PROJECTE PAUTA-V

ESTUDI PER A LA INTEGRACIÓ D'UN MODEL D'EVENTS EN UNA APLICACIÓ WEB BASADA EN EL MODEL MVC

- MEMÒRIA I DOCUMENTACIÓ -

Universitat Oberta de Catalunya

Treball Fi de Carrera dels estudis de l'Enginyeria Tècnica en Informàtica de sistemes

Jordi Martín Caballero
Consultor: Claudi Paniagua Macià
26 de març de 2006

0 – Índex

0 – ÍNDEX **2**

1 - PRESENTACIÓ **5**

2 - LA PROPOSTA INICIAL **5**

3 - L'EVOLUCIÓ DEL PROJECTE **8**

4 - MOTIVACIÓ DEL TREBALL **8**

5 - ABAST DEL PROJECTE **8**

6 - FASE PRÈVIA **9**

6.1 - ENQUESTA 9

Redactors: *10*

Corrector: *11*

Dissenyadors / Maquetadors: *11*

Fotògraf: *12*

6.2 - ANÀLISI DEL MAQUINARI EXISTENT A L'EMPRESA 12

7 - FASE D'ESPECIFICACIÓ **14**

7.1 - DISSENY PRELIMINAR 14

Pauta: *14*

Pàgina: *14*

Notícia: *15*

Fotografia: *15*

Anunci: *15*

Corrector: *15*

Fotògraf: *16*

Director: *16*

Maquetador: *16*

Redactor: *16*

Administrador: *17*

7.2 - FUNCIONALITATS DE L'APLICACIÓ 17

7.3 - ACTORS I CASOS D'ÚS 19

7.4 - APROXIMACIÓ A L'INTERFÍCIE DELS USUARIS 29

8 - APLICACIÓ WEB COL·LABORATIVA A TRAVÉS DE EVENTS **43**

8.1 - STRUTS 43
8.2 - EL MODEL CLÀSSIC 44
8.3 - EL MODEL AMB EVENTS 44
8.4 - EL PROCESSAMENT D'EVENTS 45
8.5 - ELS EVENTS APLICATS AL NOSTRE PROBLEMA 46

9. LÈXIC I SINTAXI DELS EVENTS 48

9.1 - EVENTS 48
CAS D'ÚS: Login 48
CAS D'ÚS: Alta Usuari 51
CAS D'ÚS: Llistar Usuaris 52
CAS D'ÚS: Baixa usuari 53
CAS D'ÚS: Modificació d'usuari 54
CAS D'ÚS: Llistar Pàgines 55
CAS D'ÚS: Visualitzar Pàgina 56
CAS D'ÚS: Alliberar pàgina 59
CAS D'ÚS: ModificarEstatArticle 60
CAS D'ÚS: ModificarEstatFotografia 63
CAS D'ÚS: ModificarEstatAnunci 65
CAS D'ÚS: Borrar Article 67
CAS D'ÚS: Borrar Fotografia 67
CAS D'ÚS: Borrar Anunci 68
CAS D'ÚS: CrearArticle 69
CAS D'ÚS: CrearFotografia 71
CAS D'ÚS: CrearAnunci 72
CAS D'ÚS: ModificarEstatPagina 73
CAS D'ÚS: Borrar Pàgina 76
CAS D'ÚS: Crear Pagina 78
CAS D'ÚS: Crear Pauta 79
CAS D'ÚS: NumerarPàgines 80
Events Genèrics 81
Events de control 83
9.2 – JERARQUIA D'EVENTS 84
9.3 – MAPA DE COMPONENTS 88
9.4 – ESPECIFICACIÓ DE COMPONENTS 93
9.5 - FUNCIONS 99

10 - DIAGRAMA ESTÀTIC DE DISSENY 103

11 - DISSENY DE LA BASE DE DADES RELACIONAL 106

12 - LA GESTIÓ DELS MISSATGES 108

13 - CREACIÓ DE LA BASE DE DADES 110

14 – UN PROJECTE AMB FUTUR 119

15 - BIBLIOGRAFIA 121

16 – AGRAÏMENTS 122

1 - Presentació

Em dic Jordi Martín Caballero i tot estar estudiant informàtica i haver tingut anteriorment treballs en dues empreses relacionades amb el tema avui dia em trobo treballant en un setmanari com a redactor.

Hom podria pensar que amb la tecnologia existent avui en dia la majoria de la feina d'una publicació hauria d'estar automatitzada, però realment encara es realitza molta feina de forma manual, amb la qual cosa no s'aconsegueix una gran eficiència, sobretot quan una tasca s'ha de fer en grup de forma col·laborativa.

Havent triat com a temàtica el treball col·laboratiu i veient que podria ser una bona idea desenvolupar una eina que s'ajustés a les necessitats d'un setmanari de les nostres característiques vull presentar la proposta de creació d'un entorn que permeti solucionar part dels problemes existents en una empresa d'aquestes característiques.

2 - La proposta inicial

Actualment la feina del setmanari es desenvolupa al voltant del que coneixem com a pauta. La pauta és avui dia un conjunt de fulls grapats en paper, dels quals existeix una única còpia, i que tots els treballadors de l'empresa (redactors, dissenyadors fotògrafs, director...) han de compartir. La pauta serveix com a guia i com a registre per saber quins espais estan omplerts del setmanari i amb quin material. A més és l'eina que s'utilitza per tal de saber quin material ja ha estat col·locat pels maquetadors i per realitzar la numeració de les pàgines un cop s'ha d'encetar el procés final de producció d'una nova edició del setmanari.

Com es pot comprovar, són nombroses les persones que habitualment han de treballar, pràcticament de forma constant, amb la pauta:

El redactor l'utilitza per marcar els espais lliures, o ocupats amb articles ja fets i la seva temàtica, així com per marcar els espais que ocupen les fotos.

El fotògraf comprova quines són els fotos que falten per fer i la utilitza per indicar quines es poden col·locar.

El corrector la utilitza per saber quines pàgines pot corregir i deixar llestes per a la fase de maquetació.

Els maquetadors són els encarregats de col·locar les fotos i arreglar les pàgines, així com de fer el darrer procés, que és el de generar les pàgines finals que aniran a impremta.

El director la utilitza per col·locar els anuncis on creu convenient i per realitzar la numeració final que permetrà produir el setmanari.

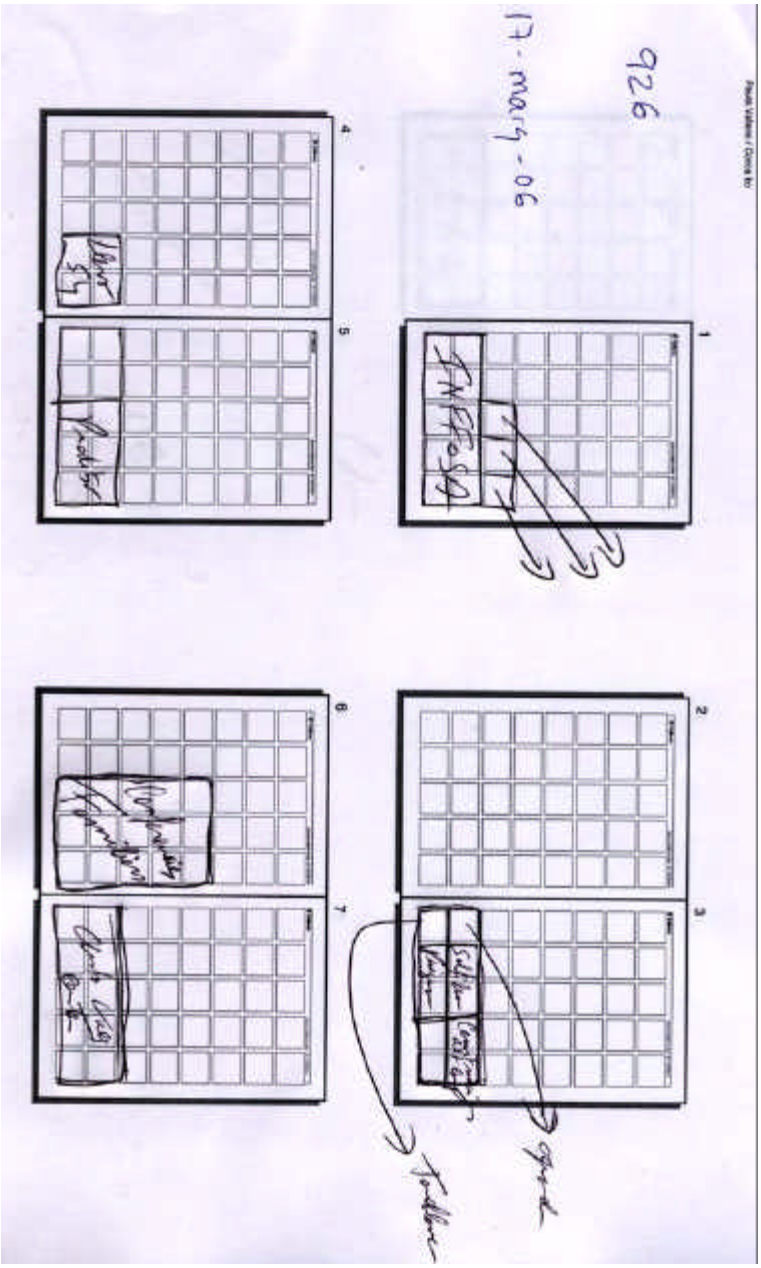
Davant d'aquesta situació la idea és crear una eina que permeti a tots aquests actors poder treballar en un entorn virtual i mitjançant una base de dades amb una pauta informàtica que permeti evitar que la possessió en un moment de la pauta física impedeixi continuar la feina a la resta d'actors.

Creiem que una eina d'aquest tipus pot fer que a curt termini la pauta física, de paper, desaparegui i el treball col·laboratiu de l'empresa s'optimitzi notablement.

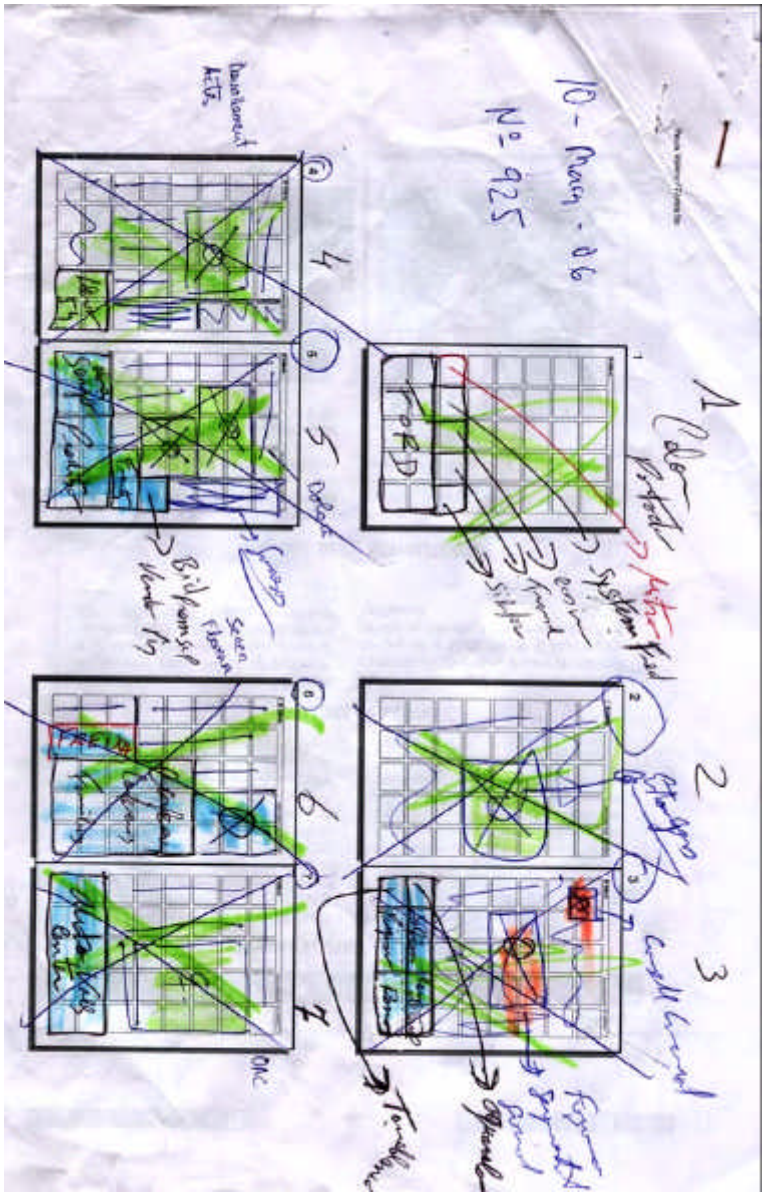
Actualment desconeixem l'existència d'una eina que permeti realitzar aquesta feina. La consulta prèvia que hem realitzat a tres mitjans de comunicació de característiques semblants al que s'utilitzarà com a base pel desenvolupament del treball ens ha permès comprovar que tots utilitzen encara pautes de paper.

A continuació adjuntem la imatge de la pauta a principi de setmana i la imatge un cop finalitzada la setmana per poder comprovar que s'està fent servir un mètode poc òptim que es podria optimitzar notablemente amb una eina d'aquestes característiques:

Imatge d'una pàgina de la pauta a inici de setmana:



Imatge d'una pàgina de la pauta a final de setmana:



Com es pot comprovar la pauta es força caòtica tot i així se segueixen una sèrie de normes, símbols i senyals per tal que tothom comprengui què hi ha a cada pàgina.

3 - L'evolució del projecte

La proposta inicial del projecte va servir com a base per desenvolupar el treball. Val a dir, però, que aquest ha evolucionat enormement des dels seus inicis. A través de la proposta inicial, després de les diferents converses mantingudes amb el consultor de l'assignatura, Claudi Paniagua, el projecte va evolucionar enormement.

Actualment el model MVC integrat en una aplicació web per un entorn col·laboratiu és quelcom força estudiat i que ja es pot veure en nombroses aplicacions. És per aquest motiu que, després de pensar com es podria millorar i innovar aquest sistema, vam arribar a la conclusió que podria resultar molt interessant un estudi sobre la introducció dels Events dins el model MVC.

Partint de la idea base original, en aquest projecte, s'ha volgut estudiar, finalment, quin seria l'impacte i com podria implementar-se el model d'Events en el model clàssic MVC de Struts.

4 - Motivació del treball

Una de les principals motivacions d'aquest treball és que ens trobem davant d'un cas real de treball col·laboratiu que necessita urgentment una millora substancial. La primera impressió al trobar-nos davant els problemes plantejats per aquest treball és que a l'hora de desenvolupar-lo podrem posar en pràctica els coneixements adquirits sobretot en matèria d'enginyeria de la programació orientada a objectes i a més podrem adquirir coneixements sobre el model Model-View-Controller (MVC) gràcies a l'ús de Struts com a mitjà per a desenvolupar el treball davant el que ens trobem.

A més a més, a l'integrar el model d'Events a la proposta inicial, descobrim una part de l'enginyeria que no s'estudia dins la carrera i que actualment no està massa desenvolupada. Buscant a la xarxa podem descobrir que actualment poques eines utiltizen aquest model i que aquest pot tenir un gran potencial que encara està per descobrir. L'aplicació del model a la proposta de la pauta virtual ens pot servir com a model bàsic de comportament i permetre en un futur, l'extrapolació del sistema de funcionament per crear fins i tot un software que permetés la integració del model d'una forma genèrica i senzilla a qualsevol aplicació.

5 - Abast del projecte

Degut a les limitacions de temps és molt difícil poder desenvolupar un prototip complert. Per tant en aquest treball es realitzarà bàsicament l'estudi preliminar d'investigació i disseny de l'aplicació.

En una primera part del treball estudiem el cas davant el que ens trobem, la pauta virtual. A partir d'aquest estudi veiem quines són les necessitats i com s'aplicaria el model MVC a la base de treball: planes, accions, formularis, ...

En una segona part del projecte, la més important, veurem com darrera del model clàssic MVC podem integrar un model d'Events que serà realment el que s'encarregarà de gestionar tota l'aplicació i de crear la pauta. Així doncs, la pauta virtual, en lloc de ser fruit d'una sèrie d'accions directes sobre una aplicació web i la seva base de dades, serà fruit d'una sequència d'events. S'ha realitzat un estudi sobre què és el model d'events i el seu funcionament.

En la tercera part del projecte s'ha desenvolupat en primer lloc el sistema d'events i a continuació la capa dle model MVC a partir de Struts deixant la feina pràcticament llesta per poder, en un futur, passar a programar l'aplicació.

6 - FASE PRÈVIA

6.1 - Enquesta

Per a poder realitzar el disseny de tota aplicació en primer lloc hem de saber quines són les característiques i funcionalitats que aquesta ha de cobrir. Hem de tenir en compte que la principal limitació amb la que ens hem trobat ha estat el temps i per tant la fase prèvia ha estat molt més limitada de tal i com hauria de ser realment aqeusta em cas d'una aplicació real.

Una de les principals limitacions ha estat la impossibilitat de realitzar una enquesta exhaustiva a un grup extens de possibles usuaris i hem hagut de centrar-nos en un grup reduït i proper al qual se li ha realtzat una sèrie de preguntes per tal d'extreure les funcionalitats que tindria la nostra aplicació.

L'empresa El Vallenc compta actualment amb quatre redactors, dos maquetadors, un corrector, un fotògraf i un director. Aquestes són les persones que es veuen implicades directament en l'ús diari de la pauta ja que a banda d'aquests treballadors hi ha secretàries, comercials i una gestora entre d'altres.

Enquesta:

Nom del treballador: _____

Feina que realitza: _____

Quines són les feines bàsiques que realitza amb la pauta:

*

Hi hauria alguna funcionalitat que li agradaria o cregués totalment necessària en una pauta «virtual»?

*

Anotació: Penseu que es crearà tan sols una versió bàsica de la «pauta». Tot i així es tindran en compte les seves possibles ampliacions futures

Per tal de complimentar aquesta enquesta vam dunar una setmana de temps als treballadors. En un primer tots van mostrar-se molt favorables a la realització d'aquest projecte ja que el veien molt útil. Com veurem més endavant, tal i com sol succeir, la motivació dels treballadors va diminuir notablement ja que consideraven feina addicional haver de comprendre el funcionament d'una aplicació d'aquest tipus.

Pasasda la setmana d'espera, durant la qual es va aprofitar per cercar informació sobre el funcionament del model MVC i de Struts, alguns dels treballadors ens varen fer arribar la seva enquesta resposta tal i com presentem a continuació:

Redactors:

Ricard Cañellas Batet

Feines bàsiques que realitza amb la pauta:

- Mirar els espais lliures de les pàgines
- Controlar quantes pàgines existeixen de les seccions que treballo
- Controlar les meves fotos
- Mirar on puc picar articles i quin espai tinc

Funcionalitat addicionals o necessàries que creu convenients a l'aplicació:

- El mateix que una pauta tradicional, manual i de paper
- Que amb un cop d'ull es veiés com està a cada moment
- Si pot ser que sigui molt àgil.

Jordi Martín Caballero

Feines bàsiques que realitza amb la pauta:

- Comprovar quins espais lliures existeixen i quines pàgines es troben en blanc per poder redactar notícies
- Assenyalar els llocs on hi ha anuncis i fotografies
- Reservar espais per a picar notícies.

Funcionalitat addicionals o necessàries que creu convenients a l'aplicació:

- Que el sistema de marcar espais o buscar-los sigui el més àgil possible
- La possibilitat de crear notes per a referència dels altres treballadors.

Corrector:

Josep M. Garcia Requena

Feines bàsiques que realitza amb la pauta:

- Consulta de la pauta per saber quines pàgines estan tancades o parcialment tancades per tal de corregir els textos

Funcionalitat addicionals o necessàries que creu convenients a l'aplicació:

- Quan una pàgina o una part d'una pàgina està corregida, el corrector ho hauria de poder marcar a la pauta perquè qui la consulti sàpiga que el text es pot donar com a bo
- Com que l'espai de la pauta es divideix en mòduls sria interessant que es pogués marcar aquests espais com a corregits. Així es podria saber quins textos són ja correctes qencara que no estigués la pàgina completament tancada. És a dir, marcar text corregit d'una manera semblant com es marca que una foto ja està col·locada.
- El corrector fa servir un full de control de pàgines corregides. Convindria vincular aquest full de correcció amb la pauta general, perquè n'hi hagués prou amb un sol document de treball.
- Hi hauria d'haver una pauta específica per a cada publicació: El Vallenc, Nova Conca, Suplement, L'Opció, Especials, etc… I cada pauta s'hauria de poder trobar amb els arxius de la publicació que pertoqui

Dissenyadors / Maquetadors:

Esther Valldosera Amoroós

Feines bàsiques que realitza amb la pauta:

- Col·locar fotos que prèviament han marcat els redactors i marcar-les com a entrades
- Col·locar els anuncis que prèviament han marcat els redactors i marcar-los com a entrats
- Sapiguer quines són les capçaleres finals de cada pàgina
- Sapiguer quines són les pàgines que vna a color.
- Barallar-me amb els redactors perquè sempre la volen i no me la deixen mai només quan ja és dijous i clar com ells ja ho han fet tot llavors les que pringuem som nosaltres. Tot això provoca una pèrdua de temps vital si es vol tenir vida pròpia. Com poder sopar amb la família els dijous.
- Buscar on està la pauta, perds molt temps per localitzar-la
- Arribar a poder entendre de què va un article desxifrant la cal·ligrafia en què s'ha escrit a la pauta

Funcionalitat addicionals o necessàries que creu convenients a l'aplicació:

- Amb una pauta virtual sol·lucionaríem dos dels problemes anteriorment exposats com el de les baralles i la cal·ligrafia
- Si el fotògraf també pogués accedir a la pauta podria mirar amb quin nom està apuntat al lloc on ha d'anar la fotografia i així no fer-me perdre el temps buscant la foto amb tots els noms possibles en què pot haver estat entrada.
- Poguer marcar d'una forma o altre si la pàgina està tancada, corregida o oberta i així no tindria que anar preguntant al correctorcada cinc minuts.
- No contribuir a la tala indiscriminada d'arbres ja que les pautes no es poden aprofitar i no són de paper reciclat.

Mireia Rafí Català

Feines bàsiques que realitza amb la pauta:

- Buscar la pauta, quí la té?, on és?
- Arreglar i col·locar fotos, després les senyalem a la pauta per indicar que han estat col·locades
- Localització i nom de les fotos que s'han de col·locar encara que de vegades el nom de la pauta no es correspon amb el de la foto i es perd temps
- Mirar la col·locació i ordre en què s'han de posar els anuncis
- Baralles entre el departament de disseny i el de redacció per possessió de la pauta, això provoca retrassos en la feina i mal rotllos entre tots.
- Desxifrar la caligrafia d'alguns redactors

Fotògraf:

Marc Lladó Cuadrat

Feines bàsiques que realitza amb la pauta:

- Marcar o senyalitzar amb diversos signes les fotos que anteriorment m'han demanat per escrit els redactors, que o bé s'han d'anar a fer o s'han de buscar a l'arxiu fotogràfic

Funcionalitat addicionals o necessàries que creu convenients a l'aplicació:

- Que un cop he entrat la fotografia a la carpeta corresponent quedi marcat a la pauta virtual com a foto entrada, bé amb el número o codi del camp associat a la pàgina.

6.2 - Anàlisi del maquinari existent a l'empresa

Actualment els ordinadors de la gent que treballa amb l'empresa model està distribuït en quatre sales: redacció, maquetació, direcció i arxiu, lloc on es troba el corrector.

L'empresa compta amb servidor que es troba situat a la sala de redacció completament desprotegit i a l'abast de tothom. La majoria dels treballadors no tenen coneixements d'informàtica però qualsevol persona amb uns mínims coneixements podria alterar la configuració del servidor sense cap tipus de problema. Desconeixem el model del servidor ja únicament es visible el teclat i el monitor i només es permet a l'informàtic de l'empresa accedir al mateix.

Els redactors treballen amb ordinadors iMAC que compten amb un processador PowerPC G3 a 500 Mhz i 256 M de memòria RAM. Actualment els ordinadors funcionen amb MacOS 9.1 tot i que també tenen instal·lat el sitema MacOS 10 i es preveu fer el canvi permanent durant els propers mesos. Es tracta d'equips antiquats i per tant haurem de fer servir programari més antic que no necessiti grans recursos ja

que fins i tot tenen problemes per suportar el MacOS 10.

En total són cinc els ordinadors que hi ha a redacció, un dels quals té sortida a internet i no està ocupat normalment per ningú. Queden per tant quatre ordinadors fixes pels redactors.

A la sala de maquetació hi ha un total de cinc ordinadors tot i que només hi ha dos maquetadors en actiu. Aquests ordinadors són Power PC G4 a 466 Mhz i compten amb una Giga de memòria RAM. La resta de característiques són iguals als ordinadors de redacció.

A l'arxiu hi ha un únic ordinador, el del corrector, que és un Power PC G4 a 366 Mhz amb 712 Mb de RAM. La resta de característiques són iguals als ordinadors de redacció.

Finalment al despatx del director hi una un iMAC de nova generació les característiques del quals desconeixem concretament ja que el director no ens ha donat permís per a comprovar-les.

Tots aquests ordinadors estan connectats entre ells a través d'una xarxa muntada a través de dos switch, un Fast Ethernet i un Gigabit Ethernet. La configuració de la xarxa és força dolenta ja que duplica serveis a més d'utilitzar material d'una qualiat superior a la realment necessària ja que els ordinadors compten amb tan sols targes Fast Ethernet.

La xarxa compta a més amb quatre servidors de disc dur dedicats a imatges, pàgines de la publicació, anuncis i treballs diversos respectivament.

Com que no disposem d'un servidor web i l'empresa no està disposada a realitzar una inversió per a adquirir-ne un, l'única solució que veiem seria situar el servidor web al mateix servidor existent i limitar les connexions al mateix a través d'un Firewall i taules d'enrutament.

Tot i que aquest darrer punt no és l'objectiu del treball creiem que és convenient tenir-ho en compte per si en algun moment hem de tractar la connectivitat.

7 - FASE D'ESPECIFICACIÓ

7.1 - Disseny preliminar

Amb les enquestes realitzades i els nostres coneixements personals hem entrat en la primera fase de disseny. A partir de la informació que tenim podem fer un enunciat de l'aplicació assenyalant quines característiques quedaran dins del treball i quines deixarem per una versió posterior que en cas de no haver-hi temps quedarien fora del treball. Analitzem a continuació els elements que formen part de la pauta real:

Pauta:

- Una pauta és un esquema que mostra la disposició d'una sèrie d'objectes en la pàgina d'un diari
- Les pàgines estan dividides en mòduls: un mòdul és una casella de mida fixa dins una pàgina. Els objectes s'ajusten a la mida dels mòduls
- Tot i que la pauta comença amb un nombre de pàgines fixes es poden obrir i tancar pàgines segons les necessitats
- Els objectes que poden anar en una pàgina són fotografies, anuncis i articles

Pàgina:

- Una pàgina està dividida de forma fixa en 40 mòduls en forma de graella de 5x8
- Les pàgines han de tenir una numeració final ja que poden imprimir-se en diferent ordre al de la pauta.
- Les pàgines poden estar reservades per un redactor, fotògraf, corrector o maquetador i mentre la té reservada ningú no hi pot treballar a excepció d'ell, ni tan sols el director. En cas de necessitar fer alguna modificació el director ordena al redactor que faci els canvis pertinents.
- Malgrat que tots els mòduls d'una pàgina poden estar plens i no quedar cap mòdul lliure, una pàgina pot restar oberta ja que a vegades falta confirmar dades o esperar que hi pugui haver una darrera modificació.
- Les pàgines poden ser en color o blanc i negre i poden ser dretes, esquerres o indiferents.
- Les pàgines poden ser continuació d'una altra.
- Les pàgines tenen una capçalera que pot ser:
  - Actualitat
  - Cultura
  - Opinió
  - Comarques
  - Societat
  - Esports
  - Especial
- Les pàgines poden estar:
  - Obertes: la pàgina està creada però no està plena
  - Tancada: la pàgina està plena però no està corregida
  - Corregida: la pàgina està corregida però no té tots els anuncis i fotografies posats i no està numerada.
  - Impresa: la pàgina s'ha imprès després de que s'hagin col·locat tots els anuncis, s'hagi corregit i estigui numerada..

Notícia:

- Una notícia té un titular, té un autor i una pàgina assignada
- Una notícia pot ocupar un nombre indeterminat de mòduls dins una o més pàgines
- Els estats d'una notícia són:
  - Sense corregir
  - Corregides

Fotografia:

- Una fotografia té un nom i una pàgina assignada
- Una fotografia pot ocupar un número de mòduls indeterminat dins una sola pàgina
- A vegades les fotografies no s'han d'entrar per poder-les col·locar ja que existeixen fotos habituals que sempre estan entrades i per tant són fotografies que passen directament de l'estat "sense fer" a "col·locada".
- Els estats d'una fotografia són:
  - Sense fer: la fotografia encara s'ha de fer o s'ha de buscar a l'arxiu
  - Entrada: la fotografia ha estat entrada al sistema de fitxers
  - Col·locada: els maquetadors han col·locat la fotografia a la pàgina

Anunci:

- Un anunci té un nom i una pàgina assignats
- Un anunci pot ocupar un número de mòduls indeterminat dins una sola pàgina
- Els estats d'un anunci són:
  - Sense entrar: l'anunci encara no ha estat col·locat a la pàgina
  - Entrat: la fotografia ha estat entrada al sistema de fitxers

A continuació passarem a analitzar les diferents feines i les necessitats que realitza cadascun dels tipus d'empleat de l'empresa i es relacionen amb la pauta en base a les respostes que ens han donat i als nostres coneixements propis. Només inclourem les funcionalitats que inclourem en aquesta primera versió del programa.

Corrector:

- Ha de poder marcar una notícia com a corregida.
- Necessita poder controlar quines notícies estan corregides i quines estan per corregir.
- Necessita poder distingir entre pàgines obertes i tancades ja que aquestes últimes tenen més prioritat a l'hora de ser corregides.

Fotògraf:

- Ha de poder llistar quines són les fotografies que estan per fer o per buscar
- Ha de poder canviar l'estat d'una fotografia (no entrada a entrada)

Director:

- Ha de poder crear i destruir pàgines per tal que quadri el nombre final de pagines
- És l'encarregat de crear la nova pauta a principi de setmana
- Ha de poder numerar les pàgines per que aquestes puguin ser impreses

Maquetador:

- Ha de poder canviar l'estat d'un anunci (per col·locar<->col·locat)
- Ha de poder canviar l'estat d'una fotografia (per fer <->entrada-<>col·locada,)
- Ha de poder canviar l'estat d'una pàgina (corregida->impresa) si tots els anuncis i fotografies de la pàgina estan col·locats, la pàgina està corregida i aquesta té numeració final.
- Ha de poder llistar els anuncis que estan per col·locar, les fotografies que estan per col·locar i les pàgines que estan llestes per imprimir

Redactor:

- Ha de poder canviar l'estat d'un anunci (per col·locar<->col·locat)
- Ha de poder canviar l'estat d'una fotografia (per fer <->entrada-<>col·locada,)
- Ha de poder canviar l'estat d'una pàgina (corregida->impresa) si tots els anuncis i fotografies de la pàgina estan col·locats, la pàgina està corregida i aquesta té numeració final.
- Ha de poder llistar els anuncis que estan per col·locar, les fotografies que estan per col·locar i les pàgines que estan llestes per imprimir
- Ha de poder llistar quines són les fotografies que estan per fer o per buscar
- Ha de poder canviar l'estat d'una fotografia (no entrada a entrada) ja que quan el fotògraf no hi és o està fent una altra feina, són els redactors els encarregats de fer aquesta feina. A més els redactors a vegades fan ells mateixos les fotografies si és necessari.
- Ha de poder crear anuncis, fotografies i notícies
- Ha de poder destruir anuncis, fotografies i notícies
- Ha de poder canviar una pàgina d'estat (oberta<->tancada), color<-> blanc i negre, sense reservar<->reservada, comarques<->actualitat<->…..)
- Ha de poder visualitzar la distribució d'una pàgina per saber quins espais queden lliures.
- Ha de poder crear pàgines ja que de vegades la pauta s'omple i se'n necessiten més
- Ha de poder llistar les pàgines per veure els seus estats

A més d'aquests empleats, en una aplicació d'aquestes característiques, és necessària la figura d'un administrador del sistema. Les seves funcionalitats series:

Administrador:

- Modificar dades de l'aplicació
- Crear base de dades d'usuari
- Llistar usuaris
- Donar d'alta, baixa o modificar un usuari
- Ser capaç de veure el log de l'aplicació
- Gestionar el processador d'events de l'aplicació

7.2 - Funcionalitats de l'aplicació

Tenint en compte les característiques que hem pogut extreure de la informació obtinguda a continuació prenem com a decisió de disseny les següents limitacions degut al poc temps disponible i la complexitat que comporta la creació d'una aplicació d'aquestes característiques sense pràcticament coneixements previs:

Els anuncis i fotografies s'inclouran sempre en una única pàgina, en canvi les notícies poden ocupar pàgines contínues.

Mentre una pàgina estigui reservada (s'està treballant en ella) ningú més que el qui l'ha reservat hi podrà treballar.

Els diferents articles els diferenciarem per colors i no es marcaran les separacions (línies reals existents a la pàgina) a l'hora de visualitzar una pàgina.

Degut a que no tenim permís per instal·lar el programa al sistema de l'empresa, i que tan sols un ordinador té sortida a internet, realitzarem les proves i tests a l'empresa en portàtils.

L'aplicació disposarà d'un sistema de missatgeria bàsic. Constarà de diversos temes (pàgines, notícies, fotografies, anuncis) en els quals s'insertaran missatges a mida que es vagin generant canvis a la pauta. Els usuaris es podran subscriure's als temes i rebran els missatges que es generin a cada tema.

Algunes funcionalitats que creiem interessants i que seran incloses tan sols si hi ha temps disponible, ja que es disposa d'un temps molt limitat, serien les següents:

Creació d'un sistema d'un sistema de log d'accions per tal de permetre en un futur crear noves funcionalitats com podria ser undos, punts de recuperació de la base de dades, ...

Seria útil poder traslladar i modificar objectes d'una pàgina a una altra ja que a vegades és necessari realitzar aquest tipus de modificacions sense la necessitat d'esborrar i tornar a crear un objecte

Finalment incloem un llistat de funcionalitats que no s'inclouran en aquesta versió del programa però que es podrien incloure en una versió futura o al final de la pràctica si ha sobrat temps:

A vegades una publicació es distribueix en més d'una pauta, sobretot quan hi ha especials. S'hauria de poder treballar en més d'una pauta per publicació. A més no es treballa amb una única publicació sinó que es pot treballar amb diverses a la vegada

Les publicacions han de complir alguns requisits (tenir pàgines múltiples de quatre o de vuit segons la publicació, les pàgines a color no poden estar a qualsevol lloc, …)

Seria útil un buscador de notícies / fotografies / anuncis

Les pàgines no tenen la mateixa mida en totes les publicacions ni tenen el mateix nombre de mòduls

A vegades un objecte no ocupa un mòdul sencer sinó només mig i això s'hauria de poder controlar (potser fen subdivisions del mòduls)

Els redactors, en lloc de reservar pàgines senceres el que fan es reservar espais segons una aproximació.

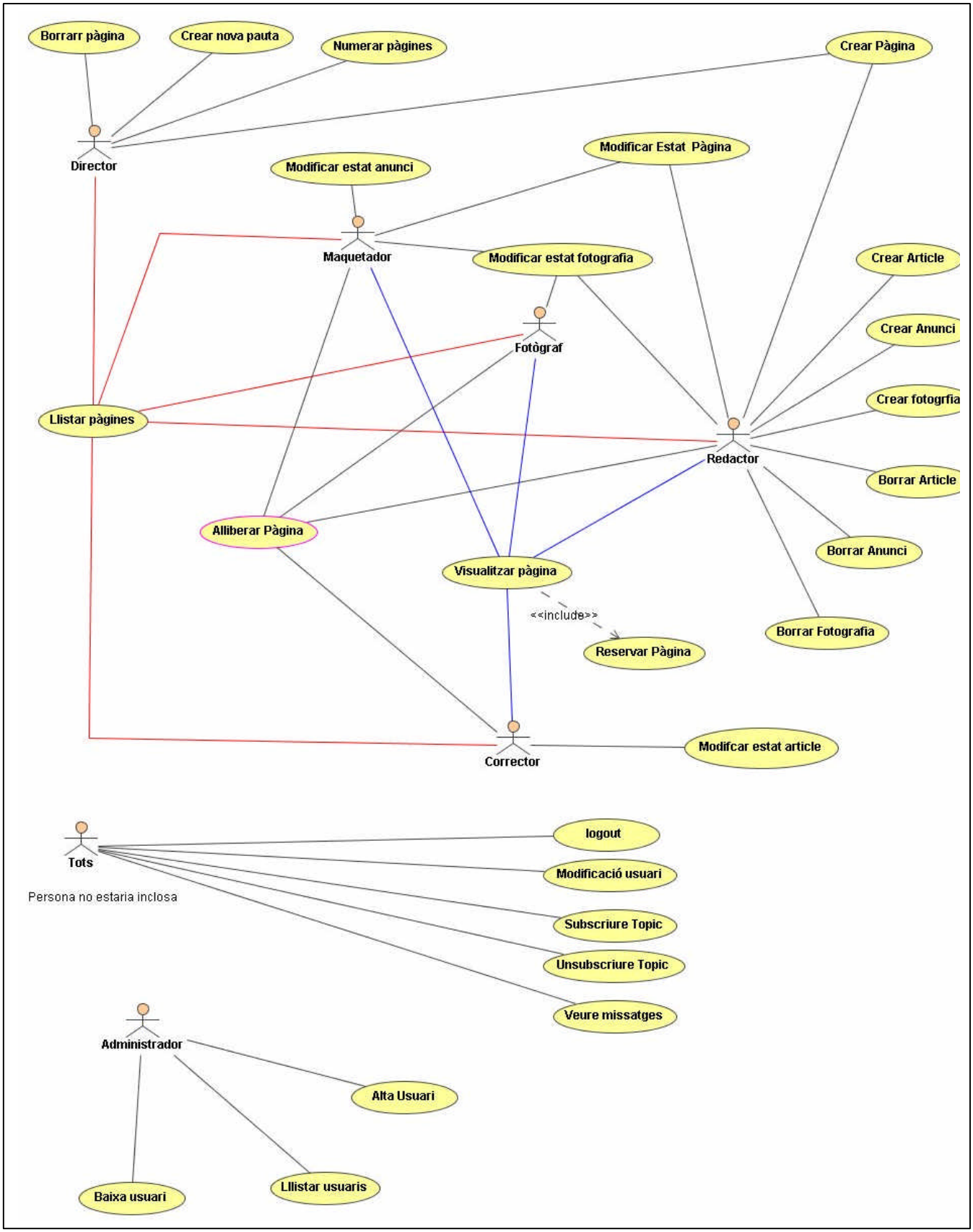
La visualització i possibilitat d'imprimir la versió final de la pauta en la seva totalitat seria una característica molt útil pel programa.

Les pautes haurien de permetre registrar una sèrie de control de pàgines. En la pauta real per exemple, no totes les pàgines poden anar en color ni es poden crear tantes pàgines com es vulguin. A més els anuncis, per exemple, han d'estar col·locats en cert ordre.

