programació però per al disseny ofereix una gran llibertat als desenvolupadors de Java i a més proveeix solucions fàcils i ben documentades. Mentre les funcionalitats

del nucli de l'entorn són usables en una aplicació Java, hi ha diferents extensions i millores per a construir aplicacions web damunt d'una plataforma Java EE, encara que això fa canviar la manera de programar respecte del model tradicional, pel que feia als EJB, ja que abstrau molts de conceptes com part de 'objectius de disseny per a integrar-se fàcilment amb estàndards J2EE existents i eines ja fabricants.

Algunes característiques importants són:

- Gestió de la configuració basada en JavaBeans, aplicant-hi principis d'Inversió de Control, més específicament usant la tècnica d'Injecció de Dependència. Això ajuda a reduir les dependències de components, en implementacions específiques, sobre altres components.
- Capa genèrica d'abstracció per la gestió de transaccions de la base de dades i Estratègies preincorporades per la especificació JTA i un sol *DataSource* de JDBC. Això elimina la dependència en un entorn JEE pel suport a les transaccions
- Integració amb entorns de persistència com Hibernate, JDO, iBatis i db4o.
- Entorn d'aplicació web MVC, construït al nucli de la funcionalitat de Spring. suportant moltes tecnologies per generar vistes, incloent-hi JSP, FreeMaker, Velocity, Tiles, iText i POI.
- Entorn extensiu de programació orientada a l'aspecte per proveir serveis, com ara gestió de les transaccions. Amb això es millora la modularitat dels sistemes.
- Factoria de *Beans* central, que és usada globalment.

Aquestes característiques elimina en part la necessitat de definir les seves funcionalitats en un document d'especificació i controlat per un comitè oficial, cosa que també ha estat criticada i es pot veure com una desavantatge.

Per altra banda, Spring Framework està constituït per una sèrie de mòduls que estan dividits en blocs de construcció típics d'aplicacions complexes:

- Contenidor d'Inversió de Control : configuració de components d'aplicació i gestió del cicle de vida d'objectes Java.
- Entorn de Programació Orientada a l'Aspecte (AOP).
- Entorn d'accés a les dades: treballant amb sistemes de gestió de bases de dades relacionals, sobre la plataforma Java usant JDBC i eines de mapeig d'objectes relacionals aportant solucions a reptes tècnics que són reusables en una multitud d'entorns basats en Java.
- Entorn de gestió de transaccions: harmonització de diferents APIs de gestió de transaccions i orquestració de transaccions configuratives per objectes Java.
- Entorn model-vista-controlador: basat en HTTP i Servlet proveïnt moltes eines per la millora i la personalització.
- Entorn d'accés remot: importació i exportació d'objectes java a la manera de l'RPC (informàtica) configurable, sobre xarxes informàtiques que suporten protocols basats en HTTP, com ara RMI, CORBA, i serveis web (SOAP.
- Entorn d'autenticació i personalització: orquestració configurativa d'autenticació i processos d'autorització que suporten molts estàndards de

la indústria, protocols, eines i pràctiques via el subprojecte Acegi security framework.

- Entorn de missatgeria: registre configurable d'objectes que reben missatges per la consumpció transparent de missates des de cues de missatges via [JMS]], millora d'enviament de missatges sobre APIs JMS estàndards.
- Entorn de testeig: suporta classes per la creació d'unitats de testeig i proves d'integració.

Destacar també que amb el contenidor d'inversió de control (punt central d'Spring Framework) é aporta recursos consistents de configuració i gestió d'objectes Java. A més l'entorn SOAP proveeix integració amb Spring per exportació amb RPC d'objectes de la part servidora. Tant el client com la configuració del servidor, per tots els protocols sobre RPC i productes suportats per l'entorn d'accés a Spring Remote (excepte pel suport a Apache Axis) és configurat pel contenidor Spring Core.

4.1.5 Apache Structs

Struts és un marc de treball que es desenvolupava com a part del projecte Jakarta de l'organització Apache que actualment es desenvolupa com a projecte independent i que en ajuda a definir el patró Modelo-Vista-Controlador (MVC) en aplicacions web sobre la plataforma *JEE*.

En aquest model Struts dóna solució implementant un sol controlador anomenat ActionServlet que avalua les peticiones dels usuaris mitjançant un arxiu configurable en format *xml* i que s'anomena *struts-config.xml*. D'aquesta manera es separa la presentació de la lògica de negoci, el model que normalment accedirà a altres sistemes com la base de dades. D'altra banda els components de control són els encarregats de coordinar les activitats de l'aplicació; recepció de dades de l'usuari, les verificacions de forma i les crides dels components del model, els quals envien al control els seus resultats o errors de manera que es pugui continuar amb altres accions de l'aplicació. Així doncs a la vista les pàgines JSP no han d'incloure maneig d'errors mestres els components de control decideixen la següent acció.

Poden distingir les diferents característiques:

- Configuració del control centralitzada.
- Interrelacions entre acciones i pàgina o altres acciones s'especifiquen per elements XML en lloc de codificar-les en cada pàgina.
- Components d'aplicació que són el mecanisme per a compartir informació bidireccional entre l'usuari de l'aplicació i les accions del model.
- Llibreries d'entitats per a facilitar la majoria de les operacions que generalment realitzen les pàgines JSP.
- Struts conté eines per a validació de camps de plantilles bé generant JavaScript i fent una validació local o bé validacions a nivell d'accions.

4.1.6 JavaServer Faces

JavaServer Faces (JSF) també és un marc de treball per aplicacions web basades en Java que simplifica el desenvolupament d' interfícies d'usuari en aplicacions JEE.,

utilitzant JSP com a tecnologia per fer el desplegament de les pàgines, encara que també pot emprar d'altres com per exemple XUL.

Podem veure les següents característiques:

- Un conjunt d'APIs per representar components d'una interfície d'usuari i administrar el seu estat, manejar esdeveniments i la validació d'entrada, definir un esquema de navegació de les pàgines i donar suport per a internacionalització i accessibilitat.
- Un conjunt per defecte de components per a la interfície d'usuari.
- Dues llibreries d'etiquetes personalitzades per a JavaServer Pages (JSP) que permeten representar una interfície JavaServer Faces dins d'una pàgina JSP.
- Un model d'esdeveniments en el costat del servidor.
- Administració d'estats.
- Managed Beans (JavaBeans creats amb injecció de dependència).

Finalment en la següent taula resum podem veure les principals característiques de cada framework:

Marc de treball	Llicencia	Model MVC	Migració BDD	Suport presentació/ Ajax	Característiques	Desavantatges
ZK FRAMEWORK	GPL			Sí	-Proporciona molts components IU generats en java. -Programació ràmpida de la interficie d'usuari gràcies al lleguatge de marcació ZUML i -Control d'errors de la IU en java -Abstracció del codi JavaScript	-Codi html generat pobre i poc òptim -Sobrecàrrega del servidor per les crides entre components -No compleix directrius per a l'accessibilitat -Pot derivar en codi espagueti
OpenLaszlo	GPL			Sí	- Interfície d'usuari molt àgil -Desenvolupament ràpid amb llenguatge de marcació LZX -Interactua amb serveis web	-Els navegadors han de tenir instal·lat el flash player -No té un bon suport per a tractaments d'error. -És necessari que el servidor web contingui el servidor Laszlo -No compleix directrius per a l'accessibilitat
Hibernate	GPL		Sí		-Eina ORM completa basat en el mapeig de POJO's (Plain Old Java Objects) -Possibilitat la portabilitat de BD en la capa de persistència, gràcies al seu lleguatge HQL que abstrau SQL -Permet treballar amb objectes tenint una base de dades relacional i mapetjant les taules mitjançant arxius xml. -No obliga a implementar interfaces complexes, utilitza mecanisme de	-No és adient per a aplicacions senzilles on hi ha poca lògica ja que no val la pena la sobrecàrrega de feina. -No és òptim quan s'ha de manejar gran de volums a dades a la base de dades ja que amb JDBC la sentència és més directa. -No s'aprofiten certs recursos no estàndards que ofereix un SGBD en concret. -Corba d'aprenentatge mitja alta

					reflexió de Java. -Utilitza el generador de codi d'alt rendiment CGLIB en temps d'execució.	per a usuaris no experts per a utilitzar tot el seu potencial
Spring framework	GPL	Sí		Sí	-Redueix les dependències entre components. -Capa genèrica d'abstracció per la gestió de transaccions de la base de dades -Integració amb entorns de persistència com Hibernate -Suport per a implementar un entorn d'aplicació web MVC -Entorn extensiu de programació orientada a l'aspecte per proveir serveis	-Part del codi es trasllada a fitxers en XML de configuració. Càrrega inicial de sevlet elevada. Abastareu molts aspectes de programació fora especificació oficial. -Canvia la forma de programació tradicional, la qual cosa incomoda a programadors ja experts en EJB.
Apache Structs	Apache	Sí		Sí	-Proporciona un marco de treball pel patró de disseny MVC. -Fluxe de la aplicació dirigit per un Controlador central. -Accions definides en XML -llibreries (struts-tablibs) de funcions comuns	-Altres marc de treball ofereixen millors solucions
JavaServer Faces	GPL	Sí		Sí	-Proporciona components IU en java en el servidor amb etiquetes JSP. -Components IU extensibles - Maneig d'events i estats dels components	-

4.2 Presa de decisions

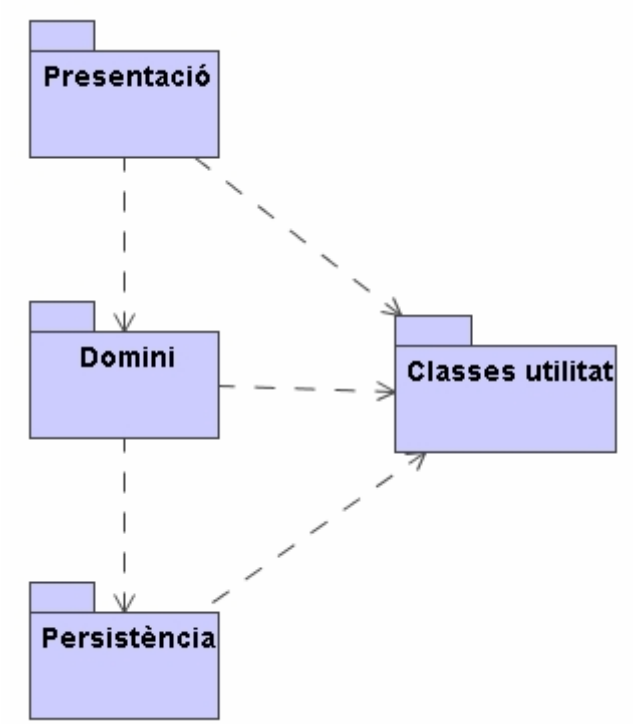
4.2.1 Descripció de l'arquitectura

L'arquitectura emprada per aquest projecte és un l'arquitectura de tres capes on podem distingir els següents nivells:

- Nivell presentació : nivell encarregat de generar la interfície d'usuari depenent de les accions dutes a terme per aquest.
- Nivell de negoci: conté la lògica que modela els processos de negoci i és on es du a terme tot el processament necessari per a atendre les peticions de l'usuari.
- Nivell persistència: és l'encarregat de fer persistent tota la informació, així com de subministrar i emmagatzemar la informació per al nivell de negoci.

