Desenvolupament d’un Controlador de Processos de Llarga Durada mitjançant tecnologia J2EE

Jorge Torrente Tomas
ETIG

Joan Vicent Orenga Serisuelo

25/06/2008

Al meu iaio Joaquim

Agraïments

Primer de tot he de donar les gràcies d'una forma molt i molt especial a la Mercè, sense ella aquests estudis que ara estan arribant a la seva fi, ni tan sols haguessin començat, però sobretot per la paciència que ha demostrat en molts dels innumerables dies (i nits) que hem hagut de passar dedicats exclusivament a les PACs i pràctiques de la UOC.

Als meus pares, avis, germana i sogres per tot l'ajut i suport prestat durant la realització dels estudis.

I finalment a tota la comunitat de docents, estudiants i altre personal de la UOC per fer possible una educació a distància de tanta qualitat.

Resum

El projecte final de carrera objectiu del present document, inclou l'anàlisi, disseny, implementació i documentació d'una aplicació mitjançant tecnologia J2EE la finalitat de la qual és la gestió dels processos o tasques de llarga durada que formen part dels fluxos de negoci de moltes organitzacions.

Funcionalment parlant, es tracta d'implementar una proposta de solució a una problemàtica molt freqüent en projectes reals i que normalment sempre s'aborda de forma particular dins de cadascun d'ells. Per tant podem dir que l'objectiu final del TFC és el de proporcionar un component integrable dins d'aquelles aplicacions que tenen el requeriment o necessitat de gestionar aquest tipus de tasques o processos, de forma que els esforços dels recursos assignats al desenvolupament de la solució puguin centrar-se en implementar la lògica dels processos i no en la codificació de components d'infraestructura de suport.

Cal tenir present que el treball fet és la culminació dels estudis d'una enginyeria i en conseqüència ha de il·lustrar el fet que s'han comprès els conceptes i coneixements adquirits durant els estudis, a més a més de saber aplicar-los a la resolució d'un problema real. En aquest sentit, a nivell de metodologia s'ha utilitzat un cicle de vida en cascada, recorrent a la notació UML en l'elaboració dels documents d'anàlisi i disseny.

Índex

1 Introducció 8
1.1 Justificació i context del TFC 8
1.2 Objectius 8
1.3 Enfocament i mètode seguit 9
1.4 Planificació 10
1.5 Productes obtinguts 11
1.6 Descripció dels capítols 12
2 Documentació de Requisits 13
2.1 Introducció 13
2.2 Domini de l'aplicació 13
2.3 Guions 14
2.3.1 Guió per a la definició 14
2.3.2 Guió per a la planificació 14
2.3.3 Guió per a l'execució 14
2.4 Casos d'ús 15
2.4.1 Actors 15
2.4.2 Diagrama general 15
2.4.3 Documentació textual 17
2.5 Requisits no funcionals 25
2.6 Requisits de la UI 25
3 Anàlisi de Requisits 26
3.1 Revisió de casos d'ús 26
3.2 Paquets d'anàlisi i serveis 26
3.3 Classes d'entitat 26
3.4 Diagrama d'estats 29
3.5 Classes de frontera, de control i de les operacions 29
4 Disseny 36
4.1 Arquitectura 36
4.2 Frameworks i programari base 37
4.3 Patrons 37
4.4 Persistència 38
4.4.1 Entitats i atributs 38
4.4.2 Diagrama ER 39
4.5 Revisió model estàtic 40
4.6 Classes de disseny 40
4.6.1 Lògica de negoci 40
4.6.2 Localitzador de serveis 42
4.6.3 Presentació 43
4.6.4 Propagació dels errors 44
4.7 Disseny de la UI 44
5 Implementació 50
5.1 Introducció 50
5.2 Aspectes a destacar de la solució 50
5.2.1 Invocació als components de negoci 50
5.2.2 Execucions de tasques 51
5.2.3 API per la codificació de tasques o processos 51
5.2.4 Sessions Hibernate 52
5.2.5 Transaccionalitat 52
5.2.6 Mappings Hibernate 53
5.2.7 Cerca d'entitats de negoci 53
5.2.8 Context d'execució 54
5.2.9 Log 54
5.2.10 Internacionalització 54

5.2.11 Struts....54
5.2.12 JSF....55
6 Conclusions....57
7 Glossari....58
8 Bibliografia....59

Índex de figures

Figura 1: Distribució de tasques 10
Figura 2: Planificació del TFC 11
Figura 3: Diagrama del model del domini 13
Figura 4: Diagrama d'Actors 15
Figura 5: Diagrama general de casos d'ús 16
Figura 6: Diagrama de paquets d'anàlisi 26
Figura 7: Diagrama de classes d'entitat 27
Figura 8: Diagrama d'estats d'una tasca 29
Figura 9: Diagrama de Seqüència de "Definir Tasca" 30
Figura 10: Diagrama de Seqüència de "Cercar Tasca" 30
Figura 11: Diagrama de Seqüència de "Detall Tasca" 31
Figura 12: Diagrama de Seqüència de "Modificar Tasca" 31
Figura 13: Diagrama de Seqüència de “Inhabilitar tasca” 32
Figura 14: Diagrama de Seqüència de “Esborrar tasca” 32
Figura 15: Diagrama de Seqüència de “Nova Execució” 33
Figura 16: Diagrama de Seqüència de “Aturar Execució” 34
Figura 17: Diagrama de Seqüència de “Avortar Execució” 34
Figura 18: Diagrama de Seqüència de “Re-arrancar Execució” 35
Figura 19: Diagrama de desplegament 36
Figura 20: Diagrama Entitat/Relació 39
Figura 21: Jerarquia de classes de les planificacions 40
Figura 22: Diagrama de classes de negoci 40
Figura 23: Diagrama de classes de negoci associades a Tasques 41
Figura 24: Diagrama de classes de negoci associades a Planificacions 41
Figura 25: Diagrama de classes de negoci associades a Execucions 42
Figura 26: Diagrama de Seqüència de la invocació a la lògica de negoci 42
Figura 27: Diagrama de classes de presentació associades a Tasques 43
Figura 28: Diagrama de classes de presentació associades a Planificacions 43
Figura 29: Diagrama de classes de presentació associades a Execucions 44
Figura 30: Jerarquia de classes de les Excepcions 44
Figura 31: Finestra de benvinguda a l'aplicació 45
Figura 32: Finestra de creació d’una nova tasca 45
Figura 33: Finestra modal que informa del resultat correcte d’una inserció 45
Figura 34: Finestra on introduir els paràmetres de cerca de tasques 46
Figura 35: Finestra on es mostren de forma paginada els resultats d’una cerca 46
Figura 36: Finestra on es mostra tota la informació d’una tasca 47
Figura 37: Finestra de modificació d’una tasca 47
Figura 38: Finestra modal que informa del resultat correcte d’un esborrament 48
Figura 39: Finestra de consulta de les notificacions de canvi d’estat 48
Figura 40: Finestra modal amb tota la informació d’una notificació 49
Figura 41: Flux d'execució d'un mètode de negoci 50

1 Introducció

1.1 Justificació i context del TFC

Cada cop hi ha més aplicacions empresarials que requereixen executar i controlar tasques que per la seva complexitat o bé no poden executar-se en certs moments del dia, o bé la seva durada és tan llarga que fa inviable la obtenció síncrona dels seus resultats.

Com a exemple real podem pensar en les administracions públiques: la integració dels sistemes d'informació d'aquest tipus d'entitats, molts cops, està basada en la generació/recepció de fitxers que poden arribar a ser molt grans. En conseqüència no és desitjable que el tractament d'aquests fitxers es faci en el moment que tots els usuaris del sistema es troben connectats i treballant, ja que és molt probable que això provoqui una caiguda del rendiment. Per tant, cal planificar-lo a una hora del dia on la seva execució no tingui tan impacte. Però un cop en marxa, els administradors del sistema sí que necessiten poder consultar l'estat del procés, o parar-lo si es dóna alguna situació anòmala i tornar-lo a posar en marxa un cop solucionat el problema.

Per tal de solucionar aquest tipus de problemàtiques es planteja el desenvolupament d'una aplicació que de forma integrada ofereixi els serveis de definició, planificació, execució i control d'aquest tipus de tasques.

També cal tenir en consideració que per tal que tot això sigui possible, és necessari que els processos controlats per l'aplicació hagin estat programats en un llenguatge i amb una estructura concretes: ens limitarem al llenguatge JAVA i s'oferirà una API totalment documentada que faciliti la feina als programadors que implementen aquests processos.

1.2 Objectius

Òbviament el primer objectiu del treball realitzat és l'obtenció de l'aplicació o component que satisfaci els requeriments funcionals anteriorment esmentats: definició, planificació, execució i control de tasques de llarga durada.

Ara bé, al mateix temps podem identificar uns objectius secundaris tals com:

- Aplicar de forma pràctica els coneixements de programació orientada a objectes per tal de resoldre un problema real, i de forma anàloga en l'ús de la notació UML per especificar i dissenyar la proposta d'una solució.

- Aprofundir en el coneixement de la tecnologia J2EE i en l'ús alguns dels *frameworks* i tecnologies: Struts, EJB, JSP, JMS

- Aprendre l'ús de *frameworks* i tecnologies: Hibernate, Quartz, JSF, RichFaces

- Estudiar la possible integració de dos dels *frameworks* de presentació més utilitzats en l'actualitat: Struts i JSF. En aquest sentit, cal tenir present que moltes de les aplicacions empresarials que s'utilitzen actualment estan basades en Struts, però arran de l'aparició de l'especificació JSF i de les seves llibreries de components gràfics, resulta molt interessant per a les organitzacions adaptar la UI de les seves aplicacions a aquestes noves llibreries minimitzant els canvis a realitzar.

- Aplicar de forma pràctica molts dels patrons de disseny més *populars*.

1.3 Enfocament i mètode seguit

El cicle de vida del programari està constituït per la programació i tot el conjunt d'etapes que la precedeixen i la succeeixen. Per l'elaboració del TFC s'ha seguit el cicle de vida clàssic o en cascada, el qual es caracteritza perquè en cada etapa s'obtenen uns documents (*deliverables*) que són les bases de partida de l'etapa següent (que per tant, no pot començar abans que hagi acabat l'anterior) i mai no es torna a etapes passades.

Les fases per les que s'ha passat durant tot el TFC són:
Anàlisi prèvia. Es defineixen els grans trets del sistema de programari a desenvolupar. Es correspon amb la PAC1.

Anàlisi (de requisits). Definició detallada de les necessitats d'informació que haurà de resoldre el programari, sense tenir en compte els mitjans tècnics amb que s'haurà de dur a terme el desenvolupament del programari. Es correspon a la meitat de la PAC2.

Disseny. S'especifica "com el programari ha de fer la seva funció": arquitectura general, estructures de dades (base de dades), interfícies d'usuari. Es correspon a l'altra part de la PAC2.

Programació o codificació. Es tradueix el disseny a codi processable per l'ordinador. Es correspon amb la PAC3.

Prova. Consisteix a provar el programari des de diversos punts de vista d'una manera planificada i naturalment, localitzar i corregir els errors que es detectin.

Pel que fa a l'aprenentatge de noves tecnologies, *frameworks* o components a utilitzar dins el TFC, sempre s'ha seguit el mateix procés:

- Llegir la documentació associada.
- Cercar i seguir varis tutorials sobre el component, tecnologia o *framework* .
- Integració dins l'aplicació.

1.4 Planificació

La llista de tasques i la seva duració en les que es va dividir el projecte és:

Nombre de tarea	Duración	Comienzo	Fin
Inici Semestre	**0 días**	**jue 28/02/08**	**jue 28/02/08**
⊞ **Fase Prèvia**	**14 días**	**vie 29/02/08**	**jue 13/03/08**
Entrega de la PAC1	**0 días**	**vie 14/03/08**	**vie 14/03/08**
⊟ **Anàlisi**	**8 días**	**sáb 15/03/08**	**sáb 22/03/08**
Subsistema de definició	2 días	sáb 15/03/08	dom 16/03/08
Subsistema de planificació	2 días	lun 17/03/08	mar 18/03/08
Subsistema d'execucions	2 días	mié 19/03/08	jue 20/03/08
Motor d'execució	2 días	vie 21/03/08	sáb 22/03/08
⊟ **Disseny**	**22 días**	**dom 23/03/08**	**dom 13/04/08**
Model estàtic del domini	2 días	dom 23/03/08	lun 24/03/08
Subsistema de definició	5 días	mar 25/03/08	sáb 29/03/08
Subsistema de planificació	5 días	dom 30/03/08	jue 03/04/08
Subsistema d'execucions	5 días	vie 04/04/08	mar 08/04/08
Motor d'execució	3 días	mié 09/04/08	vie 11/04/08
Disseny de la BBDD	2 días	sáb 12/04/08	dom 13/04/08
Entrega de la PAC2	**0 días**	**lun 14/04/08**	**lun 14/04/08**
⊟ **Implementació**	**43 días**	**mar 15/04/08**	**mar 27/05/08**
Configuració de l'entorn de desenv	5 días	mar 15/04/08	sáb 19/04/08
Classes comunes i d'utilitat	4 días	dom 20/04/08	mié 23/04/08
Subsistema de definició	8 días	jue 24/04/08	jue 01/05/08
Subsistema de planificació	8 días	vie 02/05/08	vie 09/05/08
Subsistema d'execucions	8 días	sáb 10/05/08	sáb 17/05/08
Motor d'execució	8 días	dom 18/05/08	dom 25/05/08
Encapsulament de la API	2 días	lun 26/05/08	mar 27/05/08
Entrega de la PAC3	**0 días**	**lun 19/05/08**	**lun 19/05/08**
⊟ **Producte & Documentació**	**21 días**	**mié 28/05/08**	**mar 24/06/08**
Empaquetament de la solució	2 días	mié 28/05/08	jue 29/05/08
Redacció memòria	14 días	vie 30/05/08	mar 17/06/08
Redacció presentació	5 días	mié 18/06/08	mar 24/06/08
Entrega Final	**0 días**	**mié 25/06/08**	**mié 25/06/08**

Figura 1: Distribució de tasques

És important esmentar el fet que les dates clau del treball estan prefixades per la direcció docent de la UOC, de forma que la planificació de la resta de tasques cal adaptar-la a elles.

I el diagrama de Gantt associat:

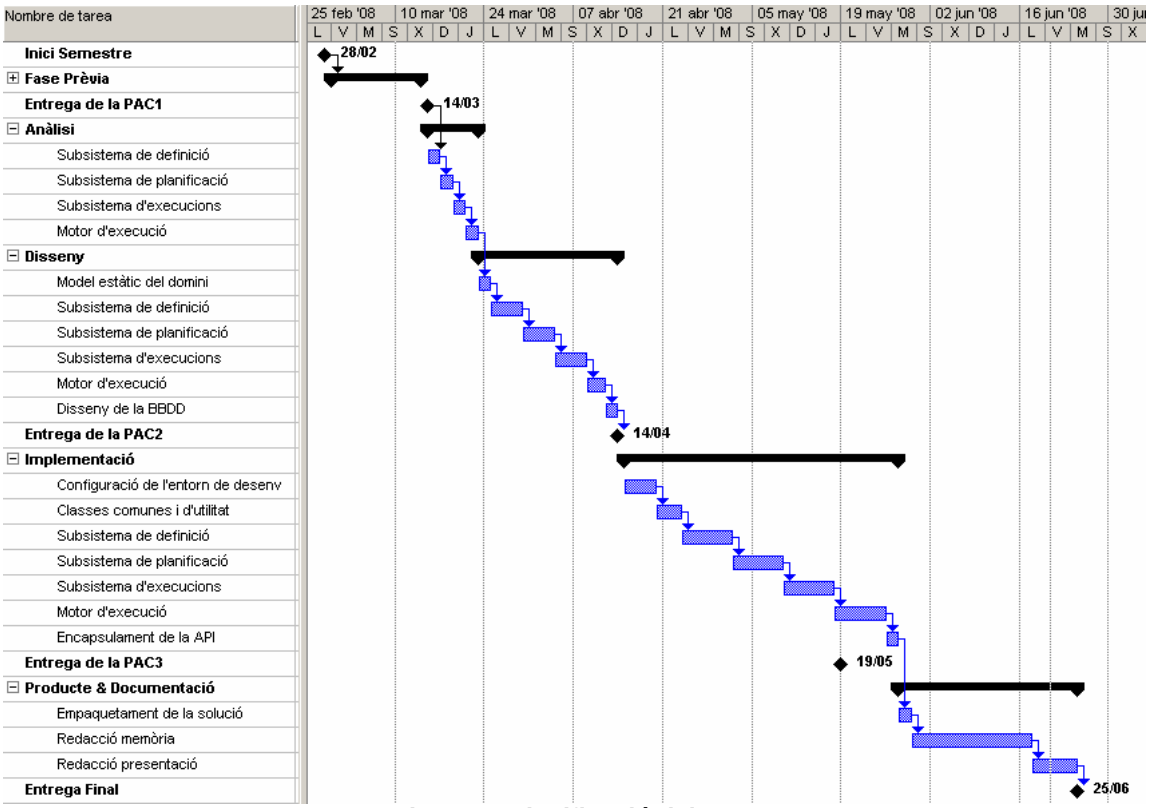


Figura 2: Planificació del TFC

Respecte aquesta planificació inicial cal esmentar que la fase d'implementació ha sofert un retard de dues setmanes que ha afectat al temps destinat a redactar la documentació. El motiu d'aquest retard el trobem en els problemes de configuració de l'entorn de desenvolupament (eclipse 3.3, JBoss 4.2.2) i en la valoració optimista de l'esforç a dedicar a les classes comunes i d'utilitat. Tot i això sempre s'han respectat les dates clau de les diferents entregues.

1.5 Productes obtinguts

Els productes obtinguts com a fruit del treball són:

- Fitxer de distribució .ear que inclou el .war de la capa web, el .jar de la capa ejb, el .jar amb les classes de la lògica de negoci, i el .jar amb les classes comuns.
- El codi font de totes les classes utilitzades.
- Documentació detallada i normalitzada en format javadoc de totes les classes.
- Els fitxers de desplegament i de configuració utilitzats, conjuntament amb les llibreries necessàries pel seu funcionament.
- El fitxers de creació de les taules i la inserció de dades necessàries a la BD a utilitzar.
- Els diversos manuals d'usuari que podeu consultar.