A través del registre de log es podria implementar accions UNDO en un futur.

7.3 - Actors i casos d'ús

Amb les decisions de disseny preses anteriorment i les funcionalitats descobertes podem procedir doncs a crear el diagrama de casos d'ús:



Procedim a descriure detalladament cadascun dels casos d'ús:

Login:
Actor: Persona
Postcondició: la persona queda identificada i pot accedir a les seves accions si la identificació és correcta

-La persona introdueix el seu login i password
-En cas que quedi identificat accedeix a la pantalla apropiada als seus privilegis

-En cas que no es validi la identificació es retorna a la pantalla inicial informant de l'error.

Logout:
Actor: Tots els actors excepte Persona
Postcondició: es clou la sessió, es buiden les dades temporals i l'actor perd la identificació i els seus privilegis

-S'alliberen totes les pàgines reservades per l'actor.
-Es tanca la sessió i es buiden les dades temporals emmagatzemades.
-S'informa de l'èxit de l'operació si tot ha anat bé.
-En cas contrari s'informa dels possibles errors.

Alta usuari:
Actor: Administrador
Postcondició: L'usuari queda donat d'alta si tot és correcte

-L'administrador introdueix les dades, password i el rol de l'usuari (redactor, fotògraf…)
-Es comprova que l'identificador d'usuari sigui correcte i únic.
-Si les dades han estat introduïdes correctament i l'identificador és únic s'actualitza la base de dades
-Si alguna de les dades no és es retorna a la mateixa pantalla tot informant de l'error existent
-Si l'administrador cancel·la l'operació es retorna al menú.

Llistar usuaris:
Actor: Administrador:

-El sistema mostra per pantalla els usuaris existent. Si hi ha algun error es mostra per pantalla

Baixa usuari:
Actor: Administrador
Precondició: L'usuari existeix
Postcondició: L'usuari queda donat de baixa si tot va bé

-L'administrador selecciona l'usuari de la llista d'usuaris.
-El sistema intenta donar de baixa, a la base de dades, l'usuari.
-Si té èxit s'informa del mateix i es mostra la nova llista d'usuaris.
-Si no ha tingut èxit s'informa de l'error i es torna a mostrar la llista d'usuaris.

Modificació usuari.
Actor: Administrador, Redactor, Fotògraf, Maquetador, Corrector, Director,
Precondició: Si es tracta de l'administrador aquest ha sel·leccionat un usuari de la llista. En cas contrari apareixen les dades del propi usuari.
Postcondició: Les dades queden modificades amb èxit amb els canvis realitzats

-Segons els privilegis de cada rol, l'actor pot modificar les dades de l'usuari.
-El sistema mostra les dades de l'usuari si no hi ha hagut cap error.
-L'actor les modifica a voluntat.
-Si l'actor accepta els canvis el sistema intentar actualitzar la base de dades.
-Si hi algun error o es cancel·la l'operació finalitza informant dels possibles errors.

Llistar pàgines:
Actors: Redactor, Fotògraf, Maquetador, Director, Corrector
Postcondició: la pantalla mostra la informació demanada per l'actor.

-L'actor escull quines són les dades que vol visualitzar
-Si hi ha algun problema amb el llistat el sistema mostra l'error i finalitza el procediment
-En cas que no hi hagi errors mostra les dades demanades amb les diferents opcions a poder realitzar segons els privilegis de cada actor i permet realitzar una nova cerca.

Visualitzar pàgina
Actor: Redactor, Fotògraf, Maquetador, Corrector
Precondició: existeix un llistat de pàgines en pantalla i la pàgina a visualitzar no està reservada.
Postcondició: es mostra un disseny visual de la pàgina seleccionada.

-L'actor seleccionar una pàgina de la llista.
-L'actor prem el botó de visualitzar la pàgina
-El sistema comprova que la pàgina existeix i que no està reservada
-Si hi ha algun error el sistema el mostra per pantalla i finalitza el procés.
-El sistema mostra el resultat de la conversió de les dades a format visual per pantalla i marca la pàgina com a reservada.

Alliberar pàgina
Actor: Redactor, Fotògraf, Maquetador, Corrector
Precondició: S'està visualitzant una pàgina reservada
Postcondició: es mostra un disseny visual de la pàgina seleccionada.

-L'actor prem el botó d'alliberar pàgina
-El sistema comprova que la pàgina existeix i que està reservada per l'actor
-El sistema intenta alliberar la pàgina.
-Si tot va bé el sistema torna al menú de l'usuari.

Modificar estat article:
Actors: Corrector
Precondició: S'ha seleccionat un article prèviament
Postcondició: es realitzen els canvis a la base de dades i es mostren novament la pàgina actualitzada

-L'actor realitza els canvis que creu oportuns
-El sistema comprova que la pàgina estigui reservada per l'actor.
- Si hi ha algun problema el sistema mostra l'error i finalitza el procediment
-El sistema actualitza les dades de l'article.
-El sistema comprova si a la pàgina està tancada.
-Si és així i si tots els articles estan corregits i la marca com corregida
-Si està tancada però no tots els articles estan corregits la marca com no corregida
-El sistema marca la pàgina com no impresa
-El sistema mostra els canvis realitzats a la pàgina.

Modificar estat fotografia
Actors: Fotògraf, Maquetador, Redactor
Precondició: S'ha seleccionat una fotografia prèviament
Postcondició: es realitzen els canvis a la base de dades i es mostren novament la pàgina actualitzada

-L'actor canvia l'estat de la fotografia segons els seus privilegis
-El sistema comprova que el canvi d'estats segons els privilegis és possible i que la pàgina estigui reservada per l'actor en qüestió.
-Si hi ha algun problema amb el llistat el sistema mostra l'error i finalitza el procediment
-El sistema actualitza les dades de la fotografia
-El sistema marca la pàgina com no impresa.
-En cas que no hi ha errors es torna a mostra els canvis realitzats.

Modificar estat anunci
Actor: Maquetador
Precondició: S'ha seleccionat un anunci
Postcondició: es realitzen els canvis a la base de dades i es mostren novament la pàgina actualitzada

-L'actor selecciona l'anunci del que vol canviar l'estat
-El sistema comprova que la pàgina estigui reservada per l'actor
-Si hi ha algun problema el sistema mostra l'error i finalitza el procediment
-El sistema actualitza les dades de l'anunci
-El sistema marca la pàgina com no impresa
-Es mostren els canvis realitzats.

Borrar article:
Actor: redactor
Precondició: S'ha seleccionat un article
Postcondició: es realitzen els canvis a la base de dades i es mostren novament la pàgina actualitzada

-El sistema comprova que la pàgina estigui reservada per l'actor en qüestió.
-El sistema intenta donar de baixa l'article a la base de dades.
-El sistema marca la pàgina com oberta, no corregida i no impresa
-Si hi ha algun problema el sistema el mostra per pantalla.
-El sistema mostra la vista de la pàgina actualitzada

Borrar fotografia:
Actor: redactor
Precondició: S'ha seleccionat una fotografia
Postcondició: es realitzen els canvis a la base de dades i es mostren novament la pàgina actualitzada

-El sistema comprova que la pàgina estigui reservada per l'actor en qüestió.
-El sistema intenta donar de baixa la fotografia a la base de dades.
-El sistema marca la pàgina com oberta i no impresa
-Si hi ha algun problema el sistema el mostra per pantalla.
-El sistema mostra la vista de la pàgina actualitzada

Borrar anunci:
Actor: redactor
Precondició: S'ha seleccionat un anunci
Postcondició: es realitzen els canvis a la base de dades i es mostren novament la pàgina actualitzada

-El sistema comprova que la pàgina estigui reservada per l'actor en qüestió.
-El sistema intenta donar de baixa l'anunci a la base de dades.
-El sistema marca la pàgina com oberta i no impresa
-Si hi ha algun problema el sistema el mostra per pantalla.
-El sistema mostra la vista de la pàgina actualitzada

Crear Article:
Actor: redactor
Precondició: S'està visualitzant una pàgina
Postcondició: es realitzen els canvis a la base de dades i es mostren novament la pàgina actualitzada

-L'actor selecciona els mòduls lliures que vol ocupar amb l'article.
-L'actor prem el botó de crear article
-El sistema comprova que la pàgina està reservada per l'actor.
-Si hi ha algun error el sistema el mostra per pantalla i finalitza el procés.
-El sistema comprova que els mòduls estan lliures.
-Si hi ha algun error el sistema el mostra per pantalla i finalitza el procés.
-El sistema intentar actualitzar els mòduls i marca l'article com no corregit
-Si hi ha algun error el sistema el mostra per pantalla i finalitza el procés.
-El sistema mostra la visualització de la pàgina actualitzada

Crear Fotografia
Actor: redactor
Precondició: S'està visualitzant una pàgina
Postcondició: es realitzen els canvis a la base de dades i es mostren novament la pàgina actualitzada

-L'actor selecciona els mòduls lliures que vol ocupar amb la fotografia.
-L'actor prem el botó de crear fotografia
-El sistema comprova que la pàgina està reservada per l'actor.
-Si hi ha algun error el sistema el mostra per pantalla i finalitza el procés.
-El sistema comprova que els mòduls estan lliures.
-Si hi ha algun error el sistema el mostra per pantalla i finalitza el procés.
-El sistema intentar actualitzar els mòduls i marca la fotografia com per buscar
-Si hi ha algun error el sistema el mostra per pantalla i finalitza el procés.
-El sistema mostra la visualització de la pàgina actualitzada

Crear anunci
Actor: redactor
Precondició: S'està visualitzant una pàgina
Postcondició: es realitzen els canvis a la base de dades i es mostren novament la pàgina actualitzada

-L'actor selecciona els mòduls lliures que vol ocupar amb l'anunci
-L'actor prem el botó de crear anunci
-El sistema comprova que la pàgina està reservada per l'actor.
-Si hi ha algun error el sistema el mostra per pantalla i finalitza el procés.
-El sistema comprova que els mòduls estan lliures.
-Si hi ha algun error el sistema el mostra per pantalla i finalitza el procés.
-El sistema intentar actualitzar els mòduls i marca l'anunci com per col·locar
-Si hi ha algun error el sistema el mostra per pantalla i finalitza el procés.
-El sistema mostra la visualització de la pàgina actualitzada

Modificar estat pàgina:
Actor: redactor, maquetador
Precondició: S'està visualitzant una pàgina
Postcondició: es realitzen els canvis a la base de dades i es mostren novament la pàgina actualitzada

-L'actor modifica la capçalera, i els diferents estats de la pàgina (segons els seus privilegis)
-El sistema comprova que la pàgina està reservada per l'actor.
-Si hi ha algun error el sistema el mostra per pantalla i finalitza el procés.
-El sistema comprova que es compleixen les normes de privilegis i les regles de cada estat
-Si hi ha algun error el sistema el mostra per pantalla i finalitza el procés.
-En cas contrari intentar actualitzar la pàgina
-El sistema mostra la visualització de la pàgina actualitzada

Borrar pàgina:
Actor: Director
Precondició: existeix un llistat de pàgines a pantalla
Postcondició: la pàgina ha quedat destruïda i es mostra el llistat complert de pàgines

-L'actor sel·lecciona la pàgines que vol destruir
-L'actor prem el botó d'actualitzar.
-El sistema comprova que les pàgina existeix i que no està reservada per cap actor.
-Si hi ha algun problema el sistema el mostra per pantalla i finalitza el procés
-El sistema esborra tota la informació relacionada amb la pàgina seleccionada existent a la base de dades.
-El sistema mostra el llistat de pàgines actualitzat per pantalla

Crear pàgina
Actors: Director, Redactor
Postcondició: la pàgina es crea amb la capçalera escollida i es mostra el llistat complert de pàgines

-L'actor selecciona quina capçalera li vol donar a la nova pàgina.
-L'actor prem el botó de crear.
-El sistema comprova que les pàgines no s'hagin numerat ja que no es poden crear més pàgines un cop numerades.
-Si n oestan numerades el sistema intenta crear una nova pàgina oberta amb la capçalera escollida al final de les pàgines existents la marca amb la capçalera adequada i la marca com oberta, blanc i negre, sense corregir i no impresa.
-Si hi ha algun problema el sistema el mostra per pantalla i finalitza el procés
-El sistema mostra el llistat actualitzat de pantalles

Crear nova pauta:
Actors: Director
Precondició: la pauta amb el número seleccionat no existeix
Postcondicióó: la base de dades s'ha creat i s'han donat d'alta les pàgines bàsiques.

-L'actor introdueix el número de la nova pauta
-El sistema comprova que la pauta no existeix anteriorment a la base de dades
-El sistema crea la pauta i crea les pàgines bàsiques:: 1-25 actualitat, 26-27 economia, 28-31 cultura, 32-45 opinió, 46-65 esports, 66-89 Comarques, 90-94 societat, 95-99 escola, 100-103 societat, 104 actualitat. Totes les pàgines son marcades com obertes, no corregides, blanc i negre i no impresa.
-Si hi ha algun error el sistema el mostra per pantalla i finalitza el procés.
-El sistema mostra el llistat de pàgines per pantalla.

Numerar pàgines:
Actors: Director
Precondició: la pauta amb el número seleccionat no existeix
Postcondicióó: la base de dades s'ha creat i s'han donat d'alta les pàgines bàsiques.

-L'actor introdueix el nombre per a cada pàgina
-El sistema comprova que no hi ha números repetits.
-Si hi ha algun error el sistema el mostra per pantalla i finalitza el procés.
-El sistema intenta actualitzar la base de dades i si la numeració ja existia la reemplaça.
-El sistema mostra el resultat per pantalla

Subscriure Topic:
Actors: Administrador, Director, Fotògraf, Maquetador, Redactor, Corrector
Postcondicióó: L'usuari queda registrat de forma persistent a un topic.

-L'actor sel·lecciona el topic al que es vol subscriure
-El sistema intenta actualitzar la base de dades
-Si hi ha algun error el sistema el mostra per pantalla i finalitza el procés.
-Es crea la subscripció al topic en qüestió.
-Si hi ha algun error el sistema el mostra per pantalla i finalitza el procés.

Unsubscriure Topic:
Actors: Administrador, Director, Fotògraf, Maquetador, Redactor, Corrector
Postcondicióó: L'usuari destrueix la subscripció a un topic

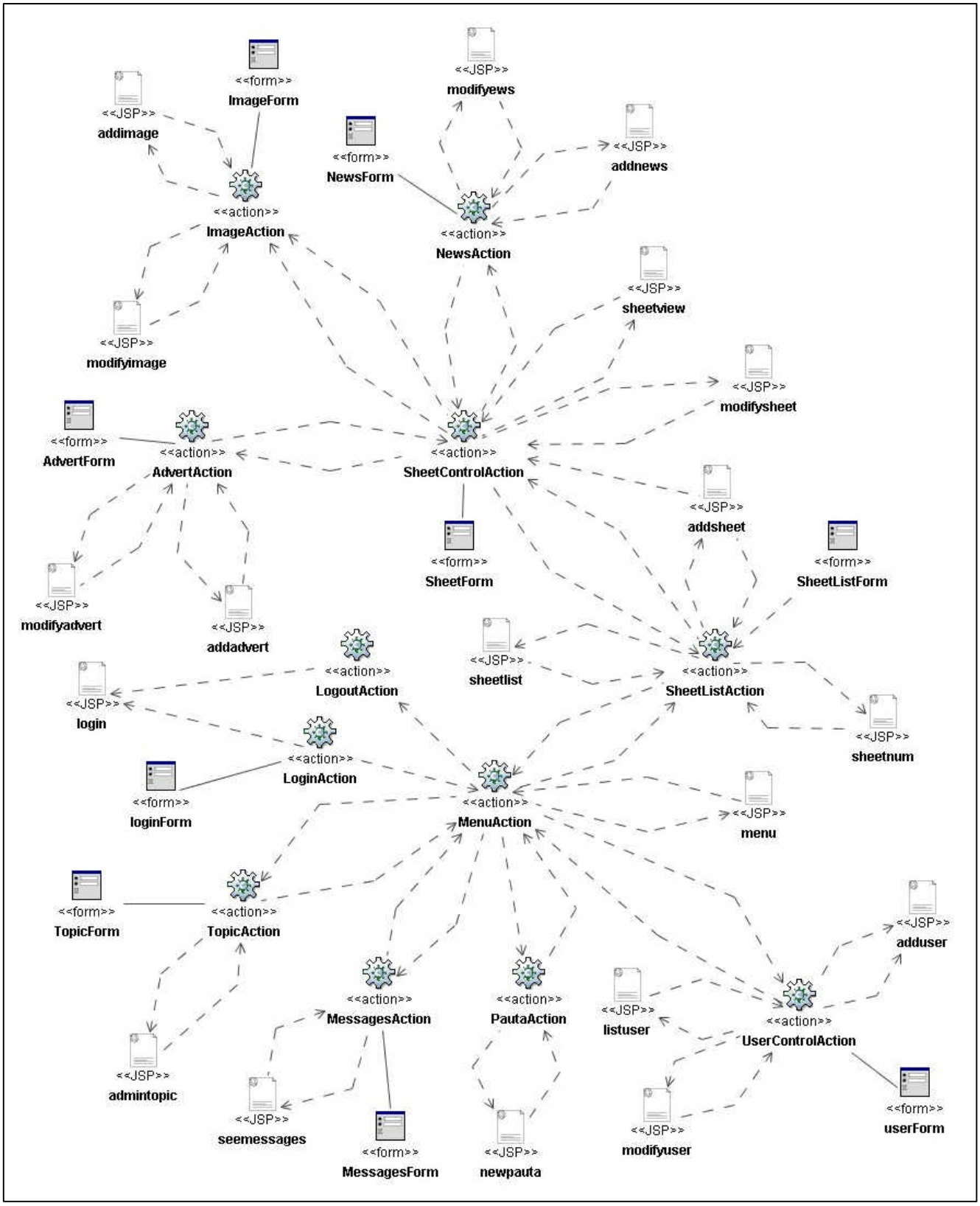
-L'actor sel·lecciona el topic al que es vol unsubscriure
-El sistema intenta actualitzar la base de dades
-Si hi ha algun error el sistema el mostra per pantalla i finalitza el procés.
-Es crea la subscripció al topic en qüestió.
-Si hi ha algun error el sistema el mostra per pantalla i finalitza el procés.

Veure missatges:
Actors: Administrador, Director, Fotògraf, Maquetador, Redactor, Corrector
Postcondicióó: L'usuari veu els missatges

-El sistema mostra per pantalla el missatges dels temes als quals l'actor està subscrit

7.4 Diagrama de flux

Un cop hem detallat els diferents casos d'ús de l'aplicació podem dissenyar el diagrama de flux provisional.



Pot semblar a primer cop de vista que en aquest diagrama no es troben representats tots els casos. L'explicació la trobem en una de les decisions de disseny que hem pres en aquest punt, i és que les accions similars, que comparteixen un mateix formulari o una funcionalitat similar les agruparem sota accions derivades del tipus DispatchAction que ens ofereix Struts.

Així doncs els diferents casos d'ús quedarien repartits de la següent manera:

LoginAction:
-Login

LogoutAction:
-Logout

MenuAction:
Cap en especial, simplement ens portarà a diferents seccions de l'aplicació

UserControlAction:
-Alta usuari
-Baixa usuari
-Modificar usuari
-Llistar usuaris

SheetListAcion:
-Llistar pàgines
-Numerar pàgines

SheetControlAction:
-Reservar pàgina
-Alliberar pàgina
-Borrar Pàgina
-Modificar estat Pàgina
-Visualitzar Pàgina
-Crear pàgina

ImageAction:
-Crear fotografia
-Esborrar fotografia
-Modificar estat fotografia

NewsAction:
-Crear article
-Esborrar article
-Modificar estat article

AdvertAction:
-Crear anunci
-Esborrar anunci
-Modificar estat anunci

TopicAction:
-SubscriureTopic
-Unsubscriure Topic

MessageAction:
-Llistar missatges

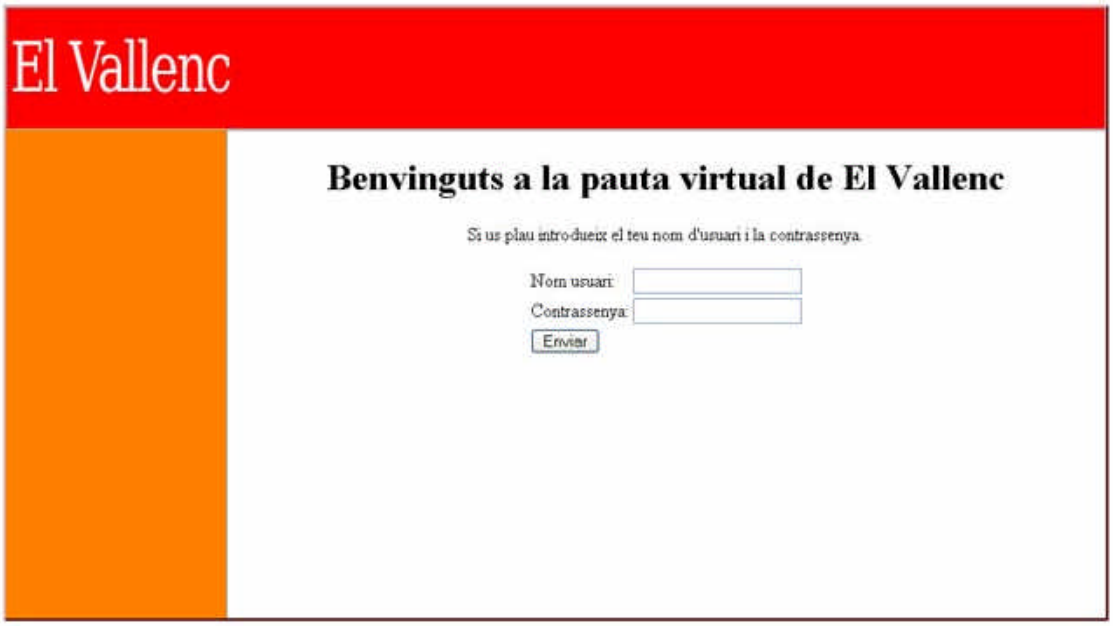
PautaAction:
-Crear nova pauta

Com podem veure tots els casos d'ús estan considerats dins les diferents accions i només s'hauran de controlar dins la navegació. El que si que falta, en aquest diagrama de flux, és el control d'errors que en aquest moment ens és difícil de precisar.

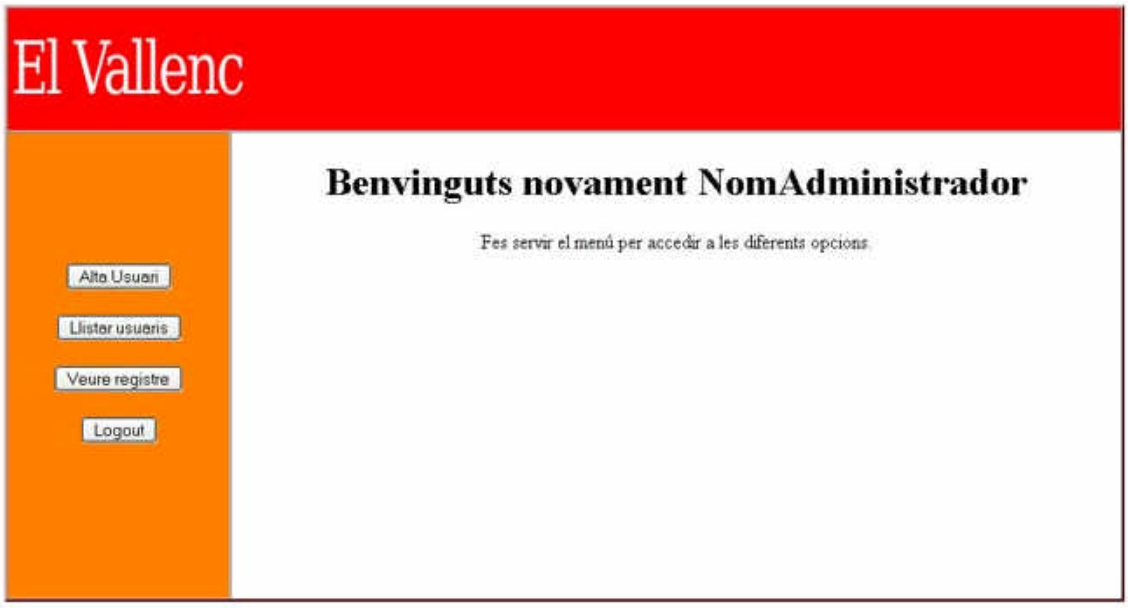
7.4 - Aproximació a l'interfície dels usuaris

Ara que tenim els casos d'ús i el diagrama de flux podem dissenyar esquemàticament com seran les pantalles que mostrarà l'aplicació. Cal tenir en compte que segons els privilegis de cada usuaris les opcions mostrades poden canviar a cada pàgina:

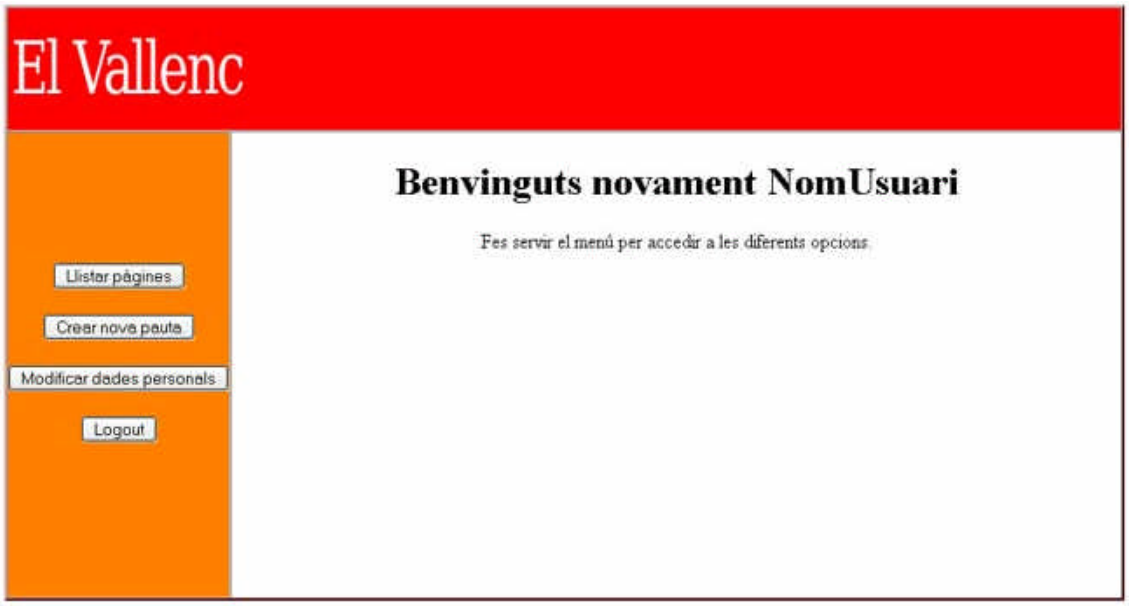
-Login



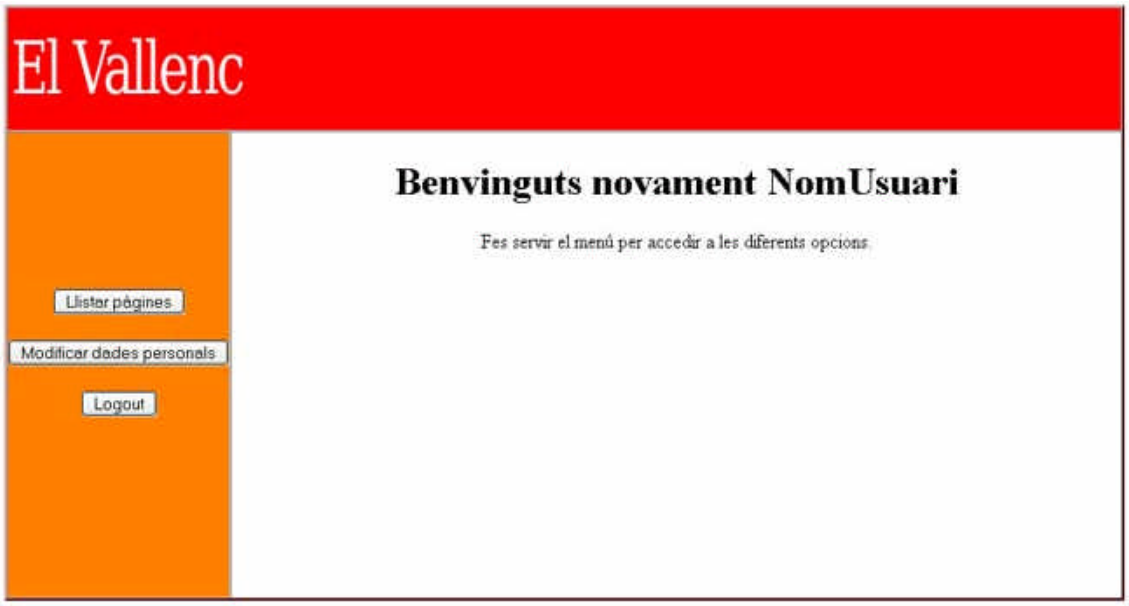
-Menú administrador



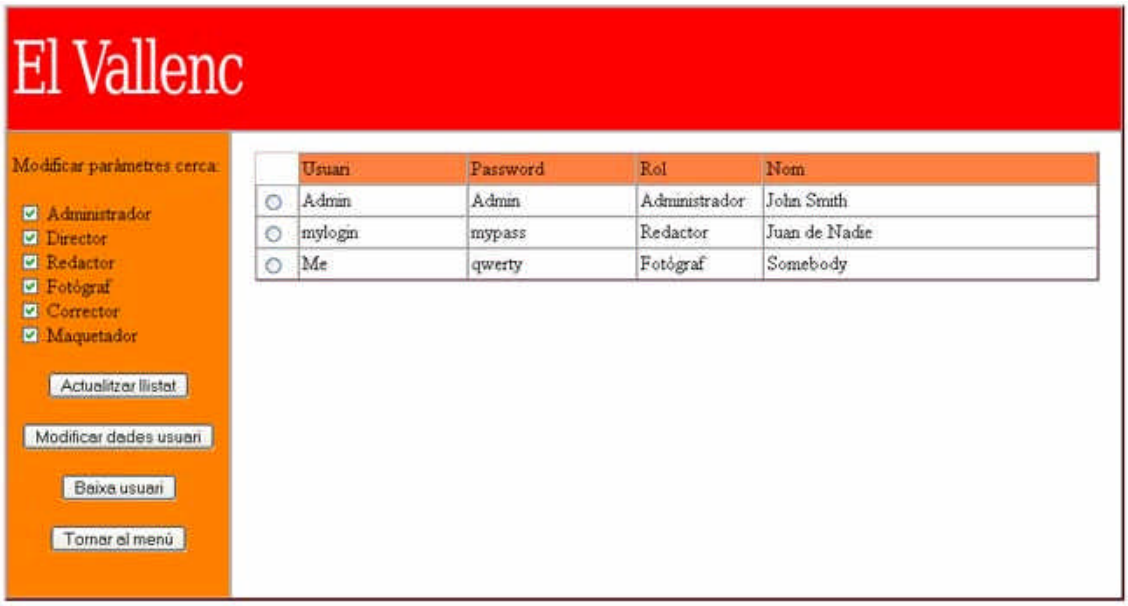
-Menú director,



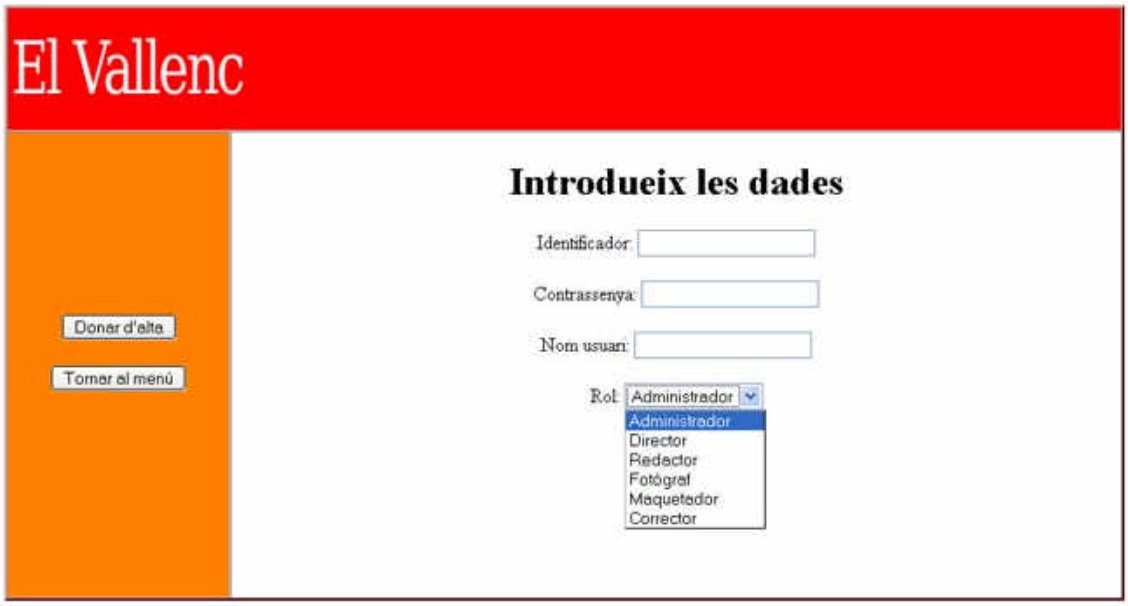
Menú redactor, corrector, fotògraf, redactor