Així la col·laboració i l'acoblament és des de les capes més altes cap a les més baixes.

Per tant ens basarem en el patró de capes:



Imatge 11 – patró arquitectònic en capes

D'aquesta manera es redueix la complexitat del sistema, ja que cada component treballa a un únic nivell d'abstracció definit per la capa que el conté i es podrien reutilitzar les capes senceres si es defineixen bé les interfícies entre elles. A més si per

exemple es decideix canviar una capa implementa amb Hibernate per EJB llavors no hauríem de modificar les altres capes.

4.2.2 Decisions de disseny de la capa de presentació

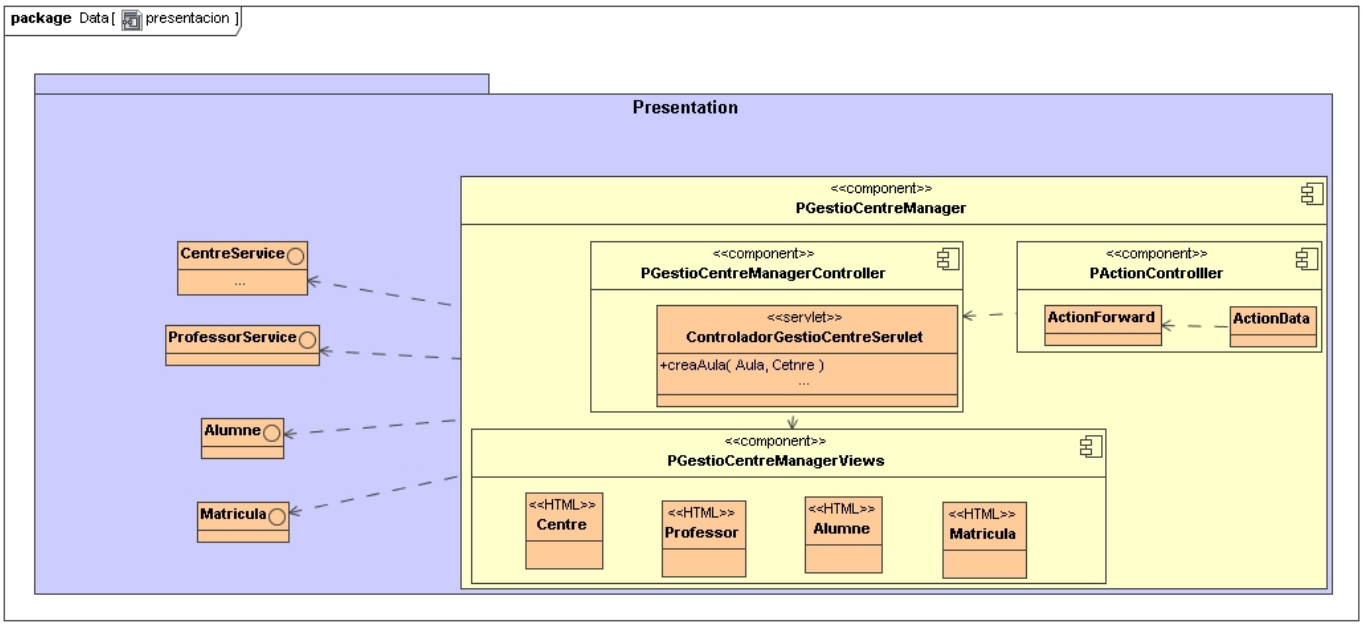
La meva intenció és fer un client més ric pel que fa a la iteració amb l'usuari i alliberar el servidor de peticions inútils, fent així un client pensant amb major riquesa a la interfície gràfica, amb controls d'usuari més complexos que els tradicionals.

També vull separar totalment la tecnologia de presentació de les altres, i per aquest motiu no utilitzaré els marcs de treball de presentació basats en Java com per exemple ZK o Openlazlo descrits anteriorment, per a tenir major control del codi presentats als clients, encara que això suposi més complexitat i més coneixements per part dels desenvolupadors.

Per a implementar aquesta capa utilitzaré AJAX (Asynchronous JavaScript and XML) que realemnt és una combinació de diverses tecnologies, basades en JavaScript, crides asíncrones i XML, que permet refrescar el contingut de parts d'una pàgina HTML sense haver de tornar-la a enviar tota. Utilitzaré llibreries desenvolupades completament en JavaScript que proporcionen la abstracció necessari per a funcionar amb varis navegadors i components addicionals, com per exemple Sarissa i JQuery.

Pel que fa a l'aspecte gràfic s'utilitzarà CSS aplicat sobre XHTML complint els estàndards proposats per l'organització W3C d'accessibilitat.

Podríem representar aquesta capa en el següent diagrama de components:



Imatge 12 – Capa de presentació

Com es pot veure a la imatge anterior, les vistes es realitzen en HTML ja que es vol evitar la dependència de qualsevol llenguatge de programació de servidor per a la interfície d'usuari. Tota la informació que es necessiti es demanarà amb peticions asíncrones al controlador que executarà la classe acció corresponent que encapsula la operació a realitzar i tornarà la resposta en format XML.

4.2.3 Decisions de disseny de la capa de negoci

L'accés a la capa de negoci es farà mitjançant els consum de XML que proporcionen serveis de la capa de negoci, que tindran un comportament asíncron. S'utilitzarà la tecnologia dels JSP i Servlets quins seran els encarregats de comunicar-se amb la capa de persistència i servir les dades a la capa de presentació.

Utilitzarem el patró d'arquitectura model, vista i controlador (MVC), on:

- La vista presenta les dades i recull les interaccions, enviant-les al contolador.
- El model encapsula l'estat del sistema i implementa la lògica de negoci, notificant dels canvis d'estat al controlador.
- El controlador establirà la correspondència entre les accions i els esdeveniments del sistema. Recordem que també utilitzem una arquitectura basada en capes i que per tant la capa de domini no pot estar acoblada a la capa de presentació, llavors no actualitzarà la vista directament sinó que serà el controlador l'encarregat de notificar els canvis a la vista.

Els serveis seran els beans que proporcionaran la informació necessària a la capa de presentación on

4.2.4 Decisions de disseny de la capa de persistència

En aquesta capa una bona opció és la utilització del marc de treball *hibernate* ja que s'esdevé una eina de mapeig objecte/relacional consistent i amb certes propietats que s'han descrit anteriorment i que elimina certes complexitats dels EJB. A més utilitzant JBoss com a servidor d'aplicacions, amb el qual Hibernate s'integra perfectament.

A més, utilitzarem Spring per a l'encapsulament d'aquesta capa ja que proporciona una bona integració amb entorn de persistència Hibernate i els seus mòduls ens serviran per a connectar les diferents capes i la gestió de transaccions.

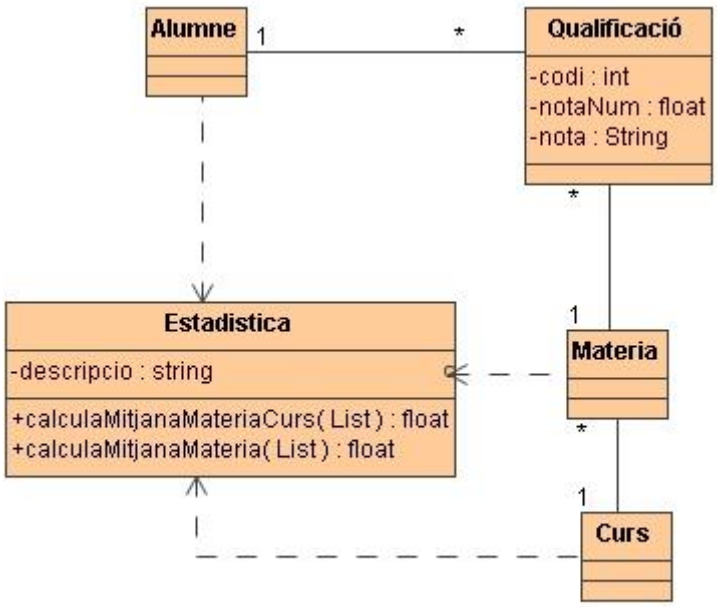
5 Patrons de disseny

5.1.1 Objecte compost

Podem veure que les diferents classes alumne, tutor, professor i en definitiva els usuaris del sistema, són una especialització de persona.

5.1.2 Fabricació pura

Per a la generació de estadístiques sobre les qualificacions del alumnes és convenient utilitzar aquest patró, pel qual crearem una nova classe a la qual assignarem, per exemple, les responsabilitats com la mitjana d'un matèria d'un curs.



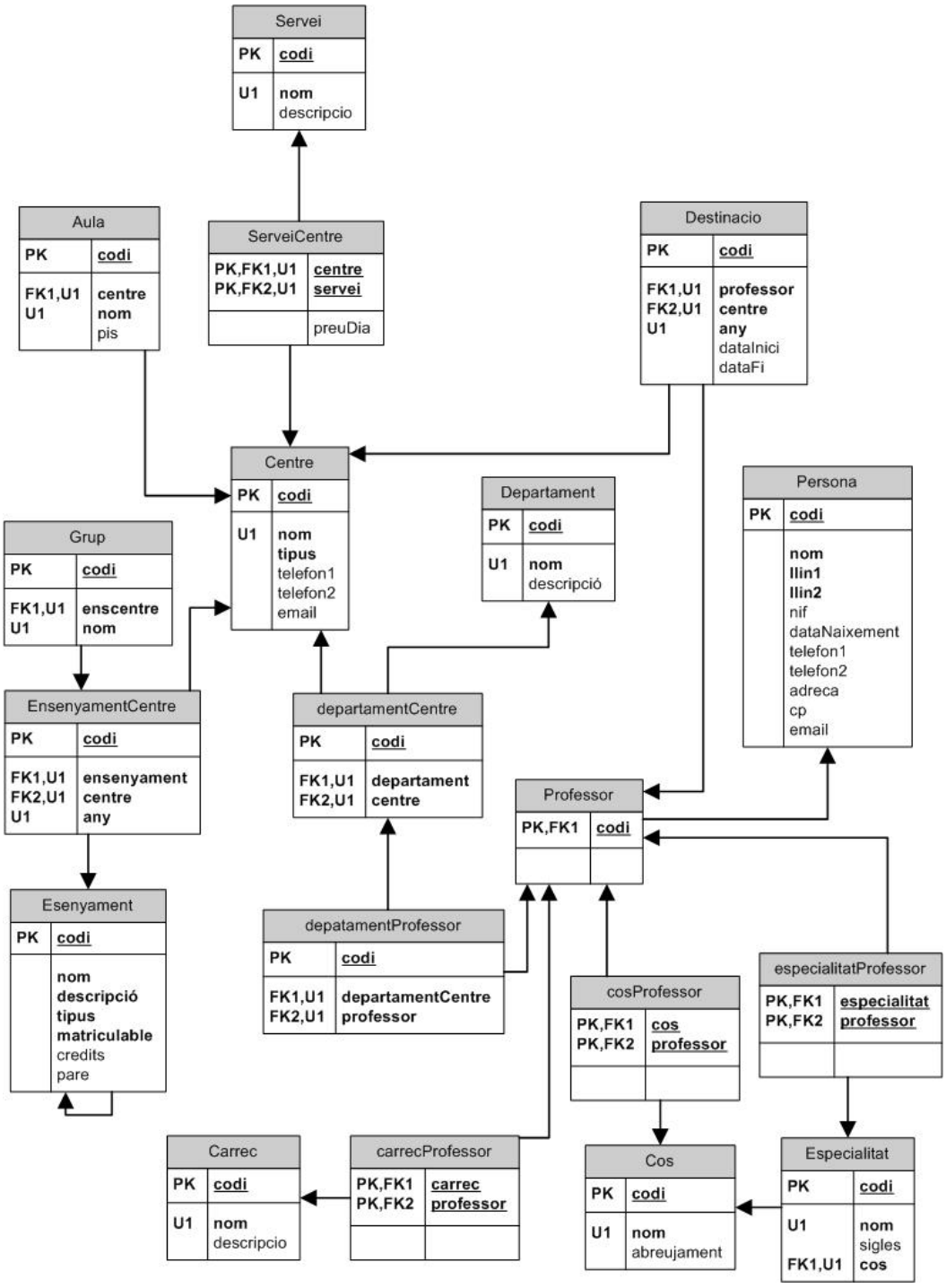
Imatge 13 – Aplicació del patró fabricació pura