1.6 Descripció dels capítols

En la resta de capítols de la memòria es comentaran les fases d'anàlisi, disseny i implementació per les quals ha passat el desenvolupament d'aquest TFC:

- Capítol 2. Documentació de requisits
- Capítol 3. Anàlisi
- Capítol 4. Disseny
- Capítol 5. Implementació

2 Documentació de Requisits

2.1 Introducció

Recordem que l'objectiu principal del nou sistema serà facilitar a les possibles aplicacions un mòdul integrable dins seu que els permeti definir, planificar, executar i monitorar tasques de llarga durada.

No es considera part del sistema proposat, la codificació mitjançant el llenguatge JAVA de les accions a executar. Si més no, serà molt important el fet d'oferir a la persona que les hagi de desenvolupar, una API amb la seva corresponent documentació, a utilitzar en aquesta codificació.

Suposant que les accions ja han estat desenvolupades correctament, el consultor de negoci que coneix el context de totes aquests tasques, les afegirà al sistema associant-les a uns paràmetres d'execució concrets. Tanmateix el sistema no només facilitarà la possibilitat d'afegir nous elements, també inclourà una gestió completa de tots ells.

Un cop definides les tasques, els operadors del sistema han de poder passar a planificar-les, és a dir, determinar en quin o quins moments de temps s'han d'executar. Anàlogament a les definicions, aquí també s'oferirà una gestió completa de les planificacions.

Com és lògic, el sistema executarà automàticament les tasques en el moment que s'arribi a les dates especificades en les corresponents planificacions i a més, serà l'encarregat d'actualitzar l'estat associat. Inicialment només es consideraran els següents estats: planificada, en execució, finalitzada correctament, finalitzada amb error, avortada, pausada. Per a totes aquelles tasques que en el moment de la seva definició s'hagi inclòs la possibilitat de notificació, en produir-se un canvi d'estat s'enviarà un missatge a l'adreça de correu electrònic corresponent.

Finalment, els operadors també podran interactuar amb les tasques que es trobin en execució per tal d'aturar-les temporalment (pausar), definitivament (avortar), o bé per consultar les seves traces de log.

2.2 Domini de l'aplicació

A primer cop d'ull s'identifiquen els objectes o classes 'acció', 'tasca', 'paràmetre', 'planificació', 'execució'. S'han posat alguns atributs a les classes de manera orientativa. Aquest diagrama no es farà servir per a etapes posteriors.

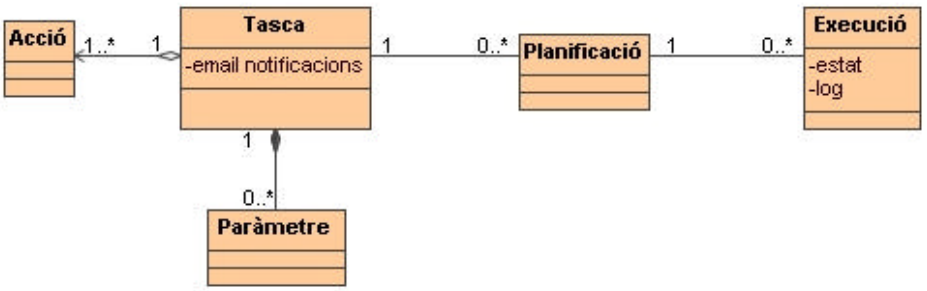


Figura 3: Diagrama del model del domini

2.3 Guions

2.3.1 Guió per a la definició

Durant la fase de construcció d'un sistema que pretén utilitzar el mòdul planificador/controlador de processos o tasques de llarga durada, un consultor del projecte analitza els processos de negoci del client, n'escriu els requeriments i demana a un desenvolupador que programi tot un conjunt d'accions (classe JAVA d'acord a l'API disponible).

Un cop acabat aquest desenvolupament el consultor defineix en el sistema la tasca i tots els paràmetres que aquesta rebrà en el moment de la seva execució. Per tal de realitzar l'operació introdueix un nom i una descripció, obtenint com a resposta l'identificador únic que el sistema li ha assignat.

Associada a la tasca també hi ha la notificació de resultats. Només es suporta la notificació per *email* de forma que en el moment de la definició és opcional la introducció de les adreces de correu que rebran les notificacions dels canvis d'estat de les tasques (en realitat de les seves execucions).
Un cop definides, el consultor por cercar, consultar, modificar, inhabilitar o esborrar qualsevol tasca.

2.3.2 Guió per a la planificació

Un cop acabat el desenvolupament del projecte (el que utilitza el mòdul planificador/controlador), i com a part del protocol d'entrada en producció, el consultor redacta un document especificant les planificacions necessàries per a totes les tasques de llarga durada que s'han incorporat.

L'equip de sistemes del client i més concretament un dels seus operadors, a partir d'aquest document i del manual d'operació de l'aplicació, és qui planifica totes les execucions.

Els tipus de planificació admesos són:
- Execució única en una data concreta.
- Execució periòdica a partir d'una data.
- Execució periòdica entre dos dates.

Pel que fa a la periodicitat es pot definir basant-se en:
- Un instant temporal concret: tots els dilluns, tots els dies a les 14:00, etc.
- Període de temps des de l'última execució: després de 15 hores, després de 20 minuts

Un cop definit el tipus de planificació i la periodicitat, també cal introduir un nom, una descripció, el ID de la tasca, i de forma opcional la possibilitat d'inhabilitar la planificació en el cas que l'execució falli. Com a resposta el sistema retornarà l' identificador únic assignat.

Un cop definides, un operador por cercar, consultar, modificar, inhabilitar o esborrar qualsevol planificació.

2.3.3 Guió per a l'execució

Les responsabilitats dels operadors inclouen el monitorar les execucions així com l'estat dels servidors on s'executen. En el cas que es detecti algun problema en els servidors, per tal de solucionar-lo, pot ser necessari inhabilitar les planificacions, aturar temporalment totes les execucions, o en funció de la criticitat del problema a una aturada definitiva.
També existeix la possibilitat que alguns usuaris funcionals del projecte global, requereixin dels operadors un informe sobre l'estat de l'execució d'una tasca, i per tant, aquests han de ser capaços d'accedir a les traces de log de qualsevol execució.

2.4 Casos d'ús

2.4.1 Actors

S'han identificat 3 actors: consultor, operador (usuaris finals directes del sistema) i el rellotge del propi sistema ja que aquest és l'encarregat d'iniciar les execucions de les tasques d'acord a les seves planificacions.

Donat que els dos usuaris finals (consultor i operador) tenen un cas d'ús en comú (el corresponent a la identificació dins el sistema) s'ha introduït un quart actor fictici (usuari) que engloba aquesta part comú:

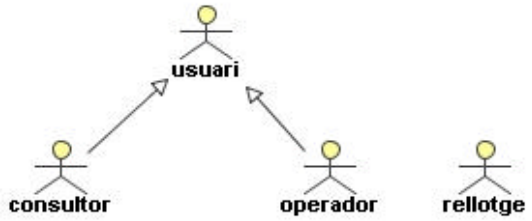


Figura 4: Diagrama d'Actors

2.4.2 Diagrama general

En la següent figura es poden observar tots els casos d'ús que s'han identificat dins la fase d'anàlisi. Cal tenir present que no s'han inclòs els casos d'ús relatius a donar d'alta els usuaris del sistema o al manteniment dels mateixos ja que s'ha considerat que aquests casos d'ús formen part del projecte global on s'integra el mòdul controlador/planificador.

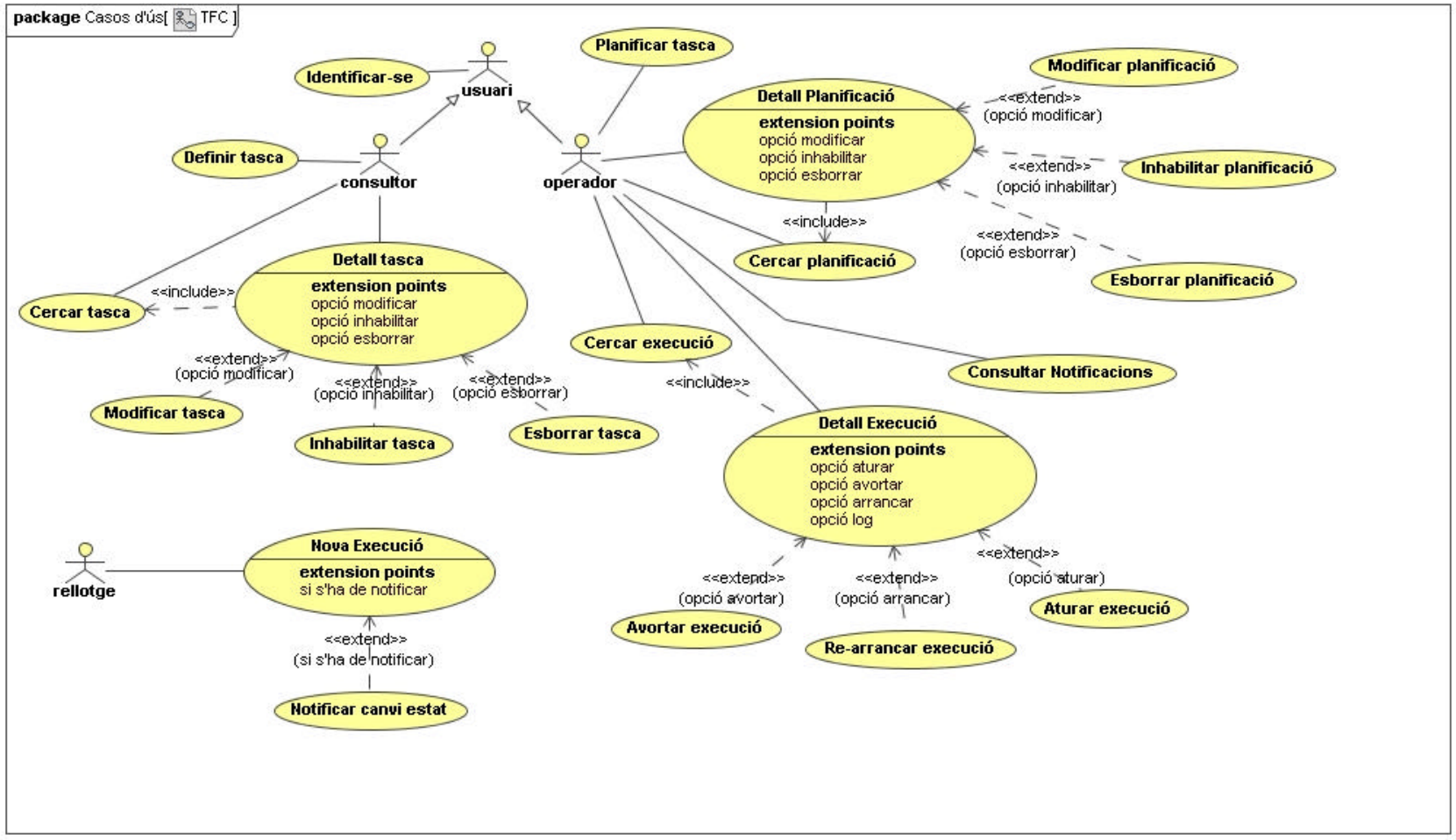


Figura 5: Diagrama general de casos d'ús

2.4.3 Documentació textual

Cas d'ús número 1. "Identificar-se".

Resum de la funcionalitat: inicia una sessió d'un usuari en el sistema

Actors: Usuari

Casos d'ús relacionats: Tots

Precondició: L'usuari ha estat donat d'alta

Postcondició: L'usuari queda registrat

Procés normal principal:

1. El sistema demana el nom d'usuari i el password de l'usuari

2. L'usuari introdueix les dades indicades al pas 1

3. El sistema verifica les dades

Alternatives de procés i excepcions:

3a. Les dades no són vàlides

3a1 Retorn al pas 1

Cas d'ús número 2. "Definir tasca".

Resum de la funcionalitat: incorpora una nova tasca dins el sistema

Actors: Consultor

Casos d'ús relacionats: Identificar-se

Precondició: El consultor està identificat, la tasca no existeix

Postcondició: S'enregistra una nova tasca en el sistema

Procés normal principal:

1. El sistema demana la classe JAVA que codifica les accions, un nom, una descripció, la llista de paràmetres d'execució i opcionalment la llista d'adreces de correu electrònic a les que notificar els canvis d'estat de les execucions associades.

2. L'usuari introdueix les dades indicades al pas 1

3. El sistema enregistra la tasca com una nova tasca en el sistema

4. El sistema informa al consultor del codi intern assignat a la nova tasca

Cas d'ús número 3. "Cercar tasca".

Resum de la funcionalitat: l'usuari a partir d'uns criteris de cerca (ID tasca, nom, classe JAVA) obté una llista paginada de totes les tasques definides que compleixen els criteris. En el cas que no s'introdueixi cap criteri el resultat serà la totalitat de les definicions. Cada registre de la llista inclou: identificador intern de tasca, nom i descripció.

Actors: Consultor

Casos d'ús relacionats: Identificar-se, Detall tasca

Precondició: El consultor està identificat

Postcondició: El consultor obté la llista paginada de totes les tasques definides que compleixen els criteris de cerca.

Procés normal principal:

1. El consultor introdueix els criteris de cerca

2. El sistema recupera i mostra una llista paginada de totes les tasques que compleixen els criteris de cerca. Cada registre de la llista inclou: identificador intern de tasca, nom i descripció.

Cas d'ús número 4. "Detall tasca".

Resum de la funcionalitat: recuperació de totes les dades que el sistema té d'una tasca.

Actors: Consultor

Casos d'ús relacionats: Identificar-se, Cercar tasca, Modificar tasca, Esborrar tasca, Inhabilitar tasca

Precondició: El consultor està identificat, la tasca existeix

Postcondició: Obtenim totes les dades d'una tasca

Procés normal principal:

1. include:: Cercar tasca

2. El consultor selecciona una tasca del llistat obtingut al pas 1

3. El sistema recupera totes les dades de la tasca

Alternatives de procés i excepcions:

2a. El llistat obtingut no conté cap element

2a1 Retorn al pas 1

3a. El consultor escull l'opció de modificar la tasca seleccionada

3a1 El sistema passa a executar el cas d'ús "Modificar tasca"

3b. El consultor escull l'opció d'esborrar la tasca seleccionada

3b1 El sistema passa a executar el cas d'ús "Esborrar tasca"

3c. El consultor escull l'opció d'inhabilitar la tasca seleccionada

3c1 El sistema passa a executar el cas d'ús "Inhabilitar tasca"

Cas d'ús número 5. "Modificar tasca".

Resum de la funcionalitat: modificació de les dades que el sistema té d'una tasca. Es pot modificar qualsevol atribut a excepció del codi intern generat i assignat automàticament pel sistema. Si en el moment de fer la modificació, la tasca està en execució els canvis no afectaran a la instància que s'està executant.

Actors: Consultor

Casos d'ús relacionats: Identificar-se, Cercar tasca, Detall tasca

Precondició: El consultor està identificat, la tasca existeix

Postcondició: El sistema emmagatzema les dades actualitzades de la tasca

Procés normal principal:

1. El consultor introdueix les noves dades

2. El sistema emmagatzema les noves dades de la tasca

Cas d'ús número 6. "Esborrar tasca".

Resum de la funcionalitat: eliminació de totes les dades que el sistema emmagatzema d'una tasca. Només serà possible si no hi ha cap planificació associada, si no hi ha cap registre d'execució associat (log) i si no està en execució.

Actors: Consultor

Casos d'ús relacionats: Identificar-se, Cercar tasca, Detall tasca

Precondició: El consultor està identificat, la tasca existeix

Postcondició: Totes les dades de la tasca desapareixen del sistema

Procés normal principal:

1. El sistema demana confirmació

2. El consultor confirma l'esborrament

3. El sistema verifica les condicions per procedir a l'esborrament

4. Les dades de la tasca desapareixen del sistema

Alternatives de procés i excepcions:

3a. La tasca té planificacions associades

3a1 El sistema informa del problema al consultor

3a2 L'execució finalitza

3b. La està en execució

3b1 El sistema informa del problema al consultor

3b2 L'execució finalitza

3c. La tasca té un registre d'execució associat (log)

3c1 El sistema informa del problema al consultor

3c2 L'execució finalitza

Cas d'ús número 7. "Inhabilitar tasca".

Resum de la funcionalitat: degut a raons de manteniment pot ser interessant que durant un temps una tasca estigui inactiva (independentment de les seves planificacions), és a dir, encara que la tasca estigui planificada, no s'executarà. El mateix cas d'ús s'utilitzarà per habilitar o inhabilitar una tasca.

Actors: Consultor

Casos d'ús relacionats: Identificar-se, Cercar tasca, Detall tasca

Precondició: El consultor està identificat, la tasca existeix

Postcondició: El sistema emmagatzema les dades actualitzades de la tasca

Procés normal principal:

1. El consultor introdueix les noves dades

2. El sistema emmagatzema les noves dades de la tasca

Cas d'ús número 8. "Planificar tasca".

Resum de la funcionalitat: permetrà donar d'alta una nova planificació d'acord als tipus d'execucions i periodicitats descrites. A més a més caldrà introduir un nom, una descripció i el ID de la tasca. També s'inclourà la possibilitat d'inhabilitar la planificació en el cas que l'execució falli. En el moment de la creació el sistema assignarà un identificador únic que es comunicarà a l'usuari.

Actors: Operador

Casos d'ús relacionats: Identificar-se

Precondició: L'operador està identificat, la planificació no existeix

Postcondició: S'enregistra una nova planificació en el sistema

Procés normal principal:

1. El sistema demana un nom, una descripció, l'ID de la tasca, la data d'execució, la periodicitat, i la possibilitat d'inhabilitar la planificació en el cas que l'execució falli.

2. L'usuari introdueix les dades indicades al pas 1

3. El sistema enregistra la planificació com una nova planificació en el sistema

4. El sistema informa al consultor del codi intern assignat a la nova planificació

Cas d'ús número 9. "Cercar planificació".

Resum de la funcionalitat: l'usuari a partir d'uns criteris de cerca (ID planificació, nom, ID de tasca) obté una llista paginada de totes les planificacions que compleixen els criteris. En el cas que no s'introdueixi cap criteri el resultat serà la totalitat de les planificacions. Cada registre de la llista inclou: identificador, nom, descripció, identificador de tasca, data de pròxima execució i estat (habilitada o no).

Actors: Operador

Casos d'ús relacionats: Identificar-se, Detall planificació

Precondició: L'operador està identificat

Postcondició: L'operador obté la llista paginada de totes les planificacions definides que compleixen els criteris de cerca.

Procés normal principal:

1. L'operador introdueix els criteris de cerca

2. El sistema recupera i mostra una llista paginada de totes les planificacions que compleixen els criteris de cerca. Cada registre de la llista inclou: identificador, nom, descripció, identificador de tasca, data de pròxima execució i estat (habilitada o no).