-ListUser



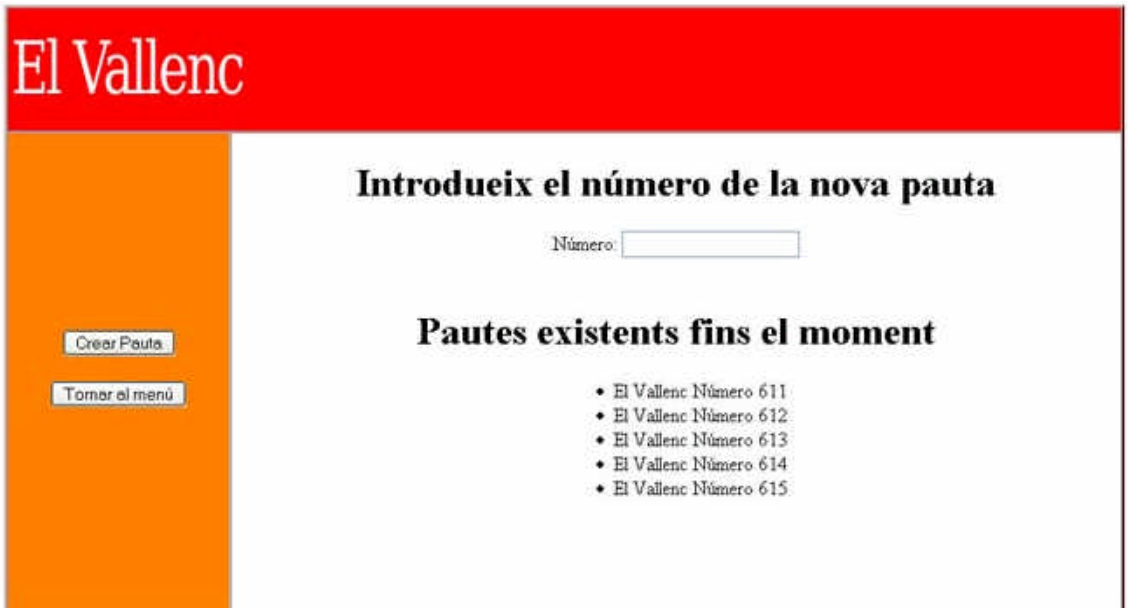
-AddUser



-ModifyUser Administrador

-ModifyUser Redactor, Fotògraf, Director, Corrector, Maquetador

-NewPauta

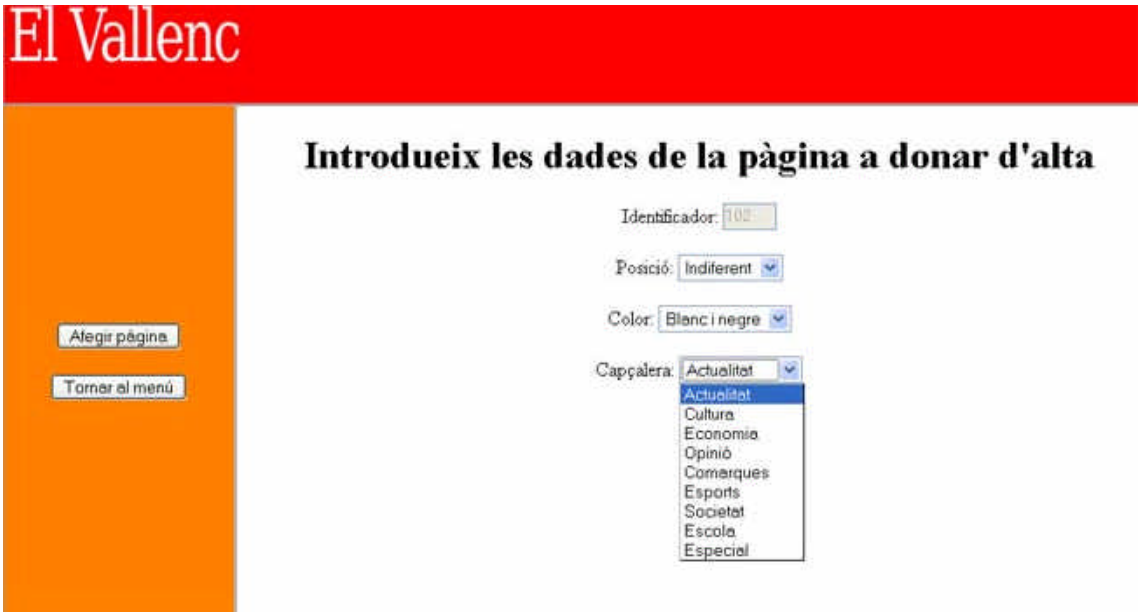


-SheetList.jsp Director

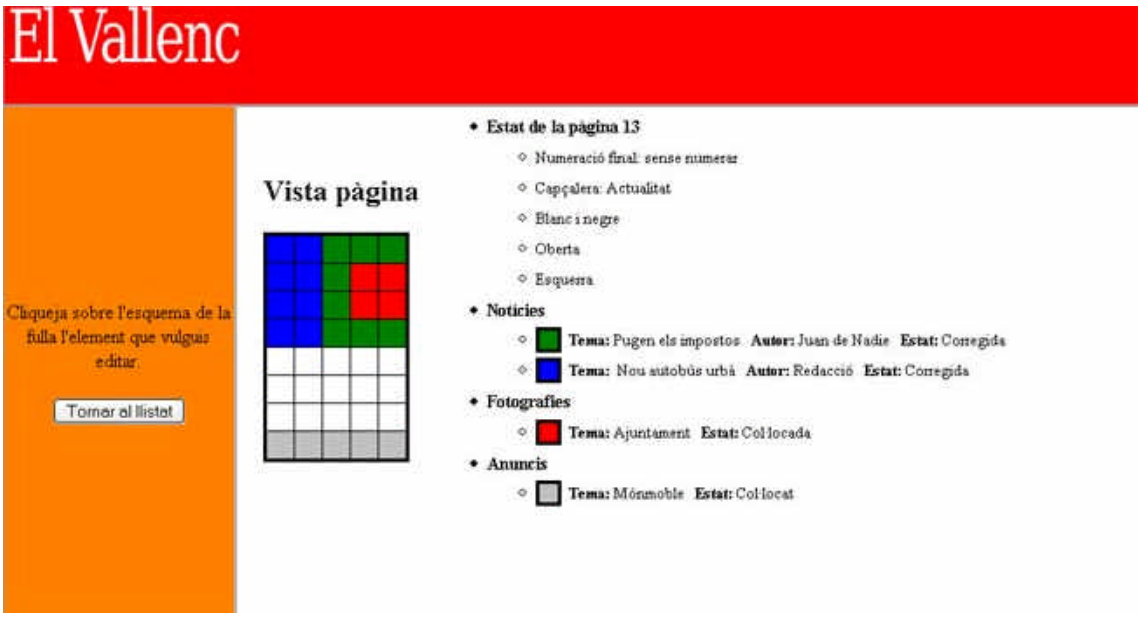
SheetList.jsp Llistat Pàgines Redactor

-SheetList.jsp Llistat Pàgines Fotògraf, Corrector, Maquetador

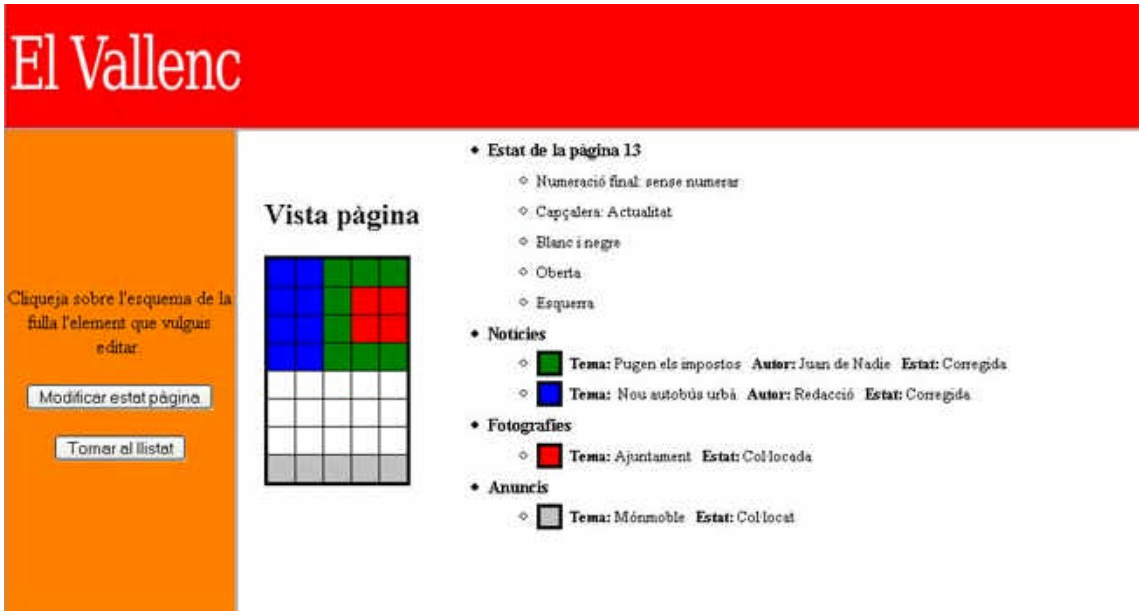
-AddSheet.jsp



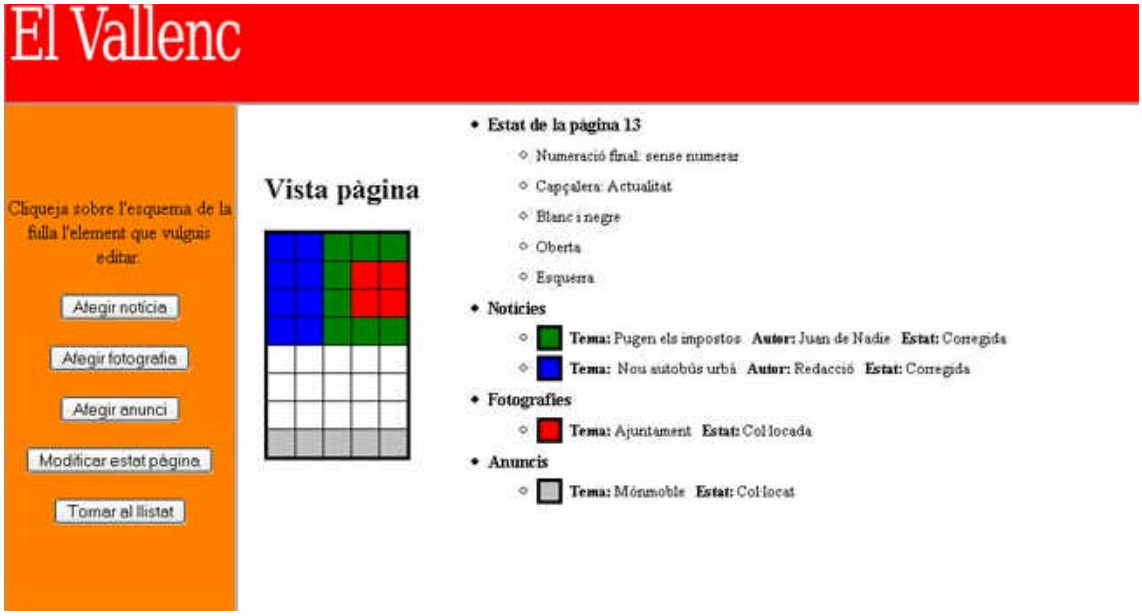
-SheetView.jsp Corrector, Fotògraf



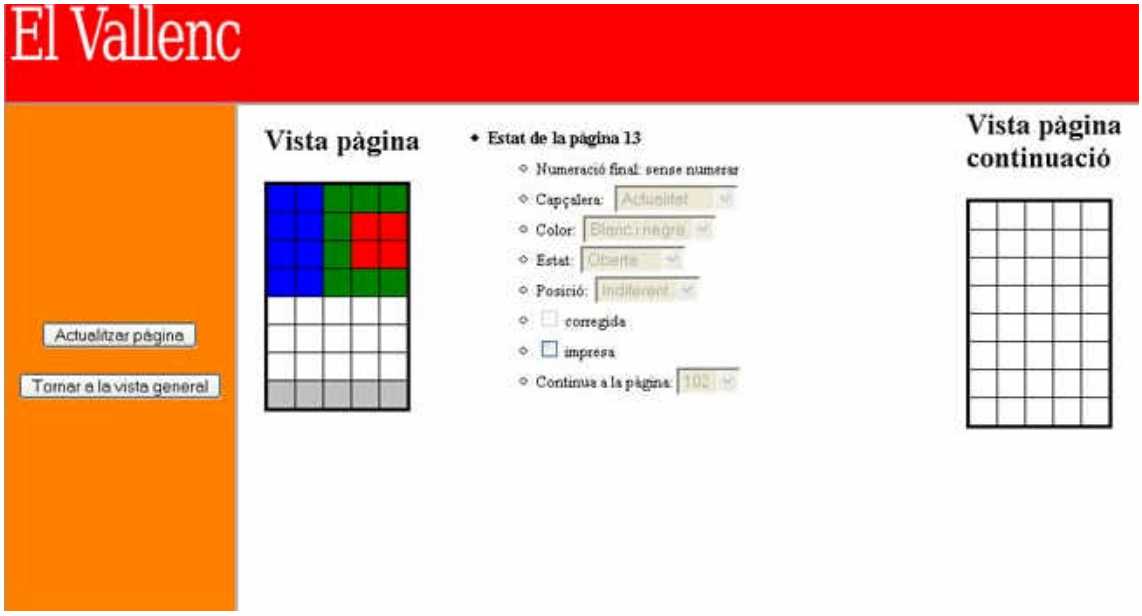
-SheetView.jsp Maquetador



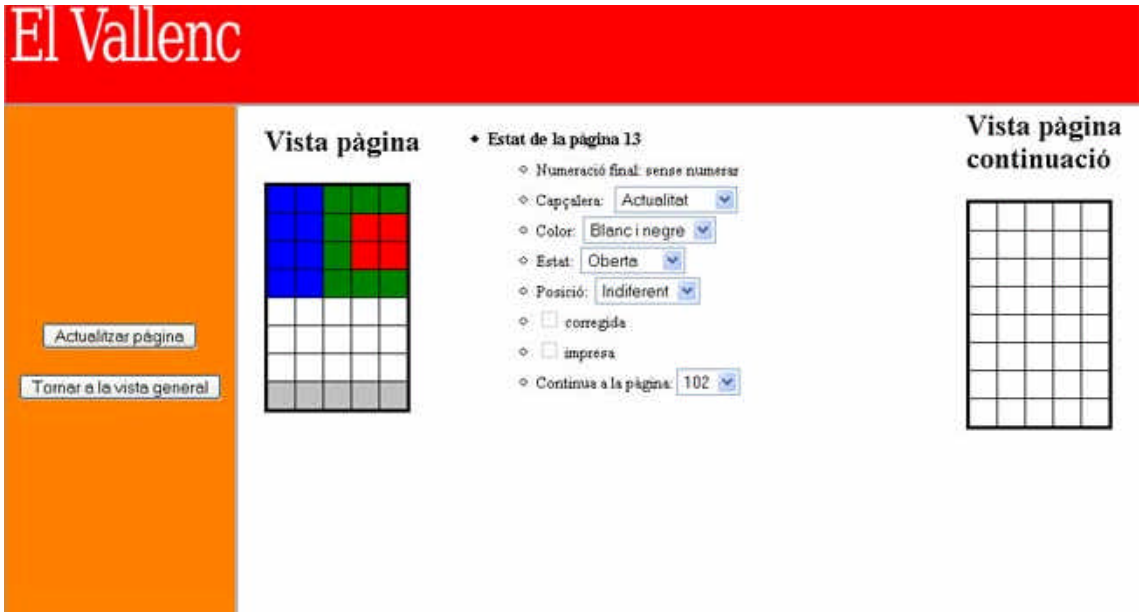
-SheetView.jsp Redactor



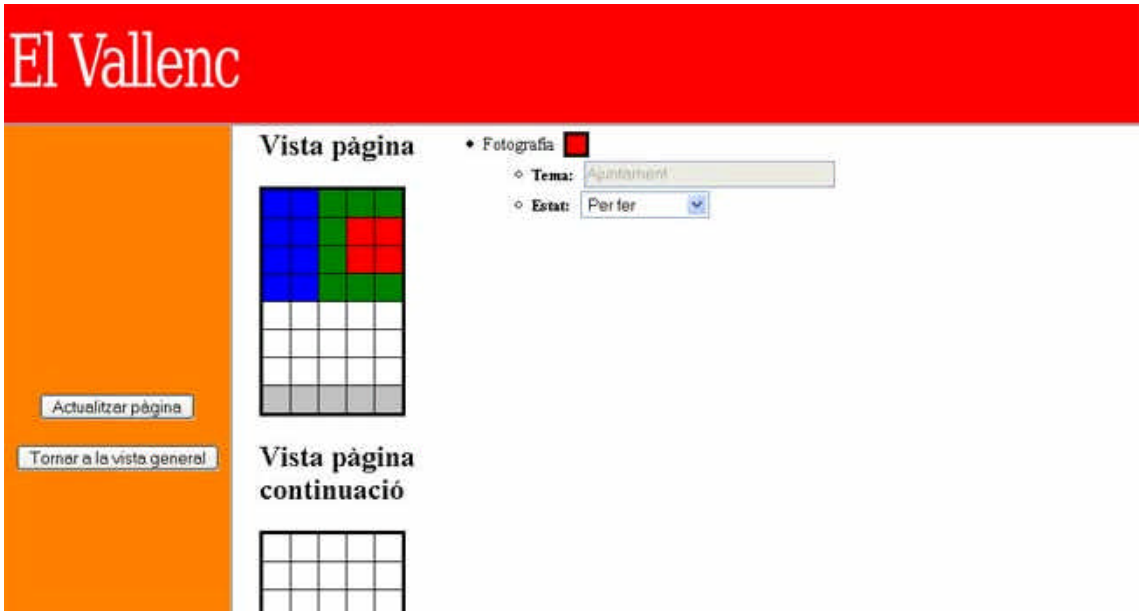
-ModifySheet.jsp Maquetador



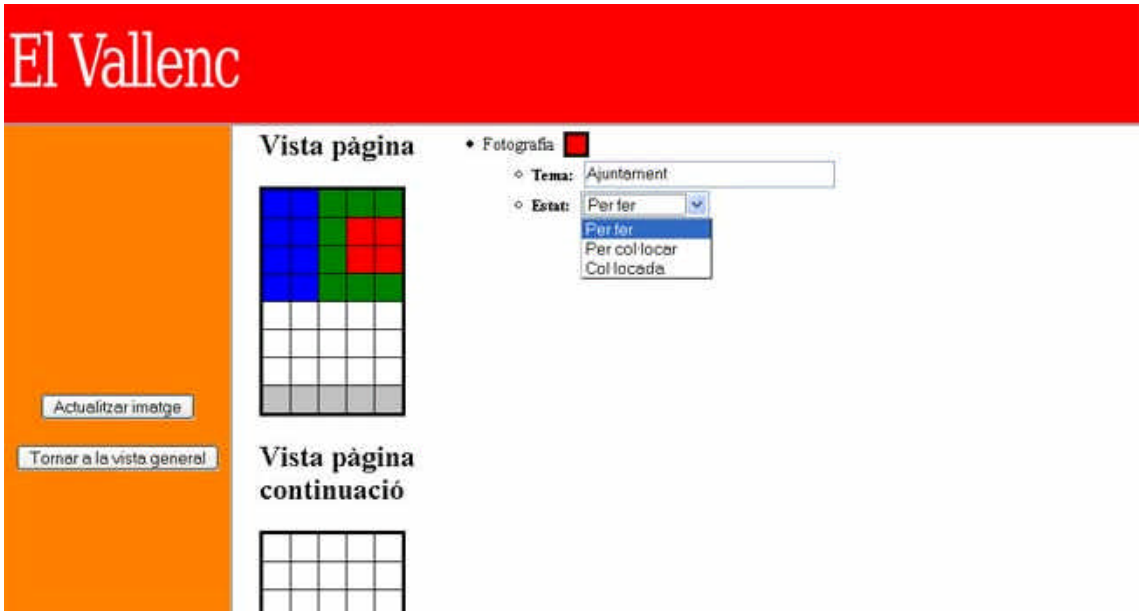
-ModifySheet.jsp Redactor



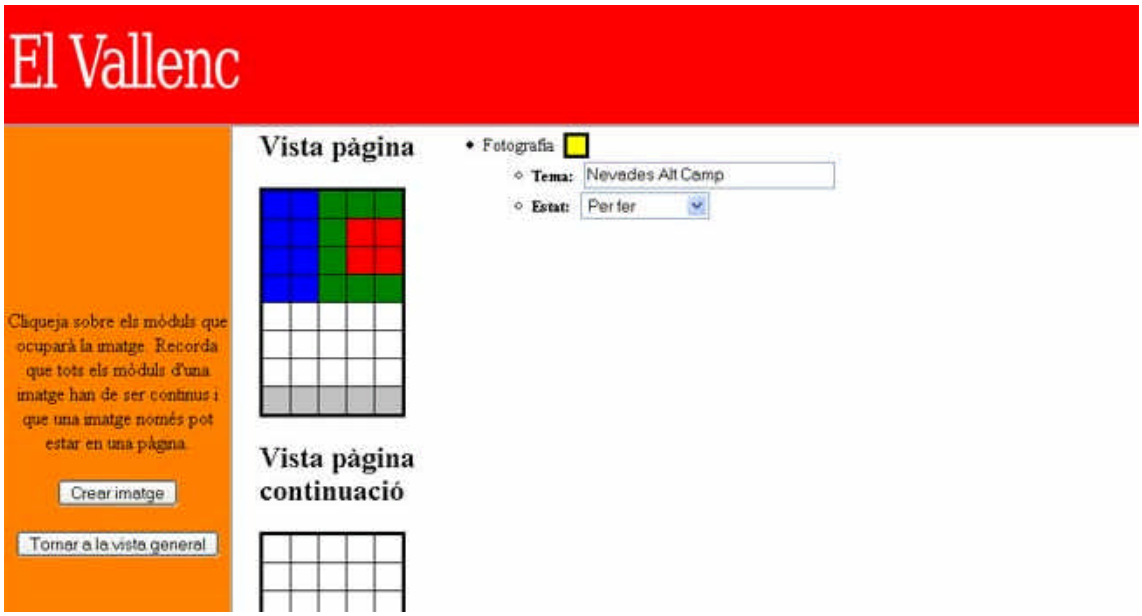
-ModifyImage.jsp Fotògraf, Maquetador



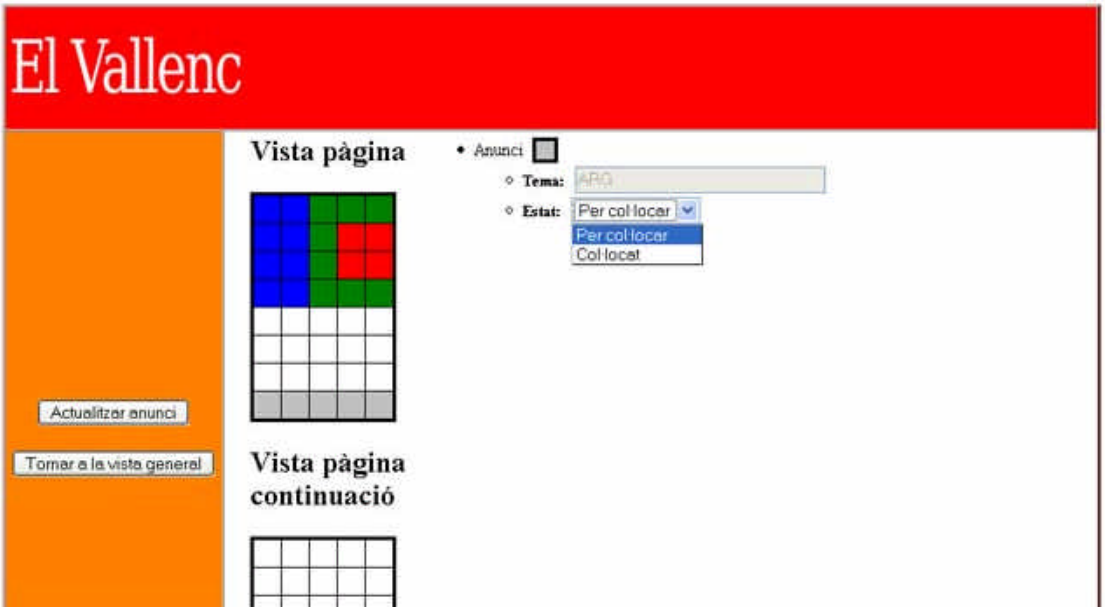
-ModifyImage Redactor



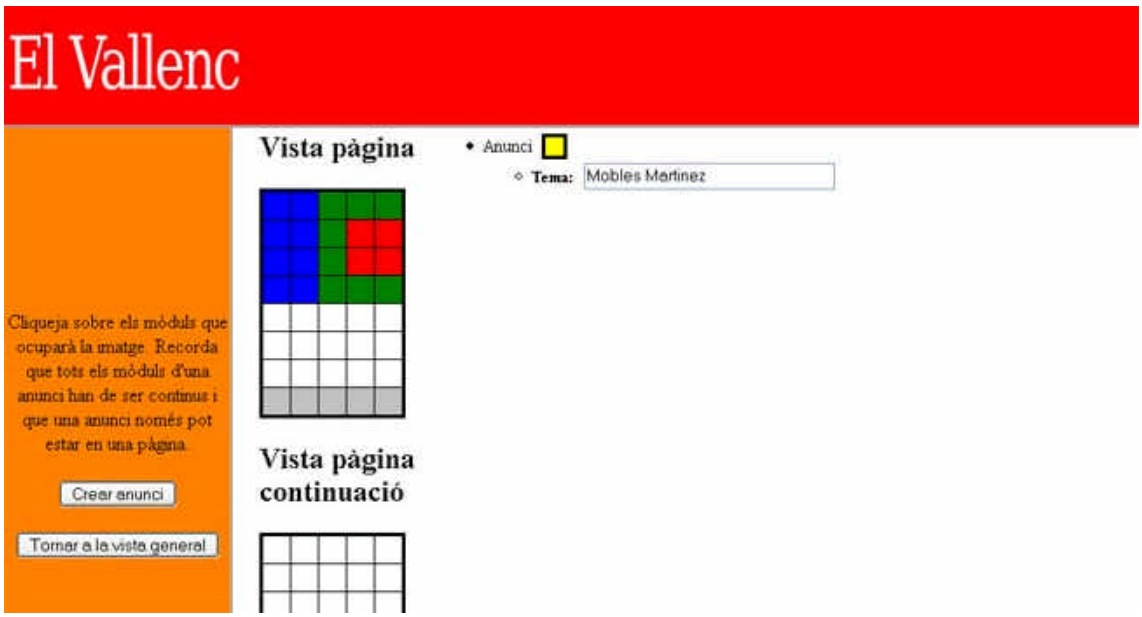
-AddImage.jsp



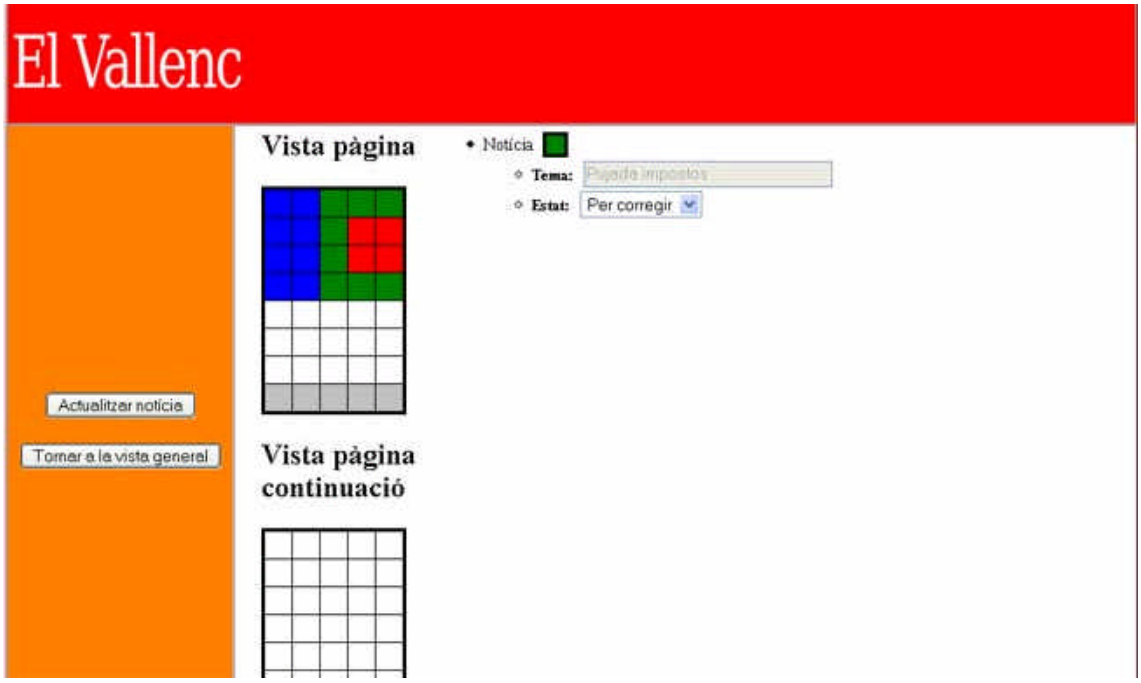
-ModifyAdvert Maquetador



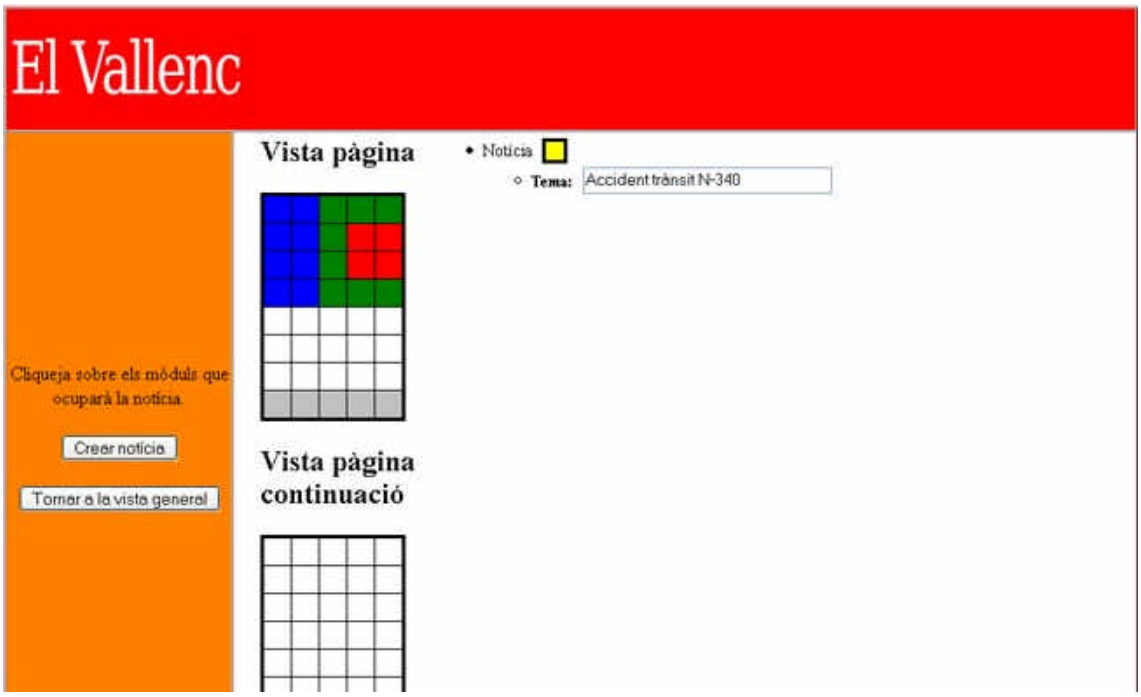
-AddAdvert.jsp



-ModNews.jsp



-AddNews.jsp



-AddminTopic

-Seemessages

Tipus	Usuari	Data	Missatge
Pàgina	Josep Maria	22-03-06	Reservada pàgina 3
Notícia	Josep Maria	22-03-06	Creada notícia "Multes ajuntament" a la pàgina 3
Notícia	Josep Maria	22-03-06	Alliberada pàgina 3

8 - Aplicació web col·laborativa a través de events

Actualment la orientació de d'una web col·laborativa es basava en moltes ocasions en el model MVC.

MVC o Model View Controller és un patró de disseny creat originalment pel llenguatge SmallTalk a la Ingenyeria del Software. El paradigme MVC consisteix en dividir les aplicacions en tres parts:

- Controlador
- Model
- Vistes

El controlador és l'encarregat de redirigir o assignar una aplicació (un model) a cada petició; el controlador ha de posseir d'alguna manera, un mapa de correspondències entre peticions i respostes (aplicacions o model) que sel's assignen.

El model seria l'aplicació que respon a una petició, és la lògica de negociació.
Una vegada realitzades les operacions necessàries el flux torna al controlar i aquest retorna els resultats a una vista assignada.

La vista presenta el model en un format adequat per interactuar a través d'una interfície d'usuari, normalment.

Els principals avantatges d'aquest model són òbviament una separació total entre lògica de negoci i presentació. A això se li pot aplicar opcions com el multillenguatge, diferents dissenys de presentació etc… sense alterar la lògica de negoci.

La separació de capes com presentació, lògica de negoci, accés a dades és fonamental pel desenvolupament d'arquitectures consistents, reutilitzables i més fàcilment mantenibles, la qual cosa al final resulta en un estalvi de temps de denvolupament en posteriors projectes.

8.1 - STRUTS

Struts és un framework per aplicacions web java que implementa el model MVC. Realment el que fa es proveir un conjunt de classes i TAG-LIBS que conformen el Controlador, la integració amb el Model (o lògica de negoci) i faciliten la construcció de vistes.
Naturalment, el Model o Lògica de negoci és la part que ens correspon desenvolupar. Per aquest motiu Struts és una plataforma sobre la que creem la lògica de negoci, i aquesta plataforma ens permet dividir la lògica de la presentació entre d'altres coses.

Naturalment, a banda d'aquestes punts positius existeixen punts negatius. Per exemple, fins i tot l'aplicació més senzilla, necessita una quantitat de fitxers més gran que no pas una aplicació que utilitza un disseny input-output tradicional.

Posarem l'exemple del tradicional "Hello World".

En un disseny web tradicional en HTML tan sols necessitaríem una plana en la que es mostrés el missatge.
En canvi fent servir el model MVC amb STRUTS es necessitaria:

- Una pàgina JSP (la presentació)
- Una classe Action (component del controlador)
- Un fitxer struts-config.xml on es defineix la classe Action.

Com que el fitxer struts-config.xml forma part del disseny bàsic el podem obviar en el còmput. Per tant fixem-nos que necessitem dos fitxers en lloc d'un per a realitzar la mateixa aplicació.

8.2 - El model Clàssic

El model clàssic de treball amb Struts deixa en mans de la part de controlador totes les operacions a realitzar. A través de l'aplicació web es crida a una sèrie d'accions i aquestes, treballant directament sobre el model i la base de dades processen tota la informació. Per a dur a terme aquesta feina les accions en si necessiten comprendre la informació que se'ls està proporcionant i per tant realitzar canvis en el model implica realitzar canvis en el controlador per tal d'entendre la informació que es proporciona.

8.3 - El model amb Events

Per poder parlar del model amb events en primer lloc hem de tenir en compte què és exactament un event.

Un Event és un succés notable que s'esdevé a l'interior o a l'exterior d'un model de negoci. Així doncs un event (de negoci o de sistema) pot significar un problema, una oportunitat, o un desviament en un moment donat.

Un event ha d'estar format per una capçalera i per un cos. La capçalera ha de contenir els elements que descriuen l'esdeveniment en si com és la ID, el tipus d'event, el nom, la data de succés, el número de succés, el creador del succés, ... Aquests elements són comuns a tots els events.

Per altra banda un event ha de tenir un cos que descriu que ha passat, per exemple, en la nostra aplicació, quan una pàgina està tancada i tots els mòduls de notícia estan corregits automàticament es generaria un event per anunciar que la pàgina es pot donar per corregida.

El cos del event hauria de contenir tota la informació necessària per tractar l'esdeveniment. En el mateix cas, per exemple, s'hauria d'informar de l'identificador de la pàgina i el seu estat actual, l'identificador de tots els mòduls, la informació que contenen i el seu estat.

Un dels avantatges un model d'events és que quan un succés notable s'esdevé, tots els interessats són informats automàticament (ja siguin persones o autòmats). D'aquesta manera tots aquells interessats poden reaccionar a un esdeveniment donat: quan s'esdevé un event de tipus "pàgina alliberada" una persona pot decidir reaccionar intentant reservar-la o en l'exemple anterior quan es crea un event "notícia corregida" un sistema central pot escoltar l'event i reaccionar mirant si la pàgina es pot donar com a corregida.

Així doncs veiem en aquest moment una del es principals diferències sobre el sistema tradicional: en el tradicional el mateix actor que generava un event sabia que passava exactament a través de les accions i a més havia d'esperar que aquestes es duguessin a terme, en canvi en un model d'events aquests poden ser tractats de forma totalment asíncrona per la qual cosa l'actor no necessita restar a l'espera que es processi la seva acció.

Existeixen dos tipus d'events:

L'event atòmic: un succés notable que inicialitza una sèrie d'accions i s'utilitza normalment per a realitzar feina en temps real.

L'event complex: un succés o conjunt de successos que es poden donar en un llarg període de temps i que es poden donar de forma casual o provocada i que normalment necessita un anàlisi complex de la situació per tal de prendre decisions. Per exemple serviria per controlar errors que es produeixen en una aplicació. Seria en definitiva un event compost d'altres events simples o complexos al seu torn.

8.4 - El processament d'events

Un flux d'un event s'enceta quan aquest és generat i finalitza amb l'execució d'una activitat particular.

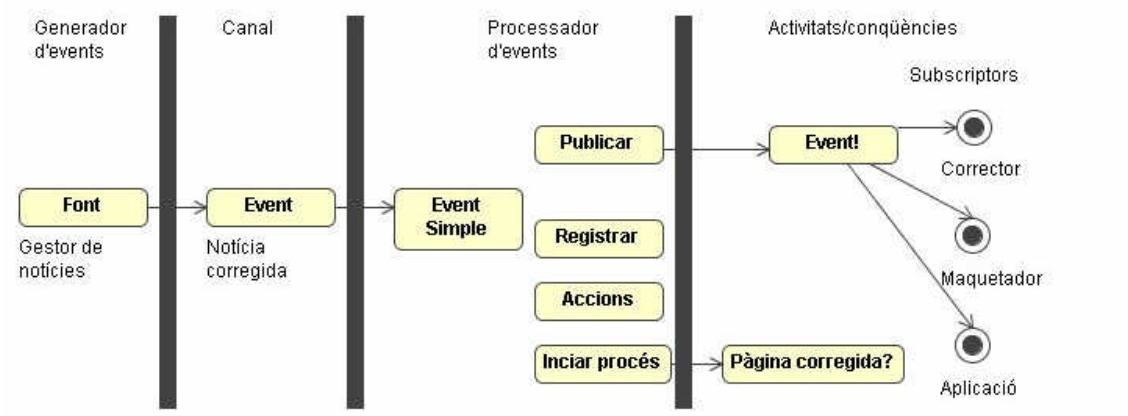
En primer lloc ens trobem amb un generador d'events. Cada event és generat per una font. Aquesta font pot ser una aplicació, un servei, un sensor o fins i tot una eina col·labroativa.

En segon lloc ens trobem amb un canal que ens servirà com a medi de transport entre els generador i els processadors d'events així com els subscriptors a un tipus d'events.

En tercer lloc ens trobem amb el processador d'events. Un event és avaluat per una sèrie de regles i a partir d'aquestes una sèrie de accions són iniciades o es generen nous events. Així doncs aquí novament veiem una diferència respecte el model clàssic. Les regles i les accions a realitzar són definides segons les necessitats dels grups implicats i no pels generadors d'events. D'aquesta manera hi ha un sistema de seguretat addicional degut a que els generadors d'events no tenen cap capacitat de decisió.

Les accions preses poden ser múltiples, inicialitzar un procés de negoci, publicar un nou event en un tema de subscripció, notificar directament a un sistema o a una persona, generar un nou event o fins i tot guardar el event en si per tenir un registre i mantenir un historial.

Exemple d'event atòmic



Ens trobem davant un event simple. Quan algú dóna una notícia per corregida el gestor de noticies ha actualitzat la base de dades genera un event avisant que una notícia ha estat corregida. Aquesta es transmet a través d'un canal i és processada pel processador d'events. Aquests s'encarrega de publicar l'event perquè el puguin llegir els subscriptors interessats, enregistra els canvis a l'historial, realitza qualsevol tipus d'acció addicional que cregui convenient i a més enceta un procés nou que és el de comprovar si ara que la notícia està corregida tota la pàgina està corregida.

8.5 - Els events aplicats al nostre problema

Tal com hem explicat anteriorment la conversió del model clàssic basat únicament en Struts a un model que uneix el millor dels Struts amb els events pot comportar diversos avantatges respecte el primer model.

El més evident és que treballarem d'una forma distribuïda, per part dels clients, però centralitzada en quant a la gestió de dades: el control i la seguretat de tota l'aplicació estarà en mans del processador d'events que validarà els diferents events que es donin a través dels missatges generats.

Un altre dels avantatges és que podrem prioritzar les accions segons la seva importància i fins i tot programar accions per que passin en un moment determinat. Imaginem que un redactor vol fer el seguiment d'una notícia a través de totes les edicions del diari. Fer la selecció d'un tema en concret quan per exemple portes cinc anys gestionats pot fer que el rendiment del sistema baixi notablement. Amb l'ús d'events un usuari pot fer la petició de la consulta i aquesta, per exemple, es podria processar de nit quan ningú treballa al diari.

Un altre cas, i aquest si que resultarà molt útil en el dia a dia, serà la monitorització de l'aplicació. Tornem a posar un exemple. En el model actual si un usuari vol reservar una pàgina i aquesta està ocupada no té més remei que esperar-se i tornar a revisar el seu estat una i una altra vegada fins que aquesta quedi lliure. En canvi amb l'ús dels events pot fer una petició de reserva de pàgina i quan es produeixi l'event d'alliberació de la mateixa, l'usuari pot rebre un avís informant que la pàgina està alliberada. Val a dir que aquesta és una de les principals diferències sobre el sistema clàssic. Tot i que en les accions podien passar moltes coses aquestes succeïen d'una manera seqüencial, en canvi amb el model d'events es poden provocar diversos processos en paral·lel com actualitzar la base de dades, enviar un avís i fins i tot realitzar un control de dades així com guardar un registre del que succeeix per tenir un historial.

8.6 - Com s'aplica el model d'events?

El principal problema de l'aplicació d'aquest nou model és situar el processador d'events. Usualment, quan tractem una aplicació s'acostuma a utilitzar un programa independent, que fins i tot moltes vegades es troba en una màquina independent, un nucli central. Nosaltres estem treballant amb una aplicació que segueix el model MVC i a més es tracta d'una aplicació web.

Tenim dos possibles solucions:

- Seguir l'estil habitual i crear una aplicació independent que faci la feina de processador d'events.
- O bé integrar el processador d'events dins el model MVC. Per exemple podem assignar la tasca d'activar la gestió d'events a un administrador. Aquest podria controlar i fins i tot desactivar en un moment donat la gestió de certes events per tal de tenir control total sobre el processador d'events.

Nosaltres optarem per aquesta darrera opció i per tant integrarem el el processador d'events a l'aplicació directament.

Per tal de controlar els events utilitzarem un sistema de missatgeria. En aquest cas a través de JMS, un MOM (middleware orientat a missatges) que ens permetrà a través la seva tecnologia de cues i temes crear un sistema semi-persistent a través de la pròpia base de dades MySQL que utilitzem per l'aplicació.

Tenint clar aquest concepte el primer pas, i realment el més complex, tal com he dit a l'inici, és la creació del lèxic per els events així com assignar prioritats i tenir en compte no només els events produïts pels propis casos d'ús, sinó els events que es produeixen pròpiament per l'ús de l'aplicació.

Un cop obtingut el lèxic, compostos per tots els possibles events i l'estructura de missatges que generen, (fixem-nos en l'exemple gràfic anterior per tenir una idea del que seria una paraula del lèxic, podrem iniciar la implementació de l'aplicació.

Per tal de poder desenvolupar una aplicació, sigui quina sigui, que es basi en events, un cop ja hem fet el disseny previ i sabem en què consistirà la mateixa s'ha de dissenyar un lèxic robust que contempli totes les possibles paraules (events) que es puguin donar mentre es treballi en l'aplicació.