5.1.3 Façana

Pel nostre cas que utilitzem l'arquitectura en tres capes serà de gran ajuda dotar d'una façana cada nivell de subsistemes i utilitzar-la com a punt d'accés a aquest. Aquesta característica es la que ens proporciona Hibernate a la capa de persistència i Spring ens ajudarà en les altres capes.

Aquí poden veure el model lògic relacional resultant de les fases anteriors:

5.1.3.1 Respecte centre i professor:

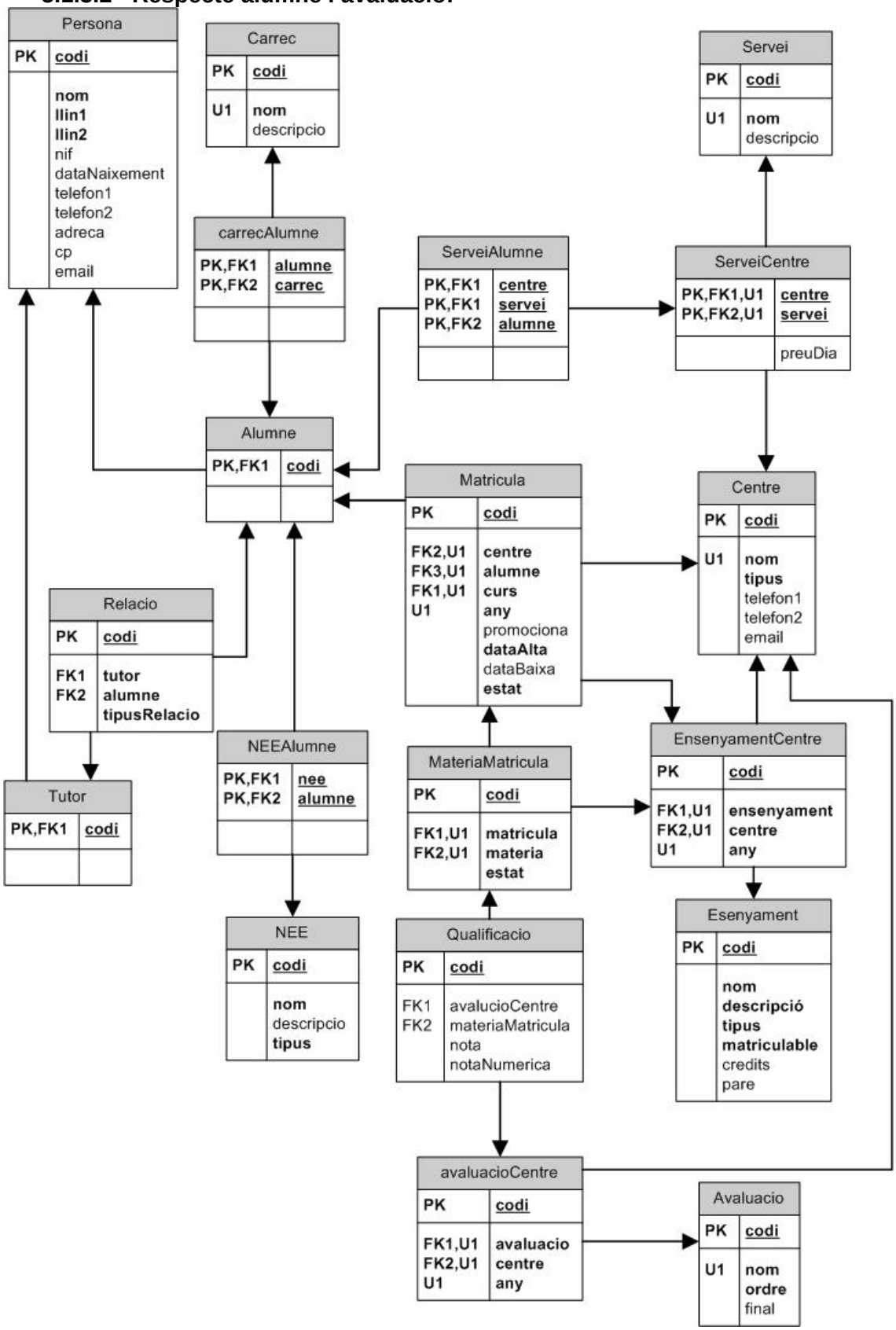


Imatge 14 – Diagrama de BD. Relacions centre i professor

Algunes restriccions a tenir en compte:

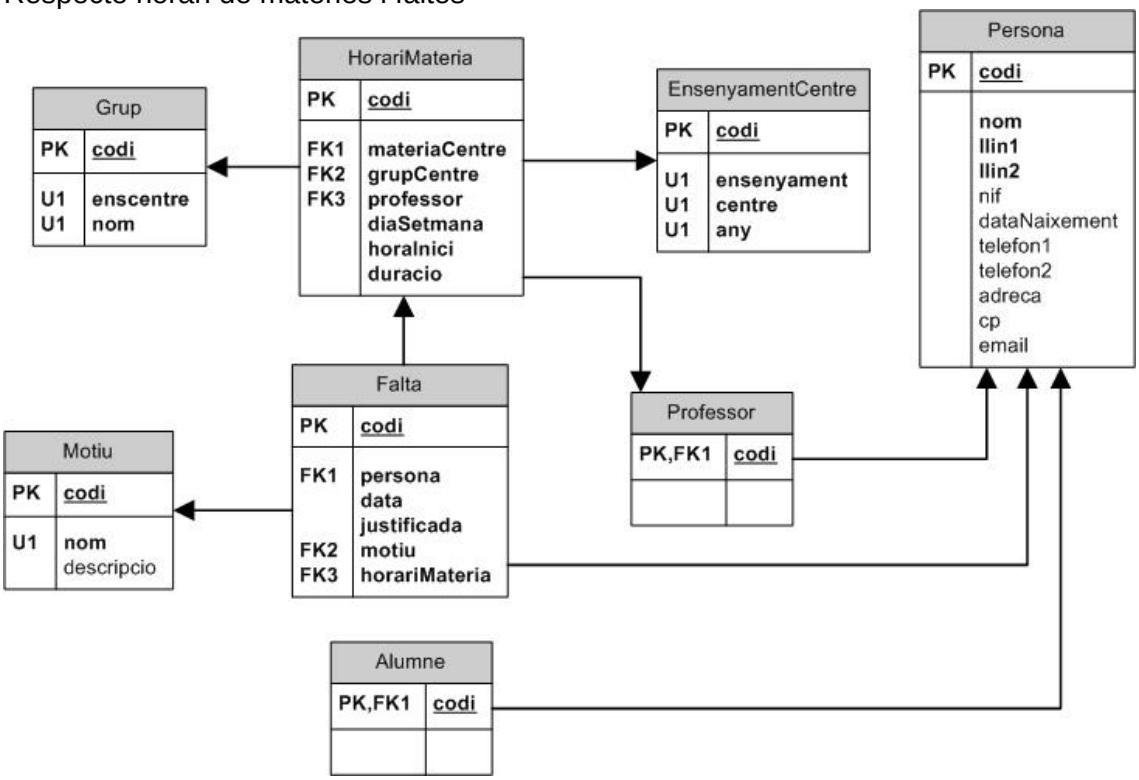
- Els grups només podran estar associats a ensenyaments que siguin de tipus curs.
- Els professor poden tenir varis cossos i especialitats, però no poden tenir especialitats de cossos que no tinguin associats.

5.1.3.2 Respecte alumne i avaluació:



Imatge 15 – Diagrama de BD. Relacions alumne i avaluació

Respecte horari de matèries i faltes



Imatge 16 – Diagrama de BD. Relacions matèries i faltes

S'han de tenir en compte que les faltes es posen per sessió horària del dia. S'ha de tenir en compte restriccions com que l'horari es fa sobre nodes matriculables de matèries. A més una falta de professor o alumne s'abstrau a una falta de persona.

He optat en posat sempre claus primàries sintètiques ja que és més òptim i eficient amb les tecnologies a implementar, menys en s associacions binaries simples.