Cas d'ús número 10. "Detall planificació".

Resum de la funcionalitat: recuperació de totes les dades que el sistema té d'una planificació.

Actors: Operador

Casos d'ús relacionats: Identificar-se, Cercar planificació, Modificar planificació, Esborrar planificació, Inhabilitar planificació

Precondició: L'operador està identificat, la planificació existeix

Postcondició: Obtenim totes les dades d'una planificació

Procés normal principal:

1. include:: Cercar planificació

2. El consultor selecciona una planificació del llistat obtingut al pas 1

3. El sistema recupera totes les dades de la planificació

Alternatives de procés i excepcions:

2a. El llistat obtingut no conté cap element

2a1 Retorn al pas 1

3a. L'operador escull l'opció de modificar la planificació seleccionada

3a1 El sistema passa a executar el cas d'ús "Modificar planificació"

3b. El consultor escull l'opció d'esborrar la planificació seleccionada

3b1 El sistema passa a executar el cas d'ús "Esborrar planificació"

3c. El consultor escull l'opció d'inhabilitar la planificació seleccionada

3c1 El sistema passa a executar el cas d'ús "Inhabilitar planificació"

Cas d'ús número 11. "Modificar planificació".

Resum de la funcionalitat: modificació de les dades que el sistema té d'una planificació. Es pot modificar qualsevol atribut a excepció del codi intern generat i assignat automàticament pel sistema.

Actors: Operador

Casos d'ús relacionats: Identificar-se, Cercar planificació, Detall planificació

Precondició: L'operador està identificat, la planificació existeix

Postcondició: El sistema emmagatzema les dades actualitzades de la planificació

Procés normal principal:

1. El consultor introdueix les noves dades

2. El sistema emmagatzema les noves dades de la planificació

Cas d'ús número 12. "Esborrar planificació".

Resum de la funcionalitat: eliminació de totes les dades que el sistema emmagatzema duna planificació.

Actors: Operador

Casos d'ús relacionats: Identificar-se, Cercar planificació, Detall planificació

Precondició: L'operador està identificat, la planificació existeix

Postcondició: Totes les dades de la planificació desapareixen del sistema

Procés normal principal:

1. El sistema demana confirmació
2. L'operador confirma l'esborrament
3. Les dades de la tasca desapareixen del sistema

Cas d'ús número 13. "Inhabilitar planificació".

Resum de la funcionalitat: degut a raons de manteniment pot ser interessant que durant un temps una planificació estigui inactiva. El mateix cas d'ús s'utilitzarà per habilitar o inhabilitar una planificació.

Actors: Operador

Casos d'ús relacionats: Identificar-se, Cercar planificació, Detall planificació

Precondició: L'operador està identificat, la planificació existeix

Postcondició: El sistema emmagatzema les dades actualitzades de la planificació

Procés normal principal:

1. El consultor introdueix les noves dades
2. El sistema emmagatzema les noves dades de la planificació

Cas d'ús número 14. "Nova execució".

Resum de la funcionalitat: periòdicament el sistema haurà d'anar comprovant si és el moment d'executar una planificació habilitada d'una tasca també habilitada. De ser així llençarà la corresponent petició al motor d'execució, i de ser necessari replanificarà el procés

Actors: Rellotge

Casos d'ús relacionats: Notificar canvi estat

Precondició: la planificació existeix i està habilitada, la tasca existeix i està habilitada

Postcondició: la tasca s'ha invocat

Procés normal principal:

1. El sistema instancia una nova execució asíncrona associada a la tasca
2. El sistema actualitza l'estat de l'execució
3. El sistema actualitza el registre d'execució associat (log)

Alternatives de procés i excepcions:

3a. En el cas que s'hagi establert la notificació via email dels canvis d'estat

    3a1 El sistema passa a executar el cas d'ús "Notificar canvi estat"

3b. En el cas que s'hagi establert una execució periòdica

    3b1 Es replanifica la tasca

    3b2 L'execució finalitza

Cas d'ús número 15. "Notificar canvi d'estat".

Resum de la funcionalitat: s'envia un *email* informant del canvi d'estat d'una tasca a totes les adreces que s'han afegit en el moment de definició de la tasca.

Actors: Rellotge

Casos d'ús relacionats: Nova Execució, Consultar notificacions

Precondició: la tasca s'ha invocat i s'han definit uns receptors de les notificacions de canvis d'estat

Postcondició: les notificacions s'han enviat a les corresponents adreces

Procés normal principal:

1. El sistema obté totes les adreces de correu electrònic associades a la tasca, i a les que cal enviar les notificacions de canvi d'estat

2. El sistema envia totes les notificacions.

Cas d'ús número 16. "Cercar execució".

Resum de la funcionalitat: l'usuari a partir d'uns criteris de cerca (ID d'execució, ID de planificació, ID de tasca, estat, data) obté una llista paginada de totes les execucions que compleixen els criteris

Actors: Operador

Casos d'ús relacionats: Identificar-se, Detall execució

Precondició: L'operador està identificat

Postcondició: L'operador obté la llista paginada de totes les execucions que compleixen els criteris de cerca.

Procés normal principal:

1. L'operador introdueix els criteris de cerca

2. El sistema recupera i mostra una llista paginada de totes les execucions que compleixen els criteris de cerca. Cada registre de la llista inclou: identificador execució, nom de planificació, nom de la tasca, data, estat.

Cas d'ús número 17. "Detall execució".

Resum de la funcionalitat: recuperació de totes les dades que el sistema té d'una execució.

Actors: Operador

Casos d'ús relacionats: Identificar-se, Cercar execució, Aturar execució, Avortar execució, Re-arrancar execució, Consultar log

Precondició: L'operador està identificat, l'execució existeix

Postcondició: Obtenim totes les dades d'una execució

Procés normal principal:

1. include:: Cercar execució

2. El consultor selecciona una execució del llistat obtingut al pas 1

3. El sistema recupera totes les dades de l'execució

Alternatives de procés i excepcions:

2a. El llistat obtingut no conté cap element

2a1 Retorn al pas 1

3a. L'operador escull l'opció d'aturar l'execució seleccionada

3a1 El sistema passa a executar el cas d'ús "Aturar execució"

3b. El consultor escull l'opció d'avortar l'execució seleccionada

3b1 El sistema passa a executar el cas d'ús "Avortar execució"

3c. El consultor escull l'opció de tornar a arrancar l'execució seleccionada

3c1 El sistema passa a executar el cas d'ús "Re-arrancar execució"

3d. El consultor escull l'opció de consultar el log de l'execució seleccionada

3d1 El sistema passa a executar el cas d'ús "Consultar log"

Cas d'ús número 18. "Aturar execució".

Resum de la funcionalitat: permet aturar temporalment l'execució d'una tasca. Per tal que aquesta petició pugui ser processada de forma correcta, cal que les accions programades dins la classe JAVA hagin previst aquesta possibilitat

Actors: Operador

Casos d'ús relacionats: Identificar-se, Detall execució

Precondició: L'operador està identificat, l'execució existeix i està en marxa

Postcondició: Es dóna ordre al motor d'execució de pausar l'execució de la tasca

Procés normal principal:

1. El sistema demana confirmació
2. L'operador confirma l'aturament
3. S'envia al motor d'execució la petició de pausa

Cas d'ús número 19. "Avortar execució".

Resum de la funcionalitat: permet aturar definitivament l'execució d'una tasca. Per tal que aquesta petició pugui ser processada de forma correcta, cal que les accions programades dins la classe JAVA hagin previst aquesta possibilitat

Actors: Operador

Casos d'ús relacionats: Identificar-se, Detall execució

Precondició: L'operador està identificat, l'execució existeix i està en marxa o en pausa

Postcondició: Es dóna ordre al motor d'execució d'aturar definitivament l'execució de la tasca.

Procés normal principal:

1. El sistema demana confirmació
2. L'operador confirma l'aturament
3. S'envia al motor d'execució la petició d'aturament definitiu

Cas d'ús número 20. "Re-arrancar execució".

Resum de la funcionalitat: permet tornar a posar en marxa l'execució d'una tasca. Per tal que aquesta petició pugui ser processada de forma correcta, cal que les accions programades dins la classe JAVA hagin previst aquesta possibilitat

Actors: Operador

Casos d'ús relacionats: Identificar-se, Detall execució

Precondició: L'operador està identificat, l'execució existeix i està en pausa

Postcondició: Es dóna ordre al motor d'execució de tornar a posar en marxa l'execució de la tasca.

Procés normal principal:

1. El sistema demana confirmació
2. L'operador confirma l'aturament
3. S'envia al motor d'execució la petició de tornar a posar en marxa la tasca

Cas d'ús número 21. "Consultar notificacions".

Resum de la funcionalitat: l'usuari obté una llista de notificacions de canvi d'estat d'execucions, rebudes en una adreça de correu concreta.

Actors: Operador

Casos d'ús relacionats: Identificar-se, Notificar canvi d'estat

Precondició: L'operador està identificat

Postcondició: L'operador obté la llista paginada de totes les notificacions enviades a una adreça de correu.

Procés normal principal:

1. L'operador introdueix l'adreça de correu
2. El sistema recupera i mostra una llista paginada de totes les notificacions de canvi d'estat d'execucions rebudes en aquesta adreça de correu.

2.5 Requisits no funcionals

L'aplicació ha de poder executar-se dins de qualsevol servidor d'aplicacions J2EE, i ha de ser susceptible de ser distribuïda en una topologia de cluster.

Els temps de resposta de totes les operacions síncrones, dutes a terme pels usuaris, no poden superar els 3 segons.

S'ha d'extremar la portabilitat de la solució, així com l'ús de components i tecnologies estàndard.

2.6 Requisits de la UI

Tots els usuaris del sistema (consultors i operadors) estan molt acostumats a treballar amb ordinadors i no presenten cap limitació ni consideració especial que hagi de condicionar el disseny de la UI.

Totes les funcions del programari s'utilitzaran molt sovint.

3 Anàlisi de Requisits

3.1 Revisió de casos d'ús

La base de partida per a l'anàlisi és la documentació sobre els casos d'ús elaborada en l'etapa anterior. El detall actual ja es considera suficient i per tant ja es poden utilitzar com a base per als passos posteriors.

3.2 Paquets d'anàlisi i serveis

Tots els casos d'ús descrits poden agrupar-se en els següents paquets d'anàlisi:

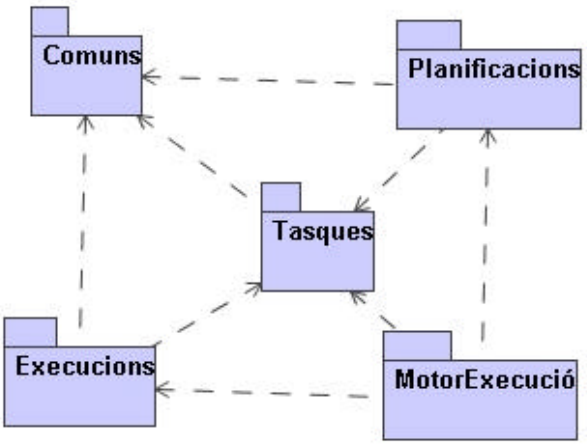


Figura 6: Diagrama de paquets d'anàlisi

Paquet "**Comuns**". Inclou els casos d'ús: Identificar-se

Paquet "**Tasques**". Inclou els casos d'ús: Definir tasca, Cercar tasca, Detall tasca, Modificar tasca, Inhabilitar tasca, Esborrar tasca

Paquet "**Planificacions**". Inclou els casos d'ús: Planificar tasca, Cercar planificació, Detall planificació, Modificar planificació, Inhabilitar planificació, Esborrar planificació.

Paquet "**Execucions**". Inclou els casos d'ús: Cercar execució, Detall execució, Aturar execució, Avortar execució, Re-arrancar execució, Consultar notificacions

Paquet "**MotorExecució**". Inclou els casos d'ús: Nova Execució, Notificar canvi estat.

Degut a la seva simplicitat, no s'ha considerat necessari afegir cap paquet de servei dins dels paquets d'anàlisi.

3.3 Classes d'entitat

De la lectura de tots els casos d'ús s'han identificat les següents classes d'entitat i les relacions entre elles:

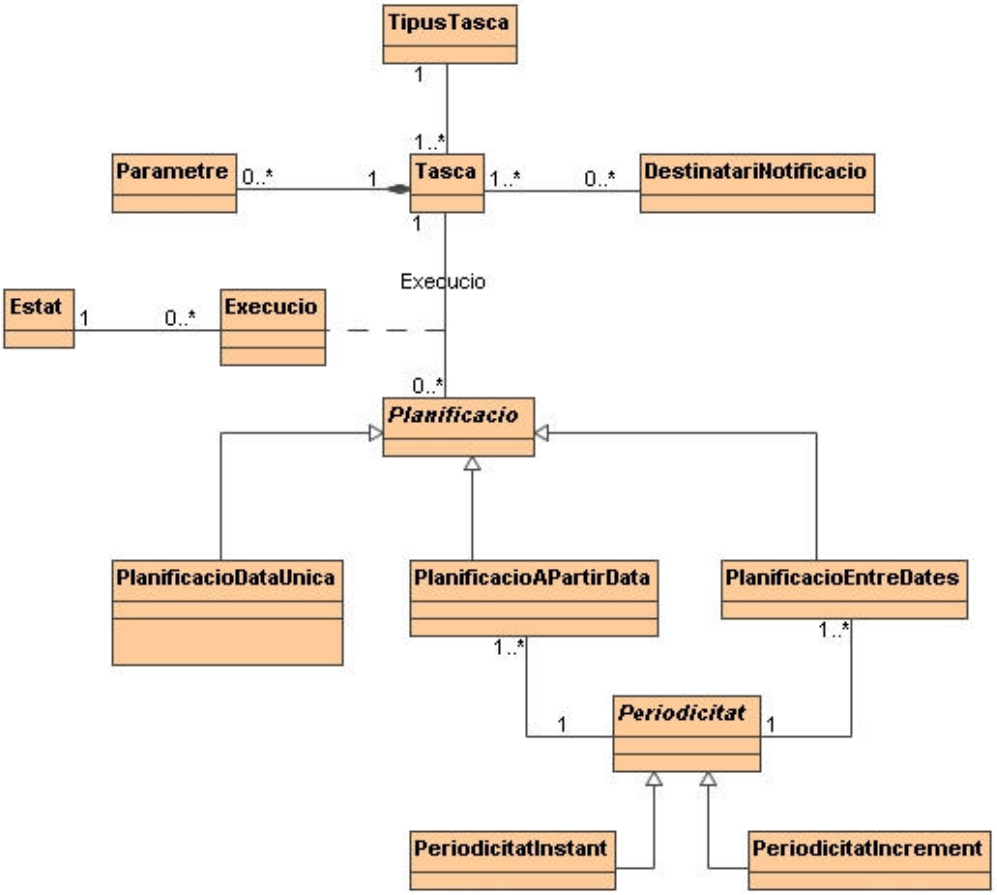
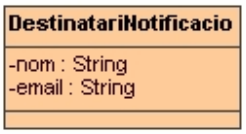
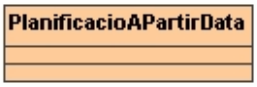
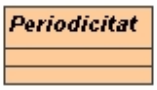
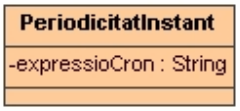
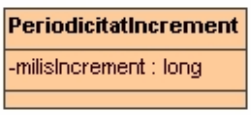
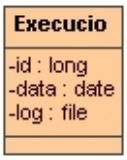
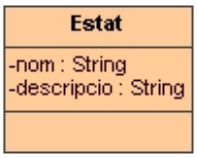


Figura 7: Diagrama de classes d'entitat

I el detall de cada entitat és:

Tasca -id : long -nom : String -descripcio : String -classeJAVA : String -habilitada : boolean	Classe que representa el conjunt d'accions a executar pel sistema.
TipusTasca -nom : String -descripcio : String	Amb l'objectiu de poder agrupar les tasques, per tal de facilitar la seva gestió s'incorpora una categorització de les mateixes.

	Modela els paràmetres d'execució associats a una tasca.
	Agrupa tota la informació necessària sobre les adreces de correu a les que s'ha de notificar els possibles canvis d'estat de les tasques.
	Representa la planificació d'una tasca, és a dir, la informació sobre l'instant o instants de temps en que una tasca s'ha d'executar. Es tracta d'una classe abstracta.
	Tipus de planificació a associar a aquelles tasques que només s'han d'executar un cop en una data concreta.
	Tipus de planificació a associar a aquelles tasques que s'han d'executar a partir d'una data concreta i d'acord a una periodicitat.
	Tipus de planificació a associar a aquelles tasques que s'han d'executar entre dos dates concretes i d'acord a una periodicitat.
	Representa la periodicitat d'execució d'una tasca. Es tracta duna classe abstracta.
	Tipus de periodicitat a establir mitjançant una *crontab expression* que indiqui tots els instants de temps en que una tasca s'ha d'executar.
	Tipus de periodicitat a utilitzar quan una tasca s'ha d'executar passat un cert temps des de l'últim cop. L'increment s'ha d'expressar en milisegons.

	Informació sobre l'execució d'una tasca
	Possible estat de l'execució d'una tasca (aturada, avortada, en execució, finalitzada ok, finalitzada amb error)

3.4 Diagrama d'estats

Els estats associats a una tasca i les seves possibles transicions són:

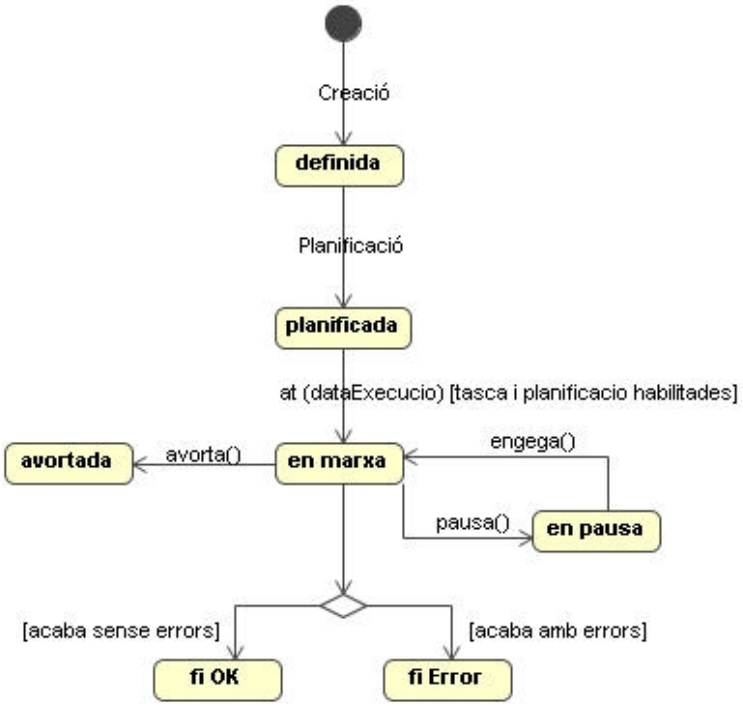


Figura 8: Diagrama d'estats d'una tasca

3.5 Classes de frontera, de control i de les operacions

Per a cada cas d'ús es farà un diagrama de seqüències.