Existeixen molts tipus de lèxic depenent el grau de control que es vulgui tenir sobre l'aplicació:

- En un dels extrems dels tpus de lèxic podem trobar els més exhaustius en que les paraules del llenguatge són totes elles primitives, no es poden dividir en events més petits, i per qualsevol activitat podem treballar amb sentències (combinacions seqüencials de paraules primitives) o creant paraules derivades que aportin nous significats a les paraules primitives.
- En l'altre extrem podem trobar els lèxics més simples en els que més que paraules partim directament de sentències que normalment es corresponen directament en les diferents accions que es poden dur a terme a l'aplicació.
- Altres lèxics poden trobar-se entre els dos extrems i aquests són els que habitualment ens trobarem a les aplicacions. Aquests lèxics combinen paraules primitives, que normalment es repeteixen i es poden reaprofitar, i paraules més complexes que s'apliquen en casos més concrets i poden ser poc reaprofitables. Aquestes paraules complexes també serien considerades 'primitives' en aquests lèxics però notificarien successos a més alt nivell.

En el disseny d'aquesta aplicació nosaltres optem per aquests darrer cas per ser el que més es pot ajustar a les nostres necessitats. Treballarem com a eina base els diferents casos d'ús del nostre llenguatge i a mida que els analitzem veurem quines paraules sorgeixen de cadascun d'ells.

Per cada event o paraula del nostre llenguatge haurem de tenir en compte quin canal utilitzarà, quin serà el contingut del missatge i la seva estructura, quina serà la feina del processador d'events i quines reaccions produirà.

Disposarem de diversos canals a la nostra aplicació:

- Canal Admin: Será el canal que farem servir per posar-nos en contacte amb el processador d'events
- Canal Log: Servirà per a controlar fàculment qui ha get què. Es tracta d'un canal que es podria eliminar ja que tenim el registre d'events. Però l'utilitzarem per ser més senzill l'obtenció de certa informació.
- Canal Pagina: Servirà per a registrar els events i canvis realitzats a les pàgines (reserves, modificacions, etc…)
- Canal Notícia: Servirà per informar dels canvis a les notícies
- Canal Anunci: Servirà per informar dels canvis als anuncis
- Canal Fotografia: Servirà per informar dels canvis a les ftografies
- Canal propi per cada usuari: com que la quantitat d'usuaris és petita tenir un canal per a cada usuari facilitarà la recepció de missatges per part d'aquest. Ens referirem a aquest canal com "usuari" a partir d'ara.

A més a més dels casos d'ús especificats en la fase prèvia, ens trobarem amb alguns events nous que sorgeixen en aquest moment de l'aplicació gràcies a l'ús dels events i que ens permetran tenir major control sobre el flux del programa. Aquests nous events poden ser provocats per moltes raons però per a fer-nos una idea senzilla posarem un exemple simple: cada quinze minuts el processador d'events generarà un event de control per si mateix en que comprovarà l'estat dels usuaris, si qualsevol usuari ha estat inactiu durant més de mitja hora es farà expirar la sessió del mateix.

9. Lèxic i sintaxi dels events

9.1 - EVENTS

CAS D'ÚS: Login

En arribar a aquest punt comprovem que ens trobem davant un problema. L'administrador serà l'encarregat de posar en marxa el processador d'events i per tal de controlar l'event login necessitem que aquest estigui en marxa. Per tant optarem per crear dos sistemes de login separats: un per l'administrador i un per la resta d'usuaris. El login de l'administrador serà completament controlat per STRUTS i en canvi el login dels usuaris es considerarà un event que ha de ser controlat. Tractarem doncs el segon cas

Event: UserLogin

S'inserta un idusuari i un password:

- Si són correctes i l'usuari no està ja en actiu l'usuari passa a estar actiu.
- Si queda identificat però ja estava actiu es produeix un FatalError avisant del problema
- Si són incorrectes s'informa de l'error

UserLogin[idevent=v, password=x, idusuari=y, usuari=z] => **UserLoggedOn** [cause={v}, idusuari=y, usuari=z] **if** **[authenticat**e (password, idusuari) /\ **!loggedOn**(idusuari)**]** **V** **UserAllreadyLogged** [cause={v}, idusuari=y, usuari=z, error="User allready logged"] **if** **[authenticat**e (password, idusuari) /\ **loggedOn** (idusuari)**]** **V** **UserLoginError** [cause={v}, idusuari=y,usuari=z, error="Bad username or password"] **if** **[!authenticat**e(password,idusuari)**]**

Event: UserLoggedOn

S'activa a la base de dades el flag LoggedIn per l'usuari i es retorna un missatge indicant que el Login ha estat correcte. Tot queda registrat al log.

UserLoggedOn [idevent=u, cause={v}, idusuari=y,usuari=z]=> **LogIn**[cause={v,u}, idusuari=y] /\ **SendMessage** [cause={v,u}, idusuari=y, usuari=z, type="UserLogin", ObjectMessage= **newObjectMessage**(**getUserInfo()**)] /\ **Log** [cause={v,u}, idusuari=y, msg="User has logged in"]

Event: LogIn

Event atòmic final activa a la base de dades el flag LoggedIn per l'usuari.

LogIn[idevent=w, cause={v,u}, idusuari=y]

Event: UserLoginError

Si el login i el password són incorrectes s'avisa a l'usuari i es guarda el registre al log.

UserLoginError [idevent=u, cause={v},idusuari=y,usuari=z, error=p] => **SendError** [cause={v,u}, idusuari=y, error=p] /\ **Log** [cause={v,u}, idusuari=y, error=p]

Event: UserAllreadyLogged

Si l'usuari ja estava identificat, envia un error de multiple login i fa un log out de l'usuari per evitar suplantacions d'identitat o per permetre que l'usuari pugui tornar a entrar en cas que l'error vingui donada degut a que l'usuari havia perdut la sessió per qualsevol motiu

UserAllreadyLogged [idevent=u, cause={v} ,idusuari=y ,usuari=z, error=p]=> **SendError** [cause={v,u}, idusuari=y, error=p] /\ **Log** [cause={v,u}, idusuari=y, error=p] /\ **UserLogout**[cause={v,u}, idusuari=y, usuari=z]

Event: PautaList

Un cop un usuari ha fet el login ha d'escollir amb quina pauta treballarà, per aquest motiu fa una petició de la llista de pautes existents.

PautaList [idevent=u, idusuari=y,usuari=z] => **SendMessage** [cause {u}, idusuari=y, usuari=z, type="PautaList", ObjectMessage= **newObjectMessage** (**getPautaList**())]

CAS D'ÚS: Logout

Event: UserLogout

L'usuari pot abandonar el sistema en qualsevol moment i s'han de dur a terme diferents accions com alliberar les pàgines que té reservades així com anular les reserves realitzades. En cas que es detecti que l'usuari no estava LoggedIn es procedirà a anunciar l'error.

UserLogout[idevent=w, cause={v,u}, idusuari=y, usuari=z] => **UserLoggedOut** [cause={v,u,w},idusuari=y,usuari=z] **if** **[loggedOn** (idusuari)**]** **V** **UserLogoutError** [cause={ v,u,w },idusuari=y,usuari=z, error="User" + z +" not logged in"] **if [!loggedOn** (idusuari)**]**

Event: UserLoggedOut

El logout és correcte. Es marca a la base de dades el logout i es creen els events que alliberaran les pàgines i esborraran les reserves del sistema. A continuació s'informa a l'usuari de l'èxit i es guarda el registre al log.

UserLoggedOut [idevent=s, cause={w,v,u},idusuari=y,usuari=z] => **LogOut**[cause={ v,u,w,s }, idusuari=y] /\ **FreeSheets**[cause={ v,u,w,s }, idusuari=x,usuari=y] /\ **DeleteReservations** **[**cause={ v,u,w,s },idusuari=x,usuari=y] /\ **SendMessage** [cause={ v,u,w,s }, idusuari=y, usuari=z, type="UserLogOut", ObjectMessage= **newObjectMessage** (**trueObject**())]/\ **Log** [cause={v,u}, idusuari=y, msg= "User has logged out"]

Event: LogOut

Event atòmic final desactiva a la base de dades el flag LoggedIn per l'usuari.

LogOut[cause={ v,u,w,s }, idusuari=y]

Event: UserLogoutError

Hi ha hagut un error a l'hora de fer el logout ja que l'usuari no es trobava loggedIn. Avisaem que no s'ha pogut realitzar la comanda i guardem el registre al log.

UserAllreadyLogged [idevent=s, cause={ v,u,w } ,idusuari=y, usuari=z, error=p]=> **SendError** [cause={ v,u,w,s } , idusuari=y , error= p] /\ **Log** [cause={ v,u,w,s }, idusuari=y, error=p]

CAS D'ÚS: Alta Usuari

Event: AltaUsuari

Un usuari demana l'alta d'un altre usuari amb un rol en concret, un id i un password. Si l'identificador ja existeix s'informa de l'error si no es crea el nou usuari.

AltaUsuari [idevent=v, idusuari=x, usuari=y, newidusuari=p, newpassword=q, newusuari=r, newrole=s] => [**NewUser** [cause={v}, idusuari=x, usuari=y ,newidusuari=p, newpassword=q, newusuari=r, newrole=s] **if [!userExist** (newidusuari) /\ **isAdmin**(idusuari)**]** **V** **UserAlreadyExists** [cause={v} idusuari=x, usuari=y ,error="User " + p + " already exists"] **if [userExist** (newidusuari)]**]** **V** **PrivilegeNotHeld** [cause={v},idusuari=x, usuari=y, privilege="admin"] **if [!isAdmin**(idusuari)**]**

Event: NewUser

Es dóna d'alta l'usuari a la base de dades i es crea el seu nou canal també a la base de dades.

NewUser [idevent=u, idusuari=x,usuari=y, cause={v},newidusuari=p, newpassword=q, newusuari=r, newrole=s] => **SendMessage** [cause={ v,u }, idusuari=x, usuari=y, type= "NewUser", ObjectMessage= **newObjectMessage** (**trueObject**())] **if** (**createUser** (newidusuari, newpassword, newusuari, newrole) /\ **createQueue** (idusuari))

Event: UserAllreadyExists

El nou usuari ja existeix per tant avisem a l'usuari creador que no s'ha pogut completar la comanda.

UserAlreadyExists [idevent=u, cause= {v}, idusuari=x, usuari=y, error=p] => **SendError** [cause={ v,u } , idusuari=x, error=p]

CAS D'ÚS: Llistar Usuaris

Event: UserList

Un usuari demana un llistat dels usuaris existents a la base de dades. Aquest event només hauria de ser produït per un administrador. Si l'usuari no és administrador es genera l'error

UserList [idevent=v, idusuari=y,usuari=z] => **SendMessage** [cause={ v }, idusuari=y, usuari=z, type= "NewUser", ObjectMessage= **newObjectMessage** (**getUserList**())] **if** [**isAdmin**(idusuari)] **V** **PrivilegeNotHeld** [cause={v},idusuari=y, usuari=z, privilege="admin"] **if** [**!isAdmin**(idusuari)]

CAS D'ÚS: Baixa usuari

Event: UserDrop

Un usuari demana la petició de la baixa d'un usuari. Aquest event només l'hauria de poder generar un administrador i a més l'usuari ha d'existir. En cas contrari ha de generar diferents errors. Es marca l'usuari com a descartat a la base de dades. L'usuari realment no s'esborra ja que es necessiten les seves dades per qüestió d'historial (les notícies que ell ha fet, les fotografies que ha col·locat, ...)

UserDrop [idevent=v, idusuari=x,usuari=y, dropid=z] => [**UserDiscard** [cause={v}, idusuari=x,usuari=y, dropid=z] **if** [**userExist** (dropid) /\ **isAdmin**(idusuari)**]** **V** **UserNotExists[cause={v}, idusuari=x, usuary=y, error="User " + z + "does not exist"]** **if** [**!userExist** (dropid) /\ **isAdmin**(idusuari)**]** **V** **PrivilegeNotHeld** [cause={v},idusuari=y, usuari=z, privilege="admin"] **if** [**!isAdmin**(idusuari)**]**

Event: UserDiscard

Es marca l'usuari com esborrat i s'envien els missatges apropiats a qui ha generat l'event

UserDiscard [idevent=u, cause={v}, idusuari=x,usuari=y, dropid=z] => [**SendMessage** [cause={ v,u }, idusuari=x, usuari=y, type= "UserDrop", ObjectMessage= **newObjectMessage** (**trueObject**())] /\ **Log** [cause {v,u}, usuari=z, msg="User dropped" + z] **if** [**dropUser**(dropid)**]**

Event: UserNotExists

El nou usuari no existeix per tant s'envia l'error apropiat

UserNotExists [idevent=u, cause= {v}, idusuari=x, usuari=y, error=p] => **SendError** [cause={ v,u } , idusuari=x, error=p]

CAS D'ÚS: Modificació d'usuari

Event: UserModify

Un usuari fa la petició de modificació de dades. La id d'un usuari no es pot canviar i un usuari, a no ser que sigui administrador, només pot canviar les seves pròpies dades tot i que no pot modificar el seu rol

UserModify [idevent=v, idusuari=x,usuari=y, idmoduser=p, newpassword=q, newusuari=r, newrole=s] => [**UserMod** [cause={v}, idmoduser=p, newpassword=q, newusuari=r, newrole=s] /\ **SendMessage** [cause={ v,u }, idusuari=x, usuari=y, type= "UserModify", ObjectMessage= **newObjectMessage** (**trueObject**())] **if [** (**userExist** (idmoduser) /\ (**isAdmin**(idusuari)) **V** (**!isAdmin**(idusuari) /\ (idmoduser=idusuari) /\ (**getRole(**idmoduser**)**==newrole) **]** **V** **UserNotExists[cause= {v}, idusuari=x, usuary=y, error="User " + z + "does not exist"] if [!userExist** (newidusuari)**] V PrivilegeNotHeld** [cause={v},idusuari=y, usuari=z, privilege="Modify user data"] **if [!isAdmin**(idusuari) /\ (**(**idmoduser!=idusuari) V (**getRole(**idmoduser**)**!=newrole))**]**

Event: UserMod

Event atòmic final que actualitza les dades d'un usuari

UserMod [idevent=u, cause={v}, idmoduser=p, newpassword=q, newusuari=r, newrole=s]

CAS D'ÚS: Llistar Pàgines

Event: SheetList

Un usuari demana la petició de la llista de pàgines amb les diferents opcions de cerca. Un objecte de la classe options estarà format pels camps :

<estatPagina>	oberta, tancada, totes
<color>	blanc i negre, color, totes
<posicio>	dreta, esquerra, totes
<estatSecundariPagina>	no corregides, corregides, impreses, totes
<capçalera>	actualitat, comarques, opinió, esports, cultura, ….
<veureNoticies>	true / false
<estatNoticies>	per corregir, corregides, totes
<veureFotografies>	true/false
<estatFotografies>	per fer, per col·locar, col·locades, totes
<veureAnuncis>	true / false
<estat anuncis>	per col·locar, col·locats, tots

Per tant l'event quedarà de la següent manera :

SheetList [idevent=v, idusuari=x,usuari=y, idpauta=z, options=o] => **SendMessage** [cause={ v }, idusuari=x, usuari=y, type= "UserModify", ObjectMessage= **newObjectMessage** (**getSheetList**(options))]

CAS D'ÚS: Visualitzar Pàgina

Event: SheetView

Un usuari fa la petició de visualitzar una pàgina i es comprova si està lliure o està reservada per algun usuari. Recordem que només pot haver-hi una reserva per pàgina tot i que un usuari pot tenir diverses reserves.

Si la pàgina està lliure i no està reservada l'ocupa per l'usuari i li envia la informació
Si la pàgina està ocupada per l'usuari que fa la petició (vol actualitzar la informació de la pàgina)
Si la pàgina està lliure i està reserverda per l'usuari , l'ocupa per l'usuari i esborra la reserva
Si la pàgina no està lliure i no està reservada informa a l'usuari per tal que pugui fer una reserva
Si la pàgina no està lliure i no està reservada informa a l'usuari
Si la pàgina està lliure però està reservada informa a l'usuari.

SheetView [idevent=v, idusuari=x,usuari=y, idpauta=z, pagina=p] =>
SendMessage [cause={ v }, idusuari=x, usuari=y, type= "SheetInfo", ObjectMessage= **newObjectMessage** (**sheetInfo**(idpauta,pagina))] **if** {
[**isSheetFree** (idpauta,pagina) ∧ !**isSheetReserved** (idpauta,pagina)] **V**
[isSheetOcuppiedBy (idusuari, idpauta,pagina)] } **V** {
SendMessage [cause={ v }, idusuari=x, usuari=y, type= "SheetInfo", ObjectMessage= **newObjectMessage** (**sheetInfo** (idpauta,pagina))] ∧ **SheetBusy**[cause={v}, idusuari=x, usuari=y, idpauta=z ,pagina=p, true] ∧ **EraseReservation** [cause={v}, idusuari=x ,usuari=y, idpauta=z ,pagina=p]] **if** [**isSheetFree** (idpauta,pagina) ∧ **hasSheetReserved** (idusuari, idpauta, pagina)] **V**
SheetViewError [cause={v}, idusuari=x,usuari=y, idpauta=z, pagina=p, error="Sheet is busy"] **if [!** **isSheetFree** (idpauta, pagina) ∧ **!isSheetReserved** (idpauta, pagina) **] V**
SheetViewError [cause={v}, idusuari=x,usuari=y, idpauta=z, pagina=p, error="Sheet is busy and reserverd"] **if [!** **isSheetFree** (idpauta, pagina) ∧ **isSheetReserved** (idpauta ,pagina) **] V**
SheetViewError [cause={v}, idusuari=x,usuari=y, idpauta=z, pagina=p, error="Sheet is reserved"] **if** [**isSheetFree**(idpauta,pagina) ∧ (**isSheetReserved**(idpauta,pagina) ∧ !**hasSheetReserved** (idusuari, idpauta,pagina))]

Event: SheetBusy

Quan s'ocupa una pàgina o s'allibera una pàgina es marca el fet a la base de dades i s'avisa del canvi al canal de notícies.

sheetTopic és el canal de missatges sobre canvis a les pàgines

SheetBusy[idevent=u, cause={v}, idusuari=x, usuari=y, idpauta=z ,pagina=p, estat=q] => **PublishMessageInTopic** [cause {v,u}, topic="sheetTopic", "La pàgina "+p+ " de la pauta " + z + " ha estat alliberada"] **if** [**changeBusyState** (idpauta,pagina, estat) /\ (estat=false)] **V** **PublishMessageInTopic** [cause {u,v}, topic="sheetTopic", "La pàgina "+p+ " de la pauta " + z + " ha estat ocupada"] **if** [**changeBusyState** (idpauta,pagina, estat) /\ (estat=true)]

Event: EraseReservation

S'esborra la reserva d'una pàgina i s'anuncia de l'event pel canal de notícies apropiat.

sheetTopic és el canal de missatges sobre canvis a les pàgines

EraseReservation [idevent=u, cause={v}, idusuari=x ,usuari=y, idpauta=z ,pagina=p] => **PublishMessageInTopic** [cause {v,u}, topic="sheetTopic", "La pàgina "+p+ " de la pauta " + z + " ja no està reservada"] **if** [**deleteReservation** (idpauta,pagina,estat)]

Event: SheetReserve

Quan una pàgina està ocupada i no existeix una reserva feta, un usuari pot intentar reservar-la. Cal tenir en compte que en el temps que un usuari fa la petició, un altre usuari es pot haver avançat i haver-la fet abans.

sheetTopic és el canal de missatges sobre canvis a les pàgines

SheetReserve [idevent=u, idusuari=x,usuari=y, idpauta=z, pagina=p] => [**CreateReservation** [cause={u}, idusuari=x, usuari=y, idpauta=z, pagina=p] **if** [!**isSheetReserved** (idpauta,pagina)] V **SheetAllreadyReservedError** [cause={v} idusuari=x ,usuari=y, idpauta=z, pagina=p, error="La pàgina ja ha estat reservada anteriorment"] **if** [**isSheetReserved** (idpauta,pagina)]

Event: CreateReservation

Es crea una nova reserva per una pàgina. S'hi s'ha pogut completar l'operació s'anuncia a l'usuari i es publica l'event al canal de notícies apropiat

sheetTopic és el cnal de missatges sobre canvisa les pàgines

CreateReservation [idevent=u, cause={v}, idusuari=x, usuari=y, idpauta=z, pagina=p] => { **PublishMessageInTopic** [cause {v,u}, topic="sheetTopic", "La pàgina "+p+ " de la pauta " + z + " ha estat reservada"] /\ **SendMessage** [cause={ v,u }, idus[ari=x, usuari=y, type= "UserDrop", ObjectMessage= **newObjectMessage** (**trueObject**())] } if [**newReservation** (idusuari,idpauta, pagina)]

Event: SheetViewError

S'avisa a l'usuari que hi ha hagut algun problema a l'hora de visualitzar una pàgina

SheetViewError [idevent=u, cause={v}, idusuari=x,usuari=y, idpauta=z, pagina=p, error=q]
=> **SendError** [cause={ v,u } , idusuari=x, error=q]

Event: SheetAllreadyReservedError

S'avisa a l'usuari que la pàgina ja ha estat reservada anteriorment

SheetAllreadyReservedError [cause={v} idusuari=x ,usuari=y, idpauta=z, pagina=p, error=q]
=> **SendError** [cause={ v,u } , idusuari=x, error=q]

CAS D'ÚS: Alliberar pàgina

Event: FreeSheet

Quan un usuari deixa de treballar en una pàgina aquesta s'ha d'alliberar. Es comprova que la pàgina realment estigui ocupada per l'usuari en qüestió i llavors s'allibera avisant pel canal pàgina i es guarda la informació al log. En cas contrari es genera un error Fatal.

sheetTopic és el canal de missatges sobre canvisa les pàgines

FreeSheet [idevent=v, idusuari=x, usuari=y, idpauta=z, pagina=p] => [**SheetBusy**[cause={v}, idusuari=x, usuari=y, idpauta=z ,pagina=p, true] /\ **SendMessage** [cause={ v }, idusuari=x, usuari=y, type= "FreeSheet", ObjectMessage= **newObjectMessage** (**trueObject**())] **if** [**isSheetOcuppiedBy** (idusuari, idpauta,pagina)] **V** **FreeSheetError** [cause={ v }, idusuari=x, usuari=y, idpauta=z, pagina=p, error="No tens ocupada la pàgina " +pagina+" de la pauta " +idpauta] if [!**isSheetOcuppiedBy** (idusuari, idpauta,pagina)]

Event: FreeSheetError

S'avisa a l'usuari que la pàgina ja ha estat reservada anteriorment

FreeSheetError [idevent=u, cause={ v }, idusuari=x ,usuari=y, idpauta=z, pagina=p, error=q] => **SendError** [cause={ v,u } , idusuari=x, error=q]

CAS D'ÚS: ModificarEstatArticle

Event: NewsModify

Un usuari pot demanar fer canvis en un article. Si es tracta d'un redactor aquest pot canviar el tema i si està corregida o no (perquè pot haver fet canvis d'última hora sobre una notícia corregida) si la notícia és seva. Si es tracta d'un corrector aquest pot donar la notícia per corregida o no i generar un event de correcció de pàgina. Si no es compleix cap dels requisits anteriors no es canvia res a la pàgina. Cal recordar que si es realitza un canvi en un article la pàgina ha de quedar marcada com a no impresa i anunciar que hi ha hagut canvis. Els canvis a més només s'han de realitzar si la pàgina estava ocupada per l'usuari i si l'objecte, és clar, existeix i és una notícia.

NewsModify [idevent=v, idusuari=x, usuari=y, idpauta=z, pagina=p, idarticle=q, tema=r, corregit=s] => [**NewsTopicChange** [cause={v}, idusuari=x, usuari=y, idpauta=z, pagina=p, idarticle=q, tema=r] /\ **SendMessage** [cause={ v,u }, idusuari=x, usuari=y, type= "UserModify", ObjectMessage= **newObjectMessage** (**trueObject**())]] **if [isEditor**(idusuari) /\ **isNewsDoneBy** (idpauta, pagina, idarticle,idusuari) /\ **isSheetOcuppiedBy** (idusuari, idpauta,pagina) /\ **isNews**(idarticle)] **V**

[NewsRevisedChange [cause={v}, idusuari=x, usuari=y, idpauta=z, pagina=p, idarticle=q, corregit=s] /\ **SendMessage** [cause={ v,u }, idusuari=x, usuari=y, type= "UserModify", ObjectMessage= **newObjectMessage** (**trueObject**())]] **if [[isSpeller**(idusuari) V **isEditor**(idusuari)] /\ **isSheetOcuppiedBy** (idusuari, idpauta,pagina) /\ **isNews**(idarticle)**] V**

PrivilegeNotHeld [cause={v},idusuari=x, usuari=y, privilege="NewsModify"] **if {[[!isSpeller**(idusuari) V **!isEditor**(idusuari)] **V !isSheetOcuppiedBy** (idusuari, idpauta,pagina) V **!isNewsDoneBy** (idpauta, pagina, idarticle,idusuari)] /\ **isNews**(idarticle)**} V**

IsNotNewsError [cause={v}. idevent=v, idusuari=x, usuari=y, idpauta=z, pagina=p, idarticle=q, error="L'element no és una notícia"] if [!**isNews**(idarticle)**]**

Event: NewsTopicChange

Event que canvia el tema d'una notícia. Un cop realitzat els canvis avisa d'aquests pel canal adequat i avisa a qui ha generat originalment l'event que els canvis són correctes.

newsTopic és el canal de missatges sobre canvisa les notícies

NewsTopicChange [idevent=u, cause={v}, idusuari=x, usuari=y, idpauta=z, pagina=p, idarticle=q, tema=r] => **PublishMessageInTopic** [cause {v,u}, "newsTopic" , "L'usuari "+usuari+ " ha canviat el tema de la seva notícia a la pàgina "+pagina+ " de la pauta "+idpauta+ " a "+ tema]
/\ **SheetPrinted** [cause={v,u}, idusuari=x, usuari=y, idpauta=z ,pagina=p, false] "] **if [newTopic** (idpauta, pagina, idarticle, tema)]

Event: NewsRevisedChange

Event que canvia el tema d'una notícia. Un cop realitzat els canvis avisa d'aquests pel canal adequat i avisa a qui ha generat originalment l'event que els canvis són correctes. Si la notícia ha estat donada per corregida es comprovarà si tota la pàgina es pot donar per corregida

newsTopic és el canal de missatges sobre canvisa les notícies

NewsRevisedChange [idevent=u, cause={v}, idusuari=x, usuari=y, idpauta=z, pagina=p, idarticle=q, corregit=s] => **PublishMessageInTopic** [cause {v,u}, "newsTopic" , "L'usuari "+usuari+ " ha donat per corregida la notícia " +idarticle+ " a la pàgina "+pagina+ " de la pauta "+idpauta]
/\ **SheetPrinted** [cause={v,u}, idusuari=x, usuari=y, idpauta=z ,pagina=p, false] /\ **CheckSheetRevised** [cause={v,u}, idusuari=x, usuari=y, idpauta=z, pagina=p]] **if [changeRevision (**idpauta, pagina, idarticle, corregit) /\ (corregit==true]] **V**
PublishMessageInTopic [cause {v,u}, "newsTopic" , "L'usuari "+usuari+ " ha donat com a no corregida la notícia " +idarticle+ " a la pàgina "+pagina+ " de la pauta "+idpauta]
/\ **SheetPrinted** [cause={v}, idusuari=x, usuari=y, idpauta=z ,pagina=p, false] /\ **SetSheetRevised**[cause={v,u}, idusuari=x, usuari=y, idpauta=z, pagina=p,false]] **if [changeRevision (**idpauta, pagina, idarticle, corregit) /\ (corregit==false]]

Event: SheetPrinted

Es canvia l'estat d'impressió d'una pàgina i s'avisa dels canvis pel canal adequat

sheetTopic és el canal de missatges sobre canvisa les pàgines

SheetPrinted [idevent=w, cause={v,u}, idusuari=x, usuari=y, idpauta=z ,pagina=p, printed=q] => { **PublishMessageInTopic** [cause {v,u}, "sheetTopic" , "L'usuari "+usuari+ " ha donat per impresa la pàgina "+pagina+ " de la pauta "+idpauta] **if [changePrinted (**idpauta, pagina, printed) /\ (printed==true]] **V**
PublishMessageInTopic [cause {v,u}, "sheetTopic" , "L'usuari "+usuari+ " ha donat per no impresa la pàgina "+pagina+ " de la pauta "+idpauta] **if [changePrinted (**idpauta, pagina, printed) /\ (printed==false)]]}

Event: SetSheetRevised

Es canvia l'estat de corecció d'una pàgina i s'avisa dels canvis pel canal adequat

sheetTopic és el canal de missatges sobre canvisa les pàgines

SetSheetRevised[idevent=w, cause={v,u}, idusuari=x, usuari=y, idpauta=z ,pagina=p, revised=q] => { **PublishMessageInTopic** [cause {v,u,w}, "sheetTopic" , "L'usuari "+usuari+ " ha donat per corregida la pàgina "+pagina+ " de la pauta "+idpauta] **if [changeRevised (**idpauta, pagina, revised) /\ (revised==true]] **V**
PublishMessageInTopic [cause {v,u,w}, "sheetTopic" , "L'usuari "+usuari+ " ha donat per no corregida la pàgina "+pagina+ " de la pauta "+idpauta] **if [changeRevised (**idpauta, pagina, revised) /\ (revised==false)]]}

Event: IsNotNewsError

S'avisa a l'usuari que l'objecte en qüestió no és una notícia

IsNotNewsError [idevent=u, cause={ v }, idusuari=x ,usuari=y, idpauta=z, pagina=p, idarticle=q , error=r] => **SendError** [cause={ v,u } , idusuari=x, error=r]

CAS D'ÚS: ModificarEstatFotografia

Event: PictureModify

Un usuari pot demanar fer canvis en una fotografia. Tan sols els fotògrafs, el redactors i el maquetadors tenen permís per fer aquests canvis. El canvi de tema d'una fotografia no és permès ja que es canviar completament una fotografia per una altra i per tant s'haurà de fer a través d'esborrar la fotografia i crear-ne una de nova. Els canvis, a més, només s'han de realitzar si la pàgina estava ocupada per l'usuari i si l'objecte existeix i es tracta d'una fotografia.
Al finalitzar es torna a visualitzar la pàgina

PictureModify [idevent=v, idusuari=x, usuari=y, idpauta=z, pagina=p, idfotografia=q, state=r] => {**PictureStateChange** [cause={v}, idusuari=x, usuari=y, idpauta=z, pagina=p, idfotografia=q, state=r] /\ **SendMessage** [cause={ v,u }, idusuari=x, usuari=y, type= "UserModify", ObjectMessage= **newObjectMessage** (**trueObject**())]} **if** **[**[**isEditor**(idusuari) **V** **isPhotographer**(idusuari) **V** **isPrinter**(idusuari)] /\ **isSheetOcuppiedBy** (idusuari, idpauta,pagina) /\ **isPicture**(idfotografia) **]** **}** **V**

PrivilegeNotHeld [cause={v},idusuari=x, usuari=y, privilege="PictureModify"] **if** **[**[**!isPhotographer**(idusuari) V **!isEditor**(idusuari) V **!isPrinter**(idusuari) **V** **!isSheetOcuppiedBy** (idusuari, idpauta,pagina)] /\ **isPicture**(idfotografia)**}** **V**

IsNotNewsError **[**cause={v}. idevent=v, idusuari=x, usuari=y, idpauta=z, pagina=p, idfotografia=q, error="L'element no és una fotografia"] if [!**isPicture**(idfotografia)**]**

Event: PictureStateChange

Event que canvia l'estat d'una fotografia. Un cop realitzat els canvis avisa d'aquests pel canal adequat i avisa a qui ha generat originalment l'event que els canvis són correctes. Realitzar un canvo en una fotografia implica donar la pàgina com a no impresa.

pictureTopic és el canal de missatges sobre canvis a les fotografies

PictureStateChange [idevent=u, cause={v}, idusuari=x, usuari=y, idpauta=z, pagina=p, idfotografia=q, state=r] => **PublishMessageInTopic** [cause {v,u}, "pictureTopic" , "L'usuari "+usuari+ " ha donat per col·locada la fotografia " +idfotografia+ " a la pàgina "+pagina+ " de la pauta "+idpauta] /\ **SheetPrinted** [cause={v,u}, idusuari=x, usuari=y, idpauta=z ,pagina=p, false] **if** **[changePictureInPlace (**idpauta, pagina, idfotografia, state) /\ (state==true]] **V**

PublishMessageInTopic [cause {v,u}, "pictureTopic" , "L'usuari "+usuari+ " ha donat com a no col·locada la fotografia " +idfotografia+ " a la pàgina "+pagina+ " de la pauta "+idpauta]

/\ **SheetPrinted** [cause={v}, idusuari=x, usuari=y, idpauta=z ,pagina=p, false]] **if** **[changePictureInPlace (**idpauta, pagina, idfotografia, state) /\ (state==false]]

Event: IsNotPictureError

S'avisa a l'usuari que l'objecte en qüestió no és una fotografia

IsNotPictureError [idevent=u, cause={ v }, idusuari=x ,usuari=y, idpauta=z, pagina=p, idfotografia=q , error=r] => **SendError** [cause={ v,u } , idusuari=x, error=r]

CAS D'ÚS: ModificarEstatAnunci

Event: AdvertModify

Un usuari pot demanar fer canvis en un anunci. Tan sols els maquetadors tenen permís per fer aquests canvis. Si es fa un canvi ens hem d'assegurar que la pàgina passa a estar no impresa i anunciar els canvis a la pàgina pel seu topic pertinent. El canvi de tema d'una anunci no és permès ja que es canviar completament aquest per una altra i per tant s'haurà de fer a través d'esborrar l'anunci i crear-ne un de nou.. A més els canvis s'han de fer només si la pàgina estava ocupada per l'usuari i si l'objecte existeix i es tracta d'un anunci.

AdvertModify [idevent=v, idusuari=x, usuari=y, idpauta=z, pagina=p, idadvert=q, state=r] => {**AdvertStateChange** [cause={v}, idusuari=x, usuari=y, idpauta=z, pagina=p, idadvert=q, state=r] /\ **SendMessage** [cause={ v,u }, idusuari=x, usuari=y, type= "UserModify", ObjectMessage= **newObjectMessage** (**trueObject**())] **if** **[**[**isPrinter**(idusuari)] /\ **isSheetOcuppiedBy** (idusuari, idpauta,pagina) /\ **isAdvert**(idadvert) **]** **}** **V**
PrivilegeNotHeld [cause={v},idusuari=x, usuari=y, privilege="AdvertModify"] **if** **[[!** **isPrinter**(idusuari) **V** **!isSheetOcuppiedBy** (idusuari, idpauta,pagina)] /\ **isAdvert**(idfotografia)]**}** **V**

IsNotAdvertError **[**cause={v}. idevent=v, idusuari=x, usuari=y, idpauta=z, pagina=p, idadvert=q, error="L'element no és un anunci"] if [!**isAdvert**(idadvert)**]**

Event: AdvertStateChange

Event que canvia l'estat d'un anunci. Un cop realitzat els canvis avisa d'aquests pel canal adequat i avisa a qui ha generat originalment l'event que els canvis són correctes. Realitzar un canvi en un anunci. implica donar la pàgina com a no impresa.

advertTopic és el canal de missatges sobre canvis als anuncis

AdvertStateChange [idevent=u, cause={v}, idusuari=x, usuari=y, idpauta=z, pagina=p, idadvert=q, state=r] => **PublishMessageInTopic** [cause {v,u}, "advertTopic" , "L'usuari "+usuari+ " ha donat per col·locat l'anunci" +idadvert+ " a la pàgina "+pagina+ " de la pauta "+idpauta] /\ **SheetPrinted** [cause={v,u}, idusuari=x, usuari=y, idpauta=z ,pagina=p, false] **if [changeAdvertInPlace (**idpauta, pagina, idadvert, state) /\ (state==true]] **V**
PublishMessageInTopic [cause {v,u}, "advertTopic" , "L'usuari "+usuari+ " ha donat com a no col·locat l'anunci " +idanunci+ " a la pàgina "+pagina+ " de la pauta "+idpauta]
/\ **SheetPrinted** [cause={v}, idusuari=x, usuari=y, idpauta=z ,pagina=p, false]] **if [changeAdvertInPlace (**idpauta, pagina, idadvert, state) /\ (state==false]]

Event: IsNotAdvertError

S'avisa a l'usuari que l'objecte en qüestió no és una fotografia

IsNotAdvertError [idevent=u, cause={ v }, idusuari=x ,usuari=y, idpauta=z, pagina=p, idadvert=q , error=r] => **SendError** [cause={ v,u } , idusuari=x, error=r]

CAS D'ÚS: Borrar Article

Event: NewsDelete

Un usuari pot realitzar la petició d'esborrar un article, com a mesura de control tan sols l'usuari que ha creat l'article, que ha d'existir, tindrà dret a esborrar-lo. L'acció, a més, només s'ha de fer si la pàgina estava ocupada per l'usuari. Al finalitzar es torna a visualitzar la pàgina

NewsDelete [idevent=v, idusuari=x, usuari=y, idpauta=z, pagina=p, idarticle=q] => **[DeleteObject** [cause={v},idevent=v, idusuari=x, usuari=y, idpauta=z, pagina=p, idarticle=q]**] if [isNewsDoneBy** (idpauta, pagina, idarticle,idusuari) /\ **isSheetOcuppiedBy** (idusuari, idpauta,pagina) /\ **isNews**(idarticle)**] V PrivilegeNotHeld** [cause={v} ,idusuari=x, usuari=y, privilege="NewsDelete"] **if [!isNewsDoneBy** (idarticle,idusuari) **V !isSheetOcuppiedBy** (idusuari, idpauta,pagina)**] V IsNotNewsError [**cause={ v }, idusuari=x ,usuari=y, idpauta=z, pagina=p, idarticle=q , error="L'element no és una notícia"] **if [!isNews**(idarticle)**]**

CAS D'ÚS: Borrar Fotografia

Event: PictureDelete

Un usuari pot realitzar la petició d'esborrar una fotografia, que ha d'existir.. Tan sols ho podra fer un redactor i si la pàgina està ocupada per aquest. Al finalitzar es torna a visualitzar la pàgina

pictureTopic és el canal de missatges sobre canvis a les fotografies

PictureDelete [idevent=v idusuari=x, usuari=y, idpauta=z, pagina=p, idfotografia=q] => **[DeleteObject** [cause={v},idevent=v, idusuari=x, usuari=y, idpauta=z, pagina=p, idfotografia=q]**] if [isEditor()**) /\ **isSheetOcuppiedBy** (idusuari, idpauta,pagina) /\ **isPicture**(idfotografia)**] V PrivilegeNotHeld** [cause={v} ,idusuari=x, usuari=y, privilege="PictureDelete"] **if [!isEditor** (idusuari) **V !isSheetOcuppiedBy** (idusuari, idpauta,pagina)**] V IsNotPictureError [**cause={ v }, idusuari=x ,usuari=y, idpauta=z, pagina=p, idfotografia=q, error="L'element no és una fotografia"] **if [!isPicture**(idfotografia)**]**

CAS D'ÚS: Borrar Anunci

Event: AdvertDelete

Un usuari pot realitzar la petició d'esborrar un anunci, que ha d'existir. Tan sols ho podra fer un redactor i si la pàgina està ocupada per aquest. Al finalitzar es torna a visualitzar la pàgina

adverteTopic és el canal de missatges sobre canvis als anuncis

AdvertDelete [idevent=v, idusuari=x, usuari=y, idpauta=z, pagina=p, idadvert=q] => **[DeleteObject** [cause={v},idevent=v, idusuari=x, usuari=y, idpauta=z, pagina=p, idadvert=q]]**] if [isEditor()**) /\ **isSheetOcuppiedBy** (idusuari, idpauta,pagina) /\ **isAdvert**(idadvert)**]** **V** **PrivilegeNotHeld** [cause={v} ,idusuari=x, usuari=y, privilege="AdvertDelete"] **if** **[!isEditor** (idusuari) **V** **!isSheetOcuppiedBy** (idusuari, idpauta,pagina)**]** **V** **IsNotAdvertError** **[**cause={ v }, idusuari=x ,usuari=y, idpauta=z, pagina=p, idadvert=q, error="L'element no és un anunci"] **if** **[!isAdvert**(idfotografia)**]**

CAS D'ÚS: CrearArticle

Event: NewsCreate

Quan es crea una notícia es comprova que la pàgina (o pàgines) estiguin ocupades per l'usuari, que els mòduls estiguin lliures i l'usuari sigui un redactor. Si ho estan es genera un event CreateObject
En cas que algun dels requisits falli s'envia un missatge a l'usuari avisant que la notícia no ha estat creada.

moduls és un array de l'objecte modul de mida variable i pot ser de pàgines diferents

NewsCreate [idevent=v, idusuari=x, usuari=y, idpauta=z, pagina=p, tema=t, moduls=M] => [idobject=**CreateObject**[cause={v}, idusuari=x, usuari=y, idpauta=z, pagina=p, tema=t, moduls=M, type="news"] **if** **[isSheetOcuppiedBy** (idusuari, idpauta,pagina) /\ **isEditor**(idusuari) /\ **areModuleFree**(moduls)] **V** **PrivilegeNotHeld** [cause={v} ,idusuari=x, usuari=y, privilege="Editor"] if (**!isEditor**(idusuari) **SheetNotOcuppied** [cause={v} ,idusuari=y,usuari=z, error="The sheet isn't ocuppied **"]** **if** [!**isSheetOcuppiedBy** (idusuari, idpauta,pagina) /\ **isEditor**(idusuari)] **V** **ModulesNotFree** [cause={v} ,idusuari=y,usuari=z, error="The modules aren't free **"]** **if** **[isSheetOcuppiedBy** (idusuari, idpauta,pagina) /\ **isEditor**(idusuari) /\ !**areModuleFree**(moduls)]

Event: CreateObject

Quan es crea un objecte i l'usuari sigui un redactor. Si ho estan es crea un nou objecte base i s'obté un nou identificador. A continuació es creida a la funció pertinent per crear l'objecte del tipus apropiat
En cas que algun dels requisits falli es genera un event d'error

moduls és un array de l'objecte modul de mida variable i pot ser de pàgines diferents

CreateObject [idevent=u, cause={v}, idusuari=x, usuari=y, idpauta=z, pagina=p, tema=t, moduls=M, type=q] => **NewObject** [cause={v,u}, idusuari=x, usuari=y, idpauta=z, pagina=p, tema=t, moduls=M, newid=**getNewObjectId**()] =>

CreateNews [cause={v,u}, idusuari=x, usuari=y, newid=s] **if** (type=="news") **V**

CreateAdvert [cause={v,u}, idusuari=x, usuari=y, newid=s] **if** (type=="advert") **V**

CreatePicture [cause={v,u}, idusuari=x, usuari=y, newid=s] **if** (type=="picture") **V**

Event: CreateNews

Crea una nova notícia i avisa a l'usuari que aquesta ha estat creada. A més avisa pel canal corresponent que hi ha una nova notícia.