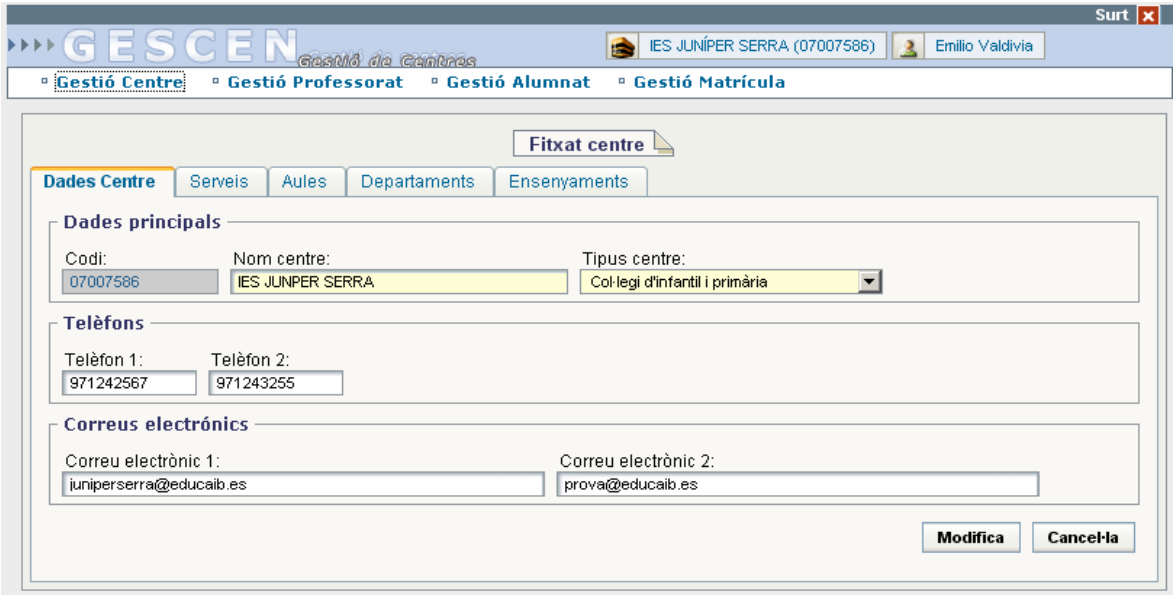
5.2 Prototipus

A continuació es presenta el prototipus confeccionat per un conjunt de pantalles que corresponent als diferents mòduls a implementar. Aquí es mostra com interactua amb el sistema un usuari amb el perfil de director o secretari del centre i pot manejar gairebé tota la informació que pertany al seu centre.

Podrem trobar aquest prototipus no funcional dins del directori "gescen/prototipus" començant per la pàgina "index.html". El prototipus que s'ha desenvolupat es prendrà com a punt de partida per a realitzar la implementació del projecte.

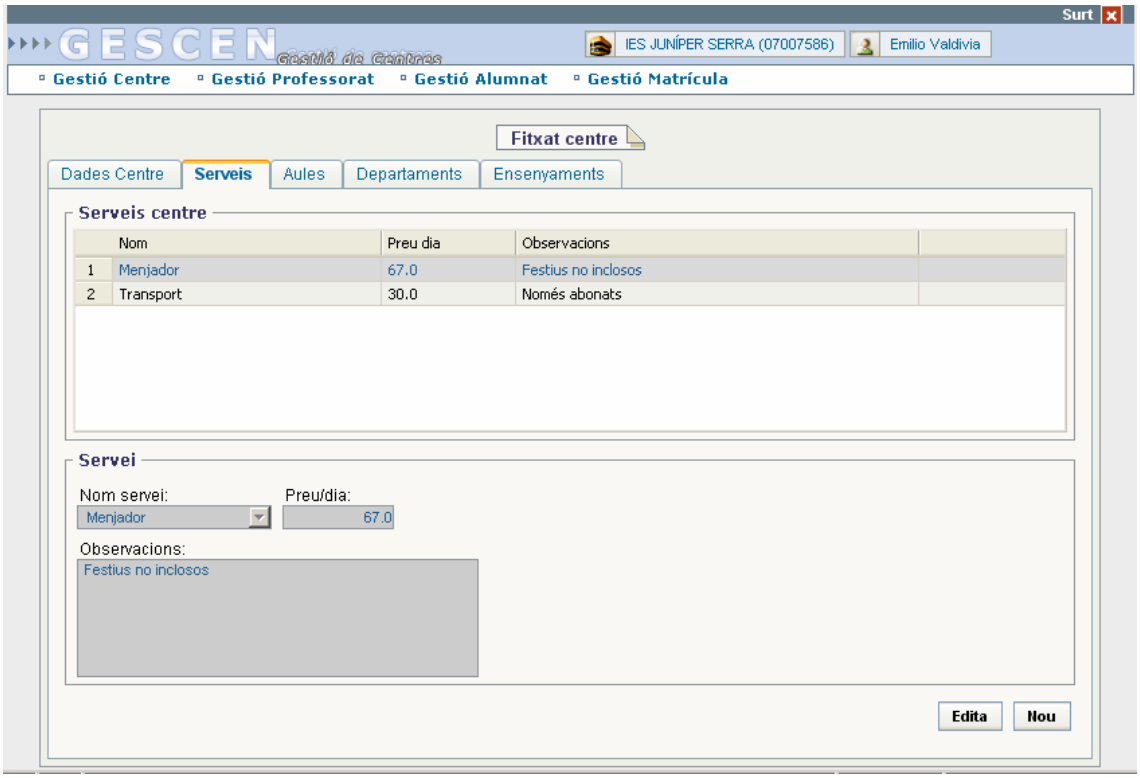
A la següent imatge podem veure la primera pàgina que es mostra a l'usuari una vegada iniciada la sessió on es veu el centre i l'usuari que s'han connectat i el menú per accedir a cada mòdul de l'aplicació.

El mòdul inicial són les dades referents al centre que l'usuari podrà modificar:



Imatge 17 – Pantalla prototipus dades centre

Aquí podem veure com es faria el manteniment dels serveis que ofereix el centre:

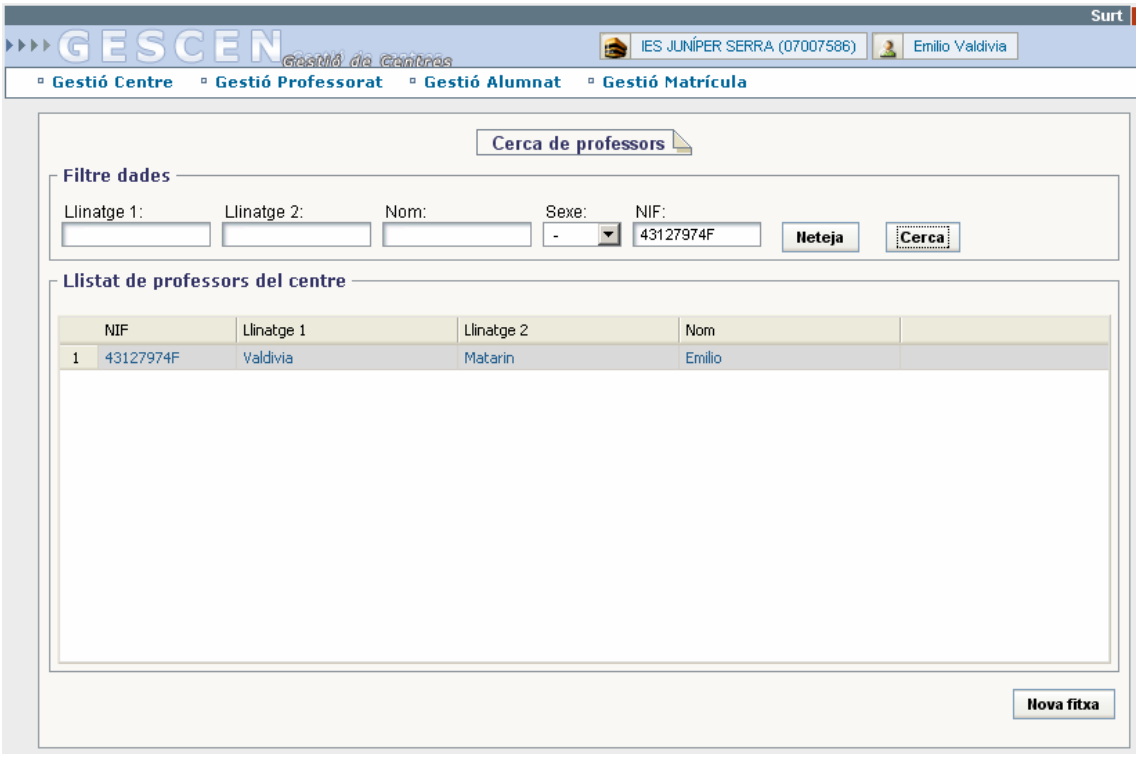


Imatge 18 – Pantalla prototipus serveis centre

A continuació es mostren el ensenyaments, cursos i matèries que el centre ofereix als alumnes a l'hora de matricular-se:

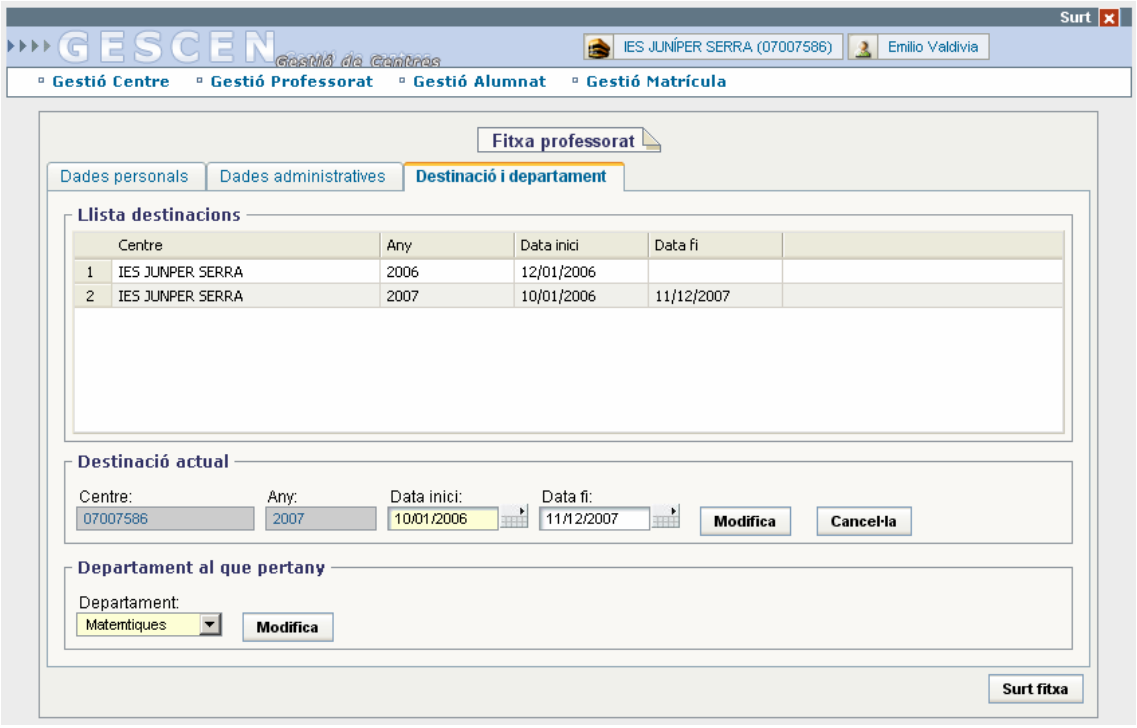
Imatge 19 – Pantalla prototipus ensenyaments centre

En aquesta imatge es veu com es pot cercar un professor en el sistema per a consultar les seves dades o modificar-les:



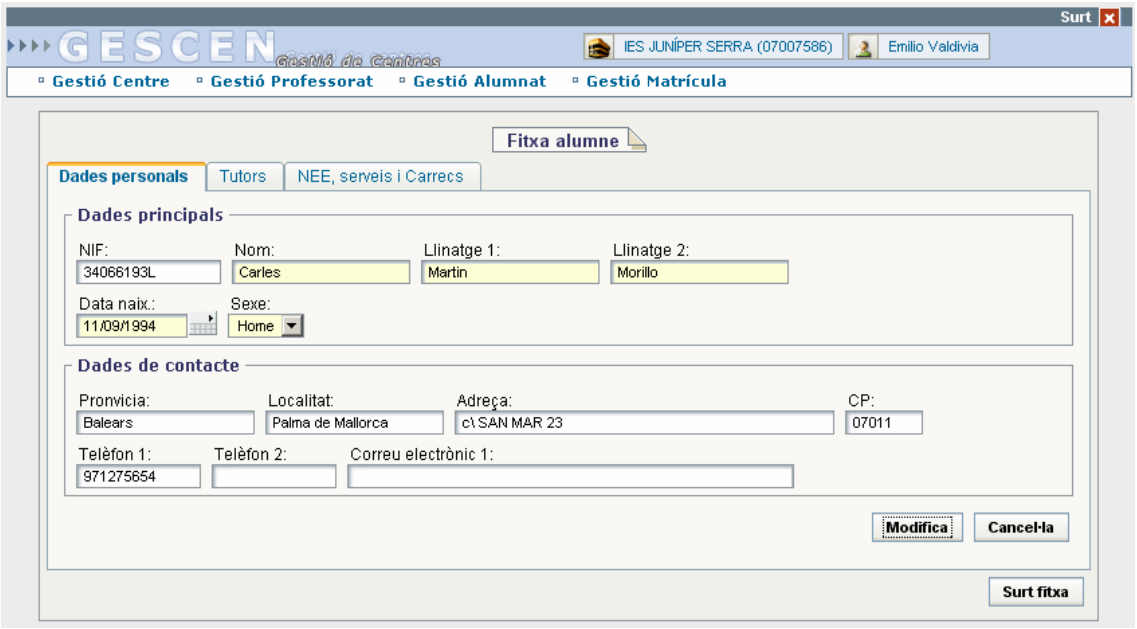
Imatge 20 – Pantalla prototipus professors amb destinacions en el centre

Una vegada en la fitxa del professor, podem consultar dades com les seves destinacions i modificar les dates del període actiu de la seva destinació en el centre:



Imatge 21 – Pantalla prototipus destinacions del professor

En aquesta pantalla podem veure la fitxa de l'alumne en la secció de les seves dades personals:

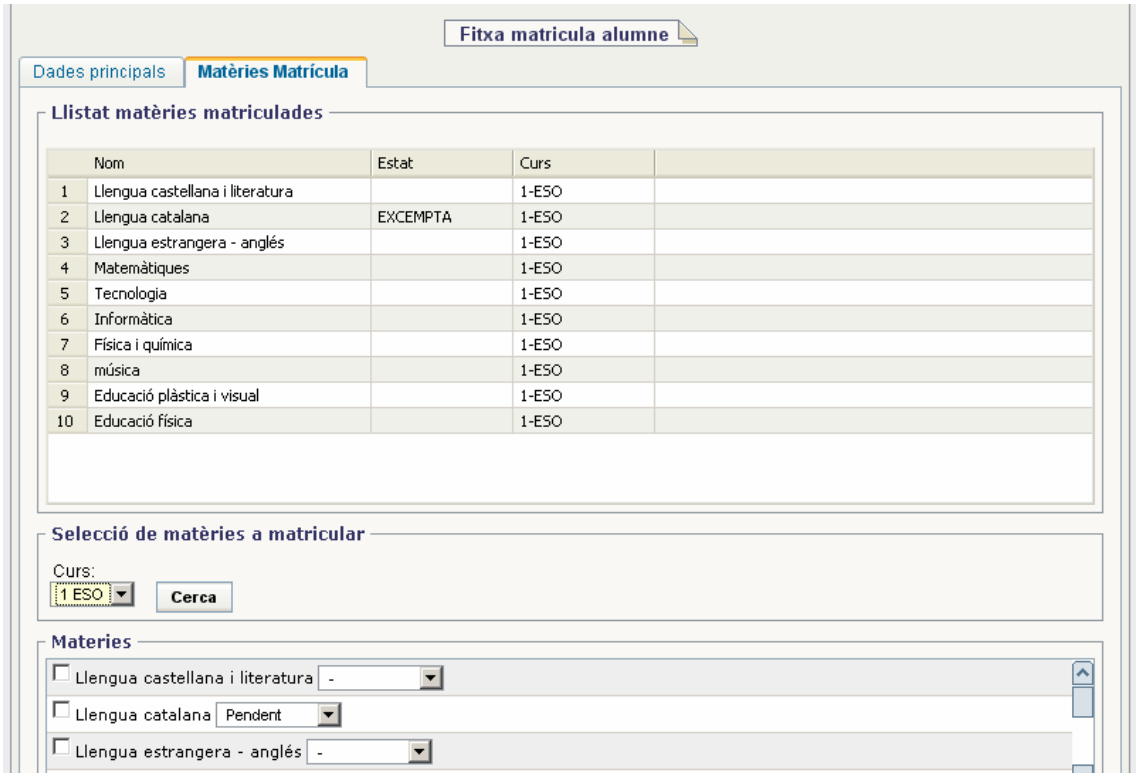


Imatge 22 – Pantalla prototipus dades personals de l'alumne

També podem veure la pantalla per associar els serveis del centre que utilitza l'alumne, o els càrrecs i necessitats educatives especials que té. S'ha de tenir en compte que no es podran eliminar serveis del centre que tinguin alumnes associats.

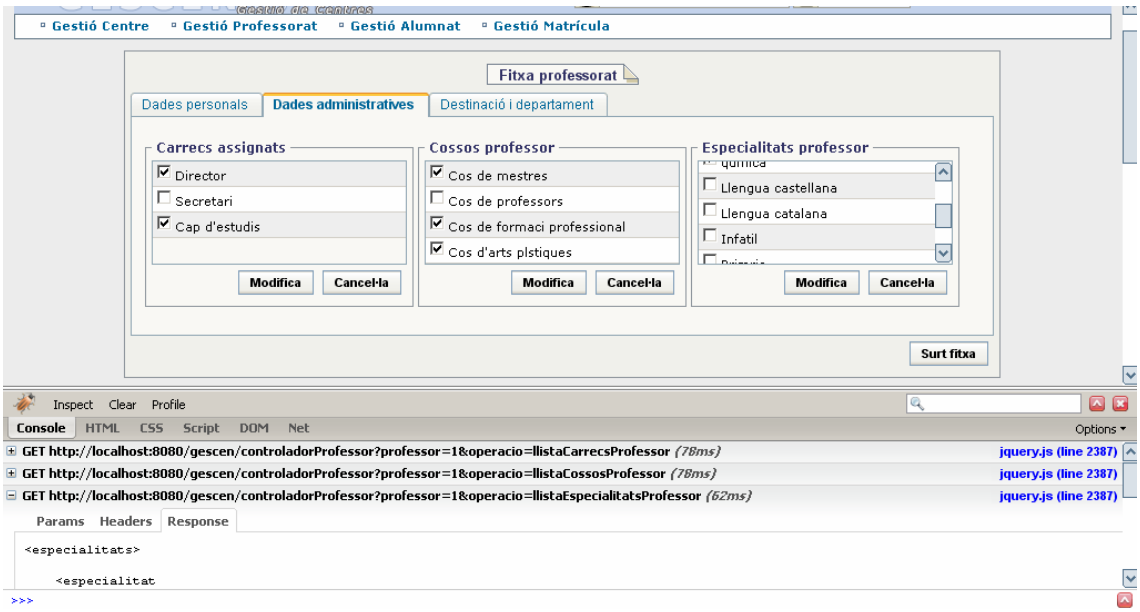
Imatge 23 – Pantalla prototipus NEE, serveis i càrrecs d l'alumne

A continuació es mostra la fitxa de la Matrícula actual de l'alumne en el centre on es poden associar noves matèries:



Imatge 24 – Pantalla prototipus matrícula alumne

Com es pot veure en la següent pantalla gràcies a l'eina Firebug, les dades es van carregant a partir de fitxers estàtics, separant així la presentació de les dades, utilitzant les llibreries JQuery per a realitzar les peticions asíncrones dels recursos corresponents.

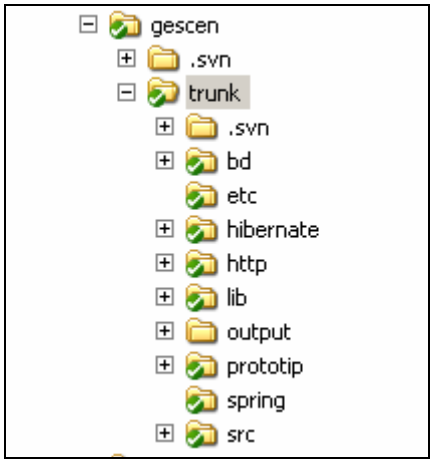


Imatge 25 – Pantalla prototipus dades administratives professor

6 *Implementació del projecte*

6.1 Organització de carpetes

L'estructura del projecte s'ha dividit carpetes dins del control de versions. Aquesta estructura bastant interessant, es basa en l'organització en directoris de les diferents parts que conformen el projecte agafant com a model d'estructura que tenen alguns projectes de codi lliure.



Imatge 26 – Estructura de directoris del projecte

A continuació s'explica breument el contingut de cada una d'aquestes carpetes:

bd

Dins d'aquesta carpeta trobem els *scripts* per a crear la base de dades i el datasource d'exemple que posarem en el JBOSS en default/deploy.

etc

Trobarem jboss.xml i el web.xml on es fan el mapeig de les URLs.

hibernate

Trobarem els arxius de mapeig de la base de dades i les consultes en HQL.

http

Trobarem la part de presentació web de l'aplicació, arxius html, css i javascript, a més dels arxius *jsp* que generen els xml segons les peticions en AJAX que es realitzen segons interactui l'usuari amb la pàgina html, de tal manera que no es faci tota la petició de la pàgina una altra vegada, ja que aquesta ja està carregada.

Més detalladament podem distingir:

- CSS: On es troben els fitxer que donen l'estil als components JavaScript, com el calendari, i a les pàgines html. Com avui en dia es consideren les imatges com a part de l'estil (imatges de fons, efectes d'intercanvi, de degradat...), les imatges relacionades a una pàgina d'estil es troben juntament amb la pàgina d'estil corresponent. Per una banda trobem un CSS amb els estils específics de cada pàgina, i per altra banda dins la carpeta comú els estils que poden trobar-se en més d'una pàgina com per

exemple els estils dels formularis. Dins de la carpeta *principal* es troba la fulla d'estil de la pàgina inicial.