A tots els casos d'ús s'ha posat que l'actor demana una opció a la classe de frontera Menu, que correspon al menú de l'aplicació, i aquesta passa a la classe de control GestorMenu, que crida a la classe de control principal del cas d'ús. Els noms dels missatges seran operacions de les classes destinatàries; aquells que van de les classes de frontera a l'actor no li demanen cap operació.

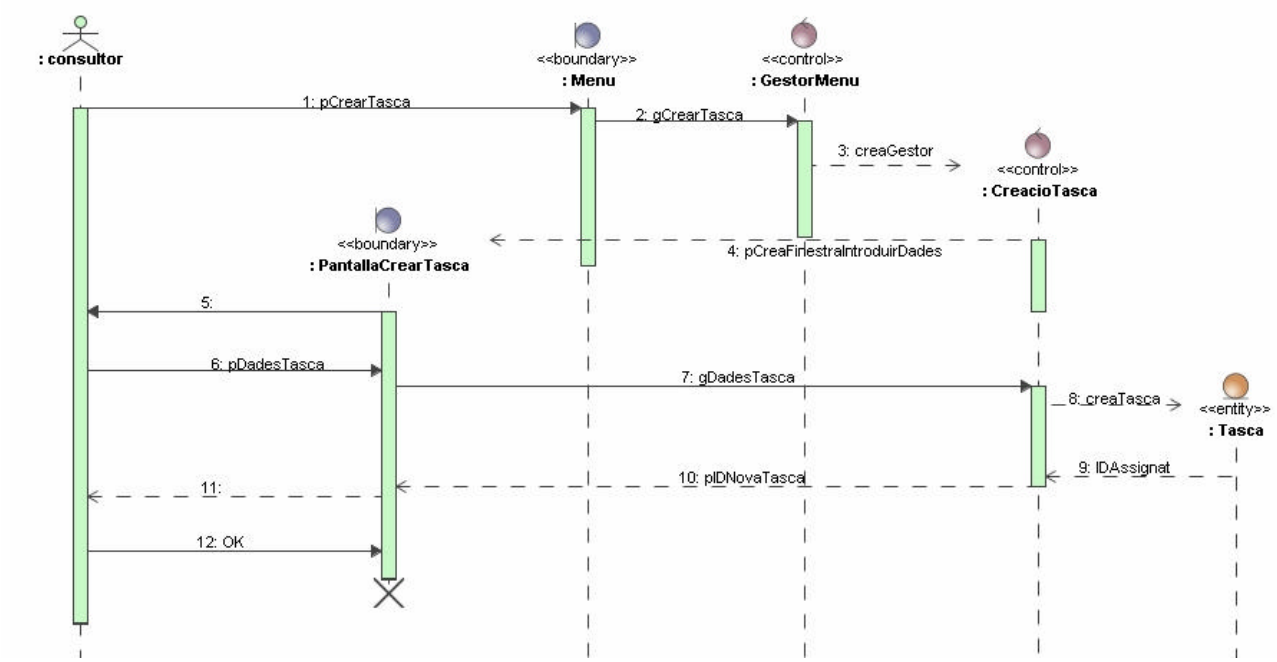


Figura 9: Diagrama de Seqüència de "Definir Tasca"

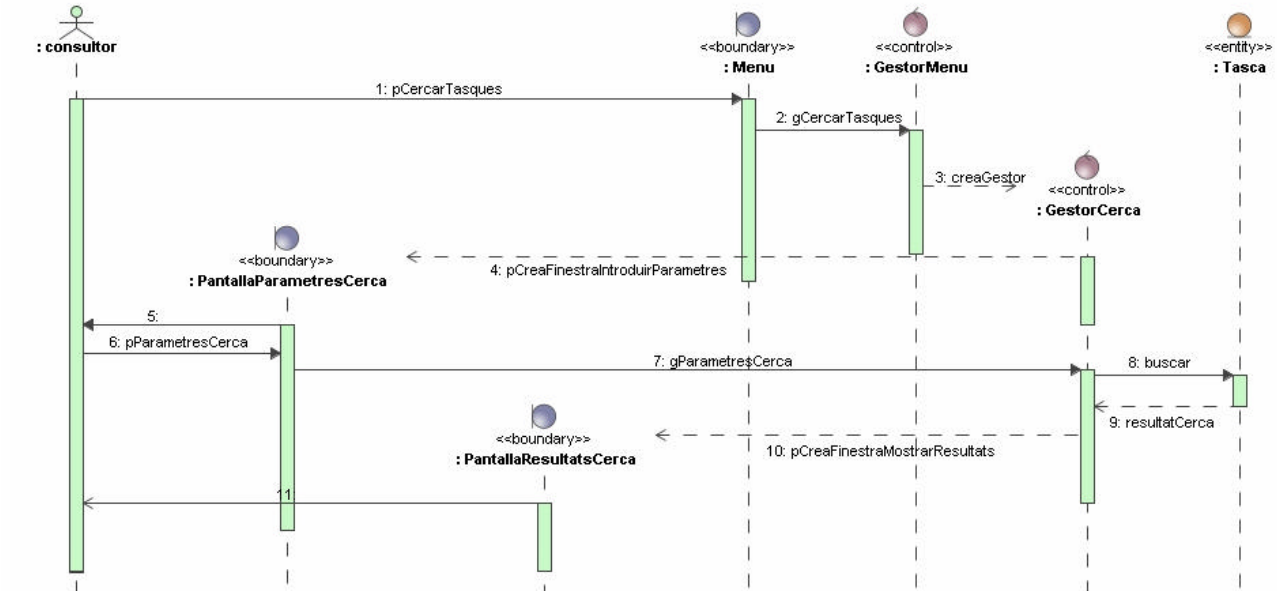


Figura 10: Diagrama de Seqüència de "Cercar Tasca"

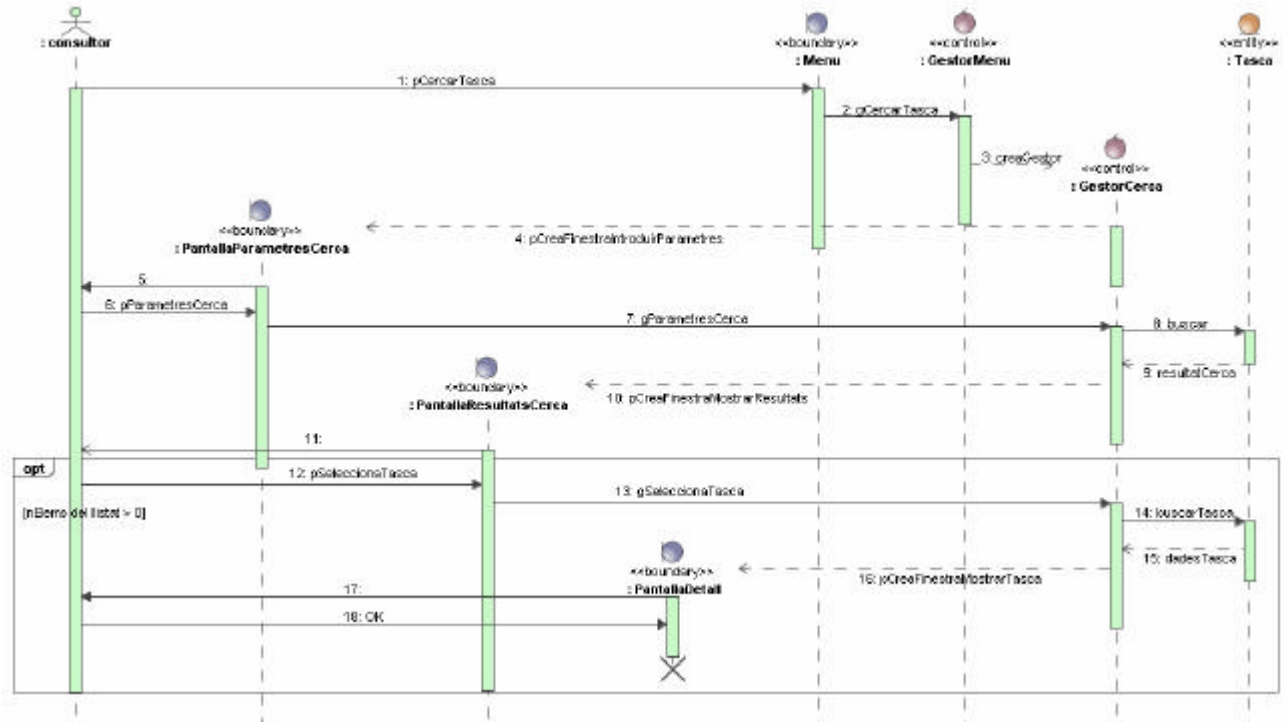


Figura 11: Diagrama de Seqüència de "Detall Tasca"

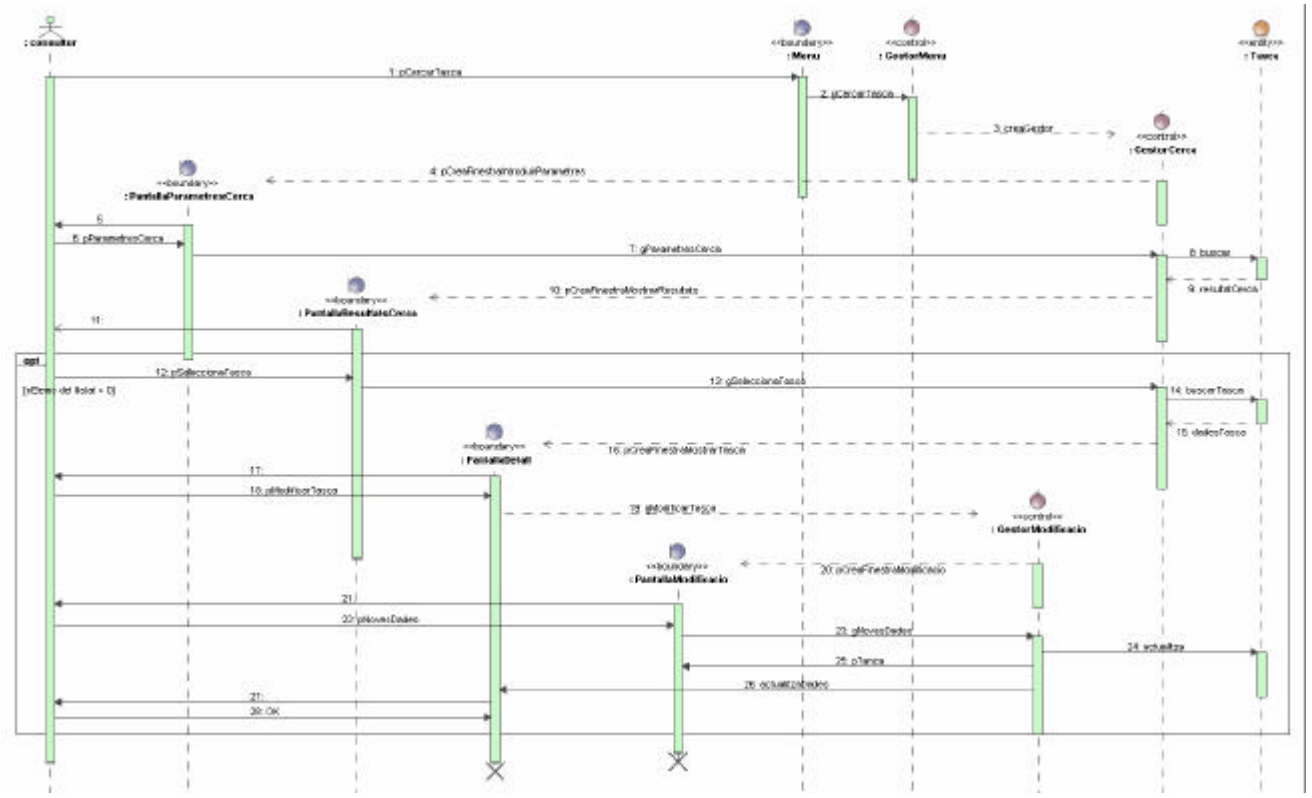


Figura 12: Diagrama de Seqüència de "Modificar Tasca"

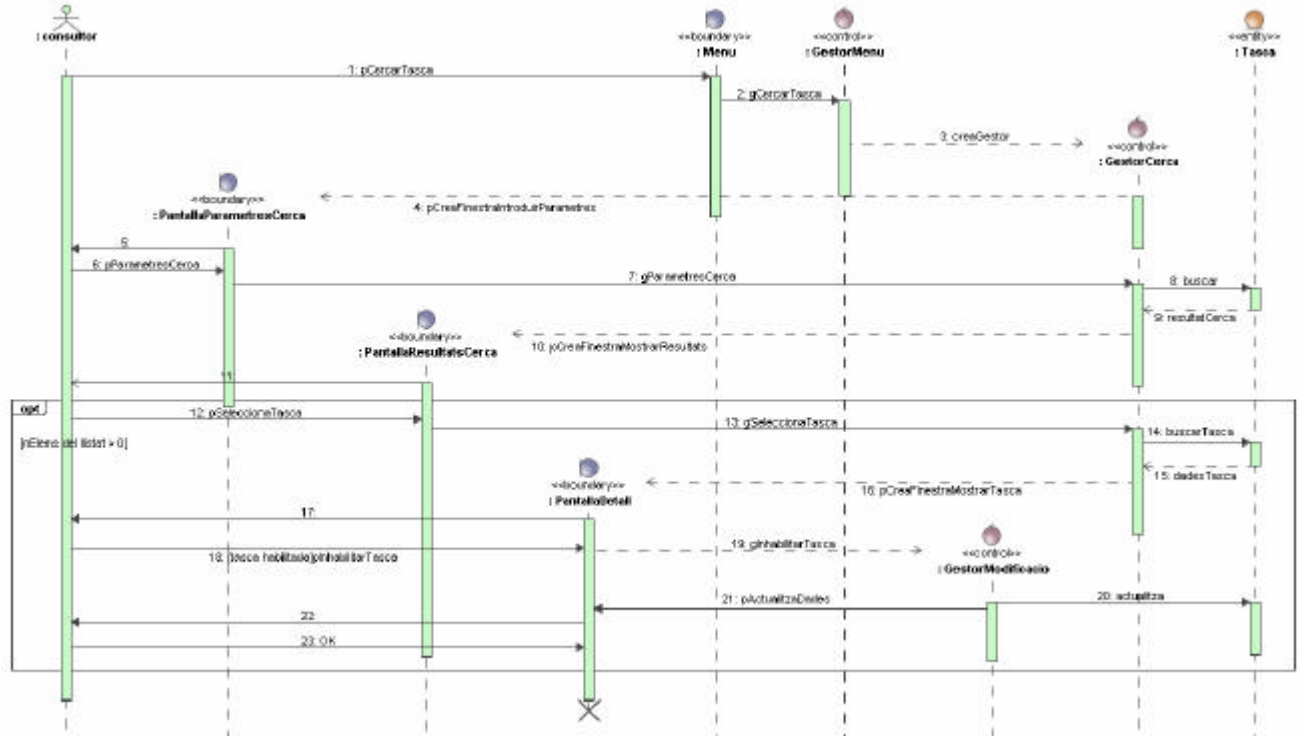


Figura 13: Diagrama de Seqüència de "Inhabilitar tasca"

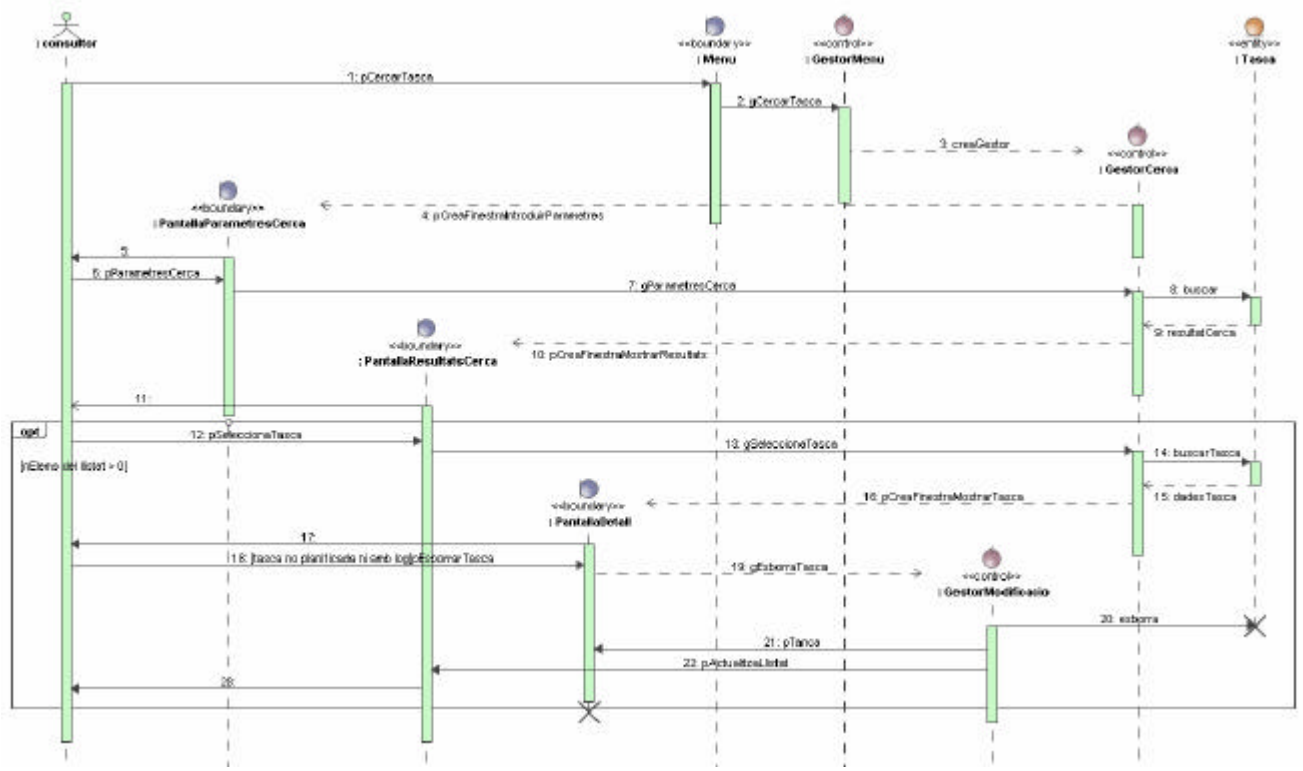


Figura 14: Diagrama de Seqüència de "Esborrar tasca"

Els casos d'ús referents a planificacions són totalment anàlegs als presentats fins ara sobre tasques i per aquest motiu no s'inclou els seus diagrames de seqüència.