CreateNews [idevent=w, cause={v,u}, idusuari=x, usuari=y, newid=s] => [**SendMessage** [cause={ v,u,w }, idusuari=x, usuari=y, type= "CreateNews", ObjectMessage= **newObjectMessage** (**trueObject**())] /\ **PublishMessageInTopic** [cause {v,u,w}, "NewsTopic" , "Nova notícia creada per "+usuari]] **if** (**NewObjectNews**(newisd, idusuari))

Event: SheetNotOcuppied

S'avisa a l'usuari que la pàgina no està ocupada per ell

SheetNotOcuppied [idevent=u, cause={v} ,idusuari=y,usuari=z, error=t]
=> **SendError** [cause={ v,u } , idusuari=x, error=r]

Event: ModulesNotFree

S'avisa a l'usuari que els moduls escollits no estan lliures

ModulesNotFree [idevent=u, cause={v} ,idusuari=y,usuari=z, error=r]
=> **SendError** [cause={ v,u } , idusuari=x, error=r]

CAS D'ÚS: CrearFotografia

Event: PictureCreate

Quan es crea una fotografia es comprova que la pàgina (o pàgines) estiguin ocupades per l'usuari, que els mòduls estiguin lliures i l'usuari sigui un redactor. Si ho estan es genera un event CreateObject
En cas que algun dels requisits falli s'envia un missatge a l'usuari avisant que la fotografia no ha estat creada.

moduls és un array de l'objecte modul de mida variable i pot ser de pàgines diferents

PictureCreate [idevent=v, idusuari=x, usuari=y, idpauta=z, pagina=p, tema=t, moduls=M] => [idobject=**CreateObject**[cause={v}, idusuari=x, usuari=y, idpauta=z, pagina=p, tema=t, moduls=M, type="picture"] **if [isSheetOcuppiedBy** (idusuari, idpauta,pagina) /\ **isEditor**(idusuari) /\ **areModuleFree**(moduls)] **V**
PrivilegeNotHeld [cause={v} ,idusuari=x, usuari=y, privilege="Editor"] if (**!isEditor**(idusuari) **SheetNotOcuppied** [cause={v} ,idusuari=y,usuari=z, error="The sheet isn't ocuppied **"]** **if** [!**isSheetOcuppiedBy** (idusuari, idpauta,pagina) /\ **isEditor**(idusuari)] **V** **ModulesNotFree** [cause={v} ,idusuari=y,usuari=z, error="The modules aren't free **"]** **if [isSheetOcuppiedBy** (idusuari, idpauta,pagina) /\ **isEditor**(idusuari) /\ !**areModuleFree**(moduls)]

Event: CreatePicture

Crea una nova fotografia i avisa a l'usuari que aquesta ha estat creada. A més avisa pel canal corresponent que hi ha una nova fotografia.

CreatePicture [idevent=w, cause={v,u}, idusuari=x, usuari=y, newid=s] => [**SendMessage** [cause={ v,u,w }, idusuari=x, usuari=y, type= "CreatePicture", ObjectMessage= **newObjectMessage** (**trueObject**())] /\ **PublishMessageInTopic** [cause {v,u,w}, "PictureTopic" , "Nova fotografia creada per "+usuari]] **if** (**NewObjectPicture**(newid))

CAS D'ÚS: CrearAnunci

Event: AdvertCreate

Quan es crea un anunci es comprova que la pàgina (o pàgines) estiguin ocupades per l'usuari, que els mòduls estiguin lliures i l'usuari sigui un redactor. Si ho estan es genera un event CreateObject
En cas que algun dels requisits falli s'envia un missatge a l'usuari avisant que l'anunci no ha estat creay.

moduls és un array de l'objecte modul de mida variable i pot ser de pàgines diferents

AdvertCreate [idevent=v, idusuari=x, usuari=y, idpauta=z, pagina=p, tema=t, moduls=M] => [idobject=**CreateObject**[cause={v}, idusuari=x, usuari=y, idpauta=z, pagina=p, tema=t, moduls=M, type="advert"] **if** [**isSheetOcuppiedBy** (idusuari, idpauta,pagina) /\ **isEditor**(idusuari) /\ **areModuleFree**(moduls)] **V** **PrivilegeNotHeld** [cause={v} ,idusuari=x, usuari=y, privilege="Editor"] if (**!isEditor**(idusuari) **SheetNotOcuppied** [cause={v} ,idusuari=y,usuari=z, error="The sheet isn't ocuppied **"]** **if** [!**isSheetOcuppiedBy** (idusuari, idpauta,pagina) /\ **isEditor**(idusuari)] **V** **ModulesNotFree** [cause={v} ,idusuari=y,usuari=z, error="The modules aren't free **"]** **if** [**isSheetOcuppiedBy** (idusuari, idpauta,pagina) /\ **isEditor**(idusuari) /\ !**areModuleFree**(moduls)]

Event: CreateAdvert

Crea un nou anunci i avisa a l'usuari que aquesta ha estat creat. A més avisa pel canal corresponent que hi ha un nou anunci

CreateAdvert [idevent=w, cause={v,u}, idusuari=x, usuari=y, newid=s] => [**SendMessage** [cause={ v,u,w }, idusuari=x, usuari=y, type= "CreateAdvert", ObjectMessage= **newObjectMessage** (**trueObject**())] /\ **PublishMessageInTopic** [cause {v,u,w}, "AdvertTopic" , "Nova fotografia creada per "+usuari]] **if** (**NewObjectAdvert**(newid))

CAS D'ÚS: ModificarEstatPagina

Event: SheetModify

Un usuari pot demanar fer canvis respecte diferents estats d'una pàgina. Depenent de la seva categoria es podran canviar uns estats o uns altres. El llistat d'estats que es poden canviar des d'aquest event són:

<color>	true/false	És la pàgina en color?	redactor
<tancada>	true/false	Està la pàgina tancada?	redactor
<posicio>	dreta/esquera/indiferent	Està en alguna posició especial?	redactor
<capçalera>	comarques, actualitat,...	A quina secció pertany?	redactor
<continuaa>	idpàgina o null	Continua a alguna pàgina?	redactor
<impresa>	true/false	Està la pàgina impresa?	Maquetador

A més existeixen altres estats que són condició per a fer alguns canvis d'estat

<vede>	idpàgina o null	És continuació d'alguna pàgina?
<corregida>	true/false	Està la pàgina corregida?

1) Una pagina impresa ha d'estar corregida
2) Una pàgina tancada ha de tenir tots els mòduls ocupats
3) Una pàgina no pot continuar a una altra que tingui el valor vede!=null
2) No es pot borrar un 'vede' o 'continuaa' si hi ha una notícia que comparteix les pàgines

SheetModify [idevent=v, idusuari=x, usuari=y, idpauta=z, pagina=p, color= q, tancada=r, posició=s, capçalera =t, continuaa= u, impresa=m] => **{**

[**SheetPrinted** [cause={v}, idusuari=x, usuari=y, idpauta=z ,pagina=p, color=q] /\

SheetCapçalera [cause={v}, idusuari=x, usuari=y, idpauta=z ,pagina=p, capçalerar=t]] **if [isEditor** (idusuari) **]** =>

SheetContinues [idevent=u, cause={v}, idusuari=x, usuari=y, idpauta=z ,pagina=p, continuaa=u] **if [isEditor** (idusuari) /\ **{[sheetComesFrom**(idpauta,continuaa)!=null /\ (continuaa!=null)**]** **V** **{**(continuaa==null) /\ !**hasSharedNews**(idpauta, pagina)**}]** =>

SheetClosed [cause={v}, idusuari=x, usuari=y, idpauta=z ,pagina=p, tancada=r] **if [isEditor** (idusuari) /\ [tancada==false V **isSheetFul**l(idpauta, pagina)]**]** =>

SheetPrinted [cause={v}, idusuari=x, usuari=y, idpauta=z ,pagina=p, printed=m]

if [isPrinter (idusuari) /\ **sheetExists**(idpauta, pagina) /\ **isSheetRevised**(idpauta, pagina)] **}** **if** **isSheetOcuppiedBy** (idusuari, idpauta,pagina) **V**

SheetNotOcuppied [cause={v} ,idusuari=y, usuari=z, error="The sheet isn't ocuppied **"]** **if** [!**isSheetOcuppiedBy** (idusuari, idpauta,pagina)

Event: SheetColor

Es canvia l'atribut color d'una pàgina i s'avisa dels canvis pel canal adequat

sheetTopic és el canal de missatges sobre canvisa les pàgines

SheetColor [idevent=u, cause={v}, idusuari=x, usuari=y, idpauta=z ,pagina=p, color=q] => { **PublishMessageInTopic** [cause {v,u}, "sheetTopic" , "L'usuari "+usuari+ " ha assignat l'atribut color a la pàgina "+pagina+ " de la pauta "+idpauta] **if [changeColor (**idpauta, pagina, color) /\ (color==true]] **V**
PublishMessageInTopic [cause {v,u}, "sheetTopic" , "L'usuari "+usuari+ " ha assignat l'atribut blanc i negre a la pàgina "+pagina+ " de la pauta "+idpauta] **if [changeColor (**idpauta, pagina, color) /\ (color==false)]]

Event: SheetCapçalera

Es canvia l'atribut capçalera d'una pàgina i s'avisa dels canvis pel canal adequat

sheetTopic és el canal de missatges sobre canvisa les pàgines

SheetCapçalera [idevent=u, cause={v}, idusuari=x, usuari=y, idpauta=z ,pagina=p, capçalerar=q] => { **PublishMessageInTopic** [cause {v,u}, "sheetTopic" , "L'usuari "+usuari+ " ha assignat la capçalera "+ capçalera + " a la pàgina "+pagina+ " de la pauta "+idpauta] **if [changeCapçalera (**idpauta, pagina, capçalera)]

Event: SheetContinues

Es canvia l'atribut que indica la pàgina que va a continuació i l'atribut "vede" d'aquesta darrera pàgina i s'avisa dels canvis pel canal adequat

sheetTopic és el canal de missatges sobre canvisa les pàgines

SheetContinues [idevent=u, cause={v}, idusuari=x, usuari=y, idpauta=z ,pagina=p, continuaa=q] => { **PublishMessageInTopic** [cause {v,u}, "sheetTopic" , "L'usuari "+usuari+ " ha assignat la pàgina "+ continuaa + " com a continuació de la pàgina "+pagina+ " de la pauta "+idpauta] **if [changeContinuaa (**idpauta, pagina, continua) /\ **changeVede** (idpauta, continua ,pagina)]

Event: SheetClosed

Es canvia l'atribut "tancada" d'una pàgina i s'avisa dels canvis pel canal adequat
sheetTopic és el canal de missatges sobre canvisa les pàgines

SheetClosed [idevent=u, cause={v}, idusuari=x, usuari=y, idpauta=z ,pagina=p, tancada=q] => { **PublishMessageInTopic** [cause {v,u}, "sheetTopic" , "L'usuari "+usuari+ " ha donat per tancada la pàgina "+pagina+ " de la pauta "+idpauta] **if [changeClosed (**idpauta, pagina, tancada) /\ (tancada==true]] **V**
PublishMessageInTopic [cause {v,u}, "sheetTopic" , "L'usuari "+usuari+ " ha donat per oberta la pàgina "+pagina+ " de la pauta "+idpauta] **if [changeClosed (**idpauta, pagina, tancada) /\ (tancada==false)]]

Event: SheetRevised

Es canvia l'atribut "corregida" d'una pàgina i s'avisa dels canvis pel canal adequat
sheetTopic és el canal de missatges sobre canvisa les pàgines

SheetRevised [idevent=u, cause={v}, idusuari=x, usuari=y, idpauta=z ,pagina=p, corregida=q] => { **PublishMessageInTopic** [cause {v,u}, "sheetTopic" , "L'usuari "+usuari+ " ha donat per corregida la pàgina "+pagina+ " de la pauta "+idpauta] **if [changeRevised (**idpauta, pagina, corregida) /\ (corregida==true]] **V**
PublishMessageInTopic [cause {v,u}, "sheetTopic" , "L'usuari "+usuari+ " ha donat per no corregida la pàgina "+pagina+ " de la pauta "+idpauta] **if [changeRevised (**idpauta, pagina, corregida) /\ (corregida==false)]]

CAS D'ÚS: Borrar Pàgina

Event: SheetDelete

Un usuari, si és redactor, pot demanar esborrar una pàgina si aquesta existeix. Primer esborrarà els diferents objectes, a continuació esborrarà els moduls que componen la pàgina, per últim esborrarà la pagina

sheetTopic és el canal de missatges sobre canvis a les pàgines

SheetDelete [idevent=v, idusuari=x, usuari=y, idpauta=z, pagina=p] => **{ [** Per tot News n de **getNews**(idpauta, pagina) => **DeleteObject** [cause={v}, idusuari=x, usuari=y ,n] **]** ∧ **[** Per tot Picture n de **getPictures**(idpauta, pagina) => **DeleteObject** [cause={v}, idusuari=x, usuari=y ,n]**]** ∧ **[** Per tot Advert n de **getAdverts**(idpauta, pagina) => **DeleteObject** [cause={v}, idusuari=x, usuari=y ,n]**] =>** **DeleteModulesFrom** [cause={v}, idusuari=x, usuari=y ,idpauta=z, pagina=p] => **DeleteSheet** [cause={v}, idusuari=x, usuari=y ,idpauta=z, pagina=p]**}** **if [isEditor**(idusuari)**]** **V** **PrivilegeNotHeld** [cause={v},idusuari=x, usuari=y, privilege="editor"] **if [!isEditor**(idusuari)]

Event: DeleteObject

S'encarrega d'itendificar l'objecte a esborrar i crear l'event adequat per a realitzar l'operació

DeleteObject [idevent=u, cause={v}, idusuari=x, usuari=y , idobject=n] =>
DeleteNews [cause={v,u}, idusuari=x, usuari=y , idobject=n] **if** (**isNews**(idobject)) **V**
DeleteAdvert [cause={v,u}, idusuari=x, usuari=y , idobject=n] **if** (**isAdver**(idobject)) **V**
DeletePicture [cause={v,u}, idusuari=x, usuari=y , idobject=n] **if** (**isPicture**(idobject))

Event: DeleteNews

Esborra una notícia de la base de dades i ho anuncia pel canal adequat

DeleteNews[idevent=w, cause={v,u}, idusuari=x, usuari=y , idobject=n] =>
PublishMessageInTopic [cause {v,u,w}, "SheetTopic" , "L'usuari "+y+ " ha esborrat la notícia "+ idobject] **if** (**deleteNewsFromDatabase**(idobject) /\ **deleteObjectFromDatabase**(idobject))

Event: DeletePicture

Esborra una fotografia de la base de dades i ho anuncia pel canal adequat

DeletePicture[idevent=w, cause={v,u}, idusuari=x, usuari=y , idobject=n] => **PublishMessageInTopic** [cause {v,u,w}, "PictureTopic" , "L'usuari "+y+ " ha esborrat la fotografia "+ idobject] **if** (**deletePictureFromDatabase**(idobject) /\ **deleteObjectFromDatabase**(idobject))

Event: DeleteAdvert

Esborra un anunci de la base de dades i ho anuncia pel canal adequat

DeleteAdvert[idevent=w, cause={v,u}, idusuari=x, usuari=y , idobject=n] => **PublishMessageInTopic** [cause {v,u,w}, "AdvertTopic" , "L'usuari "+y+ " ha esborrat la fotografia "+ idobject] **if** (**deleteAdvertFromDatabase**(idobject) /\ **deleteObjectFromDatabase**(idobject))

Event: DeleteModulesFrom

S'encarrega de cridar l'event que esborra un modul per cadascun dels mòduls d'una pàgina

DeleteModulesFrom [idevent=u, cause={v}, idusuari=x, usuari=y ,idpauta=z, pagina=p] => [Per tot Module n de **getModules**(idpauta, pagina) => **DeleteModule** [cause={v,u}, idusuari=x, usuari=y , idpauta=z, pagina=p , modul=n]]

Event: DeleteModule

Event final que esborra un mòdul de la base de dades

DeleteModule [idevent=w, cause={v,u}, idusuari=x, usuari=y ,idpauta=z, modul=n]

Event: DeleteSheet

Event que s'encarrega d'esborrar una pàgina de la base de dades i anunciar-ho pel canal apropiat

DeleteSheet [idevent=u, cause={v}, idusuari=x, usuari=y ,idpauta=z, pagina=p] => **SendMessage** [cause={ v,u }, idusuari=x, usuari=y, type= "DeleteSheet", ObjectMessage= **newObjectMessage** (**trueObject**())] /\ **PublishMessageInTopic** [cause {v,u}, "SheetTopic" , "L'usuari "+y+ " ha esborrat la pàgina "+pagina+" de la pauta "+idpauta] **if** (**deleteAdvertFromDatabase**(idobject) /\ **deleteObjectFromDatabase**(idobject))

CAS D'ÚS: Crear Pagina

Event: SheetNew

Un usuari, que pot ser redactor o director, fa la petició de creació d'una pàgina passat un missatge amb les dades necessàries

SheetNew [idevent=v, idusuari=x, usuari=y, idpauta=z, capçalera=q , posicio=r, color=s] => **{** pagina p=**getNewSheetID**(idpauta) => **CreateSheet** [cause={v}, idusuari=x, usuari=y, idpauta=z, capçalera=q , posicio=r, color=s, pagina=p] **}** **if** **[isEditor**(idusuari) **V** **isDirector**(idusuari)**]** **V** **PrivilegeNotHeld** [cause={v},idusuari=x, usuari=y, privilege="SheetNew"] **if** **[!isEditor**(idusuari) /\ !**isDirector**(idusuari)**]**

Event: CreateSheet

Event que crea la pàgina a la base de dades, anuncia a l'autor que s'ha creat i n'informa pel canal adequat

sheetTopic és el canal de missatges sobre canvis a les pàgines

CreateSheet [idevent=u, cause={v}, idusuari=x, usuari=y, idpauta=z, capçalera=q , posicio=r, color=s, pagina=p] => { For (x=0;x<**getSheetWidth();**x++) { For (y=0;y<**getSheetHeight();**y++) { **CreateModule** [cause={u,v}, idpauta=z, pagina=p, fila=y, columna=x]}} /\ **SendMessage** [cause={ v,u }, idusuari=x, usuari=y, type= "CreateSheet", ObjectMessage= **newObjectMessage** (**trueObject**())] /\ **PublishMessageInTopic** [cause {v,u}, "SheetTopic" , "L'usuari "+y+ " ha creat la pàgina "+pagina+" de la pauta "+idpauta] } **if** **createSheetInDatabase** (idpauta,pagina,capçalera,posició,color)

Event: CreateModule

Event atomic final que crea un modul a la base de dades

CreateModule [idevent=w, cause={u,v}, idpauta=z, pagina=p, fila=y, columna=x]

CAS D'ÚS: Crear Pauta

Event: PautaNew

Un usuari, que ha de ser director, fa la petició de creació d'una pauta a la base de dades amb les seves pàgines i mòduls bàsics: 1-25 actualitat, 26-27 economia, 28-31 cultura, 32-45 opinió, 46-65 esports, 66-89 Comarques, 90-94 societat, 95-99 escola, 100-103 societat, 104 actualitat. Totes les pàgines son marcades com obertes, no corregides, blanc i negre i no impresa.

PautaNew [idevent=v, idusuari=x, usuari=y, publicacio=z] => **{** idpauta p=**getNewPautaID**() => **SheetNew** [cause={v}, idusuari=x, usuari=y, idpauta=p, "Actualitat" , "indiferent", "blanc i negre"] **for** [i=0; i<25; i++] => SheetNew [cause={v}, idusuari=x, usuari=y, idpauta=p, "Economia" , "indiferent", "blanc i negre"] **for** [i=0; i<2; i++] => **SheetNew** [cause={v}, idusuari=x, usuari=y, idpauta=p, "Cultura" , "indiferent", "blanc i negre"] **for** [i=0; i<4; i++] => **SheetNew** [cause={v}, idusuari=x, usuari=y, idpauta=p, "Opinió" , "indiferent", "blanc i negre"] **for** [i=0; i<14; i++] => **SheetNew** [cause={v}, idusuari=x, usuari=y, idpauta=p, "Esports" , "indiferent", "blanc i negre"] **for** [i=0; i<20; i++] => **SheetNew** [cause={v}, idusuari=x, usuari=y, idpauta=p, "Comarques" , "indiferent", "blanc i negre"] **for** [i=0; i<24; i++] => **SheetNew** [cause={v}, idusuari=x, usuari=y, idpauta=p, "Societat" , "indiferent", "blanc i negre"] **for** [i=0; i<5; i++] => **SheetNew** [cause={v}, idusuari=x, usuari=y, idpauta=p, "Escolat" , "indiferent", "blanc i negre"] **for** [i=0; i<5; i++] =>**SheetNew** [cause={v}, idusuari=x, usuari=y, idpauta=p, "Societat" , "indiferent", "blanc i negre"] **for** [i=0; i<4; i++] => **SheetNew** [cause={v}, idusuari=x, usuari=y, idpauta=p, "Actualitat" , "indiferent", "blanc i negre"] => **SendMessage** [cause={ v,u }, idusuari=x, usuari=y, type= "PautaNewt", ObjectMessage= **newObjectMessage** (**trueObject**())]} if [**isDirector**(idusuari)] **V** => **PrivilegeNotHeld** [cause={v},idusuari=x, usuari=y, privilege="director"] if [!**isDirector**(idusuari)]

CAS D'ÚS: NumerarPàgines

Event: SheetNum

Un usuari, que ha de ser director, pot numerar les pàgines. La numeració ha de ser única per a cada pàgina.

numeració és un array de parells <idpàgina.número>

SheetNum [idevent=u, idusuari=x, usuari=y, idpauta=z, numeració[]=N] => { Per cada idpagina=p,numero=q de numeració[] **SetSheetNum** [cause={v}, idusuari=x, usuari=y, idpauta=z, idpagina=p, numero=q]} /\ **SendMessage** [cause={ v }, idusuari=x, usuari=y, type= "SheetNum", ObjectMessage= **newObjectMessage** (**trueObject**())] } if [**numeracioUnica**(numeracio) /\ **isDirector**(idusuari)] **V** **PrivilegeNotHeld** [cause={v},idusuari=x, usuari=y, privilege="director"] if [!**isDirector**(idusuari)] **V** **NumeracioNotUnicError** [cause={v} ,idusuari=y,usuari=z, error="La numeració no és única**"]** if [!**numeracioUnica**(numeracio) /\ **isDirector**(idusuari)]

Event: SetSheetNum

Event final que modifica la numeració final d'una pàgina a la base de dades

SetSheetNum [idevent?u, cause={v}, idusuari=x, usuari=y, idpauta=z, idpagina=p, numero=q]

Event: NumeracioNotUnicError

S'avisa a l'usuari que l'objecte en qüestió no és una fotografia

NumeracioNotUnicError [idevent=u, cause={ v }, idusuari=x ,usuari=y, error=r] => **SendError** [cause={ v,u } , idusuari=x, error=r]

Events Genèrics

Event: CheckReviseSheet

Es comprova si una pàgina està corregida o no i es marca com a tal. Per a que una pàgina estigui corregida ha d'estar plena i ha d'estar tancada.

CheckSheetRevised [idevent=w cause={v,u}, idusuari=x, usuari=y, idpauta=z, pagina=p] => **[SheetRevised** [cause={v,u,w}, idusuari=x, usuari=y, idpauta=z ,pagina=p, corregida="true"] **if [isSheetClosed** (idpauta, pagina) ∧ **[isNewsRevised**(n) per cada News n **getNews**(idpauta, pagina)**]** **else** **SheetRevised** [cause={v,u,w}, idusuari=x, usuari=y, idpauta=z ,pagina=p, corregida="false"]

Event: DeleteReservations

Es realitza una petició eliminar totes les reserves realitzades per un usuari.

DeleteReservations [idevent=v,idusuari=x,usuari=y]=> **DropReservation** [cause={v}, idusuari=v, idpauta=q, idpagina=p] **Per cada** (idpagina=p,idpauta=q) **de getSheetsReservedBy**(idusuari)

Event: DropReservation

Event final atòmic que esborra una reserva a la base de dades

DropReservation [idevent=u, cause={v}, idusuari=v, idpauta=q, idpagina=p]

Event: FreeSheets

Es realitza una petició per alliberar totes les pàgines ocupades per un usuari.

FreeSheets[idevent=s, cause={v,u,w}, idusuari=x,usuari=y] => **FreeSheet** [cause={v,u,w,s} , idusuari=x, usuari=y, idpauta=zq pagina=p] **Per cada** (idpagina=p,idpauta=q) **de getSheetsOccupiedBy**(idusuari)

Event: Log

Publica un missatge al topic log per control tenir un control dels usuaris

Log [idevent=w, cause {v,u}, usuari=z, msg] => **PublishMessageInTopic** [cause {v,u,w}, "Log" , msg]

Event: PrivilegeNotHeld

S'avisa a l'usuari que no compleix algun dels requisits

PrivilegeNotHeld [cause={v},idusuari=x, usuari=y, privilege=r]
=> **SendError** [cause={ v,u } , idusuari=x, privileger=r]

Event: PublishMessageInTopic

Event final atòmic que publica un missatge en un dels topics de l'aplicació

PublishMessageInTopic [cause {v,u}, topic , misssatge]

Event: SendError

Event final atòmic que publica envia un missatge d'error a un usuari

SendError [cause={v,u}, idusuari, error]

Event: SendMessage

Event final atòmic que envia un missatge de tipus Object a un receptor

SendMessage [cause {v}, idusuari=y, usuari=z, type, ObjectMessage]

Events de control

Aquests esvents es produiran cada quinze minuts i ens serviran per a mantenir el control de certs successos a les pàgines que evitaran problemes, sobretot, de llarga ocupació.
Hem de tenir en compte que gràcies al sistema de funcionament de l'aplicació, en tot moment s'està fent servir el rellotge del processador d'events i per tant se solventa un dels principals problemes de les aplicacions dedicades a events.

Event: CheckSheetOcupation

Si una pàgina fa més de dos hores que està ocupada s'alliberarà.

CheckSheetOcupation[idevent=v**]** => **[SheetBusy**[cause={v}, idusuari=x, usuari=y, idpauta=z ,pagina=p, false]] per cada <idusuari=x, usuari=y, idpauta=z, pagina=p> de **getSheetsOccupiedBy**(idusuari) if [**actualDate**()-**getSheetDateOcupattion**(idpauta,pagina)>7200 seconds]

Event: CheckSheetReserves

Si una pàgina fa més de dos hores que està reservada s'alliberarà.

CheckSheetReserves [idevent=v**]** => **[EraseReservation** [cause={v}, idusuari=x ,usuari=y, idpauta=z ,pagina=p] per cada <idusuari=x, usuari=y, idpauta=z, pagina=p> de **getSheetsReservedBy**(idusuari) if [**actualDate**()-**getSheetDateReservation** (idpauta,pagina)>7200 seconds]

Event: CheckUserIddle

Si un usuari fa més de dos hores que no fa res es fa expirar la seva sessió

CheckUserIddle [idevent=v] => **[UserLogout**[cause={v}, idusuari=x, usuari=y]
per cada <idusuari=x, usuari=y > de **GetUsersLogged**() if [**actualDate**()-**getUserIddle** (idpauta,pagina)>7200 seconds]

Event: UserIddleUpdate

Event atòmic final que actualitza l'atribut iddle d'un usuari

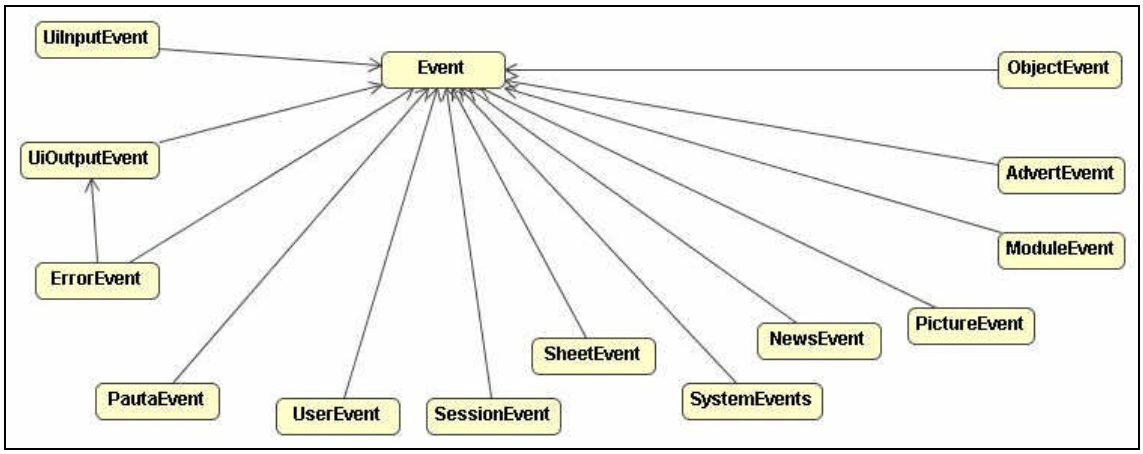
UserIddleUpdate[idevent=v, idusuari=x**]**

9.2 – Jerarquia d'events

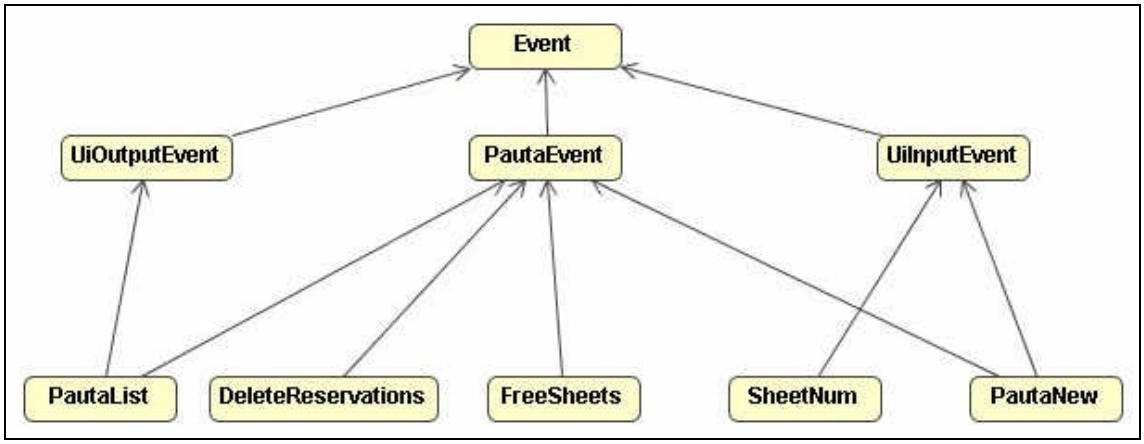
Un cop definit el lèxic podem passat a definir la jerarquia de components. Aquesta jerarquia ens permetrà en el següent pas crear el mapa de components a partir del qual definirem finalment els diferents components amb els events que generen i subscriuen.

Degut a la gran quantitat d'events hem hagut de dividir el mapa en diferents parts. En un primer diagrama veurem la branca principal de la jerarquia i en els diagrames següents veurem les diferents ramificacions d'aquestes branques.