- js: Aquí es troben tots els fitxers **js** que es poden utilitzar dins de l'aplicació. Al directori classes trobem les classes principals JavaScript fetes relacionades amb els formularis amb les funcionalitats de cada un. Així doncs, tenim les classes Centre.js, Alumne.js, Professor.js ...

Dins del directori comú tenim les funcions i components genèrics que s'utilitzaran a les pàgines. Poden destacar aquest arxius:

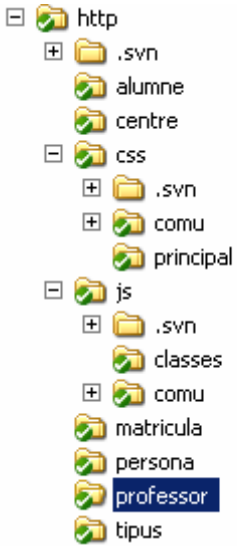
- import.js: llibreria que importarem inicialment i que ens ajudarà a realitzar les importacions dinàmiques d'altres llibreries per blocs o individuals tant llibreries JavaScript com fulles d'estils.
- net/net.js: aquesta classe especial en serveix per a realitzar les peticions asíncrones que utilitzen les llibreries jquery que abstrauen les característiques pròpies de cada navegador pel que fa a la gestió de les peticions al servidor. Podren executar una funció en finalitzar la petició correctament o un altra si aquesta no s'ha pogut realitzar correctament.
- form/check_form.js: funcions de comprovacions de camps a partir d'expressions regulars. Per a comprovar les dades d'un formulari, primer li hem d'assignar a cada camp la classe determinada (per exemple, un camp de correu electrònic li assignem la classe EMAIL). Després només fa falta utilitzar la funció checkForm passant-li com a paràmetre l'objecte formulari que es vol comprovar, o el vector d'objectes camps que volem validar.

També tenim per a cada mòdul html llibreries JavaScript específiques per inicialitzar la pàgina i manejar les graelles de dades. Així doncs, per exemple, tenim pel mòdul centre els arxius : centre_init.js, centre_grid.js i centre_carrega_llistes.js (carrega les llistes inicials que omplen els camps que mostren les llistes desplegables).

Dins del directori tipus trobarem els JSP que construeixen els XML de les llistes bàsiques utilitzades en els mòduls.

Per altra banda, s'ha separat la part de presentació de cada mòdul en els arxius centre.html, professor.html, alumne.html i matricula.html.

Aquí podem veure l'estructura de directoris:



Imatge 27 – Estructura de directoris de la part web

lib

Es troben totes les llibreries i drivers que fa servir el projecte que gràcies al build.xml faran compilar i desplegar el projecte. Dins de la carpeta drivers trobem els drivers per a que es puguem connectar amb el SGBD MySQL necessaris " mysql-connector-java-5.1.5-bin.jar"

output

Dins de product trobarem l'arxiu gescen.war que posarem dins del deploy.

prototip

Aquí es troba el prototip html que fa les peticions en AJAX a arxius xml estàtics que serveix com a base en el desenvolupament de l'aplicació.

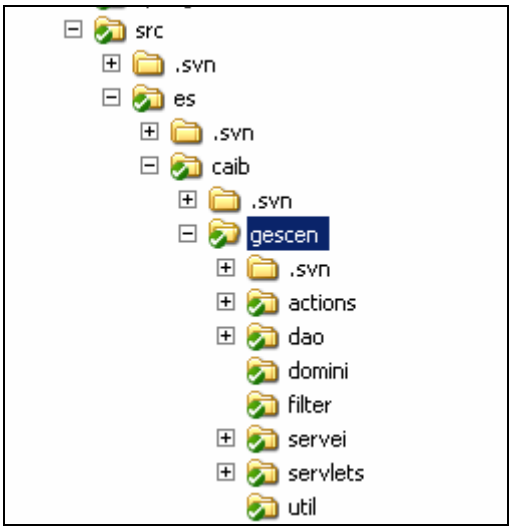
spring

Arxius de configuració de Spring que ens servei com a contenidor de la capa d'integració.

src

Tot el codi de les classes java de l'aplicació; classes de domini, dao, serveis, classes actions, servlets...

Podem diferenciar aquesta estructura:



Imatge 28 – Estructura de directoris de les classes Java

6.2 Arxius de configuració de Spring

A continuació es presenta l'arxiu XML on és veu que gràcies al Contenidor d'Inversió de Control es creen els *beans* necessaris, (realment és una factoria de beans) i també comprovem com es realitza la gestió de transaccions.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:aop="http://www.springframework.org/schema/aop"
      xmlns:tx="http://www.springframework.org/schema/tx"
      xsi:schemaLocation="http://www.springframework.org/schema/beans
                  http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
                  http://www.springframework.org/schema/aop
                  http://www.springframework.org/schema/aop/spring-aop-2.0.xsd
                  http://www.springframework.org/schema/tx
                  http://www.springframework.org/schema/tx/spring-tx-2.0.xsd
                  "
       default-autowire="no" >

   <!-- Data Source -->
   <bean id="dataSource" class="org.springframework.jndi.JndiObjectFactoryBean">
      <property name="jndiName" value="java:/gescen.db" />
   </bean>

   <!-- session factory -->
   <bean id="sessionFactory"
class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
      <property name="mappingLocations">
         <list>
            <value>classpath:/es/caib/gescen/domini/Carrec.hbm.xml</value>
            <value>classpath:/es/caib/gescen/domini/CarrecAlumne.hbm.xml</value>
            <value>classpath:/es/caib/gescen/domini/CarrecProfessor.hbm.xml</value>
            <value>classpath:/es/caib/gescen/domini/Cos.hbm.xml</value>
            <value>classpath:/es/caib/gescen/domini/CosProfessor.hbm.xml</value>
            <value>classpath:/es/caib/gescen/domini/Departament.hbm.xml</value>
            <value>classpath:/es/caib/gescen/domini/DepartamentCentre.hbm.xml</value>
            <value>classpath:/es/caib/gescen/domini/DepartamentProfessor.hbm.xml</value>
            <value>classpath:/es/caib/gescen/domini/Destinacio.hbm.xml</value>
            <value>classpath:/es/caib/gescen/domini/Ensenyament.hbm.xml</value>
            <value>classpath:/es/caib/gescen/domini/EnsenyamentCentre.hbm.xml</value>
            <value>classpath:/es/caib/gescen/domini/EspecialitatProfessor.hbm.xml</value>
            <value>classpath:/es/caib/gescen/domini/Especialitat.hbm.xml</value>
            <value>classpath:/es/caib/gescen/domini/Grup.hbm.xml</value>
            <value>classpath:/es/caib/gescen/domini/MateriaMatricula.hbm.xml</value>
            <value>classpath:/es/caib/gescen/domini/Matricula.hbm.xml</value>
            <value>classpath:/es/caib/gescen/domini/NEE.hbm.xml</value>
            <value>classpath:/es/caib/gescen/domini/NEEAlumne.hbm.xml</value>
            <value>classpath:/es/caib/gescen/domini/Persona.hbm.xml</value>
            <value>classpath:/es/caib/gescen/domini/Professor.hbm.xml</value>
            <value>classpath:/es/caib/gescen/domini/Qualificacio.hbm.xml</value>
            <value>classpath:/es/caib/gescen/domini/Relacio.hbm.xml</value>
            <value>classpath:/es/caib/gescen/domini/ServeiAlumne.hbm.xml</value>
            <value>classpath:/es/caib/gescen/domini/Tutor.hbm.xml</value>
            <value>classpath:/es/caib/gescen/dao/consultesCentre.hbm.xml</value>
```

```
            <value>classpath:/es/caib/gescen/dao/consultesTipus.hbm.xml</value>
            <value>classpath:/es/caib/gescen/dao/consultesAlumne.hbm.xml</value>
            <value>classpath:/es/caib/gescen/dao/consultesProfessor.hbm.xml</value>
            <value>classpath:/es/caib/gescen/dao/consultesMatricula.hbm.xml</value>
         </list>
      </property>
      <property name="hibernateProperties">
         <props>
            <prop key="hibernate.dialect">org.hibernate.dialect.MySQLDialect</prop>
            <prop
key="hibernate.transaction.factory_class">org.hibernate.transaction.JTATransactionFactory</pr
op>
            <prop
key="hibernate.transaction.manager_lookup_class">org.hibernate.transaction.JBossTransaction
ManagerLookup</prop>
         </props>
      </property>
      <property name="dataSource" ref="dataSource"/>
   </bean>

   <!-- dao -->
    <bean id="centreDao" class="es.caib.gescen.dao.hibernate.CentreDaoImpl">
      <constructor-arg ref="sessionFactory"/>
    </bean>
    <bean id="tipusDao" class="es.caib.gescen.dao.hibernate.TipusDaoImpl">
      <constructor-arg ref="sessionFactory"/>
    </bean>
    <bean id="alumneDao" class="es.caib.gescen.dao.hibernate.AlumneDaoImpl">
      <constructor-arg ref="sessionFactory"/>
    </bean>
    <bean id="professorDao" class="es.caib.gescen.dao.hibernate.ProfessorDaoImpl">
      <constructor-arg ref="sessionFactory"/>
    </bean>
    <bean id="matriculaDao" class="es.caib.gescen.dao.hibernate.MatriculaDaoImpl">
      <constructor-arg ref="sessionFactory"/>
    </bean>
    <bean id="personaDao" class="es.caib.gescen.dao.hibernate.PersonaDaoImpl">
      <constructor-arg ref="sessionFactory"/>
    </bean>

   <!-- serveis -->
   <bean id="centreService" class="es.caib.gescen.servei.impl.CentreServiceImpl">
      <constructor-arg  ref="centreDao" />
   </bean>
   <bean id="tipusService" class="es.caib.gescen.servei.impl.TipusServiceImpl">
      <constructor-arg  ref="tipusDao" />
   </bean>
   <bean id="alumneService" class="es.caib.gescen.servei.impl.AlumneServiceImpl">
      <constructor-arg  ref="alumneDao" />
   </bean>
   <bean id="professorService" class="es.caib.gescen.servei.impl.ProfessorServiceImpl">
      <constructor-arg  ref="professorDao" />
   </bean>
   <bean id="matriculaService" class="es.caib.gescen.servei.impl.MatriculaServiceImpl">
      <constructor-arg  ref="matriculaDao" />
   </bean>
   <bean id="personaService" class="es.caib.gescen.servei.impl.PersonaServiceImpl">
```

```
        <constructor-arg  ref="personaDao" />
    </bean>

    <!-- gestor de transaccions-->

    <bean id="transactionManager"
class="org.springframework.orm.hibernate3.HibernateTransactionManager" >
        <property name="sessionFactory" ref="sessionFactory" />
    </bean>

    <tx:advice id="txAdvice" transaction-manager="transactionManager">
        <tx:attributes>
            <tx:method name="llista*" propagation="SUPPORTS" read-only="true" />
            <tx:method name="carrega*" propagation="SUPPORTS" read-only="true" />