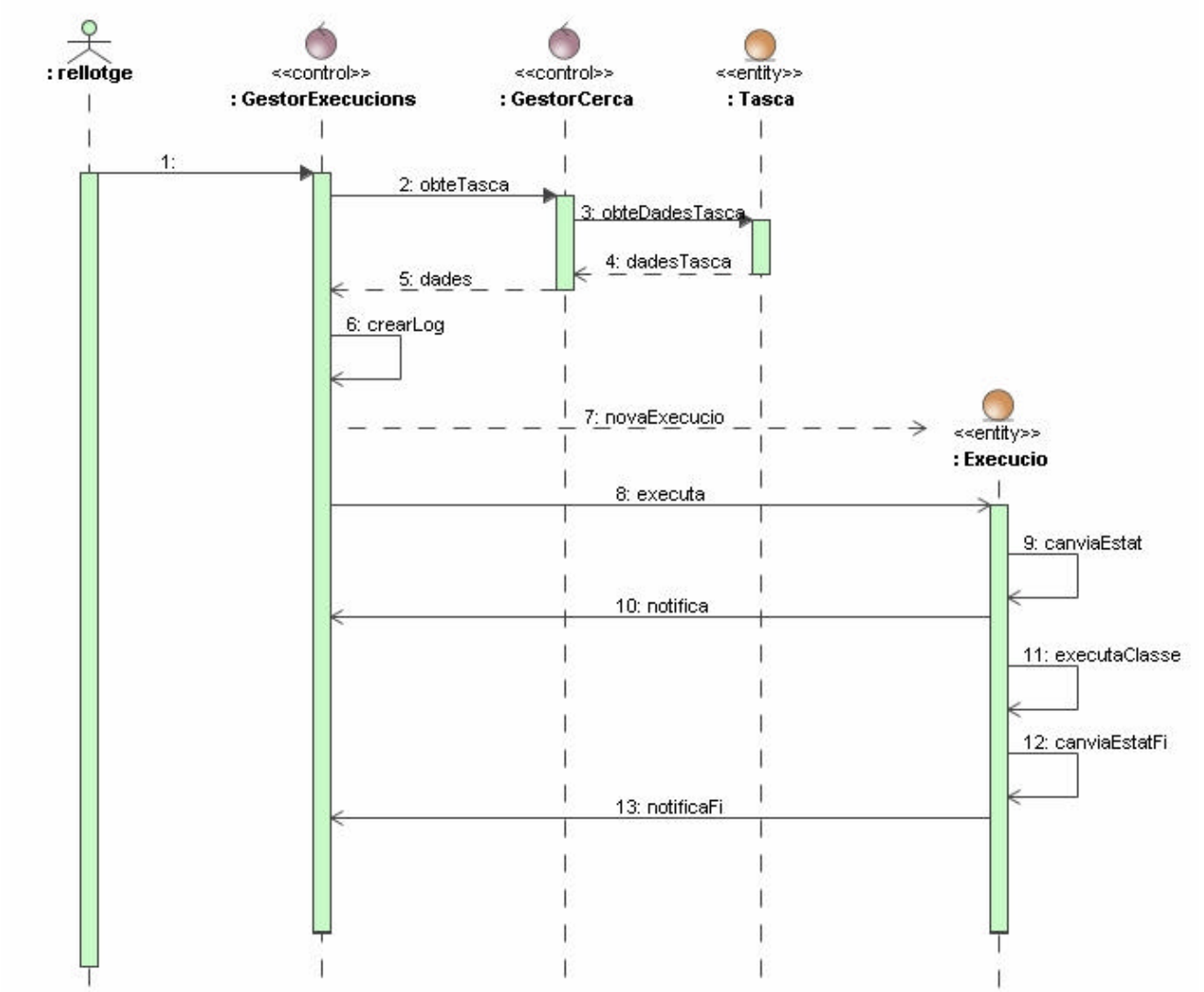


Figura 15: Diagrama de Seqüència de "Nova Execució"

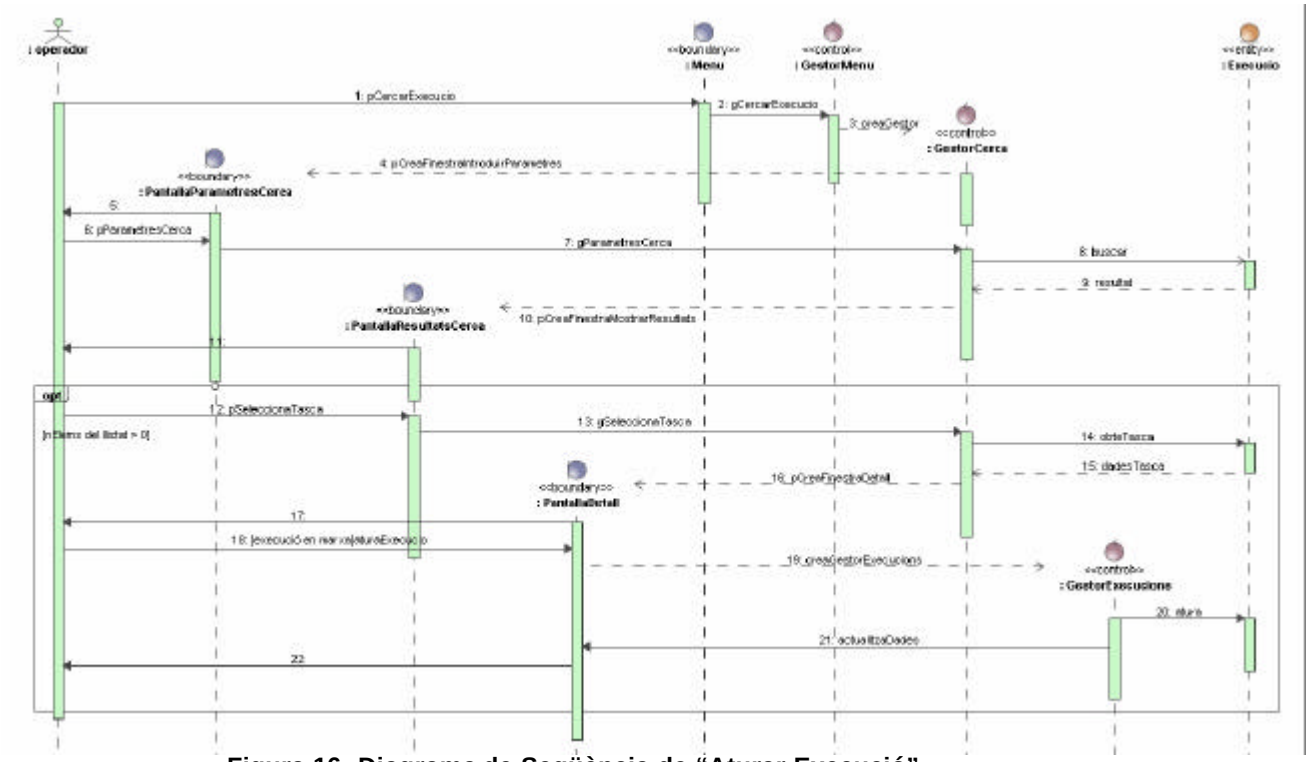


Figura 16: Diagrama de Seqüència de "Aturar Execució"

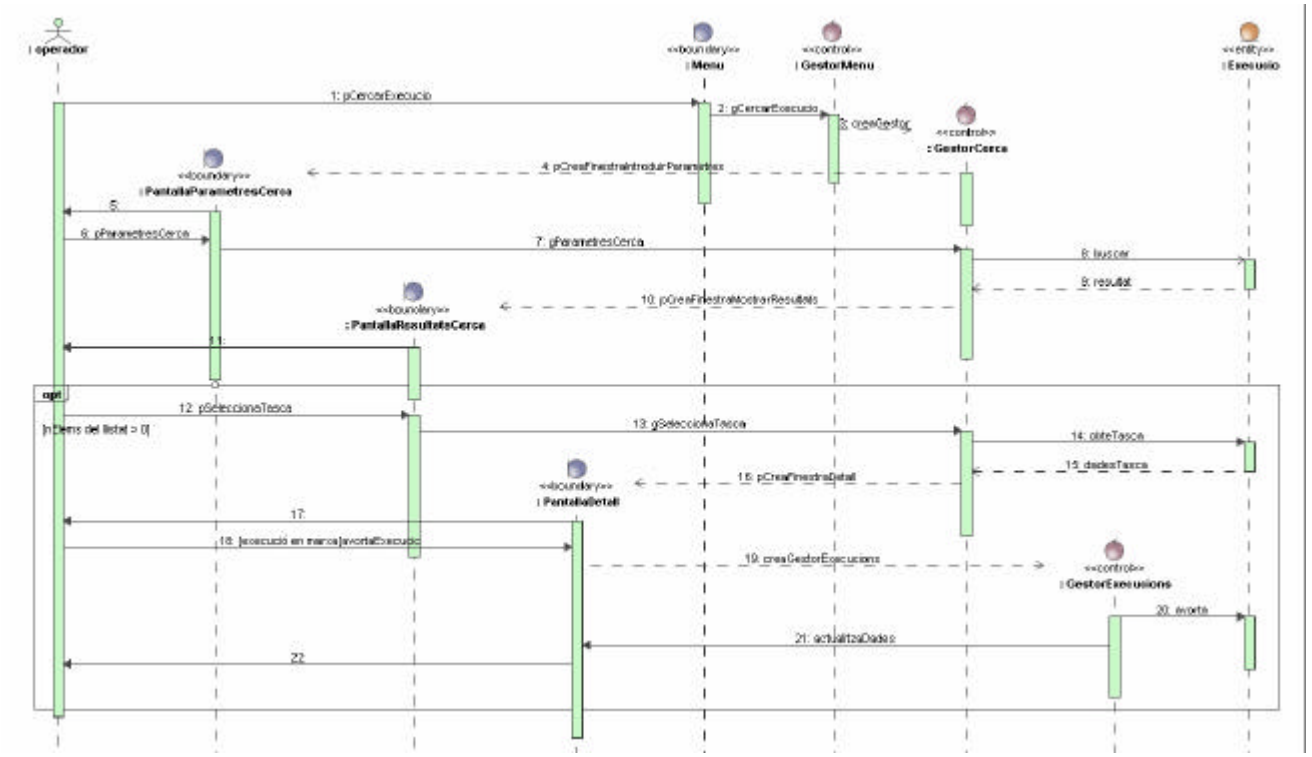


Figura 17: Diagrama de Seqüència de "Avortar Execució"

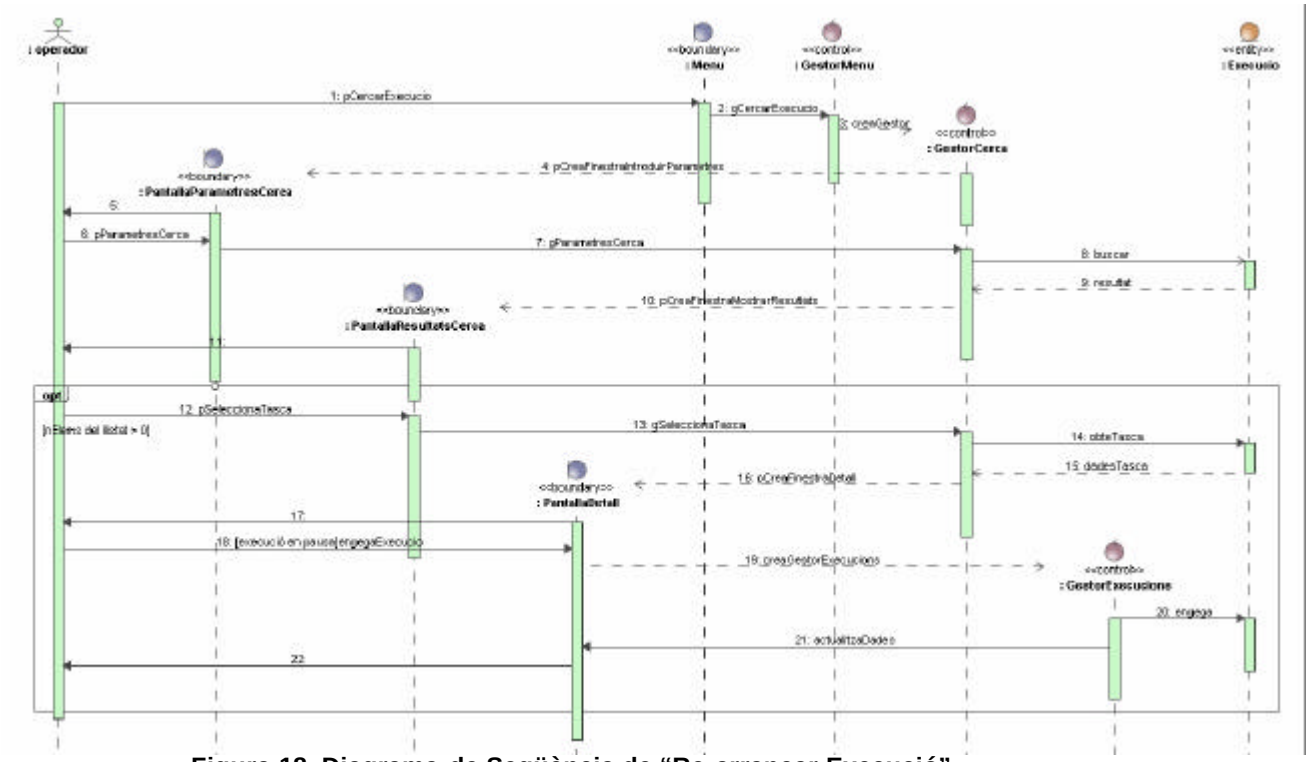


Figura 18: Diagrama de Seqüència de "Re-arrancar Execució"

4 Disseny

4.1 Arquitectura

El sistema a desenvolupar es divideix en 4 subsistemes, corresponents als 4 paquets d'anàlisi identificats anteriorment:

1. **Subsistema de definició**. Inclou totes les funcionalitats referents a la definició i gestió de les tasques de llarga durada.

2. **Subsistema gestió de la planificació**. Conjunt de funcionalitats referents a la gestió de les planificacions de les tasques.

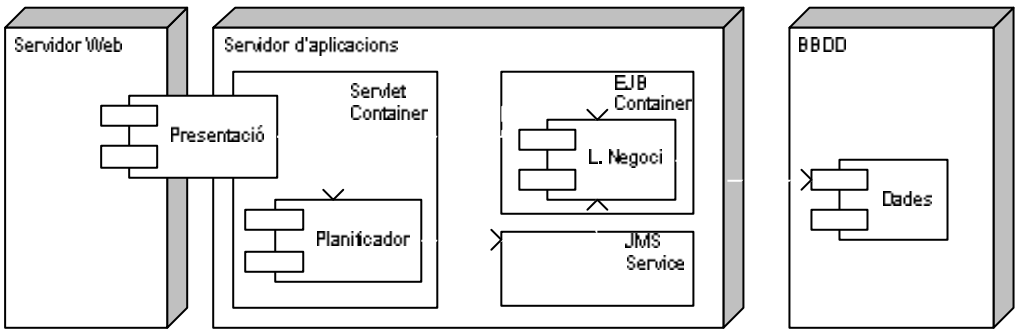
3. **Subsistema gestió d'execucions**. Inclou totes les funcionalitats que permeten interaccionar amb l'execució de les tasques: pausar, avortar ...

4. **Subsistema motor d'execució**. Funcionalitats encarregades d'executar les tasques, mitjançant missatges JMS i Message Driven Beans (MDB).

D'acord a l'especificació J2EE l'aplicació, i per tant tots els subsistemes, es dividirà en capes intentant que siguin el més desacoblades possible:

- **Presentació**. UI de l'aplicació.
- **Lògica de negoci**. Implementació de les funcionalitats.
- **Persistència**. Enregistrament perdurable de les dades.

En el següent diagrama s'il·lustra la ubicació de cada component:



4.2 Frameworks i programari base

Els *frameworks* que s'utilitzaran en el desenvolupament de l'aplicació són:

- Presentació. **JSF** integrat amb **Struts**. La llibreria de components gràfics JSF utilitzada ha estat **JBoss RichFaces**, que inclou suport per a Ajax.
- Planificador. **Quartz** *d'Opensymphony* .
- Negoci. Les invocacions des de presentació es realitzaran mitjançant **Session Stateless** i les invocacions que, mitjançant missatges **JMS**, faci el planificador s'atendran per mitjà de **Message Driven Beans**.
- Accés a dades. **Hibernate**.

El software base a utilitzar serà:

- Servidor web: **Apache 2.2** (encarregat de servir tot el contingut estàtic)
- Servidor d'aplicacions: **JBoss 4.2.2** (incorpora el servei JMS)
- Servidor BBDD: **Postgres 8.2**
- Servidor SMTP: **Java Email Server 1.6** (inclòs dins el codi de l'aplicació)

4.3 Patrons

Els patrons que s'implementaran són:

- **MVC (Model/View/Controller)**. El fet d'utilitzar *Struts* i *JSF* ja ens obliga a utilitzar aquest patró. La principal avantatja que s'obté és la separació clara de les diferents capes de l'aplicació i de les responsabilitats de cadascuna.
- **Factory**. La gestió de cada entitat de negoci s'agruparà en una classe que permetrà crear, cercar, modificar i esborrar-les.
- **DAO**. Encapsulen tots aquells accessos a components d'infraestructura o repositoris que no es facin mitjançant una Factory (entitats del domini de l'aplicació), com per exemple l'accés a un servidor de correu SMTP.
- **Façade**. Cada subsistema oferirà els seus serveis per mitjà d'una classe que amagarà si la informació s'obté per mitjà d'una Factory o d'un DAO. Quan aquesta "Façade" s'utilitzi per publicar els mètodes de negoci, en direm "**Business Façade**"
- **Command**. Per a l'execució de les classes JAVA associades a les tasques, utilitzant un únic MDB.
- **Singleton**. A utilitzar en aquelles classes que només contenen lògica i no representen cap entitat del domini.