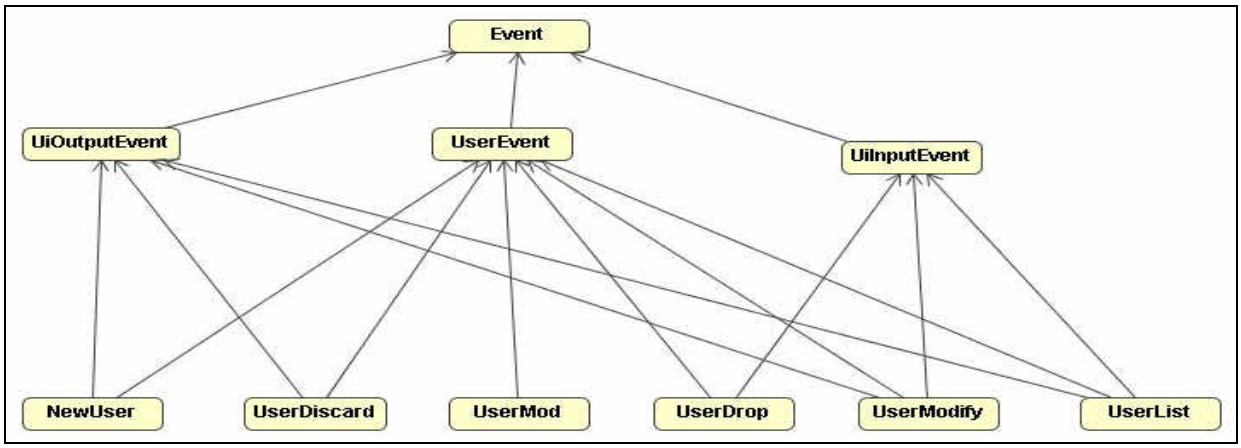
Jerarquia d'events, diagrma principal



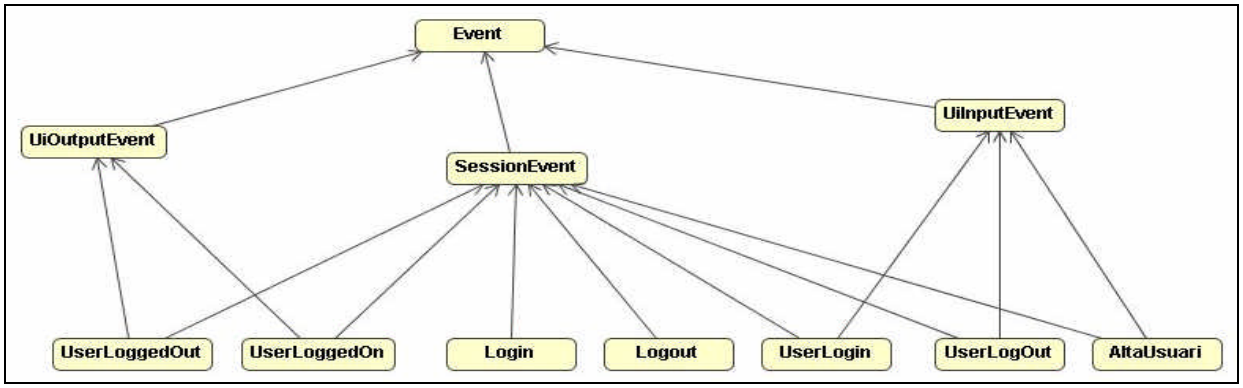
Jerarquia d'events, branca PautaEvent



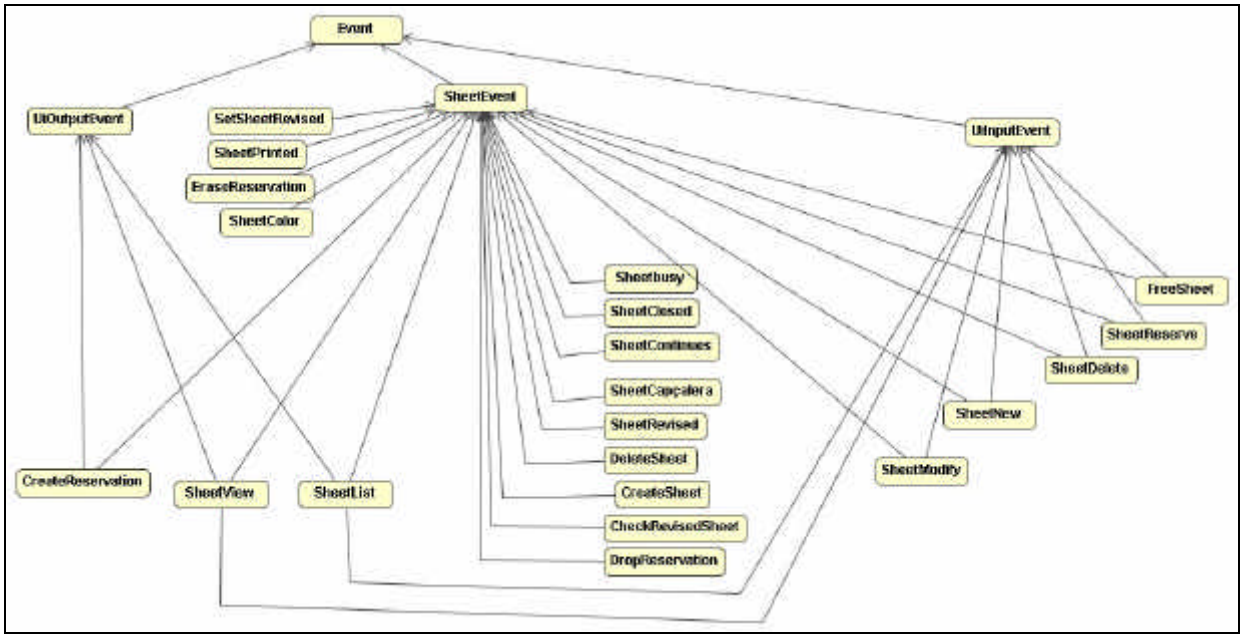
Jerarquia d'events, branca UserEvent



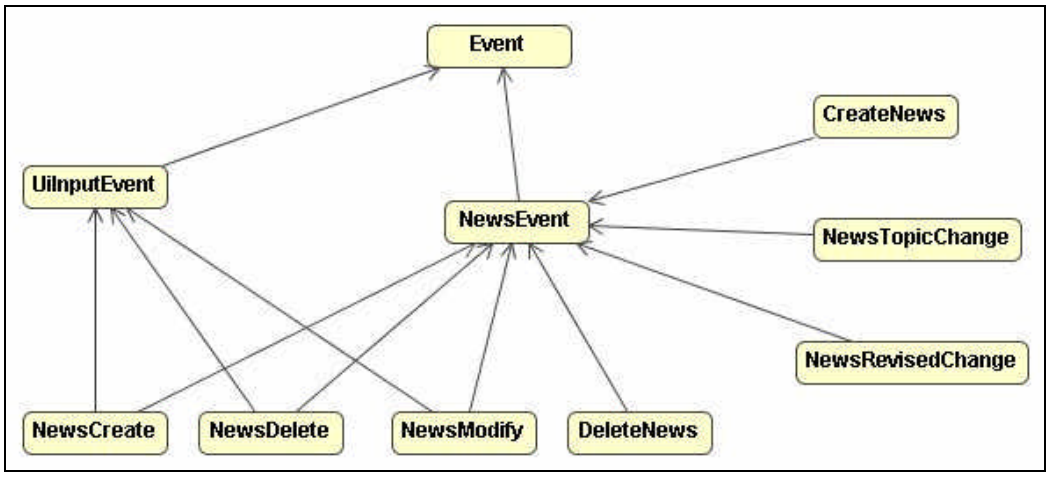
Jerarquia d'events, branca SessionEvent



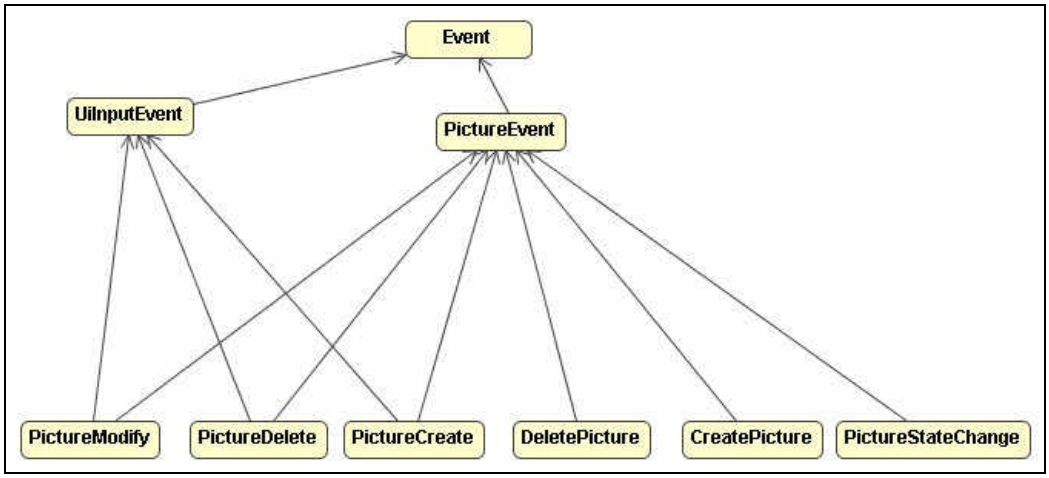
Jerarquia d'events, branca SheetEvent



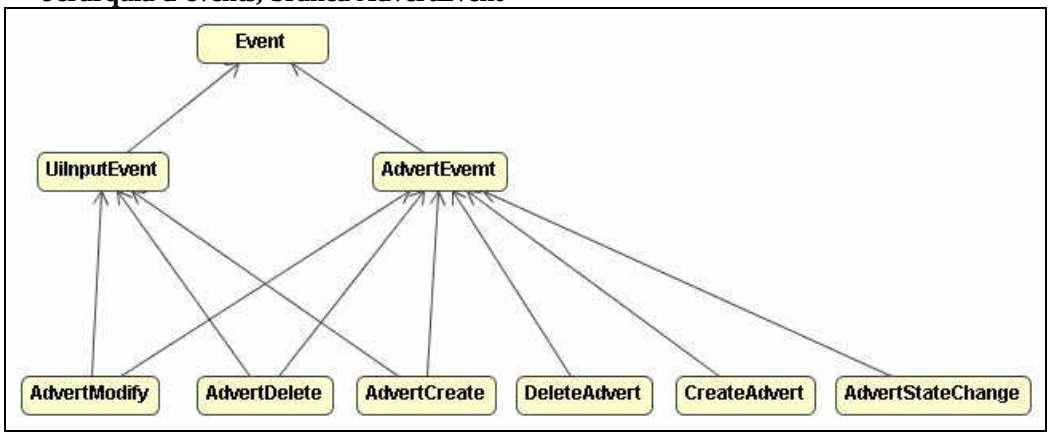
Jerarquia d'events, branca NewsEvent



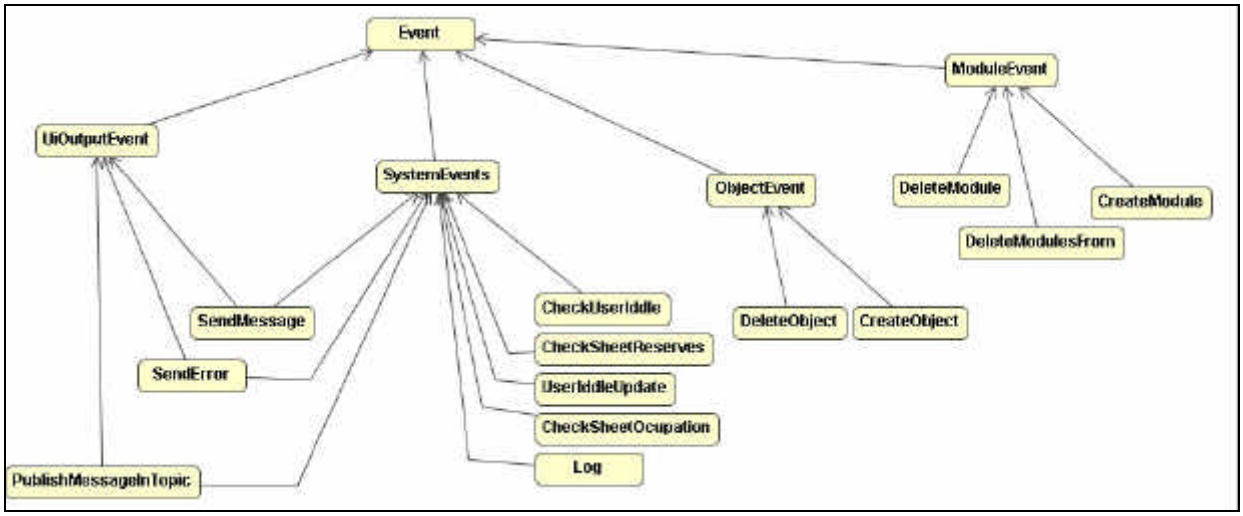
Jerarquia d'events, branca PictureEvent



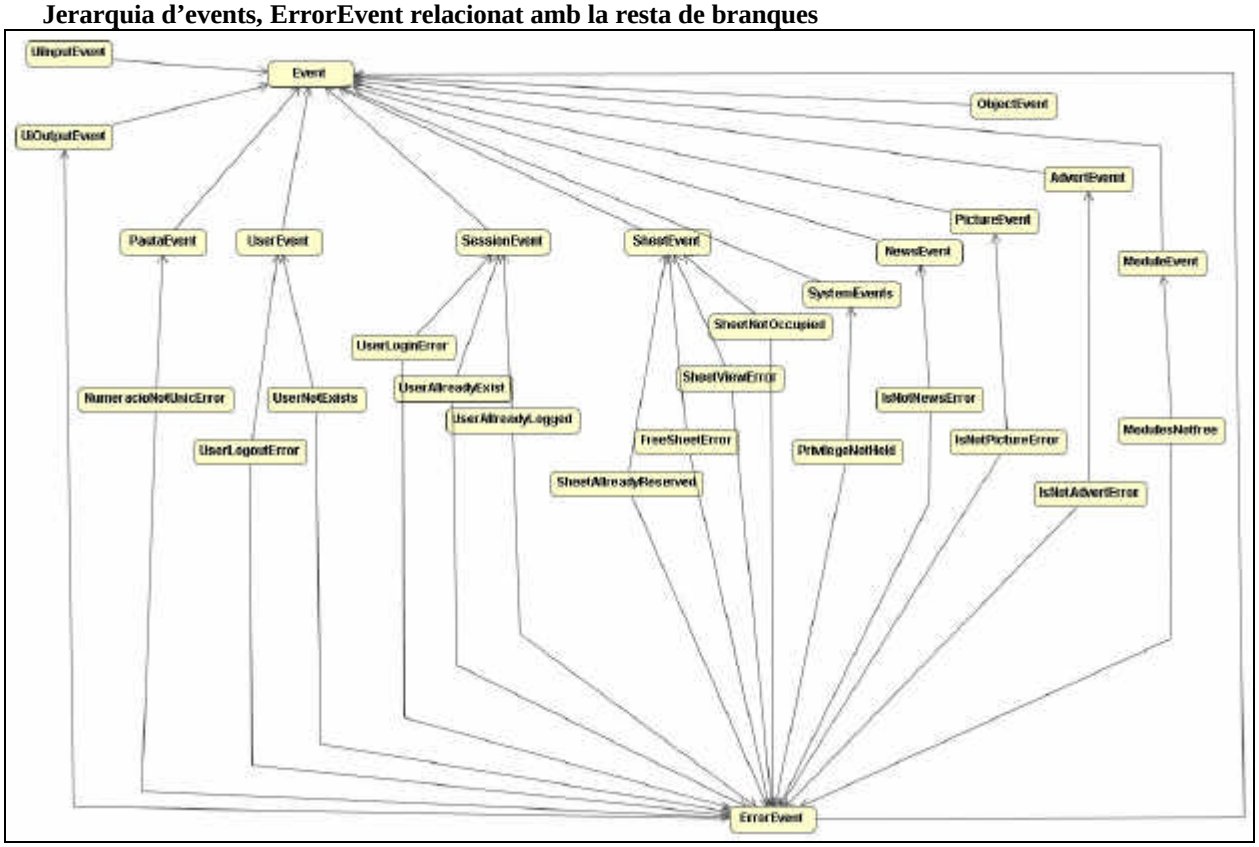
Jerarquia d'events, branca AdvertEvent



Jerarquia d'events, branques SystemEvents, ObjectEvent, ModuleEvent



Jerarquia d'events, ErrorEvent relacionat amb la resta de branques



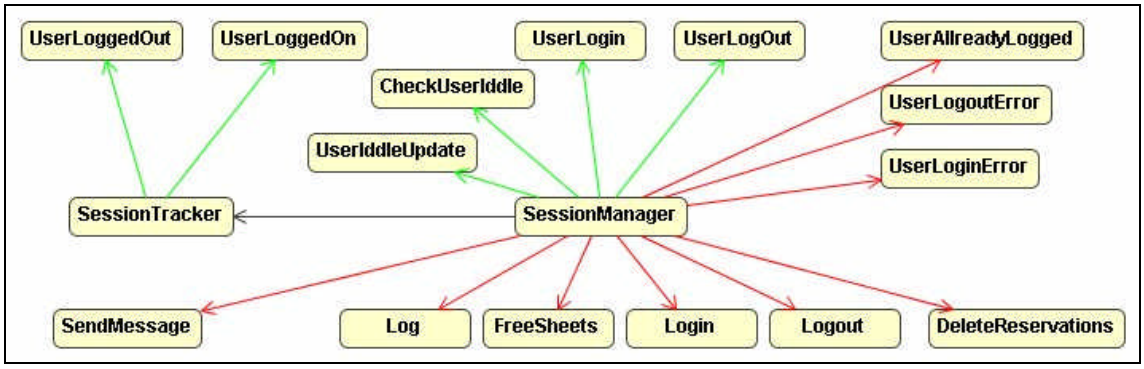
9.3 – Mapa de Components

El mapa de components ens permet assenyalar quins components hi haurà a la nostra aplicació. A partir del mapa podrem veure d'una forma senzilla i intuitiva a quins esvents se subscriu cada component i quins events emet cadascun. D'aquesta manera posteriorment podrem dur a terme l'especificació de cada component.

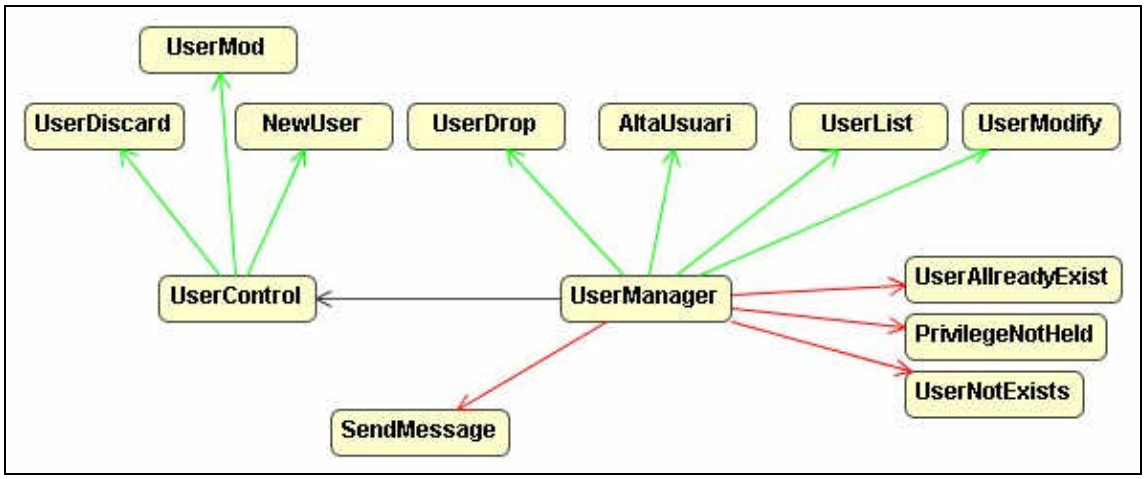
Novament per qüestions de tamany hem dividit el mapa en diversos diagrames on ja no apareix l'herència bàsica que es mostrava a la jerarquia, ens hem limitat tan sols al que és el mapa de components en si.

En els diagrames les **línees verdes** representen events d'entrada.
Les **línies vermelles** representen events de sortida
Les **línies negres** representen herència

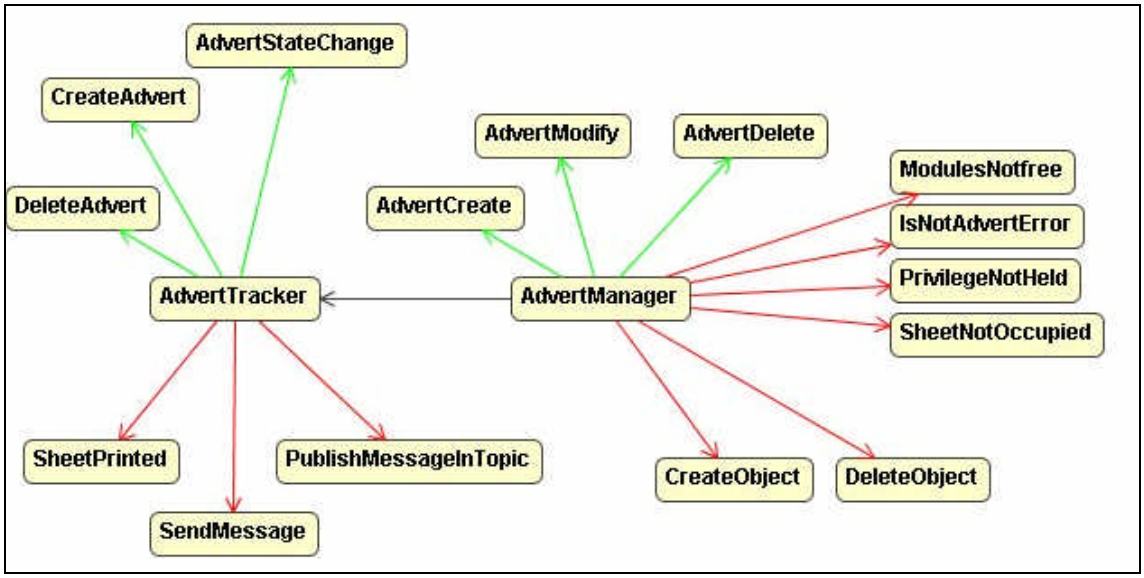
SessionManager



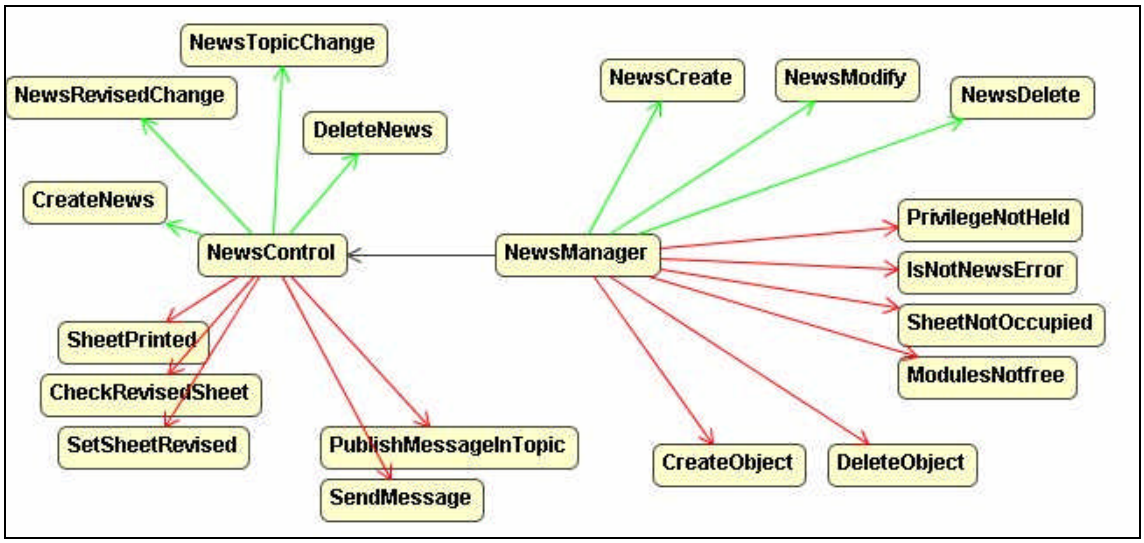
UserManager



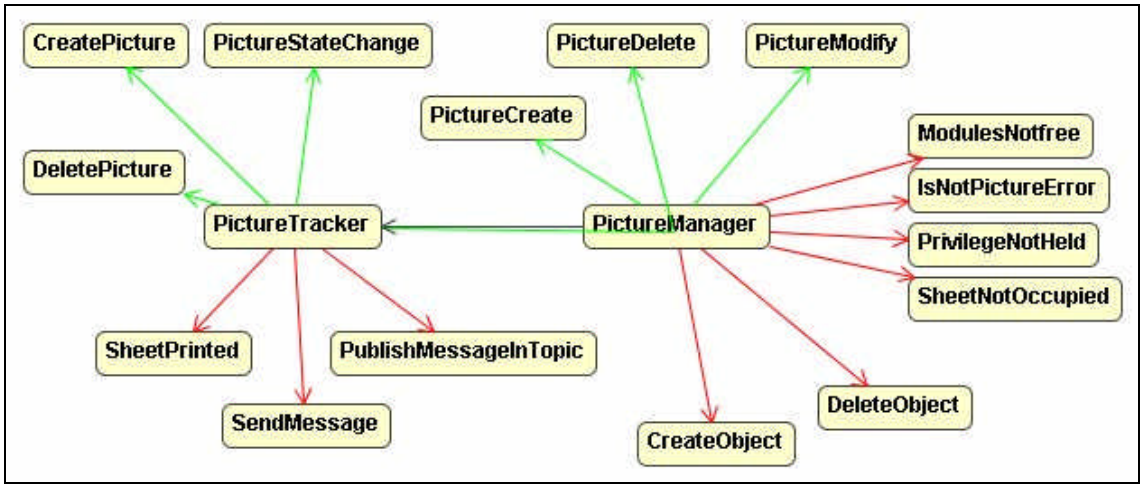
AdvertManager



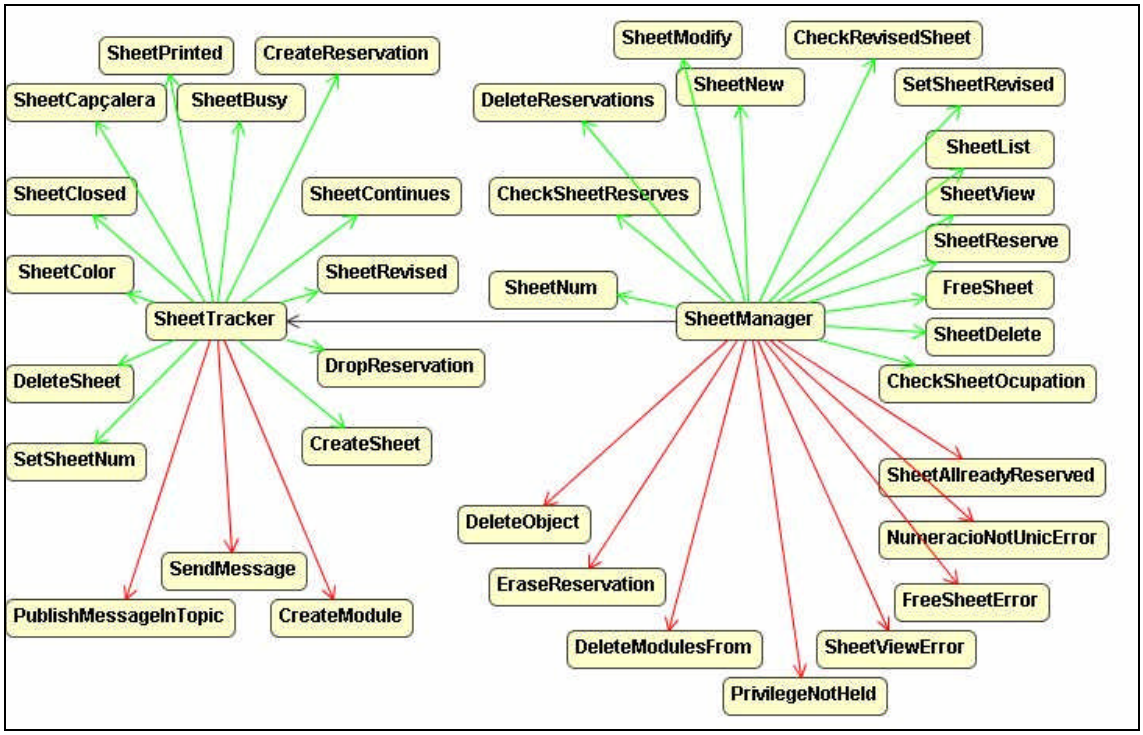
NewsManager



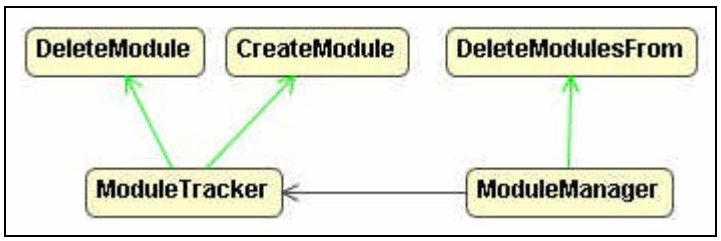
PictureManager



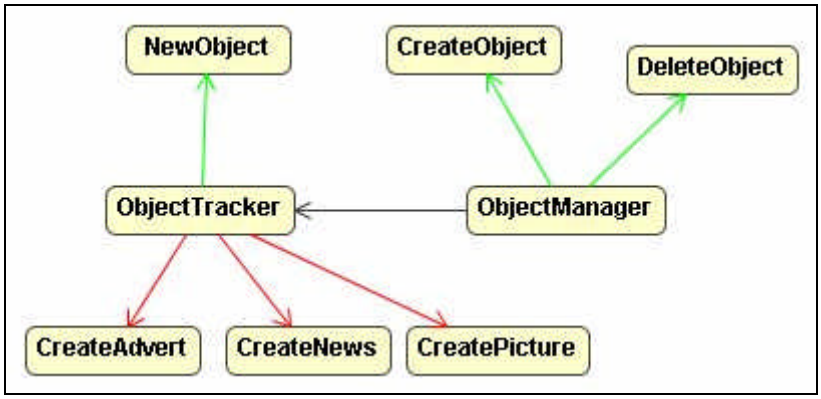
SheetManager



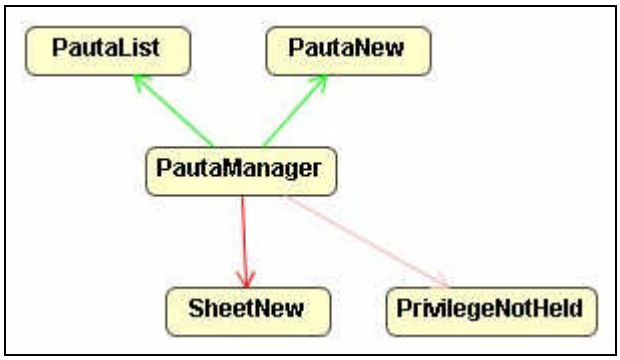
ModuleManager



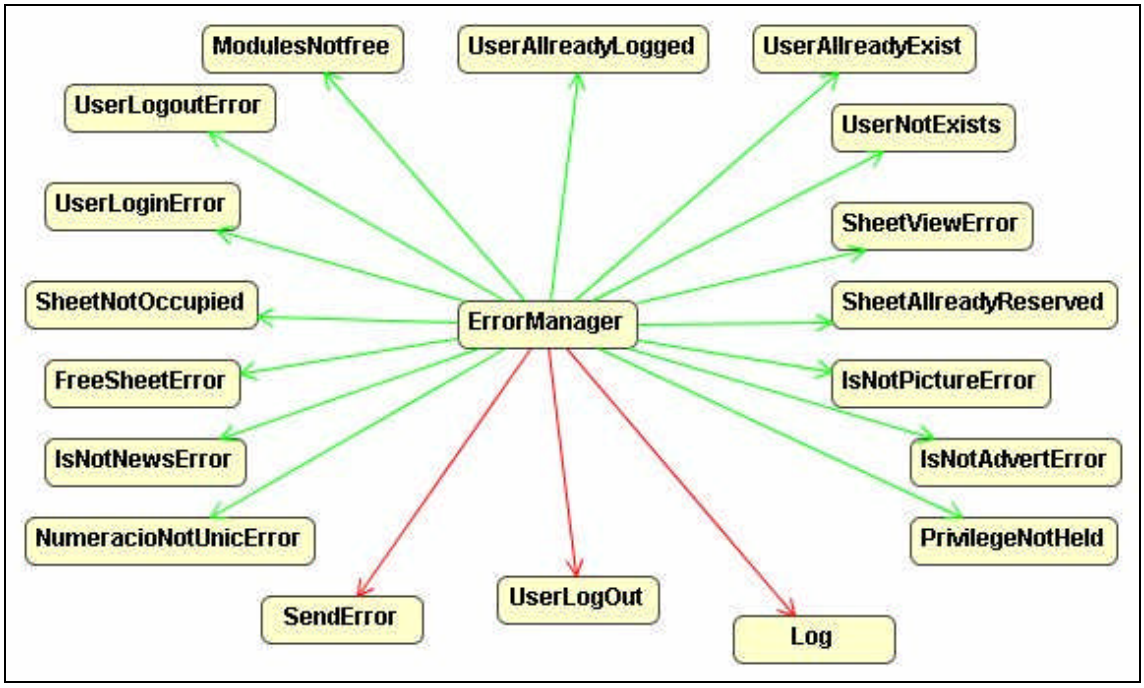
ObjectManager



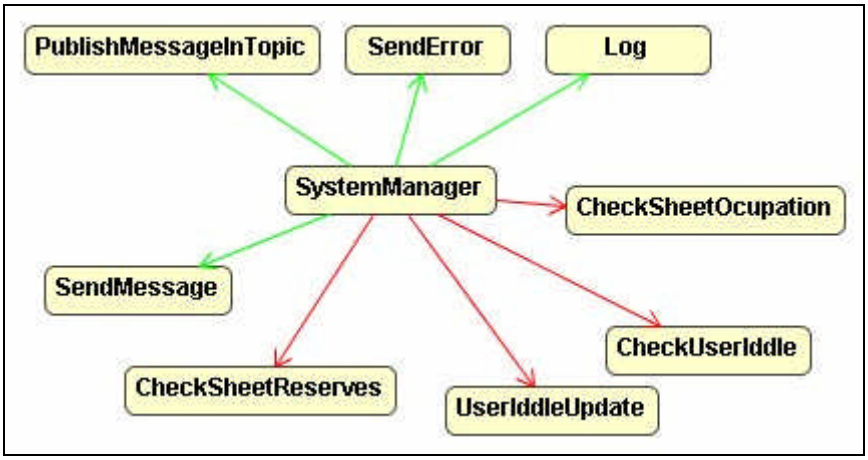
PautaManager



ErrorManager



ObjectManager



9.4 – Especificació de Components

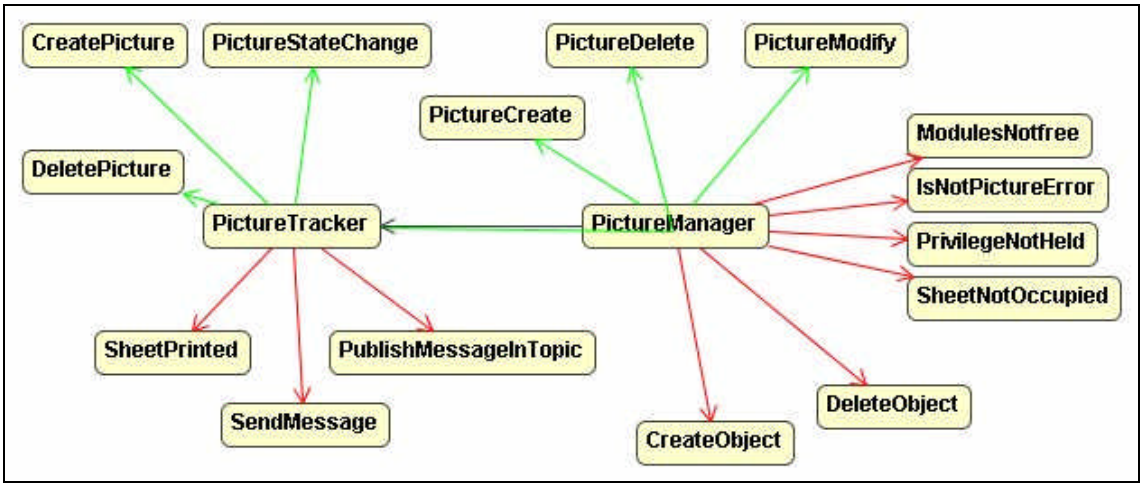
Podem dir que el funcionament de la programació orientada en events es basa en la interpretació seqüèncial d'una sèrie d'events. Aquesta seqüència no es pot produir en qualsevol ordre sinó que ha de seguir uns patrons de comportament. És per aquest motiu que s'utilitzen unes regles per tal d'especificar el comportament dels events dins la seqüencia.

Existeixen dos tipus de regles:

- *Regles de producció*: especifiquen quan un event de sortida ha de ser emès i afegit a la cua d'events.
- *Regles de reducció*: especifiquen com una certa part de la seqüència pot ser reduïda a una més simple sense que se n'alteri el resultat.

Aquestes hauran de complir amb la gramàtica d'events que hem generat anteriorment. Gràcies a la combinació de les regles i la gramàtica podrem comprobar sempre que les seqüències d'events són correctes i en cas contrari veure en quin punt fallen.

A tall d'exemple mostrarem les regles dels components d'uns dels diagrames generats en l'apartat anterior del qual tornem a mostrar a continuació:




```
entity PictureTracker is

    input: CreatePicture, PictureStateChange, DeletePicture
    output: SheetPrinted, PublishMessageInTopic, SendMessage

    reductions

        u.x.v.y.w.t.s ≅ u.v.y.w.t.s where x:CreatePicture[idevent=i]
        ∧ y:PublishMessageInTopic [i ∈ cause] ∧ t:SendMessage [i ∈
        cause]

        /* La Regla vol dir que si hem obtingut un
        PublishMessageInTopic i un SendMessage generats a partir
        del mateix CreatePicture, podem eliminar el CreatePicture
        perquè ja ha fet la feina que havia de fer*/
```

```
u.x.v.y.w.t.s ≅ u.v.y.w.t.s where x:CreatePicture[idevent=i]
∧ y: SendMessage [i ∈ cause] ∧ t: PublishMessageInTopic [i
∈ cause]

/*Repetim amb ordre invers per tal de mostrar que no ens
importa en quin ordre es generen SendMessage i
PublishMessageInTopic*/

u.x.v.y.w.t.s ≅ u.v.y.w.t.s where x:PictureStateChange
[idevent=i] ∧ y:PublishMessageInTopic [i ∈ cause] ∧
t:SheetPrinte [i ∈ cause]

u.x.v.y.w.t.s ≅ u.v.y.w.t.s s where x:PictureStateChange
[idevent=i] ∧ y: SheetPrinted [i ∈ cause] ∧ t:
PublishMessageInTopic [i ∈ cause]


/*Igual que en les reduccions anteriors*/

u.x.v.y.w ≅ u.v.y.w where x:PictureStateChange [idevent=i]
∧ y:PublishMessageInTopic [i ∈ cause]

/*Igual que en les reduccions anteriors*/

/*Veiem que en cap cas hem eliminat SheetPrinted,
PublishMessageInTopic o SendMessage, això és degut a que
aquests events encara han de realitzar funcions en un futur
a la seqüència abans de poder ser eliminats*/


productions

u. CreatePicture[idevent=i, idusuari=x, usuari=y,
newsid=s] .v ->
where
- NewObjectPicture(newid): SendMessage [cause={ i },
idusuari=x, usuari=y, type= "CreatePicture", ObjectMessage=
newObjectMessage (trueObject())] . PublishMessageInTopic
[cause={i}, "PictureTopic" , "Nova fotografia creada per
"+usuari ]

/*Si rebem un event del tipus CreatePicture i la funció
NewObjectPicture retorna 'true' llavors es generen els
events SendMessage i PublishMessageIntopic*/

u.
PictureStateChange [idevent=i, idusuari=x, usuari=y,
idpauta=z, pagina=p, idfotografia=q, state=r]. v
->
where
- changePictureInPlace (idpauta, pagina, idfotografia,
  state) /\ (state==true]: PublishMessageInTopic [cause
  {i}, "pictureTopic" , "L'usuari "+usuari+ " ha donat
  per col·locada la fotografia " +idfotografia+ " a la
```

```
    pàgina "+pagina+ " de la pauta "+idpauta]. SheetPrinted
    [cause={i}, idusuari=x, usuari=y, idpauta=z ,pagina=p,
    false]

- changePictureInPlace (idpauta, pagina, idfotografia,
state) /\ (state==false]: PublishMessageInTopic [cause {i},
"pictureTopic" , "L'usuari "+usuari+ " ha donat com a no
col·locada la fotografia " +idfotografia+ " a la pàgina
"+pagina+ " de la pauta "+idpauta]. SheetPrinted [cause={i},
idusuari=x, usuari=y, idpauta=z ,pagina=p, false]

/* De forma molt semblant al cas anterior, en aquesta
ocasió, però, cambiem el missatge depenent del valor de
state*/

u. DeletePicture[idevent=i, cause={v,u}, idusuari=x,
usuari=y , idobject=n].v ->
where
-deletePictureFromDatabase(idobject)                    /\
deleteObjectFromDatabase(idobject):   PublishMessageInTopic
[cause {i}, "PictureTopic" , "L'usuari "+y+ " ha esborrat
la fotografia "+ idobject]

/* Molt semblant a la primera producció que hem realitzat*/


/* Cal fer notar que alguns casos no estan contemplats en
l'especificació ja que no han sigut tinguts en compte a la
gramàtica. Aquest és un signe que la gramàtica no és
completa i per fant s'hauria de completar amb tots els
cassos possibles com és el cas de quan NewObjectPicture
retorna false en la primera producció*/

eentity
```

A continuació passarem al següent component del diagrama que hereta l'anterior, la qual cosa vol dir que també té les mateixes regles de producció i reducció.

```
entity PictureManager extends PictureTracker is

    input: PictureCreate, PictureDelete, PictureModify
    output: ModulesNotFree, IsNotPictureError, PrivilegeNotHeld,
    SheetNotOccupied, DeleteObject, CreateObject

    reductions

    u.x.v.y.w ≅ u.v.y.w where x:PictureCreate[idevent=i] ∧
    y:CreateObject[i ∈ cause]

    /* Aquesta regla vol dir que si tenim un CreateObject que
    ha estat generat per un Picture Create ja no cal que
    mantenim el PictureCreate*/

    u.x.v.y.w ≅ u.v.y.w where x:PictureCreate[idevent=i] ∧
    y:PrivilegeNotHeld[i ∈ cause]
```

```
u.x.v.y.w ≅ u.v.y.w where x:PictureCreate[idevent=i] ∧
y:SheetNotOccupied[i ∈ cause]

u.x.v.y.w ≅ u.v.y.w where x:PictureCreate[idevent=i] ∧
y:ModulesNotFree[i ∈ cause]

/* Aquestes tres reduccions funcionen de forma semblant a
la primera*/
u.x.v.y.w ≅ u.v.y.w where x:PictureDelete[idevent=i] ∧
y:DeleteObject[i ∈ cause]

u.x.v.y.w ≅ u.v.y.w where x:PictureDelete[idevent=i] ∧
y:PrivilegeNotHeld[i ∈ cause]

u.x.v.y.w ≅ u.v.y.w where x:PictureDelete[idevent=i] ∧
y:IsNotPictureError[i ∈ cause]

/* Exactament la mateixa reducció però aquest cop amb
l'event PictureDelete*/

u.x.v.y.w.t.s ≅ u.v.y.w.t.s where x:PictureModify[idevent=i]
∧ y:PublishStateChange [i ∈ cause] ∧ t:SendMessage [i ∈
cause]

u.x.v.y.w ≅ u.v.y.w where x:PictureDelete[idevent=i] ∧
y:PrivilegeNotHeld[i ∈ cause]

u.x.v.y.w ≅ u.v.y.w where x:PictureDelete[idevent=i] ∧
y:IsNotNews[i ∈ cause]

/* Novament tenim un cas molt semblant a l'anterior. Com, a
novetat, però, veiem que les reduccions d'un mateix event
no han de tenir sempre el mateix nombre d'elements. Veiem
que en la primera reducció intervenen tres elements mentre
que en les altres dues només hi intervenen dos*/


productions

u. PictureCreate [idevent=i, idusuari=x, usuari=y,
idpauta=z, pagina=p, tema=t, moduls=M].v ->
where
-[isSheetOcuppiedBy (idusuari, idpauta,pagina) /\
isEditor(idusuari) /\ areModuleFree(moduls)]:
CreateObject[cause={i}, idusuari=x, usuari=y, idpauta=z,
pagina=p, tema=t, moduls=M, type="picture"]

-!isEditor(idusuari): PrivilegeNotHeld
[cause={i} ,idusuari=x, usuari=y, privilege="Editor"]

-[!isSheetOcuppiedBy (idusuari, idpauta,pagina) /\
isEditor(idusuari)]: SheetNotOcuppied
[cause={i} ,idusuari=y,usuari=z, error="The sheet isn't
ocuppied "]
```

```
-[isSheetOcuppiedBy (idusuari, idpauta,pagina) /\
isEditor(idusuari) /\ !areModuleFree(moduls)]:
ModulesNotFree [cause={i} ,idusuari=y,usuari=z, error="The
modules aren't free "]


/* Com podem comprobar en aquesta ocasió tenim moltes més
sortides. A destacar el comportament que s'ha triat per la
generació d'errors. Podem veure que com a molt es generarà
sempre un sol error. Aquesta ha estat una decisió de
disseny ja que es podria generar més d'un si a cada error
tan sols li assignéssim una comprovació.*/



u. PictureDelete [idevent=i idusuari=x, usuari=y, idpauta=z,
pagina=p, idfotografia=q].v ->
where
-[isEditor()) /\ isSheetOcuppiedBy (idusuari,
idpauta,pagina) /\ isPicture(idfotografia)]:
DeleteObject [cause={i},idevent=v, idusuari=x, usuari=y,
idpauta=z, pagina=p, idfotografia=q]

-[!isEditor (idusuari) V !isSheetOcuppiedBy (idusuari,
idpauta,pagina)] : PrivilegeNotHeld [cause={i} ,idusuari=x,
usuari=y, privilege="PictureDelete"]

-[!isPicture(idfotografia)]: IsNotPictureError [cause={ i },
idusuari=x ,usuari=y, idpauta=z, pagina=p, idfotografia=q,
error="L'element no és una fotografia"]


/* Exactament el mateix que en les produccions anteriors,
res de nou a afegir*/


u. PictureModify [idevent=v, idusuari=x, usuari=y,
idpauta=z, pagina=p, idfotografia=q, state=r].v ->
where

-[[isEditor(idusuari) V isPhotographer(idusuari) V
isPrinter(idusuari)] /\ isSheetOcuppiedBy (idusuari,
idpauta,pagina) /\ isPicture(idfotografia) ] :
PictureStateChange [cause={i}, idusuari=x, usuari=y,
idpauta=z, pagina=p, idfotografia=q, state=r] . SendMessage
[cause={i}, idusuari=x, usuari=y, type= "UserModify",
ObjectMessage= newObjectMessage (trueObject())]

-[[! isPhotographer(idusuari) V !isEditor(idusuari)
V !isPrinter(idusuari) V !isSheetOcuppiedBy (idusuari,
idpauta,pagina)] /\ isPicture(idfotografia)}:
PrivilegeNotHeld [cause={v},idusuari=x, usuari=y,
privilege="PictureModify"]
```

```
-[!isPicture(idfotografia)]: IsNotPictureError [cause={ i },
idusuari=x ,usuari=y, idpauta=z, pagina=p, idfotografia=q,
error="L'element no és una fotografia"]

/* Tal com hem dit abans en aquesta producció podem
comprobar que cada una de les branques pot generar un
nombre diferent d'events*/
```

eentity

Tal i com podem veure per poder dur a terme l'especificaicó dels components hem hagut de fer servir tant els diagrames de components, que ens han permès identificar les diferents entitats així com el lèxic que s'havia realitzat anteriorment.