            <tx:method name="*" propagation="REQUIRED" read-only="false" />
        </tx:attributes>
    </tx:advice>
     <!--  -->
    <aop:config>
        <aop:advisor
                pointcut="execution(* es.caib.gescen.servei..*(..))"
                advice-ref="txAdvice" />
    </aop:config>
</beans>
```

6.3 Gestió de flux de la capa de presentació

Inicialment vaig optar pel mapeig dels *servlets* de tal forma que las URL fossin descriptives i susceptibles a un posterior filtre de seguritat, de tal manera que un determinat perfil pogués executar només certes operacions. Així doncs teníem:

```
Per a modificar l'aula d'un centre "centre/aula/modifica.do"
Per a eliminar un aula d'un centre "centre/aula/elimina.do"
Per a crear un aula d'un centre "centre/aula/crea.do"
...
```

La implementació posterior consisteix en definir en un arxiu XML la descripció de tota la navegació de l'aplicació, es a dir, la vista realitzarà la petició d'una operació al controlador corresponent, i el controlador trobarà per l'operació demanada la classe a executar i la vista següent a mostrar.

Per tant tenim l'arxiu etc/Action-config.xml :

```
<action-mappings>
  <action>
    <name>modificaCentre</name>
    <description>Modifica el centre</description>
    <action-class>es.caib.gescen.actions.centre.ModificaCentreAction</action-class>
    <forward>resultatOperacio.jsp</forward>
  </action>
  <action>
    <name>centre</name>
    <description>Dades centre</description>
    <action-class>es.caib.gescen.actions.centre.CentreAction</action-class>
    <forward>centre/centre.jsp</forward>
  </action>
...
</action-mappings>
```

Per a cada acció tenim definida la classe que encapsula la recollida de paràmetres necessaris i l'execució de l'operació, i per altra banda tenim en la etiqueta *forward* la resposta, que pot ser bé un XML amb les dades demanades o bé un XML amb l'estat de l'operació.

També tenim el controlador que gestiona las peticiones del client referent al mòdul tractat, per exemple, referents sobre peticions de centre:

(es.caib.gescen.servlets.controlador.ControladorGestioCentreServlet):

```
/**
 * Servlet que fa de controlador, reb totes les peticions d'usuari, tria la lògica de negoci,
 * i executa l'acció corresponent que invoca el mètode de negoci
 * i redirecciona a la vista adecuada en cada cas mitjançant
 * XML de configuració d'accions
 *
 */
public class ControladorGestioCentreServlet extends HttpServlet {

    private CentreService centreService;

    public void init() throws ServletException {
        WebApplicationContext context =
WebApplicationContextUtils.getRequiredWebApplicationContext(getServletContext());
```

```
      centreService = (CentreService) context.getBean("centreService");
   }

   public void service(HttpServletRequest request, HttpServletResponse response) throws IOException {
      String accion = request.getParameter("operacio");// Obtenemos la accion
      try{
         ActionForward actFor = new ActionForward();
         ActionData act = actFor.getAccion(accion);     // Obtenemos la acción a ejecutar del fichero
         String msg = "";
         //System.out.println(act.toString());
         if (act!=null && (act.getForward().indexOf("\r")!=-1 || act.getForward().indexOf("\n")!=-1 ))
            System.out.println("> fatal error accion");
         // Comprobamos que la acción ha sido definida en el XML
         if (act==null){
            msg="Acció:"+accion+" no definida en Action-config.xml";
            System.out.println("msg");
            // Si existe una clase para ejecutar, la ejecutamos
         }else if(act.getActionClass() != null){
            // Instanciamos la clase de la acción y la ejecutamos
            ( (Executor)(Class.forName(act.getActionClass()).newInstance()) ).execute(request, response,
act, centreService);

         }else{//y si no hacemos forward
            // Preparamos el forward respetando el request
            // redireccionem a la JSP adient
              System.out.println("Tenemos redirección:" + act.getActionClass() );
            RequestDispatcher rd = getServletContext().getRequestDispatcher("/"+act.getForward());
            rd.forward(request, response);
         }
      } catch (Exception e){ e.printStackTrace();}
   }
}
```

Podem veure com instanciem la classe *ActionForward* que s'encarregarà de llegir el XML i cercar l'operació dins del XML anteriorment definit, de tal forma que si l'acció està ben definida se carrega l'objecte *ActionData* i si n'hi ha una classe acció associada a executar (que implementa la interfície Executor corresponent), l'executem passant-li l'objecte acció que contindrà la següent pantalla a mostrar.

En el paquet *actions* trobarem les classes acció, por exemple:

```
package es.caib.gescen.actions.centre;
...

/**
 * Servlet que recupera les dades principals d'un centre
 *
 * @author Emilio Valdivia
 */
public class CentreAction extends Executor{
   public void execute(HttpServletRequest request, HttpServletResponse response, ActionData act,
CentreService centreService ) throws Exception{

      request.setAttribute("centre", centreService.carrega(request.getParameter("codi")));
      RequestDispatcher rd = request.getRequestDispatcher(response.encodeURL(act.getForward()));
      rd.forward(request, response);

   }
}
```

6.4 Peticions interfície d'usuari

Per a explicar un poc el funcionament, posarem com exemple la vista "centre.html", que conté tota la presentació relacionada amb la gestió pròpia de les dades del centre (aules, departaments, serveis...), de tal forma que mai haurem de refrescar la pantalla de l'usuari quan es canvia d'estat.

Per tant en una pestanya tenim el següent formulari:

```
....
<form id="formCentre" name="formCentre" action="" method="post">
            <fieldset>
              <legend>Dades principals</legend>
              <p><label>Codi:</label><br/><input id="cen_codi" name="codi" type="text"
class="readOnly" size="15" value="" readonly="readonly"/></p>
              <p><label>Nom centre:</label><br/><input id="cen_nom" name="nom" type="text"
class="FS" size="40" value="" maxlength="128"/></p>
              <p><label>Tipus centre:</label><br/>
                <select id="cen_tipus" name="tipus" class="selectObl">
                  <option value="CC">Col·legi concertat</option>
                  <option value="CEP">Col·legi de primària</option>
                  <option value="CEIP">Col·legi d'infantil i primària</option>
                  <option value="IES">Institut d'ensenyament de secundària</option>
                </select>
              </p>
              </fieldset>

            <fieldset><legend>Tel&egrave;fons</legend>
              <p><label>Telèfon 1:</label><br/><input id="cen_telefon1" name="telefon1" type="text"
class="TELO" size="12" value="" maxlength="9"/></p>
              <p><label>Telèfon 2:</label><br/><input id="cen_telefon2" name="telefon2" type="text"
class="TELO" size="12" value="" maxlength="9"/></p>
            </fieldset>

            <fieldset>
              <legend>Correus electr&oacute;nics</legend>
              <p><label>Correu electr&ograve;nic 1:</label><br/><input id="cen_email1" name="email1"
type="text" class="EMAILO" size="60" value="" maxlength="256"/></p>
              <p><label>Correu electr&ograve;nic 2:</label><br/><input id="cen_email2" name="email2"
type="text" class="EMAILO" size="60" value="" maxlength="256"/></p>
            </fieldset>

            <div class="marcBotons">
              <input type="button" id="idCentreModifica" name="b_modifica" class="boto"
value="Modifica"/>
              <input type="button" id="idCentreCancela" name="b_cancela" class="boto"
value="Cancel·la"/>
            </div>
          </form>
....
```

Tros de codi de "centre.html"

Per altra banda tenim les classes auxiliares JavaScript que encapsulen les operacions de les que disposem. Així doncs, podem trobar dins de "http/js/classes/Centre.js" l'operació:

```
...
 this.modifica=function (){
        var operacio = "&operacio=modificaCentre";
        if( !checkForm(oFormulari) ) return;
        var params = getParamsURL(oFormulari);
            params += "&codi=" + valorCamp(prefix + "_codi");
            params += operacio;
        var loader = new net.ContentLoader(urlControlador,
                    operacioRealitzada,
                    operacioRebutjada,"POST",params,null);
    }
 ...
```

Com podem veure en aquest bocí de codi, recollim els paràmetres del formulari amb la funció *getParamsURL* que disposa la URL amb els noms dels camps i els seus respectius valors amb la codificació necessària per a enviar-los mitjançant el mètode POST al controlador juntament amb l'operació que es vol realitzar.

L'objecte *ContentLoader* del paquet net encapsula les peticions AJAX asíncrones utilitzant les llibreries JQuery que s'encarreguen d'abstraure les diferències entre navegadores i els mètodes de comunicació amb el servidor. Una vebada acabada la petició amb èxit, llavors executarem la funció *operacioRealizada* que llegirà el XML de resposta, i en cas contrari la funció *operacioRebutjada*.

Altres components como el grid, se refresquen també a partir del XML utilitzant peticiones asíncrones.

6.5 Consideracions

A continuació present unes quantes consideracions estretes durant la implementació del projecte:

- A causa de les incompatibilitats entre navegadors i els problemes que aquest fet m'ha donat, he hagut de fer moltes proves per què l'aplicació funcionés en MZ I IE, encara que per a mi, Mozilla Firefox s'ajusta més als estàndard i normes de la W3C.
- Amb l'arquitectura plantejada, les capes de l'aplicació són totalment independents, es a dir, la capa de presentació està feta totalment en AJAX i s'actualitza mitjançant peticions a la capa de lògica de noticia, fent-se la comunicació integrament amb arxius XML. A més els DAO implementen l'accés a dades i amb tenint Srping com a contenidor, canviant els paràmetres adients, sense haver de refer les capes superiors, es pot canviar la implementació de la capa d'accés a dades, per exemple en JDBC o altre framework.
- Per falta de temps, no hi ha un control d'errors exhaustiu però si hi ha un control dels camps mitjançant les expressions regulars adients.
- He provat varies combinacions i l'experiència amb Spring i Hibernate ha estat profitosa però també he hagut de dedicar molt de temps en aprendre qualque cosa d'aquest marc de treballs, encara que sóc conscient que es pot treure molt més profit, ja que he vist les grans avantatges d'escalabilitat que té Hibernate i també he provat la portabilitat, és cert, funciona!

6.6 Compilació

Es pot compilar tot el projecte a partir de build.xml amb *ant* que deixa el war dins de la carpeta output/product i també al *deploy* corresponent que es representa en l'arxiu "properties local.properties.sample". Aquest arxiu s'ha de renombrar a "properties local.properties" i canviar el seu contingut perquè apunti al directori *JBoss* corresponent:

```
compile.debug=true
compile.optimize=false
deploy.dir=c:/java/jboss/jboss-3.2.7-caib1/server/default/deploy
```

De totes maneres com s'ha comentat anteriorment podrem trobar l'arxiu compilat del projecte a la carpeta "output / product / gescen.war"

Imatge 29 – Estructura de directoris del producte final

6.7 Instal·lació del projecte

6.7.1 Descripció dels passos

A continuació es descriuen els passos a seguir per a la correcta instal·lació del sistema:

PAS 1 – Creació base de dades mysql
Dins la carpeta de ***gescen/bd*** trobarem els scripts corresponents per a crear la base de dades en MySql.

PAS 2 – Instal·lació Drivers MYSQL
Assegurar-se de què el servidor JBOSS té els drivers mysql adients. A la carpeta "*gescen/lib / drivers*" poden trobar els drivers que posarem a la llibreria ***lib*** corresponent del JBOSS.

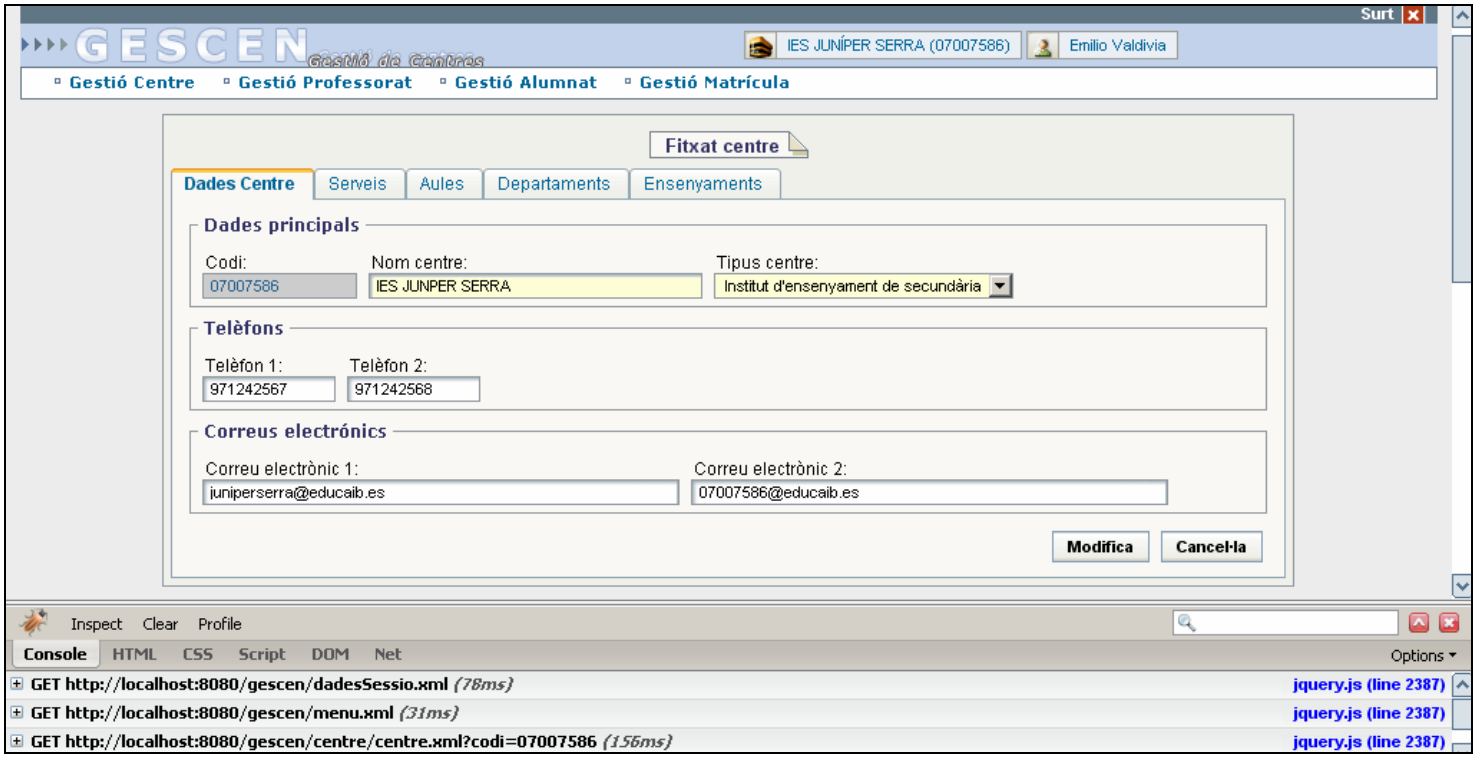
PAS 3 – Configuració Datasource
Agafar l'arxiu gescen-ds.xml de ***gescen/bd/datasorurce*** i canviar l'usuari i contrasenya per una adient a la base de dades creada i posar-la a default/deploy/

PAS 4 – Posar projecte gescen.war al deploy
Agafar l'arxiu gescen.war dins de ***output/product*** i posar-la dins default/deploy/

6.7.2 Accés a l'aplicació

Una vegada instal·lada l'aplicació només ens fa falta posar l'adreça en el navegador

http://localhost:8080/gescen/



Imatge 30 – Pantalla inicial de l'aplicació

Com es pot veure a la part inferior de la pantalla, les peticions es fan mitjançant arxius XML que gràcies al plugin firebug poden visualitzar sense problemes.

7 Conclusions

Des de el meu punt de vista, s'han assolit els objectius que es varen establir al començament del projecte tot seguint les recomanacions del consultor . Per una banda he tingut l'oportunitat de realitzar un estudi general dels marcs de treball més complerts que hi ha avui en dia en continu desenvolupament i que ens ajuda en la construcció de sistemes robustos i de gran qualitat.

Després, avaluant les característiques de cada marc de treball, he seleccionat un parell d'ells que s'adaptaven perfectament al tipus de projecte que volia implementar, de tal forma que he pogut conèixer algunes de les possibilitats Hibernate (que m'ha facilitat el mapeig d'atributs entre la base de dades relacional tradicional i el model d'objectes de l'aplicació) i també de Spring Framework aprofundint en certs aspectes que posteriorment he aplicat en el projecte proposat.

A més, amb la solució proposa en AJAX, aprofitant certes funcionalitats de la llibreria JavaScript JQuery, he donat un valor afegir al sistema ja que s'aconsegueix així un temps de resposta molt més reduït que la típica pàgina que es va recarregant a capa petició o els incòmodes *iframes*. D'aquesta manera es transmet per la xarxa només aquella informació necessària en cada moment mitjançant arxius XML amb les dades demanades en les peticions asíncrones corresponents.

També destacar el control de flux desenvolupat sense aplicar cap marc de treball però basant-se en algunes idees com l'ús d'arxius XML per encapsular les diferents accions que fan possible la navegació i l'ús de controladors per a cada mòdul que reben les peticions dels usuaris permetent així un major control de les diferents accions. D'aquesta manera podríem general un mapa de navegació a partir del XML i el manteniment en el flux de peticions és molt més senzill.

Ha estat una feina complicada a la que he dedicat moltes hores, però finalment crec haver presentat una solució elegant i innovadora respecte a la majoria d'aplicacions pel que fa a la capa de presentació. Per altra banda, sóc conscient que no he tret tot el profit que ofereixen els marcs de treball utilitzats, pot ser ni la dècima part, però en relació al temps del que es disposava estic satisfet de la feina realitzada.

En resum, podrien dir que aquesta experiència ha enriquit els meus coneixements sobre diverses tecnologies que combinades fan possible crear sistemes molt potents de diversos tipus i representen una nova visió en el món de les aplicacions web ja que disminueix molt el temps de resposta que han de esperar els clients i permeten interfícies d'usuari més riques, a més de fer el programari fàcil de mantenir i escalable gràcies a la seva arquitectura.

8 Glossari

AJAX (Asynchronous JavaScript And XML): es una tècnica de desenvolupament web per a crear aplicacions interactives o RIA (Rich Internet Applications) on s'utilitzen varies tecnologies.

Frameworks: marc de treball que ens facilita el desenvolupament i abstreu certs aspectes al programador.

GPL (General Public License): Llicència Pública General per programari que permet la copia, distribució (comercial o no) i modificació del codi, sempre que qualsevol modificació es continuï distribuint amb la mateixa llicència GPL

JBOSS: servidor de aplicacions J2EE de codi obert implementat en Java pur.

JEE (Java Platform, Enterprise Edition): plataforma de programació per desenvolupar i executar programari escrit amb el llenguatge Java, amb una arquitectura distribuïda d'n nivells, basada en components de programari.

MySQL: un sistema de gestió de base de dades relacional, multifil i multiusuari de codi obert, considerat un dels millors per les seves prestacions.

ORM (Object-Relational mapping): és una tècnica de programació per convertir dades de llenguatges de programació orientats a objectes en la seva representació en bases de dades relacionals, a través de la definició de les correspondències entre els diferents sistemes

PFC: Projecte Final de Carrera

SGBD (Sistema de Gestió de Bases de Dades): és un conjunt de programaris dissenyats per a facilitar la gestió d'un conjunt de dades en una base de dades.

SQL (Structured Query Language): Llenguatge estàndard de consulta a bases de dades relacionals.

SOAP (Simple Object Access Protocol): és un protocol dissenyat per intercanviar missatges en format XML que es fa servir per accedir a serveis web.

Xdoclet: llibreria de codi obert implementada per la generació automàtica de codi a partir dels tags Javadoc particulars inserits a les capçaleres dels mètodes

XML (Extensible Markup Language): llenguatge de marques extensible, es adir, és un metallenguatge que permet definir llenguatges de marcat adients per usos determinats.

XUL (XML-based User-interface Language): Llenguatge basat en XML per la interfície d'usuari. És un aplicació de XML feta per Mozilla.

ZK: marc de treball d'aplicacions web en AJAX, completament en Java de codi obert que permet una rica interfície d'usuari per a aplicacions web sense usar JavaScript i amb poca programació.

9 Biografia

- Life above The Service Tier (Ganesh prasad , Rajat taneja, Vikrant Todankar, October 2007).
- Java Development with the Spring Framework (Wrox).
- Hibernate in action (by Christian Baver, Gavin King. 2005 Manning Publications)
- Head Rush Ajax (by Brett McLaughlin, 2006 O' Reilly Media).
- Spring in action (by Craig Walls, 2007 Manning Publications)
- Programación avanzada con XML (Osborne McGraw-Hill).
- Professional CSS Cascading Style Sheets for Web Design (Wiley Publishing, 2005).
- Diversos matèries de la UOC relacionats amb la programació orientada a objectes i J2EE.
- http://www.w3schools.com :consulta de diverses referències pel desenvolupament CSS, HTML, XML, XPATH...
- http://jquery.com/ : Llibreries JavaScript
- http://www.onjava.com :Per a consultes sobre llenguatge Java
- http://www.activewidgets.com/active.controls.grid/ : Per consultes sobre el grid utilitzat en la interfície d'usuari.
- Altres enllaços de consulta per a l'estudi de marcs de treball i el desenvolupament:
  - http://www.zkoss.org/
  - http://www.springframework.org/
  - http://www.laszlosystems.com
  - http://www.hibernate.org/
  - http://java.sun.com/javaee/javaserverfaces/
  - http://struts.apache.org/