9.5 - Funcions

Les funcions, tot i que podrien haver estat considerats events atòmics o events complex bàsics, han estat considerats prou simples per executar ordres sense necessitat de generar events de forma directa. La gran diferència entre un event i una funció és que la funció és l'encarregada d'aportar valors a les variables que es fan servir als events.

A continuació detallem totes les funcions que apareixen a la nostra aplicació i que hauran de ser implemetnades.

actualDate()
Retorna la data i hora actual del processador d'events

authenticate(password,idusuari)
Retorna true o false si el login i el password són correctes

changeAdvertInPlace (idpauta, pagina, idadvert, state)

Canvia l'atribut de colocació d'un anunci a la base de dades

changeBusyState (idpauta,pagina, estat)

Canvia l'atribut d'ocupació d'una pàgina a la base de dades

changeCapçalera (idpauta, pagina, capçalera)

Canvia la capcelera d'una pàgina al a base de dades

changeClosed (idpauta, pagina, tancada)

Canvia l'atribut per indicar si una pàgina està tancada a la base de dades

changeColor (idpauta, pagina, color)

Canvia l'atribut de color d'una pàgina a la base de dades

changeContinuaa (idpauta, pagina, continua)

Canvia l'atribut quina pàgina ve a continuació de l'actual a la base de dades

changePictureInPlace (idpauta, pagina, idfotografia, state)

Canvia l'atribut de col·locació d'una fotografia a la base de dades

changePrinted (idpauta, pagina, printed)

Canvia l'atribut d'impressió d'una pàgina a la base de dades

changeRevised (idpauta, pagina, revised)

Canvia l'atribut de correcció d'una pàgina a la base de dades

changeRevision (idpauta, pagina, idarticle, corregit)

Canvia l'atribut de correcció d'una notícia a la base de dades

changeVede (idpauta, continua ,pagina)

Canvia l'atribut que indica de quina pàgina prove la pàgina indicada a la base de dades

createQueue (idusuari)
Crea un canal de tipus cua per a un usuari per tal que pugui rebre missatges

createUser (idusuari, password, usuari, role)
Crea un nou usuari amb un rol determinat

deleteAdvertFromDatabase(idobject)

Esborra un anunci de la base de dades

deleteNewsFromDatabase(idobject)

Esborra una notícia de la base de dades

deleteObjectFromDatabase(idobject)

Esborra un Objecte de la base de dades

deletePictureFromDatabase(idobject

Esborra una fotografia de la base de dades

deleteSheetFromDatabase (idpauta,pagina)
Esborra una pàgina de la base de dades

dropUser(dropid)
Marca un usuari com esborrat a la base de dades. No s'esborra realment per motius d'historial

falseObject()
Retorna un missatge de tipus ObjectMessage amb un boolean amb valor false;

getAdverts(idpauta, pagina)
Retorna una llista amb els objectes de tipus anunci d'una pàgina per a una pauta en concret

getNewPautaID()
Retorna un nou i únic identificador per a una nova pauta.

getNews(idpauta, pagina)
Retorna una llista amb els obejctes tipus notícia d'una pàgina i pauta concretes

getNewSheetID(idpauta)
Retorna un nou i únic identificador per a una nova pàgina.

getPautaList()
Retorna un llistat amb totes les pautes existents a la base de dades

getPictures(idpauta, pagina)
Retorna una llista amb els objectes tipus fotografia per una pàgina i pauta concretes

getRole (idusuari)
Retorna el rol d'un usuari

getSheetDateOcupattion(idpauta,pagina)
Retorna la data i hora en que va ser ocupada una pàgina

getSheetDateReservation (idpauta,pagina)
Retorna la data i hroa en que va ser feta una reserva

getSheetHeight();

Retorna el nombre de files de les pàgines d'una pauta

getSheetList(options)
Retorna un llistat amb les pàgines i la informació demanada per les opcions. Les opcions s'expliquen a l'event que crida aquesta funció.

getSheetsOccupiedBy(idusuari)

Retorna un llistat amb les pàgines que actualment està ocupant un usuari.

getSheetsReservedBy(idusuari)
Retorna un llistat amb les pàgines que estan actualment reservades

getSheetWidth();

Retorna el nombre de columnes de les pàgines d'una pauta

getUserInfo() Retorna les dades d'un usuari de la base de dades

getUserList()
Retorna un llistat amb tots els usuaris existents

getUsersLogged()
Retorna la llista d'usuaris connectats

hasSharedNews(idpauta, pagina)
Comprova si una pàgina té alguna notícia compartida amb una altra pàgina.

hasSheetReserved (idusuari,idpauta,pagina)
Comprova si un usuari té una pàgina d'una pauta en concret reservada

isAdmin(idusuari)
Comprova si l'usuari és un administrador

isAdvert(idadvert)
Comprova si l'objecte es un anunci

isDirector(idusuari)
Comprova si l'usuari és un director

isEditor(idusuari)
Comprova si l'usuari és un redactor

isModuleFree(m)
Comprova si el modul m està lliure

i**sNews**(idarticle)
Comprova si l'objecte és de tipus notícia

isNewsDoneBy (idarticle,idusuari)
Comprova si una notícia està feta per un usuari en concret

isNewsRevised(idpauta, pagina)
Comprova si una pàgina està corregida

isPhotographer(idusuari)
Comprova si l'usuari és un fotògraf

isPicture(iidfotografia)
Comprova si l'objecte és una fotografia

isPrinter(idusuari)
Comprova si l'usuari és un maquetador

isSheetClosed (idpauta, pagina)
Comprova si una pàgina està tancada

isSheetFree (idpauta,pagina) (recordar-se filla de, continua a la pàg)
Comprova si una pàgina i les seves pàgines 'mares' i 'filles' estan lliure s

isSheetOcuppiedBy (idusuari, idpauta,pagina)
Comprova si una pàgina està ocupada per un usuari en concret

isSheetReserved (idpauta,pagina)
Comprova si una pàgina està ocupada

isSpeller(idusuari)
Comprova si un usuari es un corrector

loggedOn(idusuari)
Comprova si un usuari està actiu a la base de dades

newObjectAdvert(newisd)

Crea un nou anunci a la base de dades

newObjectMessage(Object o**)**

Crea un nou objecte a la base de dades

newObjectNews(newisd, idusuari)

Crea una nova notícia a la base de dades

newObjectPicture(newisd)

Crea una nova fotografia a la base de dades

numeracioUnica(numeracio)
Comprova que donada una numeració no hi hagi cap número corregit i aquesta sigui correlativa

sheetComesFrom(idpauta,continuaa)
Retorna la pàgina de la que prové una altra

sheetExists(idpauta, pagina)
Comprova si una pàgina d'una pauta existeix

sheetInfo(idpauta,pagina)
Retorna tota la informació d'una pàgina i de les que estan relacionades amb ella per ser 'mares' o 'filles'

trueObject()
Retorna un missatge de tipus ObjectMessage amb un boolean amb valor true;

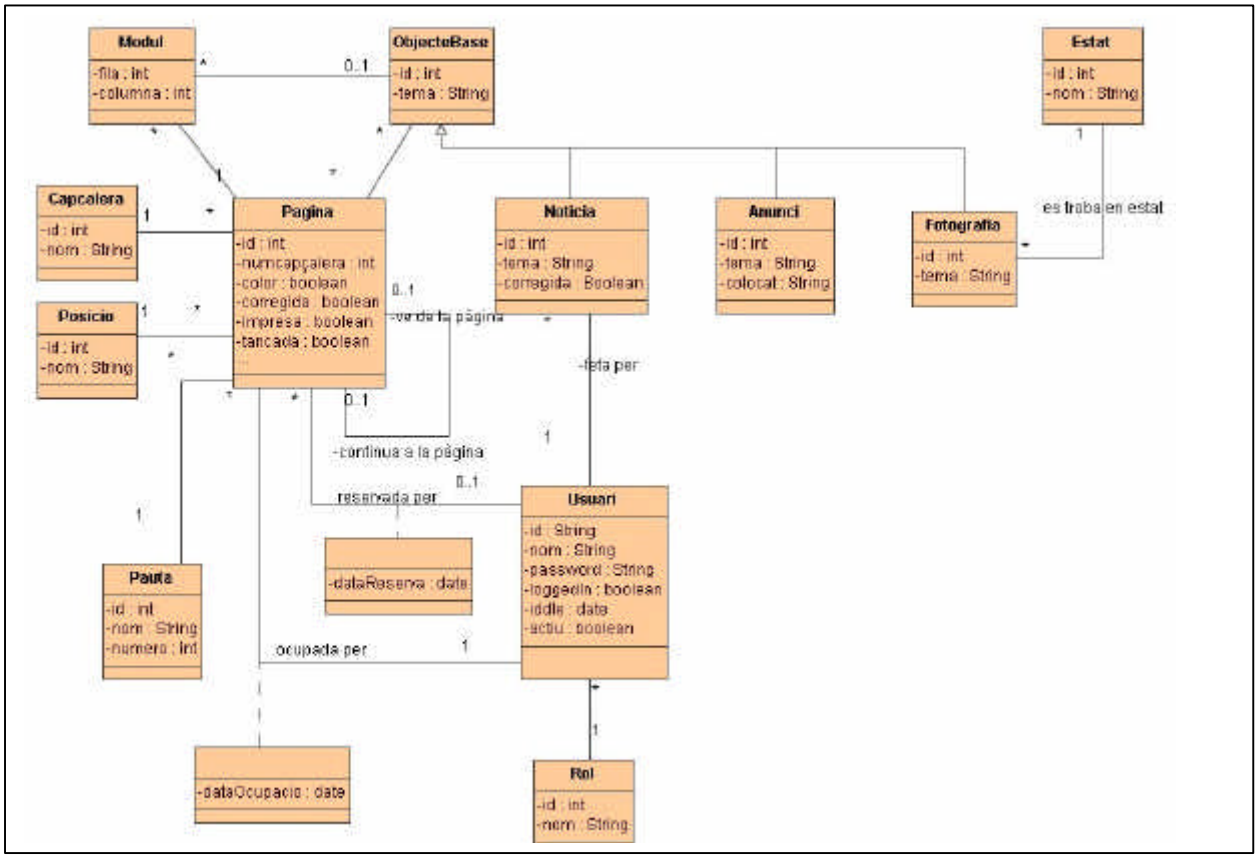
userExist (newidusuari)
Mira si un usuari en concret existeix a la base de dades

10 - Diagrama estàtic de disseny

El següent pas a realitzar ha estat realitzar el diagrama estàtic de disseny per una tecnologia orientada a objectes. Per tal d'estalviar feina, hem tingut en compte ja alguns dels aspectes que no es poden implementar en java i alguns que no es poden implementar en bases de dades MySQL per tal de simplificar l'operació.

Cal destacar que el diagrama de disseny en aquest punt tan sols ens servirà per crear el model de dades. El disseny de classes final es realitzarà posteriorment tenint en compte el present per tal d'integrar l'aplicació correctament en un model MVC.

Tenint en compte les característiques que hem especificat anteriorment podem crear el següent diagrama de classes.



Tal com hem dit a l'hora de realitzar aquest disseny ja hem tingut en compte aspectes com l'adaptació de l'herència a nivell de JAVA , la transformació de les classes associatives en relacions que si són suportades o la transformació de les associacions n-àries.

Amb tot i això existeixen dues associacions qualificades per la qual cosa les haurem d'adaptar a Java. Com que aqeustes es troben en relacions *..1 el que farem serà convertir-les en atributs de la classe de l'extrem *

Un cop fet aquests passos el disseny resultat sols conté herència simple i relacions binàries entre classes. Per tant el disseny és directament implementable per Java simplement substituint les relacions per nous atributs a les classes participants que guardin ("apuntin") els objectes de l'altra classes amb les que

l'objecte està relacionats.

Així doncs, un cop fet el diagrama de disseny, ja podem generar el codi Java corresponent. Val a dir que no a tot arreu hem assumit una navegació bidireccional perquè no ho hem cregut necessari. A més a més, per poder utilitzar alguns objectes dins missatges JMS, els hem convertit en implementacions de la interfície 'java.io.Serializable'

```
Public Class ObjecteBase implements Serializable {
        Private int id;
        Private String tema;
        Private Modul modul[];
        Private Pagina pagina[];
}

Public Class Pauta {
        Private int id;
        Private String nom;
        Private int num;

}

Public Class Modul implements Serializable{
        Private int fila;
        Private int columna;
Private Pagina pagina;
        Private ObjecteBase objecteBase;
}

Private Class Posicio {
        Private int id;
        Private String nom;
}


Private Class Capçalera  {
        Private int id;
        Private String nom;
}


Private Class Estat {
        Private int id;
        Private String nom;
}

Public Class Pagina implements Serializable {
        Private int id;
        Private Pauta pauta;
        Private Boolean color;
        Private Boolean corregida:
        Private Boolean impresa;
        Private Boolean tancada;
        Private Posicio posicio;
        Privata Capcalera capcalera;
        Private Pagina continuaalapàgina;
        Private Pagina vedelapagina
        Private Usuari reservadaPer;
        Private Date dataReserva;
```

```
        Private User ocupadaPer;
        Private Date dataOcupacio;

    }


    Public Class Noticia extends ObjecteBase {
        Private Boolean corregida;
        Private Usuari fetaPer;
    }


    Public Class Anunci extends ObjecteBase {
        Private Boolean colocat;
    }

    Public Class Image extends ObjectBase{
        Private Estat estat;
    }

    Private Class Usuari implements Serializable {
        Private int id;
        Private String nom;
        Private String password;
        Private Rol rol;
    }

    Private Class Rol  implements Serializable {
        Private int id;
        Private String nom;
    }
```

Val a dir que amb tot i que aquestes classes ens serviran d'ajuda com a guia és molt probable que en necessitem d'addicionals per poder gestionar els events, com és el cas de llistes d'objectes. El disseny bàsic que hem realitzat, però ens servirà pel següent pas que serà l'obtenció dels objectes que formaran la base de dades de l'aplicació.

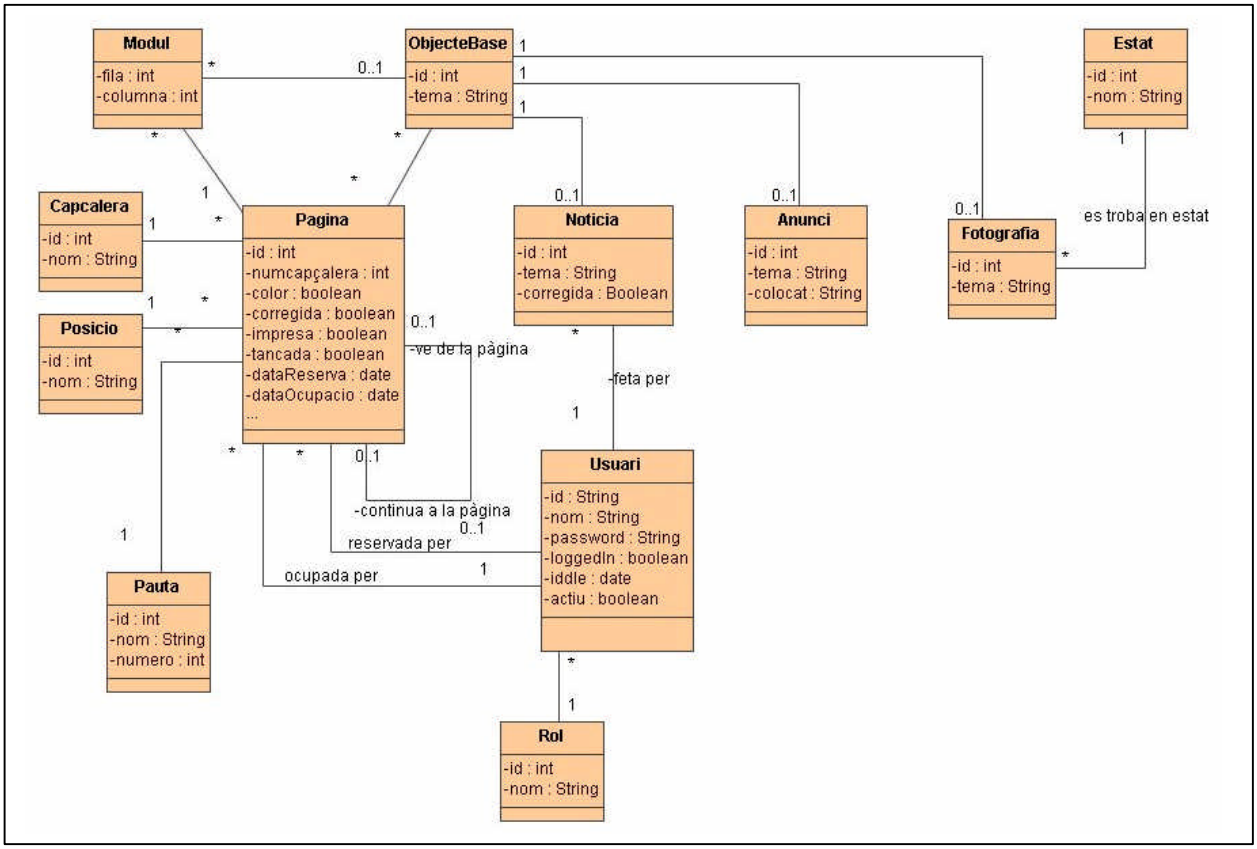
11 - Disseny de la base de dades relacional

A partir del diagrama de classes obtingut a la fase d'anàlisis, per a dissenyar la base de dades relacional hem de seguir una sèrie de passos.

Eliminació de l'herència.

Les bases de dades relacionals no ofereixen cap tipus de suport per l'herència, per tant fins i tot l'herència simple s'ha d'eliminar de l'esquema. Com que en aquest cas la classe pare ObjecteBase participa de relacions i a més la seva informació bàsica ens pot servir per fer resums senzills de continguts, hem optat per manterni-la i convertir les herències en relacions 1:*.

El diagrama ens queda doncs de la següent manera:



Un cop fet el tractament de l'herència ja estem en condicions de crear el conjunt de taules relacions que ens permetran guardar la informació del nostre sistema de forma permanent.

Per a fer-ho convertirem cada classes a una taula relacional. Els atributs de la classe passen a ser camps de la taula. Totes les taules necessita una clau primària, que és un atribut conjunt d'atributs que permeten identificar cada fila ("objecte") de la taula. El concepte de clau primària ja l'hem tingut en compte a l'hora de realitzar el primer disseny i tan sols tindrem algun cas especial com la classe "Modul" en que serà necessari que més d'un camp sigui clau primària a més de la relació amb la classe pàgina.

El següent pas és transformar les associacions. Cada associació de tipus 1:* la convertirem de tal forma que afegirem a la taula que representa la classe del costat * els atributs que formen la clau primària de la taula que representa la classe del costat 1. Aquests atributs quedaran marcats com claus foranes. Una

clau forana obliga a què el valor del nou atribut sigui vàlid, és a dir, que sigui un dels valors existents dins l'altra taula. L'altra taula queda inalterada ja que el model relacional no permet atributs amb valors múltiples (atributs de tipus vector)

Per cada associació *:* (relacions on ambdós extrems tenen una cardinalitat >1) quedaran convertires en una nova taula que té com atributs les claus primàries de les taules que representen les classes participants en l'associació. Aquests atributs es marquem com a clau forana. La suma de tots ells és la clau primària de la nova taula.

Un cop fets aquests canvis, el conjunt de taules relacions obtinguts és el següent:

Pauta (id, nom, numero)

Capcalera (id, nom)

Posicio (id, nom)

Rol (id, nom)

Estat (id, nom)

ObjecteBase (id, tema)

Usuari (id, nom, password , rol, loggedIn, iddle, actiu)
{rol} és clau forana a Rol

Pagina (id, idpauta, numero, color, corregida, impresa, tancada, capcalera, posicio, continuaA, veDe, reservadaPer, dataReserva, ocupadaPer, dataOcupacio)
{idpauta} és clau forana a Pauta
{capcalera} és clau forana a Capcalera
{posicio} és clau forana a Posicio
{continuA} és clau forana a Pagina
{veDe} és clau forana a Pagina
{reservadaPer} és clau forana a Usuari

Modul (fila, columna, pagina, idpauta, objecteBase)
{pagina, idpauta } és clau forana a Pagina
{objecteBase} és clau forana a ObjecteBase

ObjecteAPagina (idPagina, idpauta, idObjecte)
{idPagina, idpauta } és clau forana a Pagina
{idObjecte} és clau forana a ObjcteBase

Noticia (id, corregida, fetaper)
{id} és clau forana a ObjecteBase
{fetaper} és clau forana a Usuari

Anunci (id, colocat)
{id} és clau forana a ObjecteBase

Fotografia (id, estat)
{id} és clau forana a ObjecteBase
{estat} és clau forana a Estat

12 - La gestió dels missatges

A banda d'aquestes classes i taules a la base de dades en necessitarem d'addicionals per gestionar els missatges i les subscripcions. OpenJMS, el middleware que utilitzarem per al tractament de missatges, ens obliga a fer servir unes taules en concret. El codi de creació d'aquestes taules per a MySql és:

```
create table system_data (
    id              bigint not null,
    version         varchar(20) not null,
    creationDate    DATE not null
  );
create unique index system_data_pk on system_data(id);

create table seeds (
    name            varchar(20) not null,
    seed            bigint not null
);
create unique index seeds_pk on seeds(name);

create table destinations (
    name            varchar(255) not null,
    isQueue         int not null,
    destinationId   bigint not null
);
create unique index destinations_pk on destinations(name);

create table messages (
    messageId       varchar(64) not null,
    destinationId   bigint not null,
    priority        int,
    createTime      bigint not null,
    expiryTime      bigint,
    processed       int,
    messageBlob     longblob not null
);
create index messages_pk on messages(messageId);

create table  message_handles (
    messageId       varchar(64) not null,
    destinationId   bigint NOT NULL,
    consumerId      bigint NOT NULL,
    priority        int,
    acceptedTime    bigint,
    sequenceNumber  bigInt,
    expiryTime      bigint,
    delivered       int
);
create index message_handles_pk ON message_handles(destinationId, consumerId, messageId);

create table consumers (
    name            varchar(255) not null,
    destinationId   bigint not null,
    consumerId      bigint not NULL,
    created         bigint NOT NULL
  );
create unique index consumers_pk on consumers(name, destinationId);
```

```
create table users (
    username        varchar(50) not null,
    password        varchar(50) not null
);
create unique index users_pk on users(username);
```

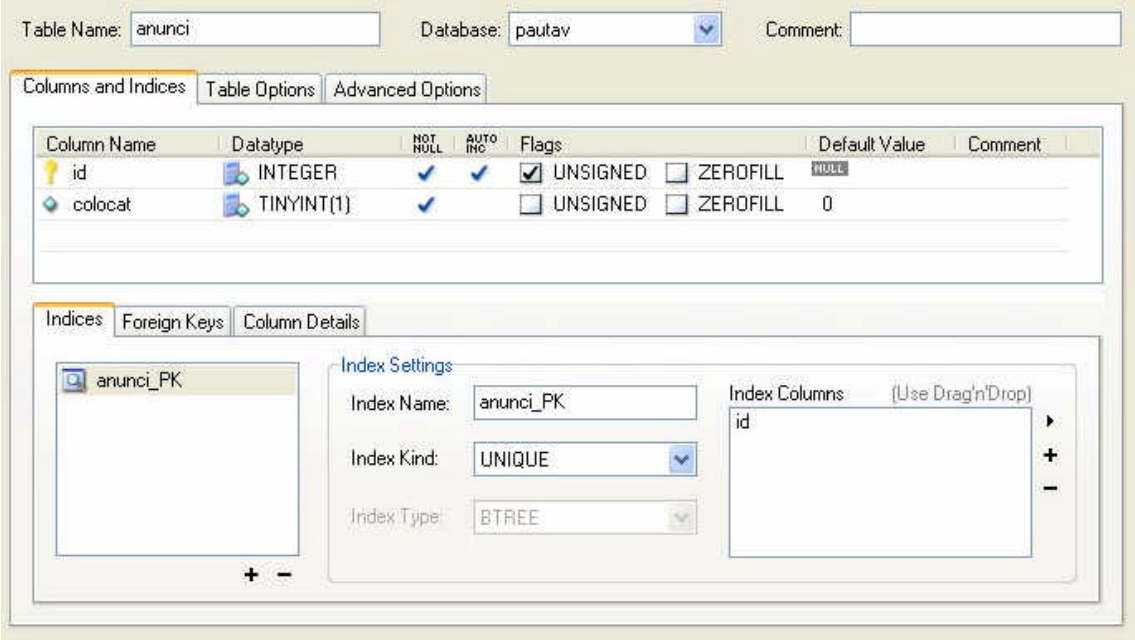
Podem observar que aquesta darrera taula s'assembla molt a la taula usuaris que nosaltres hem dissenyat prèviament. Per tant optarem per fusionar les dues utilitzant únicament aquesta darrera afegint els atributs necessaris i utilitzant al codi el nostre propi sistema de creació d'usuaris en lloc del que ofereix OpenJms.

13 - Creació de la base de dades

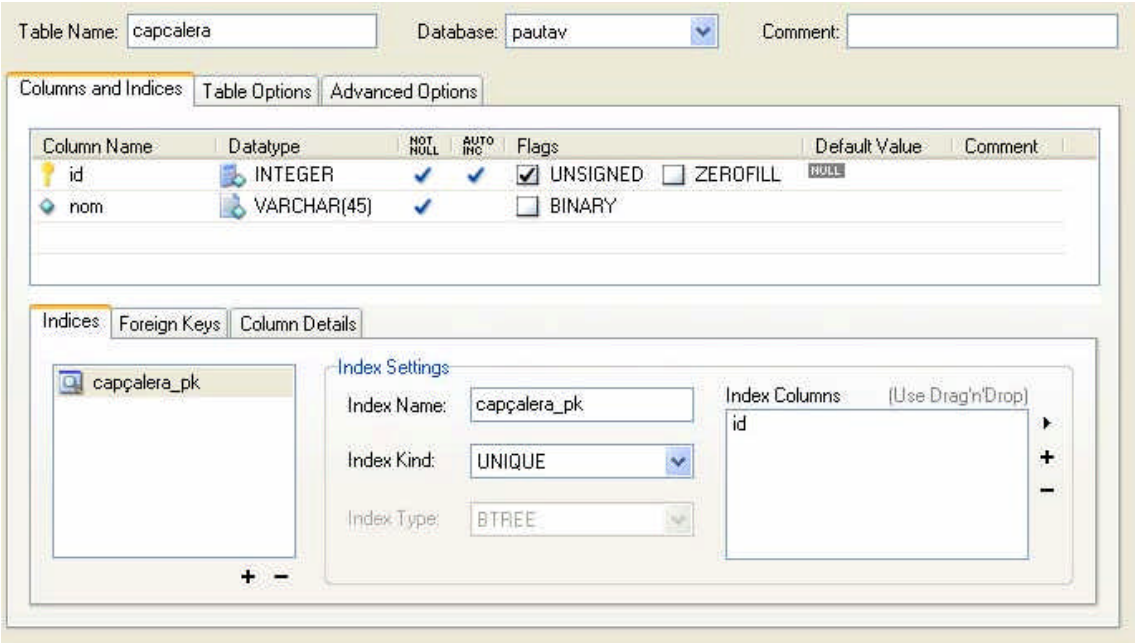
Si unim tota la informació anterior i procedim a crear la base de dades (hem utilitzat el MySQL_Browser proporcionat per la mateixa base de dades per facilitar la feina), obtenim el següent resultat:

Nota: Cal destacar que en els resums les claus foranes apareixen com a índex, les podrem distingir perquè acaben amb les sigles _FK en lloc dels índes de les claus primàries _PK

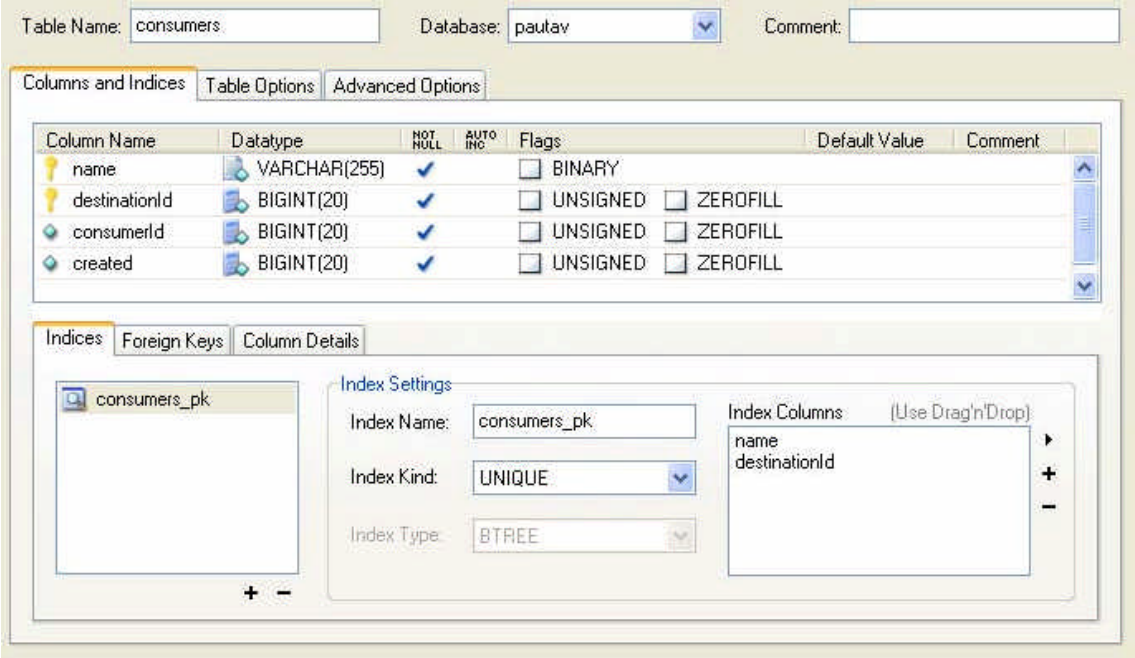
Anunci



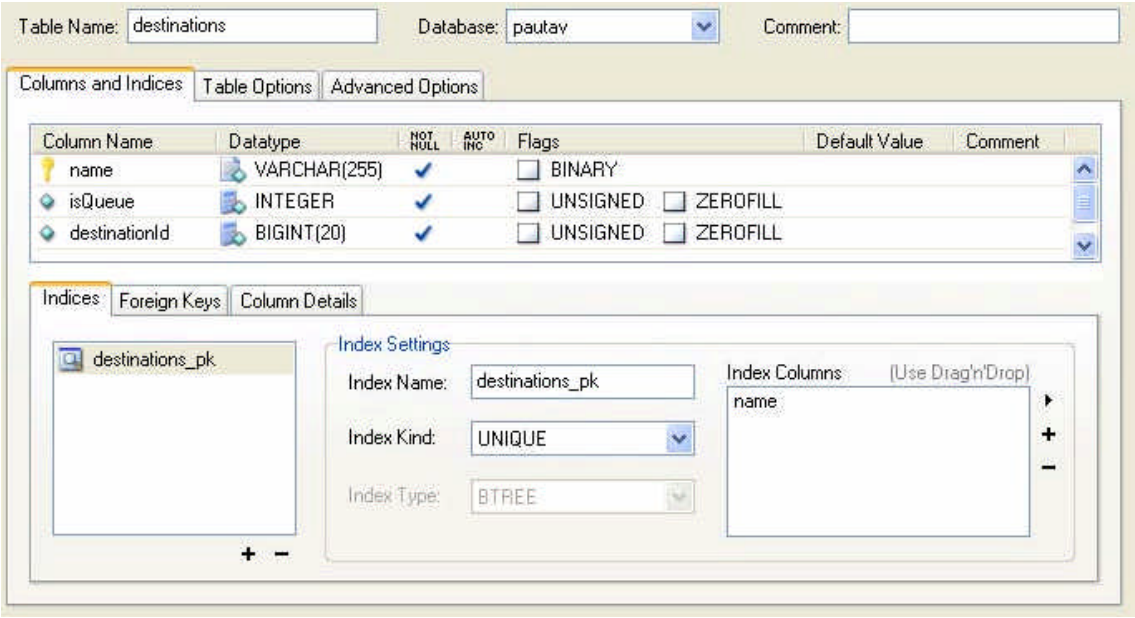
Capcalera



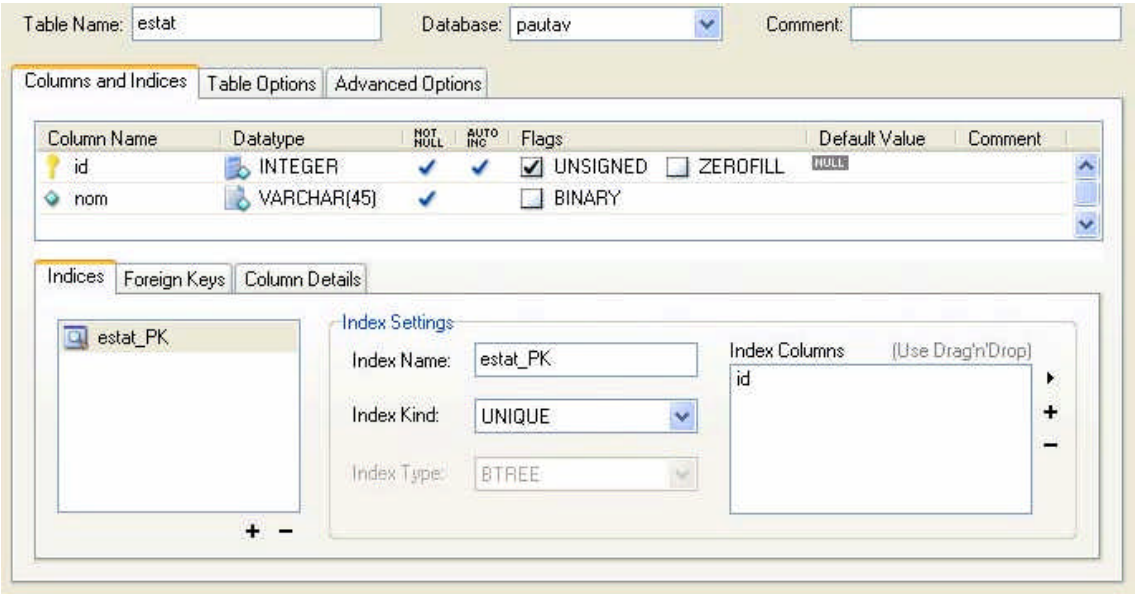
Consumers



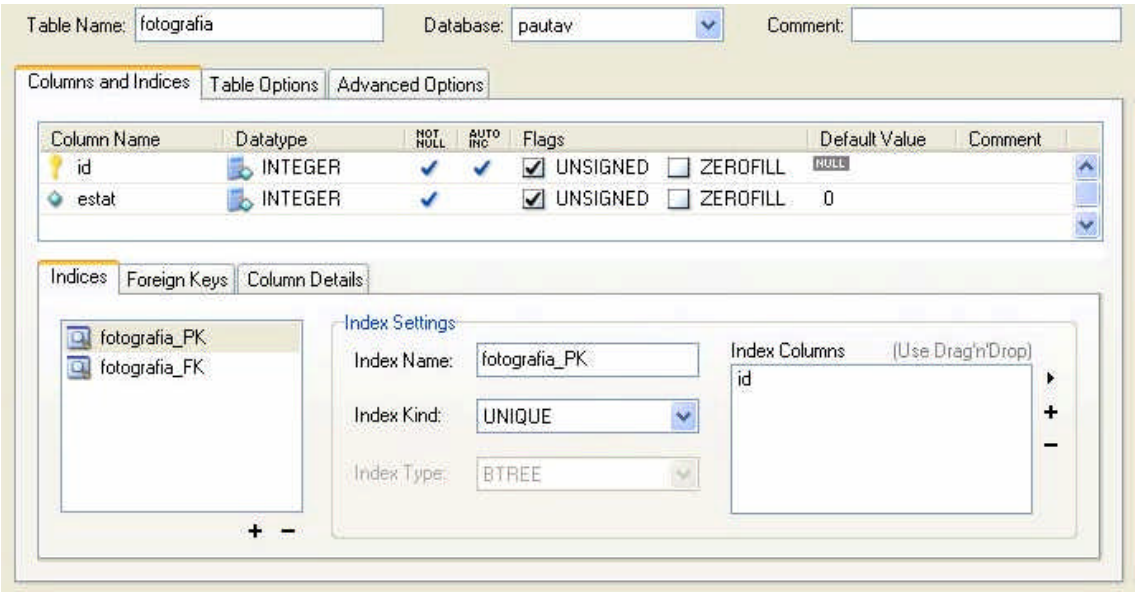
Destinations



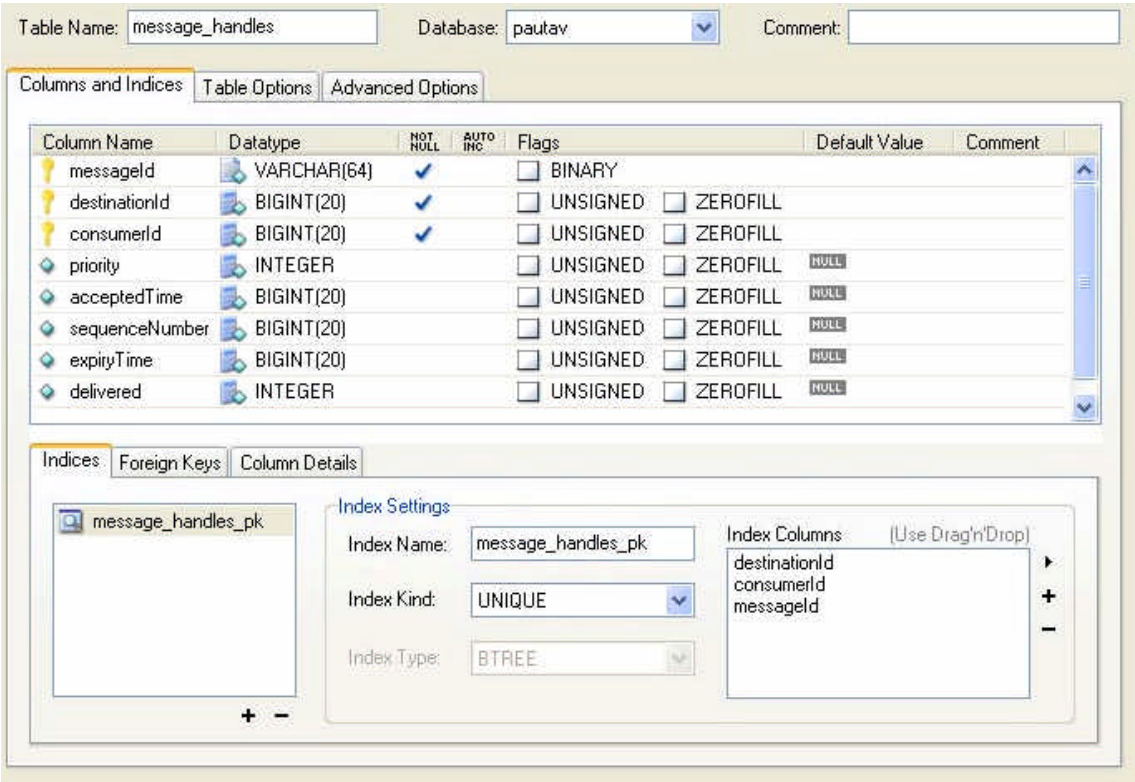
Estat



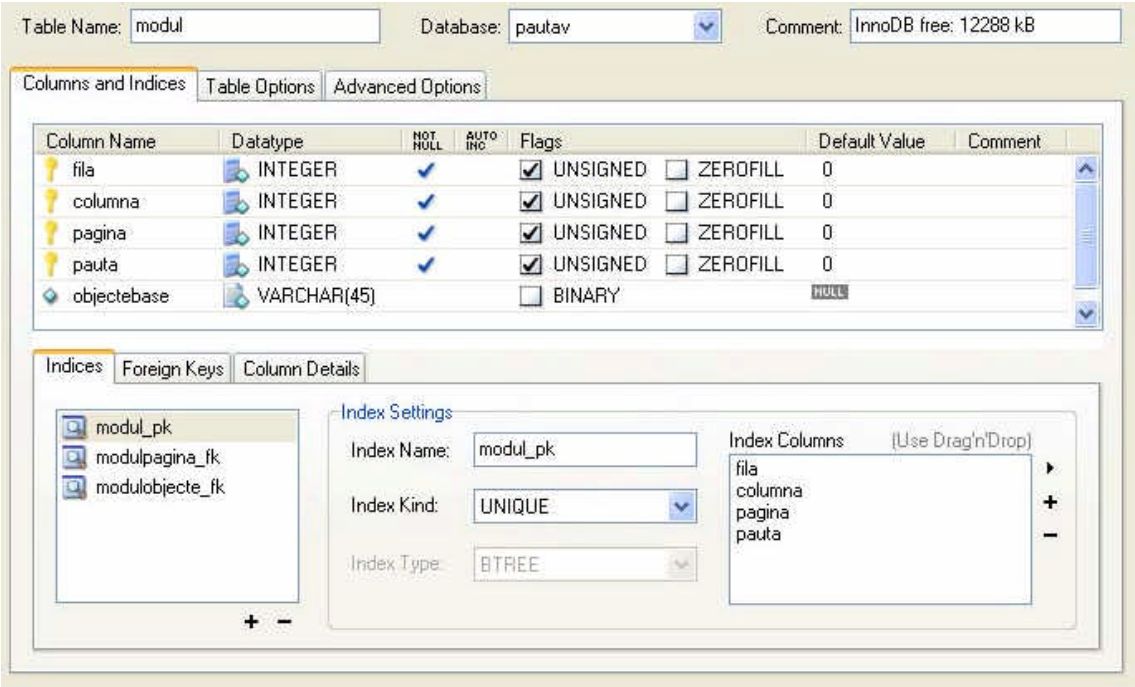
Fotografia



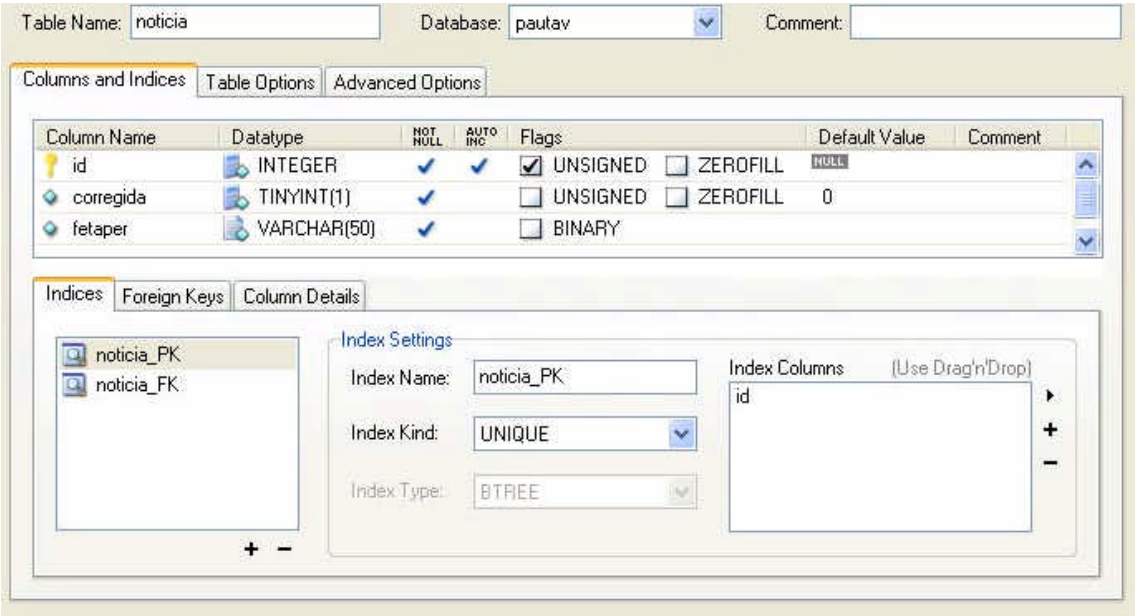
message_handles



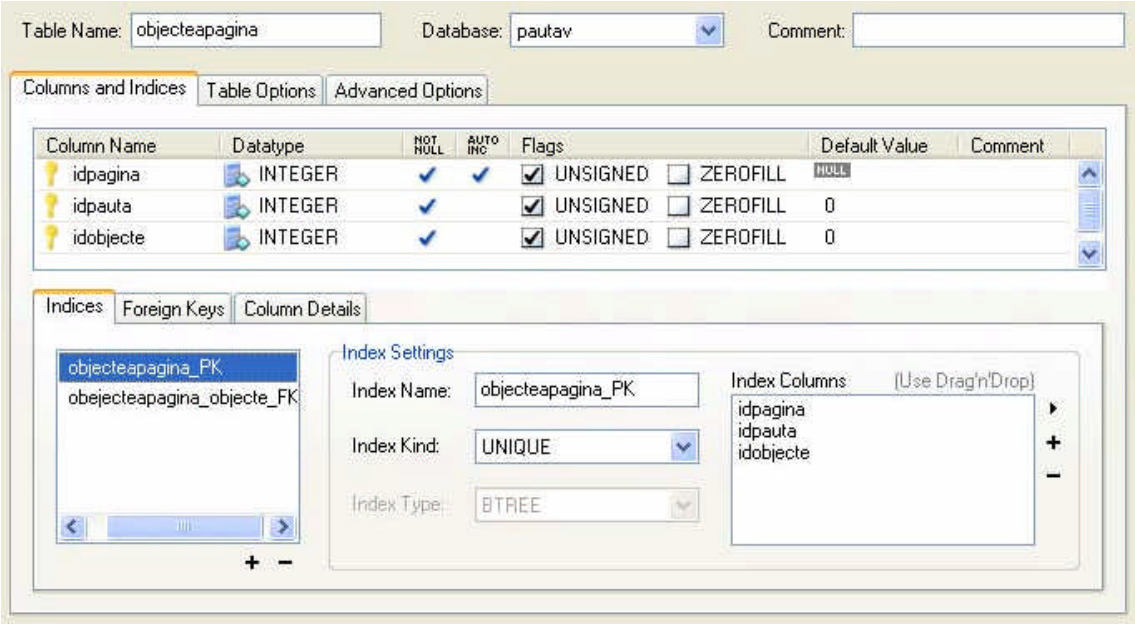
modul



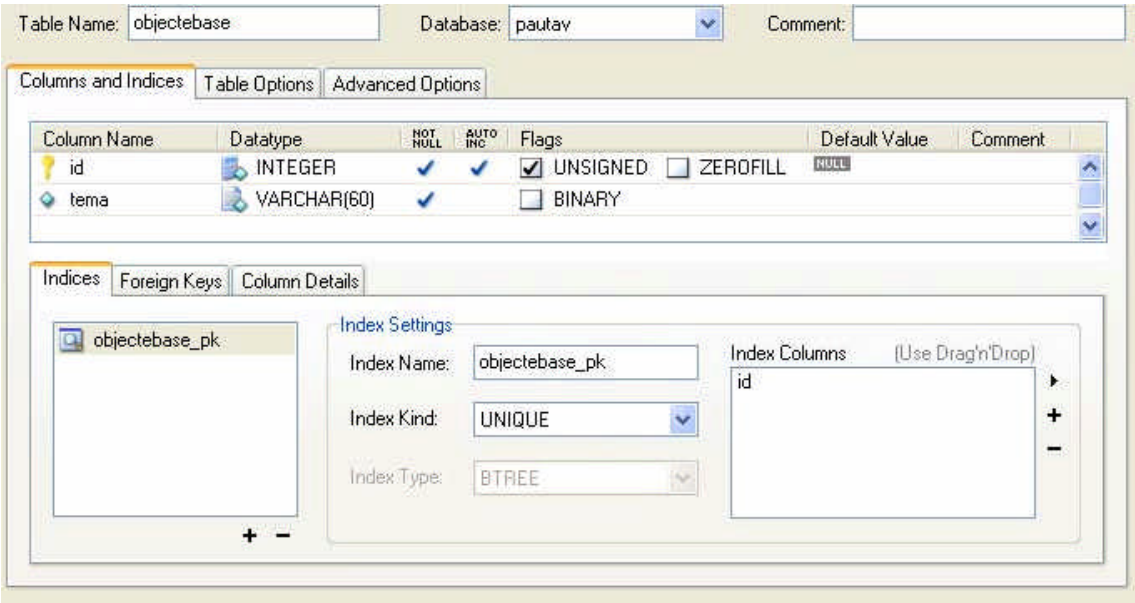
Notícia



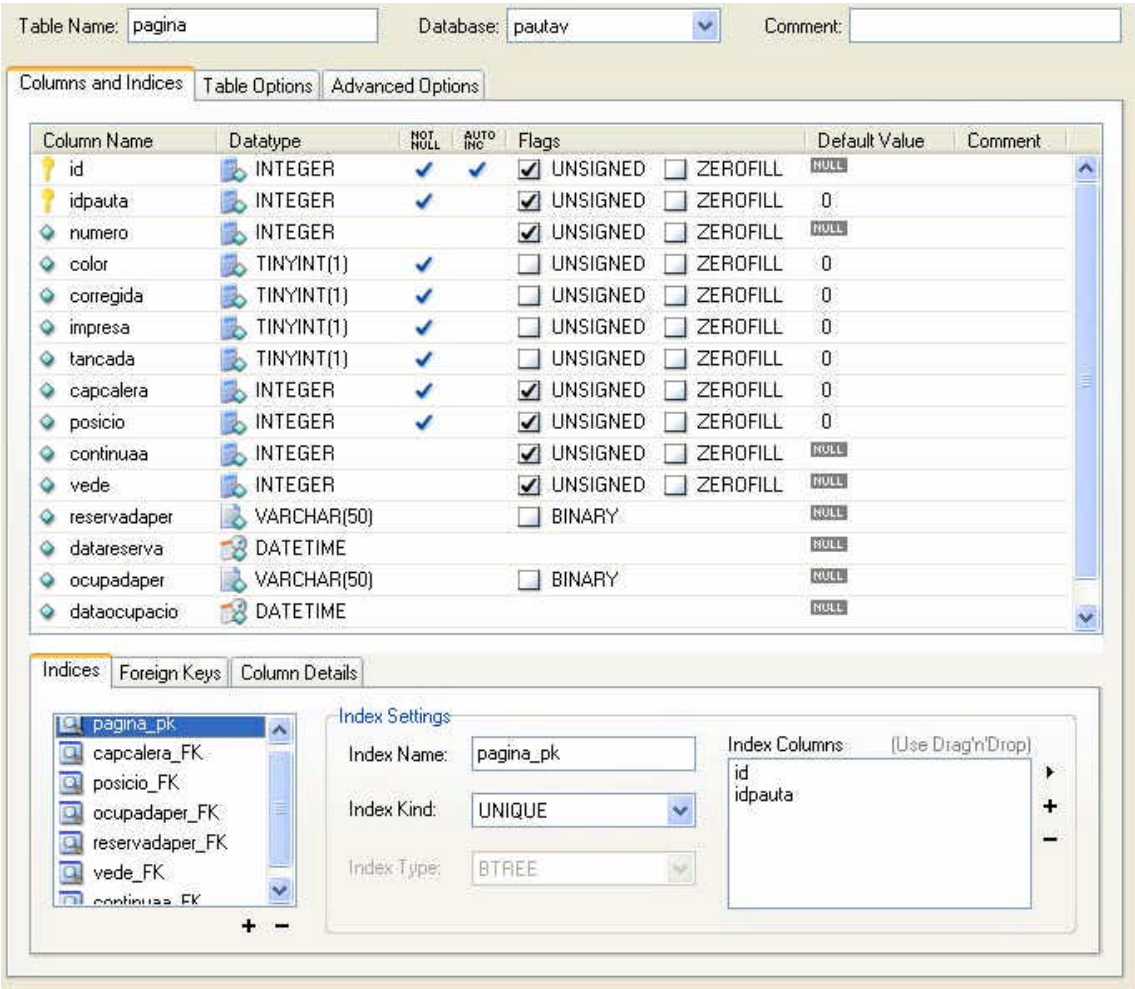
objecteapagina



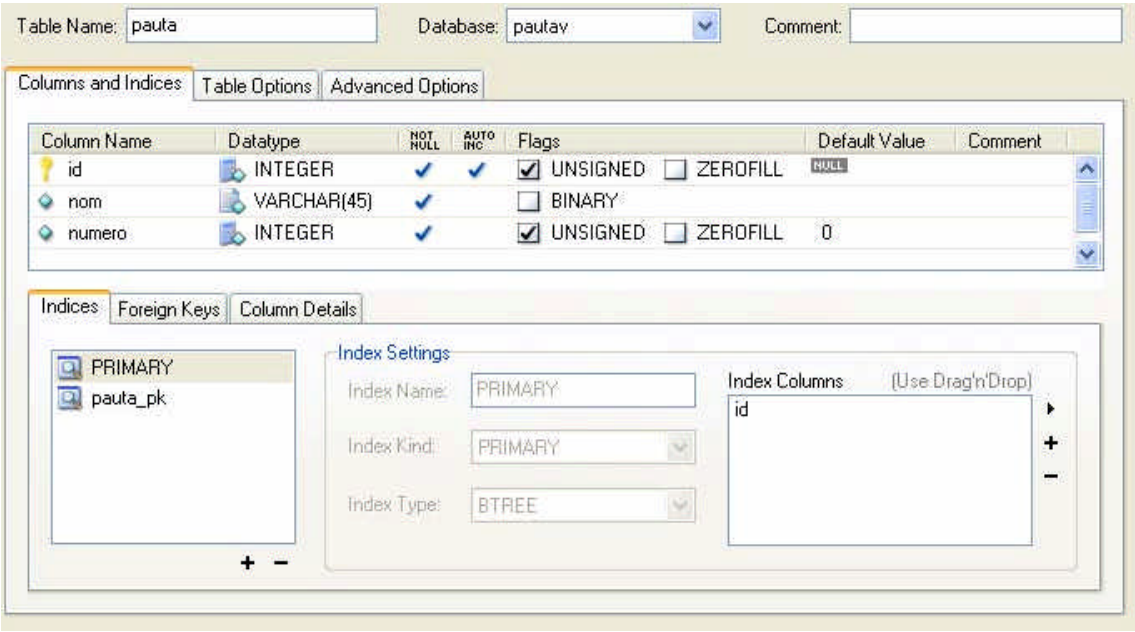
objectebase



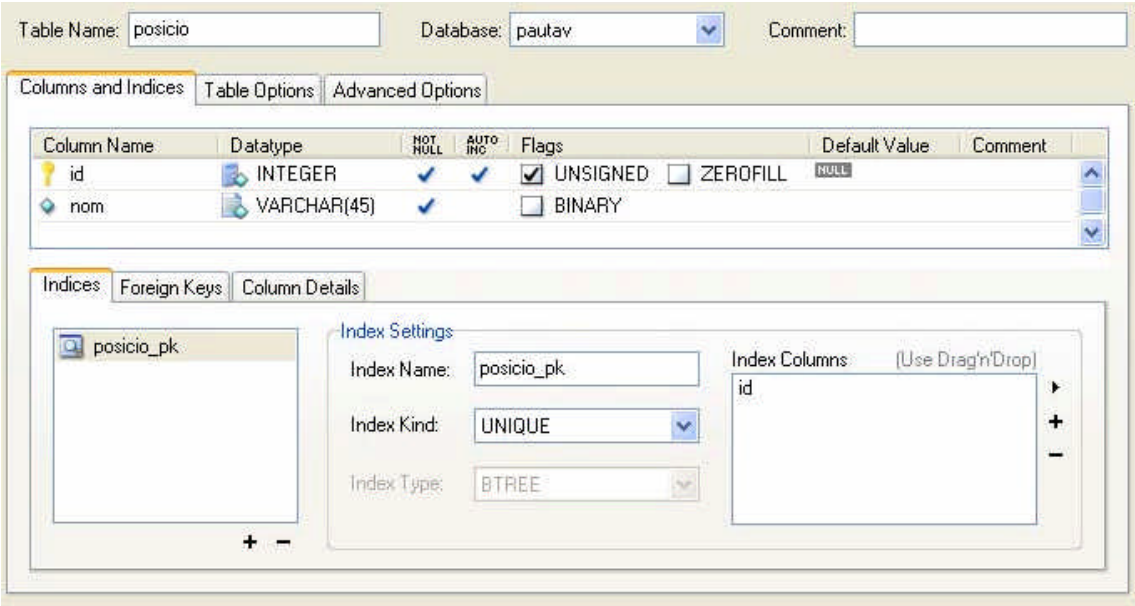
pagina



pauta



posicio



rol

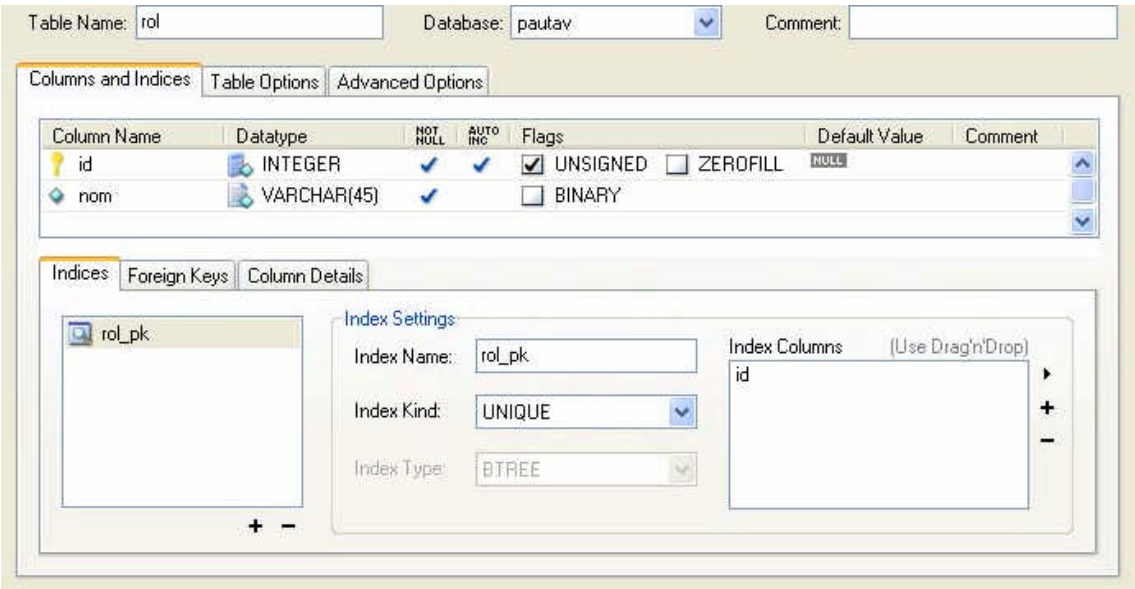


Table Name: rol
Database: pautav
Comment:
Columns and Indices
Table Options
Advanced Options
Column Name
Datatype
NOT NULL
AUTO INC
Flags
Default Value
Comment
id
INTEGER
UNSIGNED
ZEROFILL
NULL
nom
VARCHAR(45)
BINARY
Indices
Foreign Keys
Column Details
rol_pk
Index Settings
Index Name:
rol_pk
Index Kind:
UNIQUE
Index Type:
BTREE
Index Columns
(Use Drag'n'Drop)
id

seeds

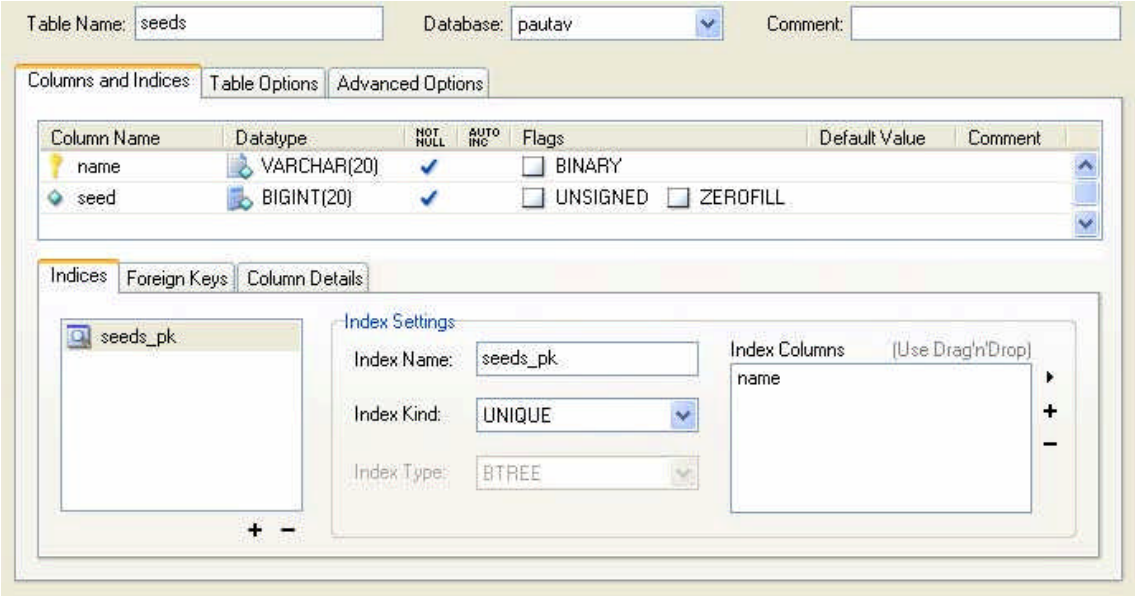
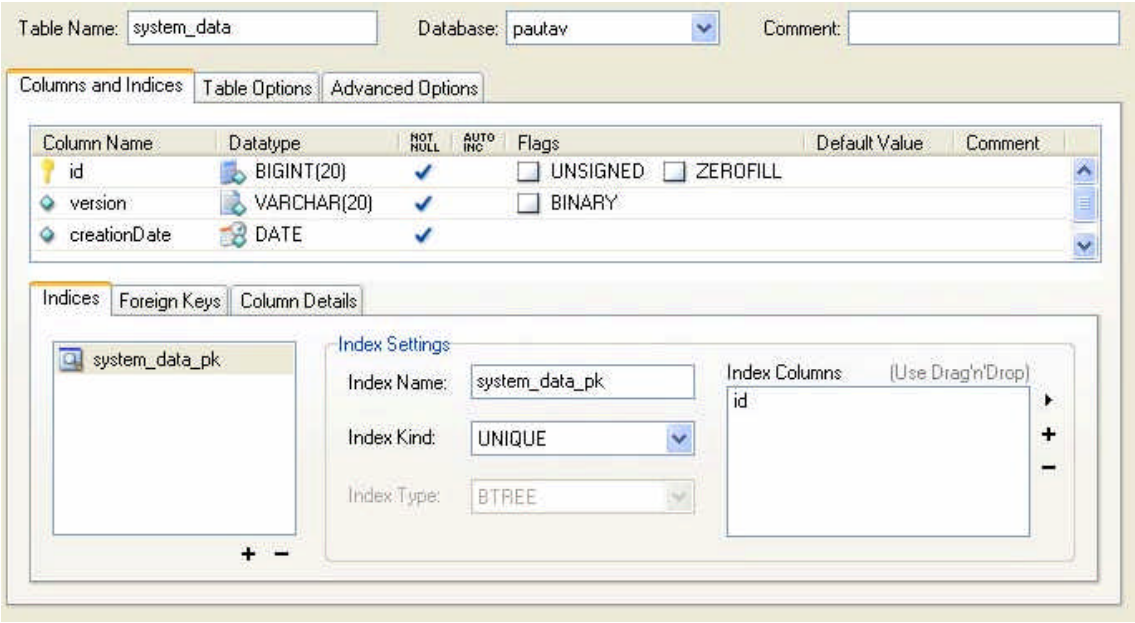
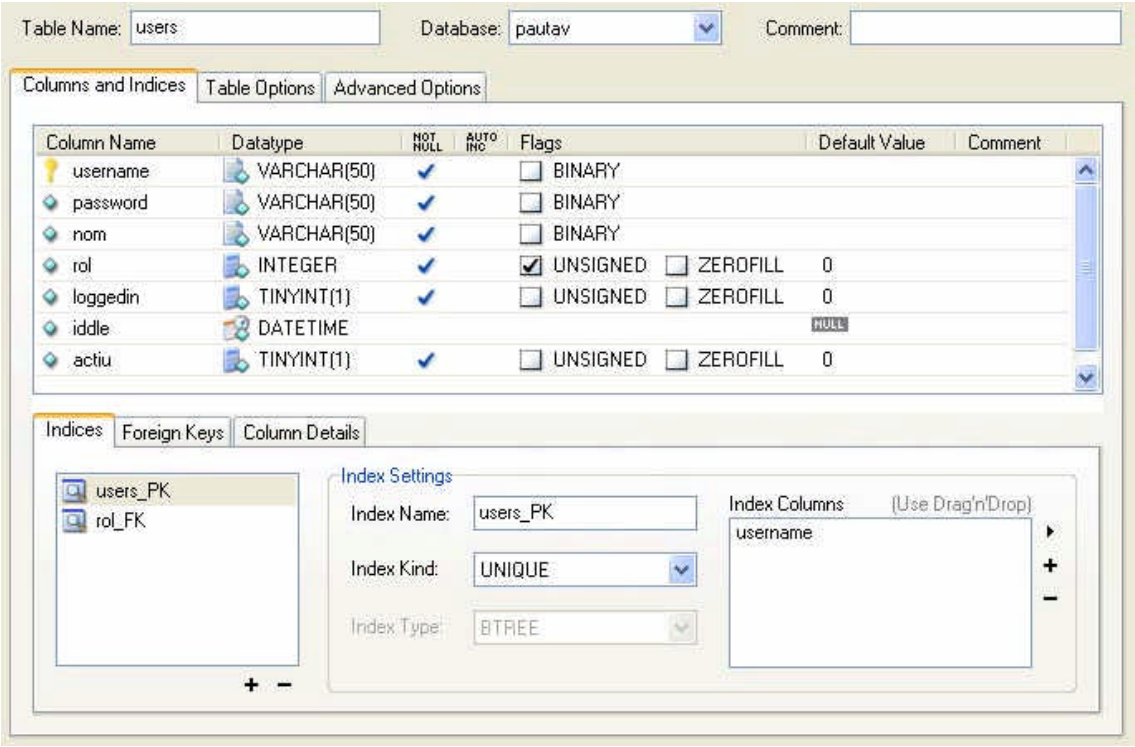


Table Name: seeds
Database: pautav
Comment:
Columns and Indices
Table Options
Advanced Options
Column Name
Datatype
NOT NULL
AUTO INC
Flags
Default Value
Comment
name
VARCHAR(20)
BINARY
seed
BIGINT(20)
UNSIGNED
ZEROFILL
Indices
Foreign Keys
Column Details
seeds_pk
Index Settings
Index Name:
seeds_pk
Index Kind:
UNIQUE
Index Type:
BTREE
Index Columns
(Use Drag'n'Drop)
name

system_data



users



14 – UN PROJECTE AMB FUTUR

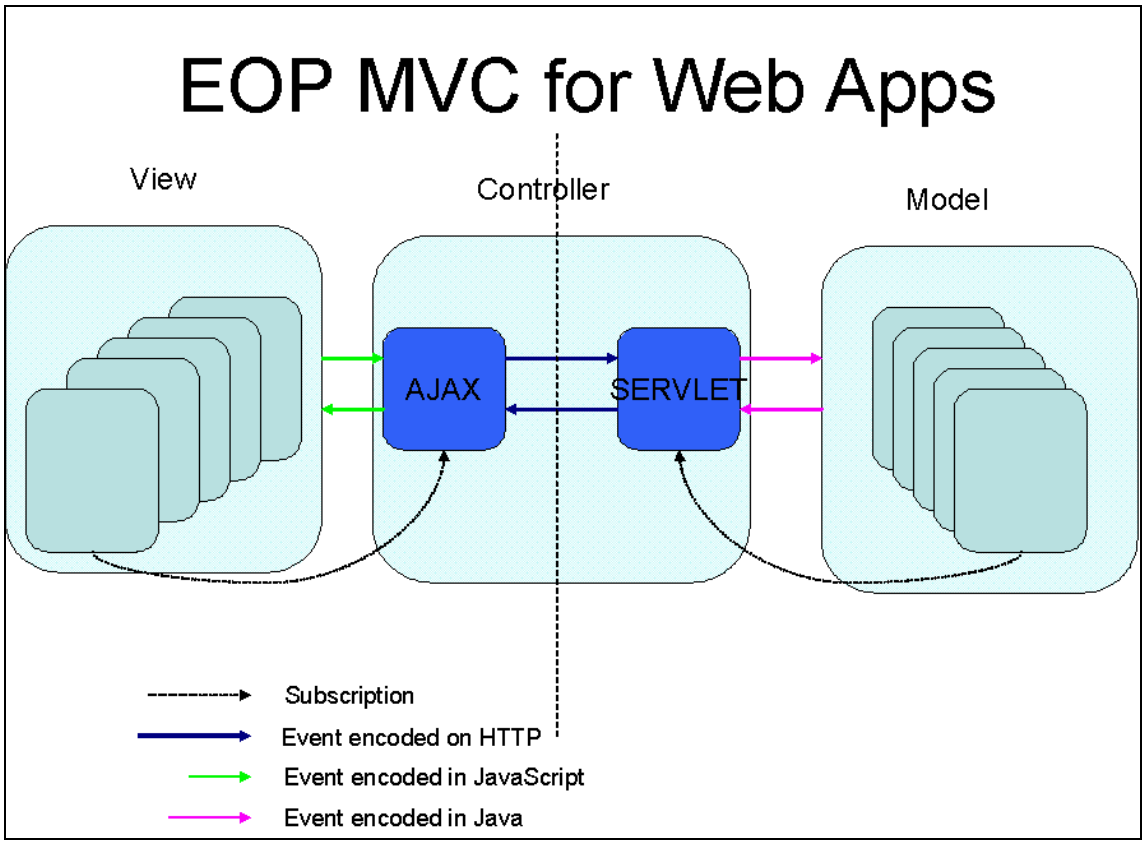
Tot i que en el projecte s'ha realitzat en base a un cas real, aquest ens ha servit per investigar la creació d'un sistema de webs col·laboratives basada en el sistema MVC que integran el model d'events. No ha estat un projecte complert ja que la programació realitzada és pràcticament inexistent degut a la limitació de temps, i el fet de realtizar un projecte de tal magnitud una única persona, i no paga la pena ni tan sols presentar-la.

El projecte, però, no tindria per què acabar aquí, en la creació d'una única aplicació. Doncs com podem veure el sistema d'events és totalment independent de l'aplicació i es podria integrat a qualsevol tipus de web.

Un projecte de futur seria doncs crear una aplicació genèrica que permetès integrar d'una forma senzilla i intuïtiva el model POE-MVC a qualsevol web d'una manera semblant a com ho fa STRUTS amb el model clàssic MVC

Hem investigat dins el possible si existeix alguna aplicació d'aquestes característiques sense trobar-ne cap, per tant creiem que seria una oportunitat única per poder desenvolupar un camp de la informàtica, com és el de la programació orientada a events, que avui dia és prou novedosa.

Durant les discusions amb el consultor de l'assignatura ha sorgit un projecte que pot ser prou interessant pel futur, la creació d'un framework orientat a events tal i com podem veure al següent diagrama:



En aquest framwork el controlador està distribuït, una part resideix al client, implementat amb Ajax, i l'altra al servidor, implementat amb un servlet. La part controladora client rep les peticions HTTP de l'usuari i les asincronitza i les transforma en events codificats en HTTP mitjançant la funcionalitat que ofereix Ajax. Aquests events arriben al servlet que és la part control.ladora servidora, aquest servlet transforma els events HTTP en events "normals" (i.e. en Java) i fa de processador d'events enviant els

events a aquelles components del model que s'hi hagin suscrit. Com pots veure el Model està format per components orientats a events que es suscriuen al control.lador servidor.

Aquestes components tornen les respostes també en forma d'events (i.e. del tipus *UIOutputEvents*) que el servlet codifica en HTTP i que envia al control.lador client. Aquest els rep i també fa de processador d'events, enviant aquests events a les vistes que s'hi hagin suscrit...

Notar que es poden tenir diverses vistes obertes alhora que anirant referscant-se de forma asíncrona i de la mateixa manera es poden fer peticions HTTP en paral.lel. Acabem de trencar el model *request-response* típic de les aplicacions web.

15 - BIBLIOGRAFIA

- ***Advanced JMS Messaging with OpenJMS*** de Kulvir Singh Bhogal
- ***Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC)*** de Steve Burbeck, Ph.D.
- ***Event-Driven Architecture Overview*** penjat per Brenda Michelson a Business-Driven Architecture, Service-oriented architecture
- ***Event-Driven Architecture para Aplicaciones Colaborativas de Aprendizaje*** de Claudi Paniagua Macià
- ***Event-driven architecture poised for wide adoption*** de Carol Sliwa http://www.computerworld.com/softwaretopics/software/appdev/story/0,10801,81133,00.
- ***How to generate traces that can be executed and automatically* verified** de Claudi Paniagua Macià
- ***Master JMS Messaging with OpenJMS*** de Kulvir Singh Bhogal
- ***O'Reilly Jakarta*** *Struts* de Chuck Cavaness. Editorial Anaya
- ***Project Esper*** http://esper.sourceforge.net/
- ***The NaradaBrokeringProject*** http://www.naradabrokering.org/
- ***Tutorial de SQL*** http://www.desarrolloweb.com/manuales/9/

16 – AGRAÏMENTS

M'agradaria agrair al meu consultor Claudi Paniagua Macia per la seva paciència i haver-me ajudat a fer evolucionar el projecte desde la seva concepció original tot i que ha amplicat que en molts moments fer-me parar mig boig i que la planificació inicial servís de poca cosa.
Amb tot i això estic molt satisfet amb el resultat i en veure com un projecte pot evolucionar.
Res hauria estat el mateix sense ell.