4.4 Persistència

4.4.1 Entitats i atributs

TASCA

codi-tasca, nom, descripció, classeJAVA, habilitada, tipus

{tipus} és clau forana a TIPUSTASCA

TIPUSTASCA

nom, descripció

PARAMETRE (entitat dèbil de TASCA)

codi-tasca, nom ,valor

{codi-tasca} és clau forana a TASCA

DESTINATARI

nom, email

AVISA (materialització de la relació M:N entre TASCA i DESTINATARI)

codi-tasca, nom

{codi-tasca} és clau forana a TASCA

{nom} és clau forana a DESTINATARI

EXECUCIO (entitat associativa entre TASCA i PLANIFICACIO)

codi-execucio, codi-tasca, codi-planificació, data, log, estat

{codi-tasca} és clau forana a TASCA

{codi-planificació} és clau forana a PLANIFICACIO

{estat} és clau forana a ESTAT

ESTAT

nom ,valor

PLANIFICACIO

codi-planificació, nom, descripció, codi-tasca, dataExecució, habilitada, errorInhabilita, tipus, dataFi, periode

{codi-tasca} és clau forana a TASCA

{ periode } és clau forana a PERIODICITAT

{ dataFi } pot ser nul

PERIODICITAT

codi-periodicitat

PERINSTANT (subclasse de PERIODICITAT)

codi-periodicitat, expressio

{ codi-periodicitat } és clau forana a PERIODICITAT

PERINCREMENT (subclasse de PERIODICITAT)

codi-periodicitat, increment

{ codi-periodicitat } és clau forana a PERIODICITAT

4.4.2 Diagrama ER

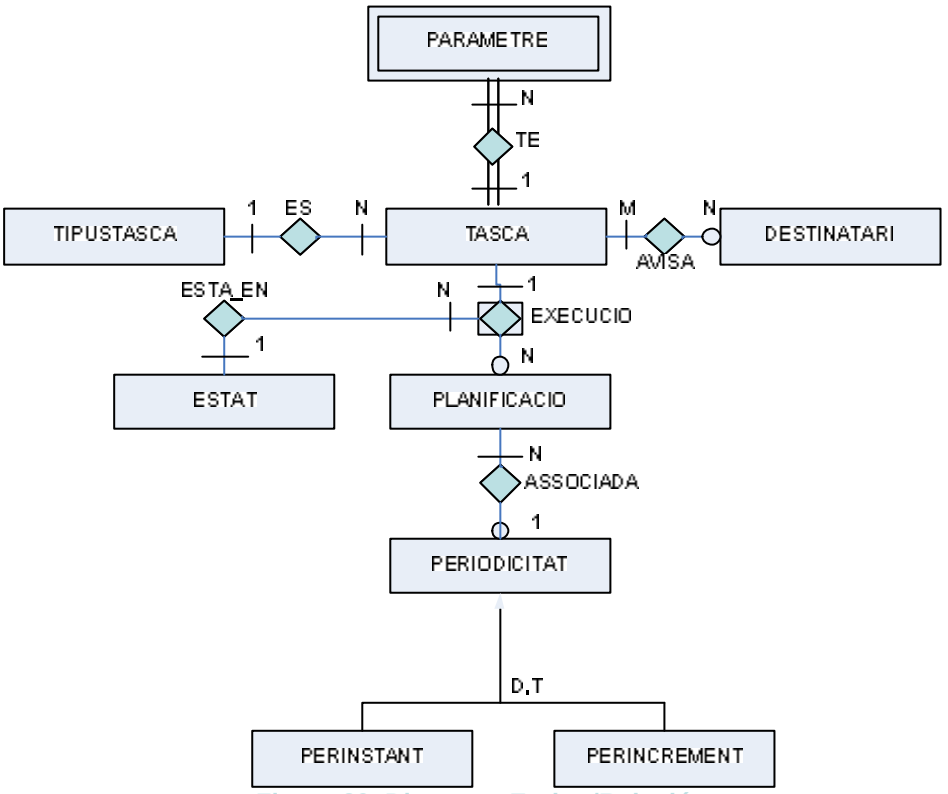


Figura 20: Diagrama Entitat/Relació

4.5 Revisió model estàtic

Respecte a les classes del domini identificades durant la fase d'anàlisi, es proposa eliminar la jerarquia de les planificacions:

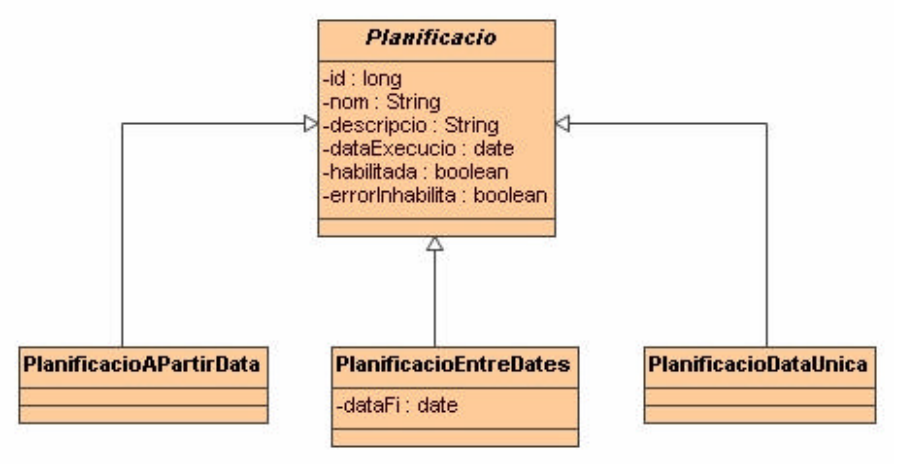


Figura 21: Jerarquia de classes de les planificacions

En el seu lloc es proposa utilitzar una única classe que inclogui tots els atributs possibles de qualsevol planificació, més un que indiqui el tipus. En el cas que algun atribut no tingui sentit per a un tipus concret cal assignar-li el valor "null".

4.6 Classes de disseny

4.6.1 Lògica de negoci

Aplicant els patrons Façade, DAO i Factory les classes resultants tindran un dels següents rols:

Component de negoci	Classes que ofereixen els mètodes públics "de negoci". Internament utilitzen factories i DAOs per obtenir i manipular les entitats del domini. Són els encarregats de validar les regles de negoci que facin falta, i d'obrir i tancar connexions i transaccions.
Factoria	Encapsula la gestió de les entitats de negoci mitjançant Hibernate
DAO	Encapsula els accessos a aquells components d'infraestructura o repositoris que no es facin mitjançant una Factoria

Si representem gràficament aquestes dependencies:

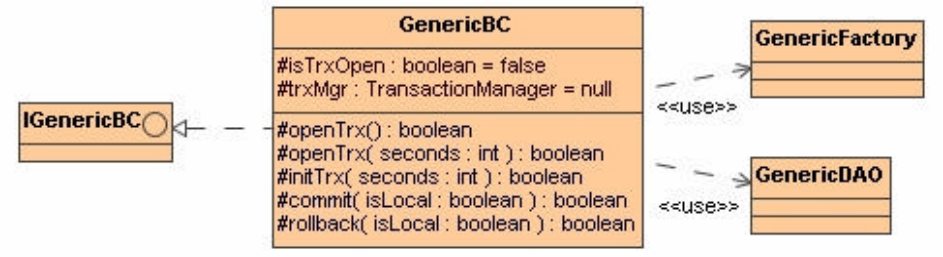


Figura 22: Diagrama de classes de negoci

I aplicant aquest mateix criteri als diferents subsistemes a implementar:

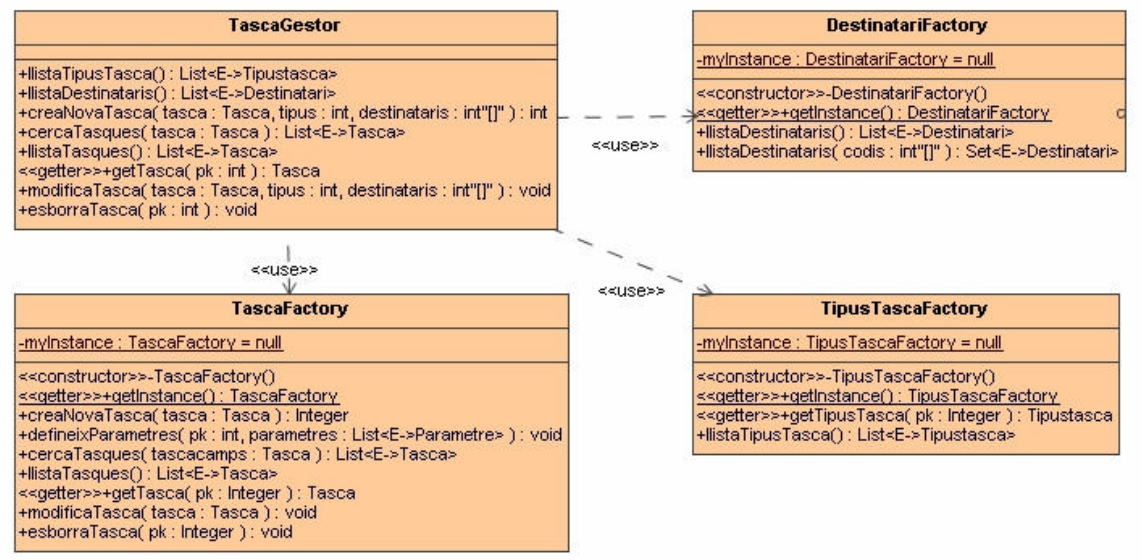


Figura 23: Diagrama de classes de negoci associades a Tasques

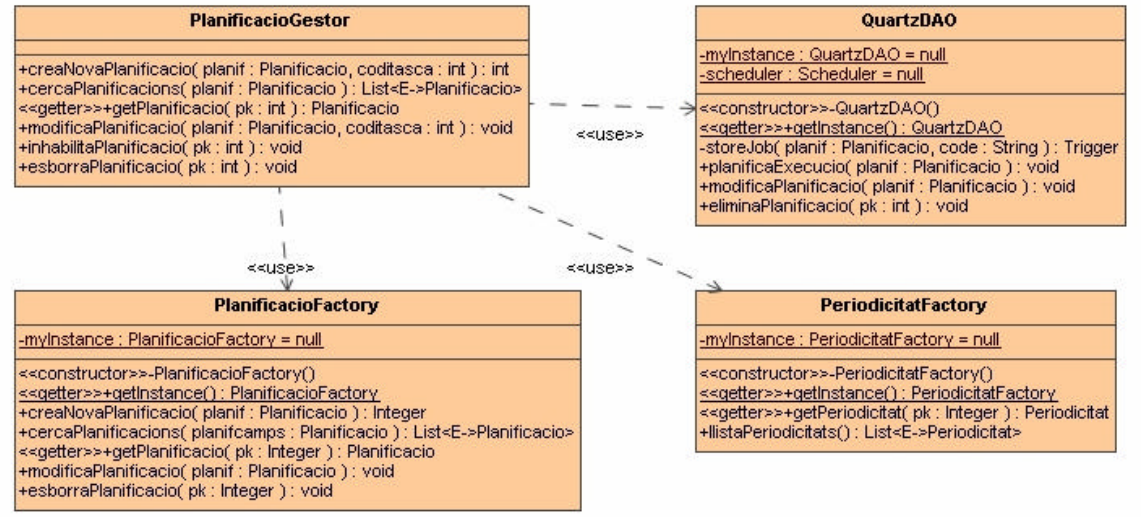


Figura 24: Diagrama de classes de negoci associades a Planificacions

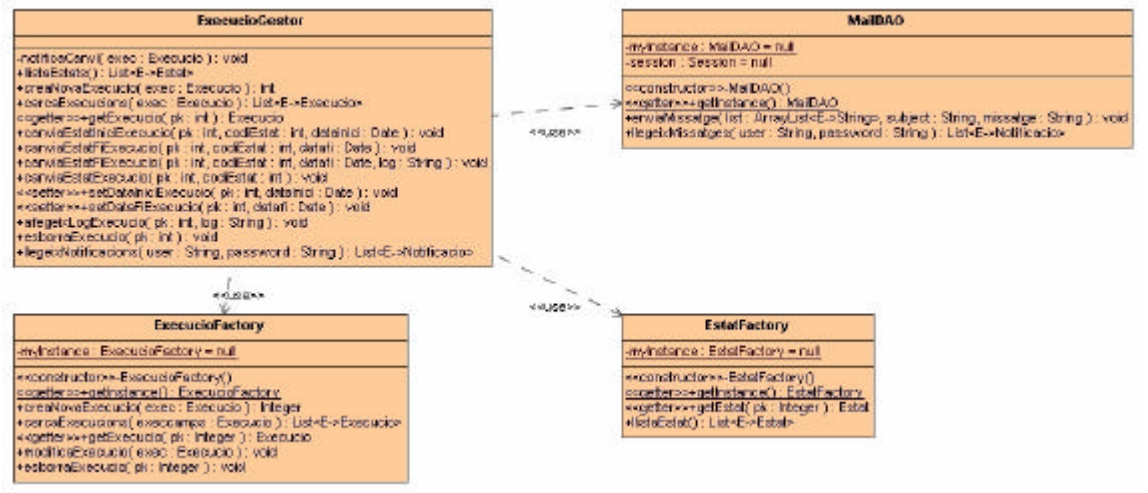


Figura 25: Diagrama de classes de negoci associades a Execucions

4.6.2 Localitzador de serveis

Aquest servei és l'encarregat d'independitzar la capa web (client) de la capa de lògica de negoci en dos aspectes:

1. Protocol utilitzat en la invocació (rmi, jms, http). Sigui quin sigui el protocol utilitzat el codi de la capa web no sofreix cap canvi.

2. Implementació del component de negoci. A nivell de compilació la capa web només depèn de la interfície de negoci, no de la seva implementació.

De forma molt esquemàtica podem representar aquest comportament com:

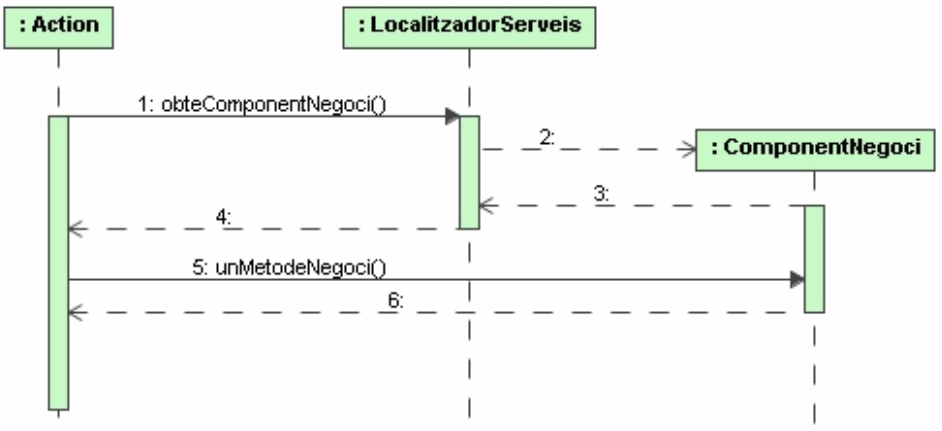


Figura 26: Diagrama de Seqüència de la invocació a la lògica de negoci

Per a la implementació d'aquest component s'utilitzarà la llibreria "cglib" i els EJB de sessió sense estat.

4.6.3 Presentació

Com a conseqüència de l'ús de Struts, les jerarquies de classes de la capa web estan condicionades pel propi *framework* , i són:

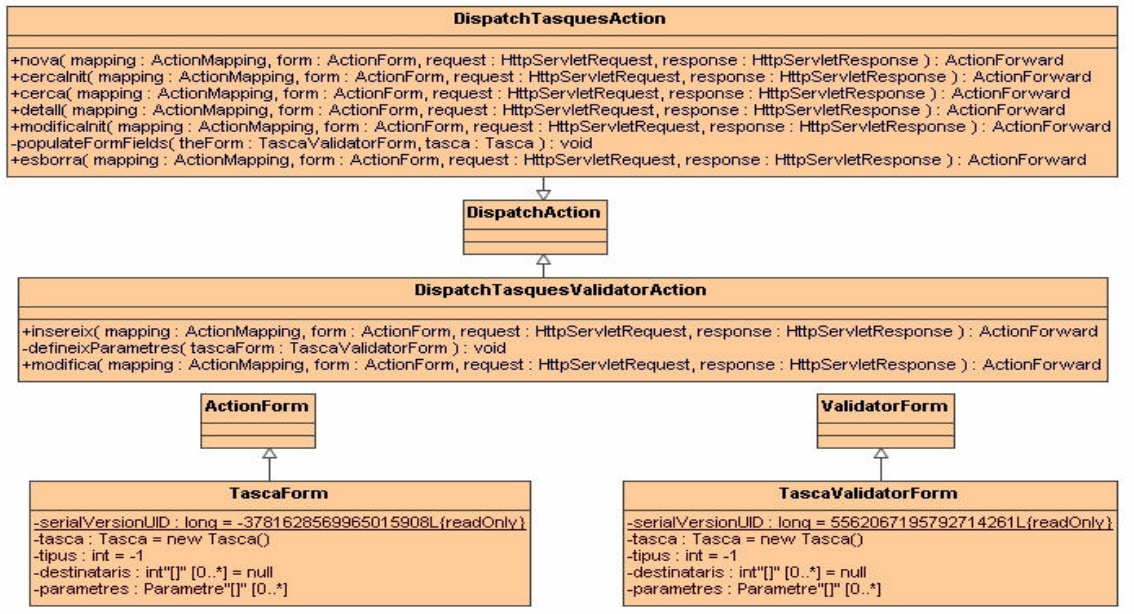


Figura 27: Diagrama de classes de presentació associades a Tasques

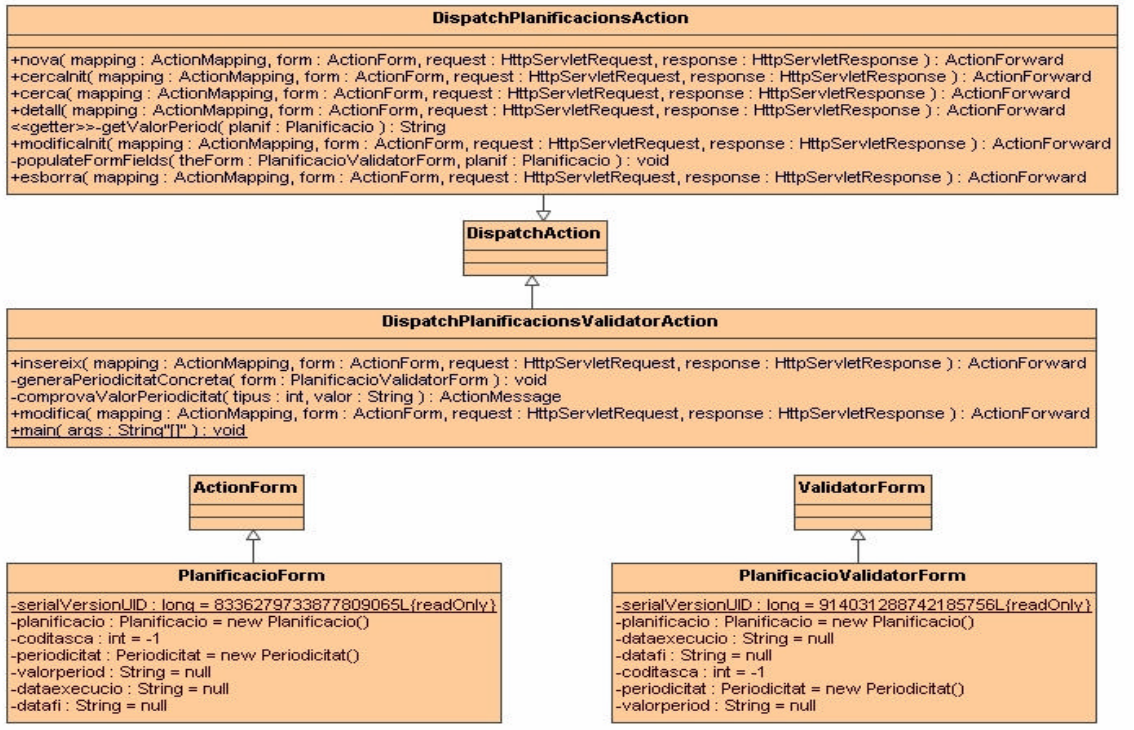


Figura 28: Diagrama de classes de presentació associades a Planificacions

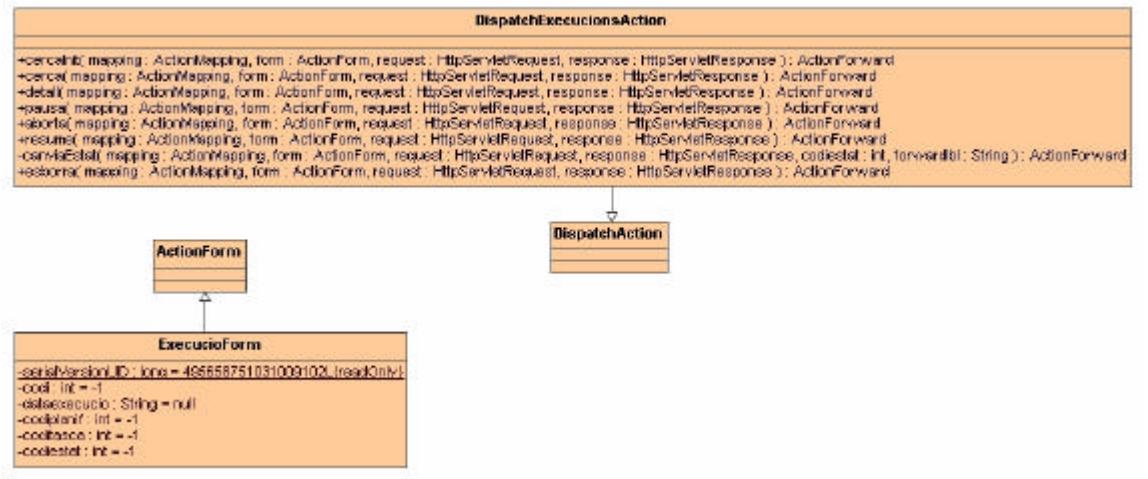


Figura 29: Diagrama de classes de presentació associades a Execucions

4.6.4 Propagació dels errors

El mecanisme a utilitzar és la propagació d'excepcions i en conseqüència s'ha definit la següent jerarquia de classes:

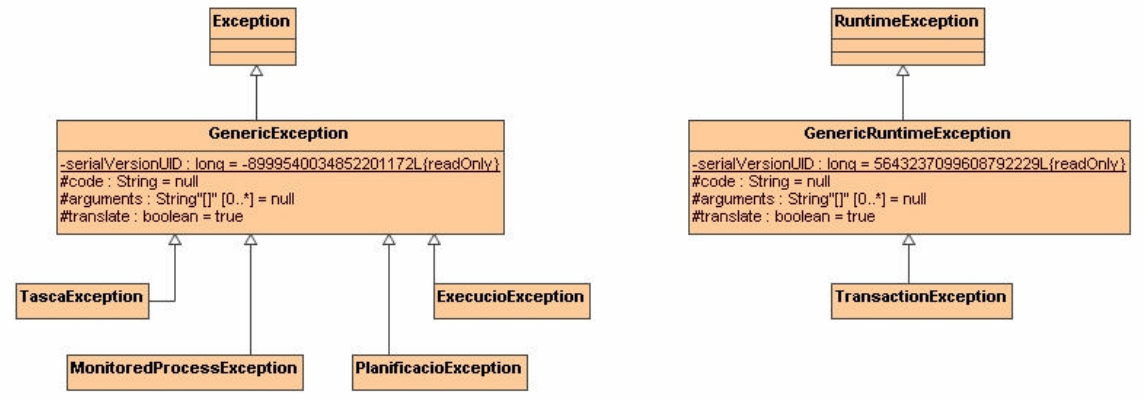


Figura 30: Jerarquia de classes de les Excepcions

4.7 Disseny de la UI

La interfície d'usuari de l'aplicació es construirà mitjançant:

- Els components gràfics de la llibreria JBoss RichFaces. Especialment útil en la creació de llistes paginades, finestres modals i gestió d'events mitjançant Ajax.
- Una fulla d'estil en cascada (CSS) comú a totes les pàgines de l'aplicació.
- La llibreria de tags html del *framework* Struts.
- L'únic navegador suportat és el Microsoft Internet Explorer 7.0 o superior.

Tot seguit es presenta un recull de les principals finestres de l'aplicació on pot observar-se l'aplicació de les diferents tecnologies:

Figura 31: Finestra de benvinguda a l'aplicació

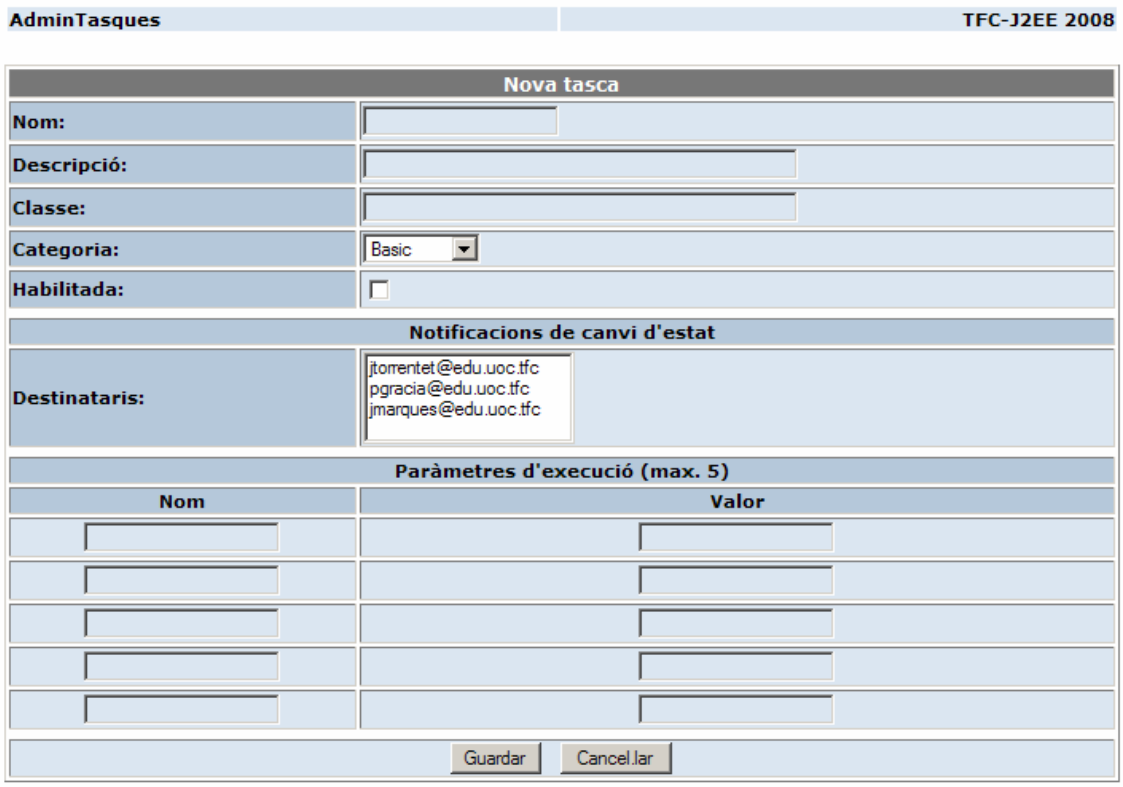


Figura 32: Finestra de creació d'una nova tasca

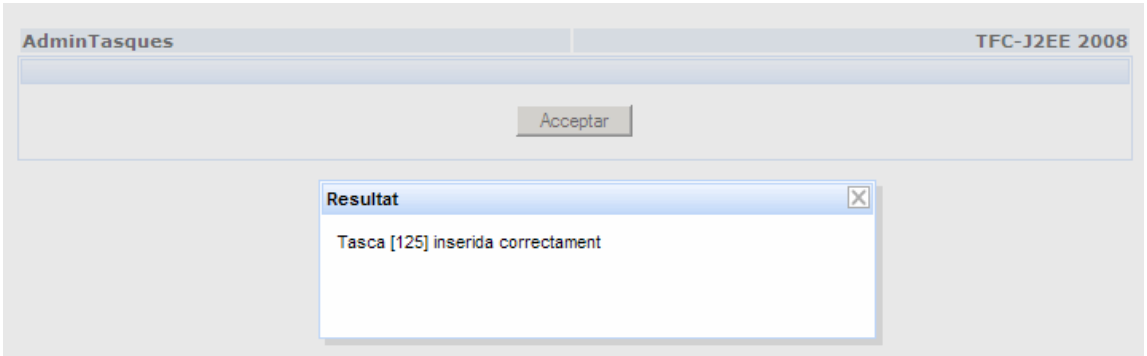


Figura 33: Finestra modal que informa del resultat correcte d'una inserció

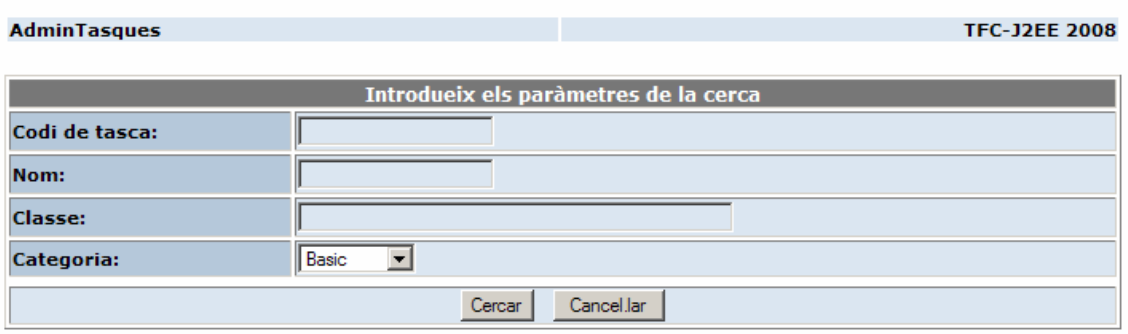


Figura 34: Finestra on introduir els paràmetres de cerca de tasques

Figura 35: Finestra on es mostren de forma paginada els resultats d'una cerca

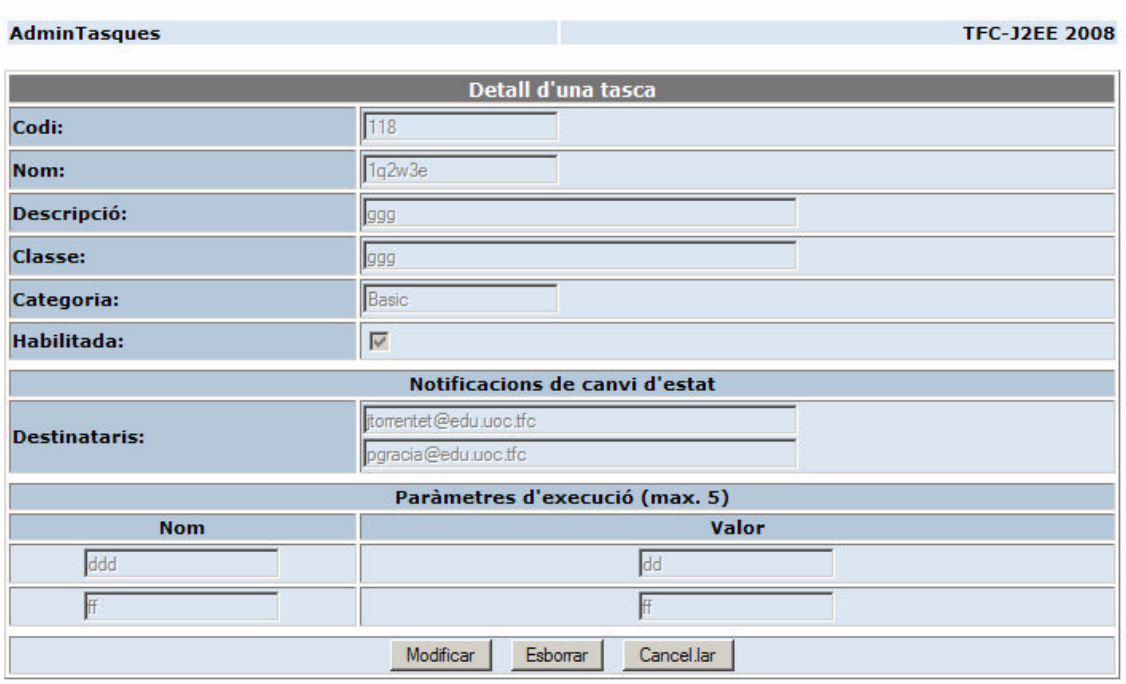


Figura 36: Finestra on es mostra tota la informació d'una tasca

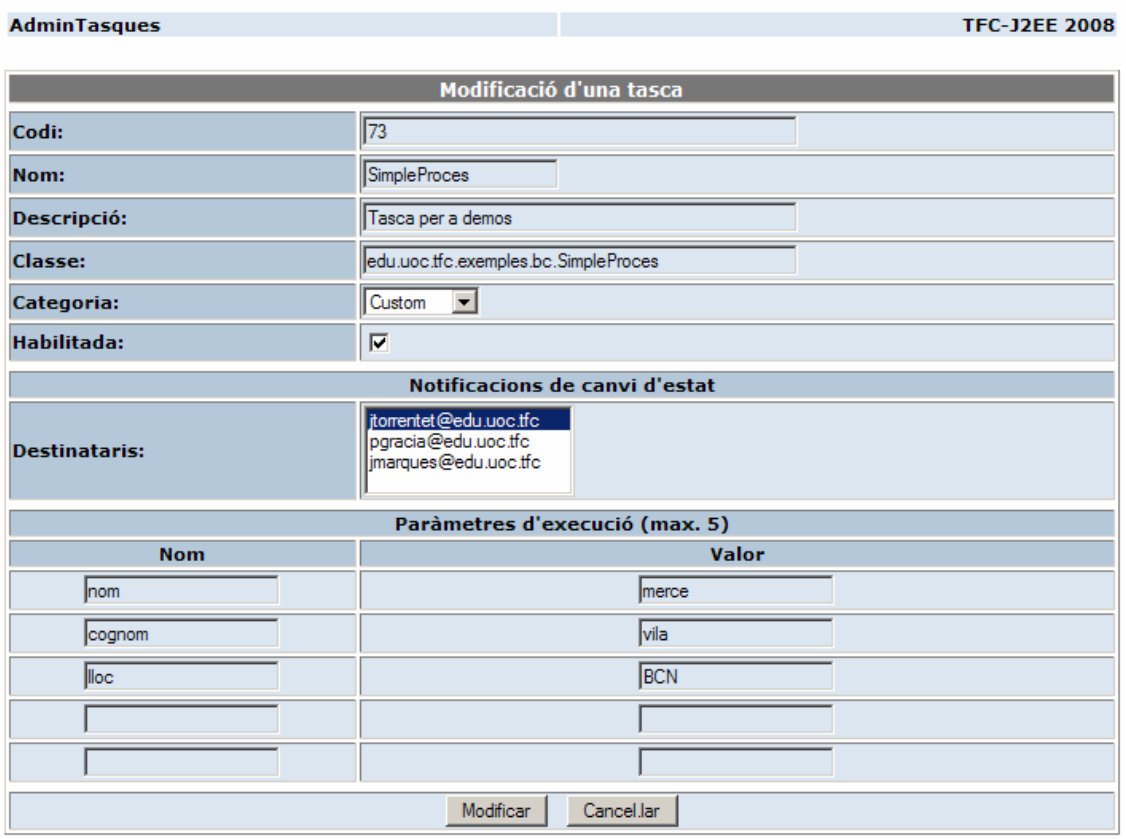


Figura 37: Finestra de modificació d'una tasca

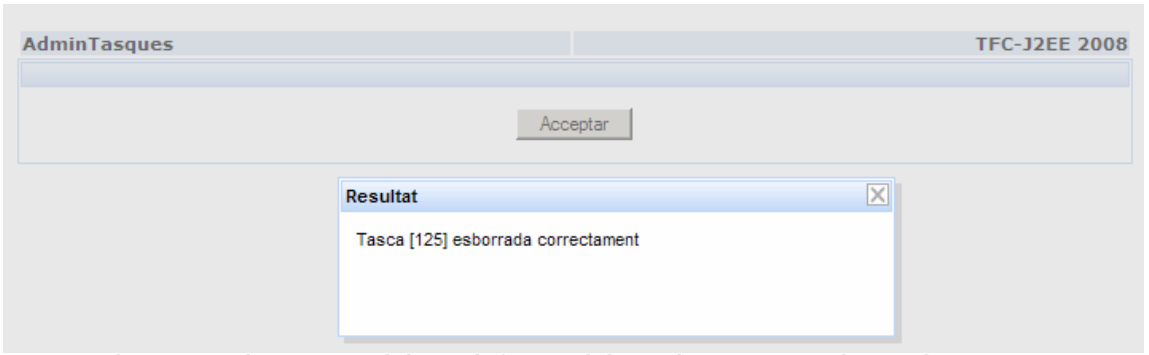


Figura 38: Finestra modal que informa del resultat correcte d'un esborrament

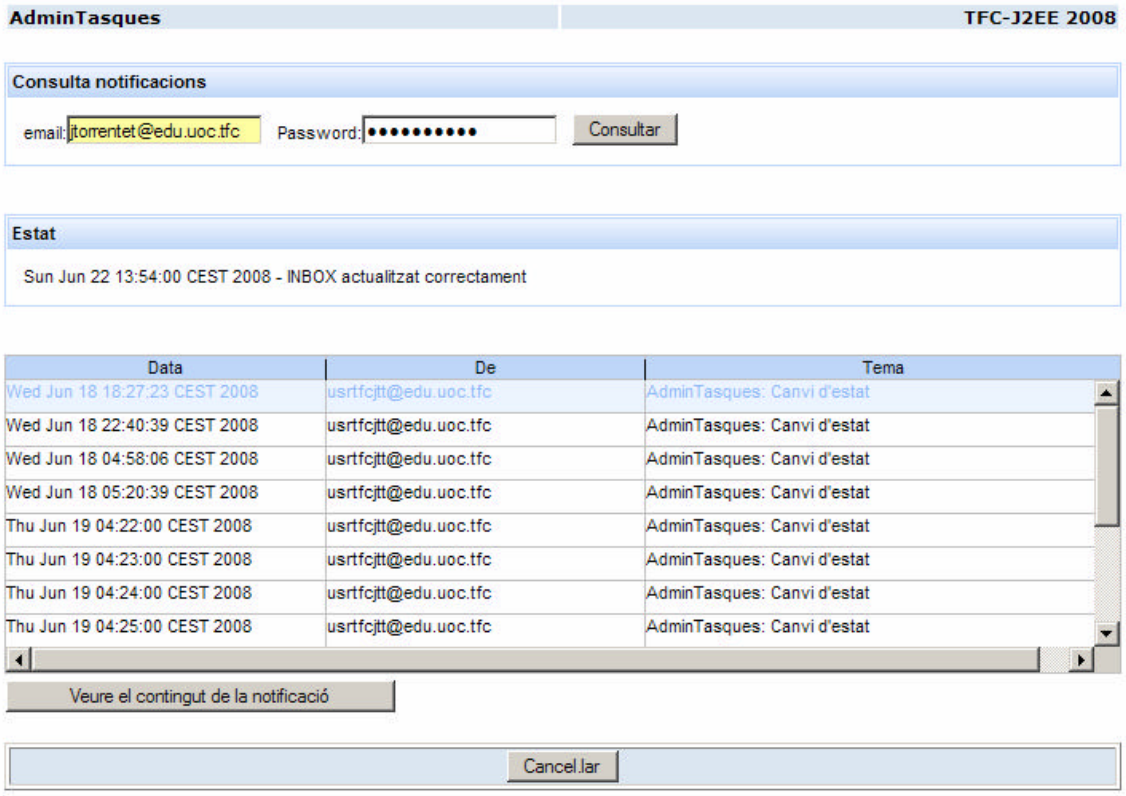


Figura 39: Finestra de consulta de les notificacions de canvi d'estat

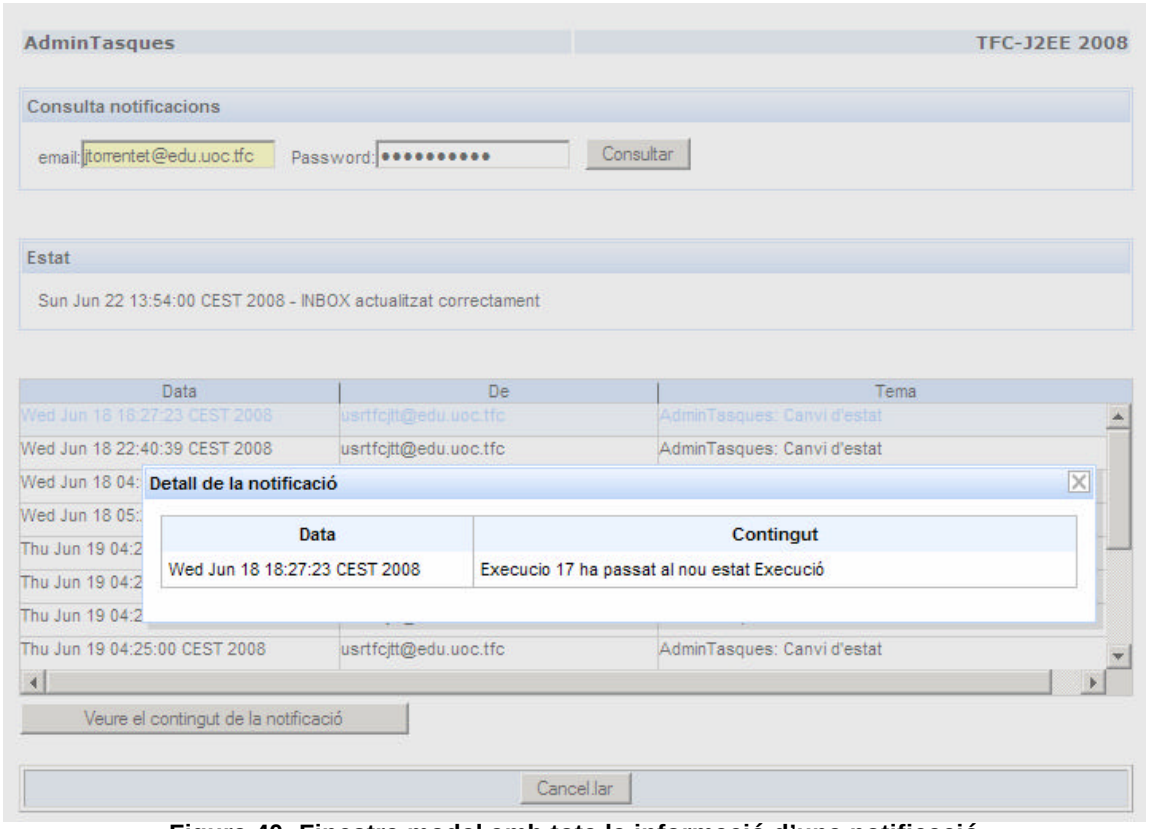


Figura 40: Finestra modal amb tota la informació d'una notificació

5 Implementació

5.1 Introducció

S'han implementat tots els casos d'ús especificats durant la fase d'anàlisi a excepció de l'anomenat "Identificar-se", és a dir, en la versió d'aplicació presentada com resultat del treball, la capa de seguretat s'ha eliminat.

Tot el codi es troba codificat en JAVA mitjançant el JDK (Java SE Development Kit) versió 1.5.

5.2 Aspectes a destacar de la solució

5.2.1 Invocació als components de negoci

Els components de negoci no s'instancien directament en la capa web (client) ja que mitjançant la llibreria *cglib* es genera un objecte que actua com a *proxy* del component de negoci (aquest *proxy* ofereix els mètodes declarats en la interfície de negoci que implementa el component). El *proxy* intercepta la crida, recull els paràmetres rebuts i tota aquella informació necessària per tal de poder invocar per mitjà de Reflexió al mètode corresponent de la classe adequada. Un cop disposa de totes les dades delega en la capa EJB l'esmentada execució per Reflexió.

Representant gràficament la invocació:

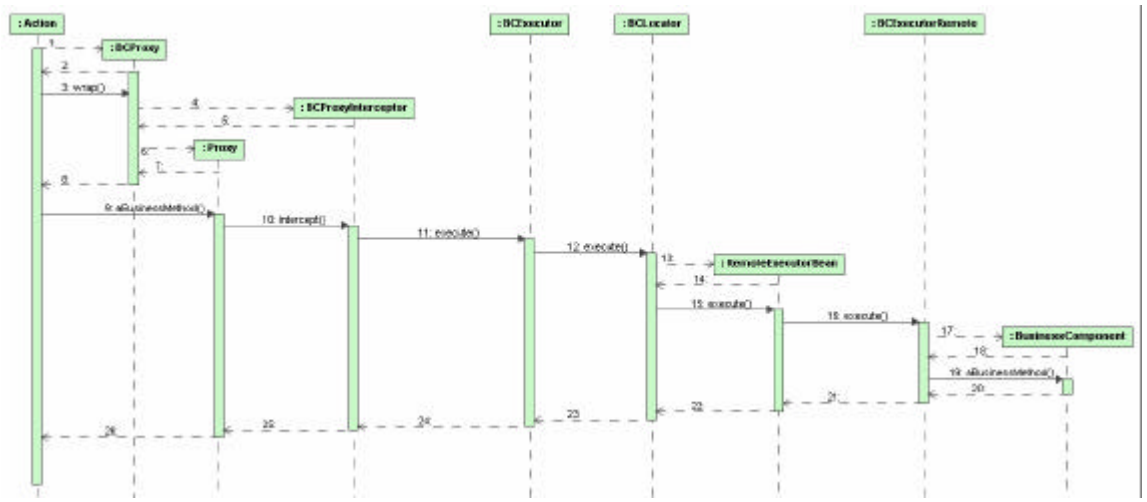


Figura 41: Flux d'execució d'un mètode de negoci

Dins la capa EJB, s'han definit dos EJBs diferents (RemoteExecutor i CMTRemoteExecutor) els quals només es diferencien en la gestió que fan de les transaccions: el primer s'ha definit com BMT (Bean Managed Transaction) i el segon com a CMT (Container Managed Transaction). La raó d'aquesta doble definició està en la necessitat de permetre la propagació d'una transacció en successives crides a components de negoci remots.

Si un component de negoci obre una transacció i dins d'aquesta s'invoca a un segon component, l'única manera que hi ha per a que la transacció es propagui és utilitzar un EJB/CMT per a la crida a l'altre component, de no fer-ho (o sigui recorrent a un EJB/BMT) la transacció no arribaria mai al segon component.

5.2.2 Execucions de tasques

Encara que per la planificació de les execucions s'ha integrat un component tecnològic extern com és el cas de *quartz*, s'ha intentat desacoblar-lo al màxim de la resta de la solució. En aquest sentit s'han pres les següents decisions:

- Encapsular l'accés al component en un únic DAO.
- Duplicar la informació. El **moment temporal** en el que llençar l'execució i la seva **periodicitat** associada, es troben duplicats dins del model de dades, ja que a més a més d'incorporar-ho en el model de l'aplicació també cal afegir-lo al model de dades de *quartz*.
- Desacoblar l'execució de la tasca, de l'execució llençada pel planificador. Un cop *quartz* detecta que ha arribat el moment d'arrancar una execució, l'única acció que fa és enviar un missatge JMS a una cua concreta. Un cop acabat l'enviament, *quartz* considera que l'execució també ha acabat, encara que la tasca continuï en marxa sota el control de l'aplicació. L'únic paràmetre que rep *quartz* en el moment de la programació, i que després propaga com atribut del missatge JMS és el identificador intern de la planificació (tasca) a executar.

5.2.3 API per la codificació de tasques o processos

Per tal que una classe JAVA pugui ser gestionada com a tasca o procés dins la solució proposada cal que extengui de "edu.uoc.tfc.execucio.bc.GenericMonitoredProcess" i que implementi la interfície "edu.uoc.tfc.comu.bci.IMonitoredProcess".

El fet d'extendre de "GenericMonitoredProcess" permet disposar dels mètodes d'utilitat següents:

boolean isAborted()	Retorna cert si la tasca es troba en estat "Abortada". Fals en cas contrari.
boolean isPaused()	Retorna cert si la tasca es troba en estat "Pausada". Fals en cas contrari.
int getState()	Retorna l'estat de la tasca
void waitSeconds(long sec)	Provoca la suspensió de l'execució del thread on s'està executant la tasca durant els milisegons rebuts com a paràmetre. No altera l'estat de la tasca que es troba emmagatzemat dins el model de dades.
void sendToLog(String loglines)	Escriu el missatge rebut en el log de la tasca (emmagatzemat en el model de dades)
Set<Parametre> getParametres()	Retorna els paràmetres d'execució introduïts en el moment de la definició de la tasca

L'actualització de l'estat de la tasca dins el model de dades es realitza mitjançant uns altres mètodes de "GenericMonitoredProcess", susceptibles de ser sobreescrits en el codi de cada tasca:

void onStart()	Mètode que es crida en el moment que arranca una tasca. Per defecte només canvia l'estat a "Execució"
void onEnd()	Mètode que es crida en el moment de finalització d'una tasca. Per defecte només canvia l'estat a "Acabada"
void onError(Throwable th)	Mètode que es crida en el moment que una tasca propaga una excepció. Per defecte canvia l'estat a "Error" i intenta escriure l'excepció en el log de la tasca

Tal com obliga el fet d'implementar "IMonitoredProcess", les accions d'una tasca són invocades mitjançant una crida al mètode: "public void start() throws MonitoredProcessException"

5.2.4 Sessions Hibernate

La responsabilitat d'obrir i tancar les sessions *Hibernate* recau en un únic punt: els components de negoci. Tots aquests que treballin amb *Hibernate* segueixen el model:

```
ThreadDbEnv.openSession();
try{
      [accés a factories, a DAOs o a altres components de negoci]
}
finally{
      ThreadDbEnv.closeSession();
}
```

Encara que l'obertura de les sessions es troba centralitzada en la classe "edu.uoc.tfc.comu.factory.DbSessionFactory", els components de negoci i les factories accedeixen a les sessions a través d'una variable de tipus "ThreadLocal" seguint el patró *per-thread Singleton*: cada *thread* disposa d'una "net.sf.hibernate.Session" independent de la dels altres *threads*, però accessible mitjançant "ThreadDbEnv" com si d'una variable compartida es tractés.

Donat que el component de negoci és l'únic punt on s'obren i tanquen les sessions, les factories segueixen el model:

```
Session session = ThreadDbEnv.getSession();
try {
      [accés a dades]
      ThreadDbEnv.flushSession();
}
catch (HibernateException e) {
      throw new BusinessException("error.xxx.yyy",e);
}
```

5.2.5 Transaccionalitat

En el cas d'haver d'usar transaccions sempre seran de tipus JTA, inclús treballant amb *Hibernate* no s'utilitzaran els mecanismes de transaccionalitat oferts per la pròpia sessió *d'Hibernate*. Per tal de facilitar la seva gestió s'ha encapsulat en una classe l'accés a l'objecte "java:comp/UserTransaction".

La responsabilitat de gestionar-les únicament estarà ubicada a nivell de component de negoci, de fet tots els mètodes de negoci que treballin amb transaccions han de seguir el model:

```
boolean rollback = true;
boolean isLocalTrx = openTrx();

ThreadDbEnv.openSession();
try{
      [accés a factories, a DAOs o a altres components de negoci]
      rollback = false;
}
finally{
      ThreadDbEnv.closeSession();

      if (rollback) rollback(isLocalTrx);
      else commit(isLocalTrx);
}
```

Els mètodes "openTrx()", "commit(isLocalTrx)" i "rollback(isLocalTrx)" es troben implementats en la classe base de la que extenen tots els components de negoci, que en el nostre cas es "edu.uoc.tfc.comu.bc.GenericBC".

Un cop oberta, la transacció és propaga a totes els crides que es fan a factories o a altres components de negoci, és a dir, la transacció és única dins un mateix flux d'execució (no existeix anidament).

5.2.6 Mappings Hibernate

Amb l'objectiu de desacoblar totalment la capa de presentació de la capa de negoci, dins *Hibernate*, per a l'obtenció dels objectes del domini, no s'ha utilitzat el mecanisme de *lazy load*. Per tant, quan un mètode negoci retorna un objecte del domini, aquest inclou la seva informació i la de les entitats amb les que està relacionat.

En aquest sentit cal tenir present que no s'han modelat totes les relacions inverses possibles, és a dir, només s'han mapejat aquelles que aporten sentit funcional. Per exemple entre Tasca i Destinatari existeix una relació 1:N, doncs bé la relació "Tasca ? Destinatari" sí s'ha mapejat, però no la "Destinatari ? Tasca".

En el cas de les relacions M:N s'ha delegat en *Hibernate* la gestió de la taula que materialitza la relació.

Finalment és important esmentar el fet que les restriccions de "NOT NULL" d'algunes columnes també s'han delegat només a *Hibernate*, ja que d'aquesta forma és capaç de gestionar correctament l'esborrament dels elements relacionats quan s'esborra la relació entre ells. Com a exemple tenim la columna "tasca" de l'entitat Paràmetre, que a nivell de model no té la restricció sí en el fitxer de mapeig:

```
<many-to-one     name="tasca"     class="edu.uoc.tfc.comu.model.Tasca"
fetch="select">
    <column name="tasca" not-null="true">
        <comment>Identificador intern de la tasca</comment>
    </column>
</many-to-one>;
```

5.2.7 Cerca d'entitats de negoci

La cerca lliure d'entitats de negoci s'ha implementat mitjançant el mecanisme *Query by example*: la cerca es realitza a partir dels atributs no nuls d'un objecte del mateix tipus del que es desitja cercar. Per exemple, per cercar una tasca pel seu nom, caldria crear un objecte de la classe Tasca, assignar-li com a nom el nom que desitgem cercar i a continuació dir-li a *Hibernate*:

```
Example exempleTasca = Example.create(tascacamps)
      .ignoreCase()
      .excludeZeroes()
      .excludeProperty("descripcio")
      .excludeProperty("habilitada")
      .enableLike(MatchMode.ANYWHERE);

Criteria criteri =
ThreadDbEnv.getSession().createCriteria(Tasca.class).add(exempleTasca);

criteri.addOrder(Order.asc("codi"));

return criteri.list();
```

5.2.8 Context d'execució

Tot el procés d'una petició, ja sigui en la capa web o en la lògica de negoci, té disponible mitjançant una variable de tipus "ThreadLocal" un objecte on es pot emmagatzemar de forma temporal qualsevol objecte serialitzable. Aquest contenidor viatja entre capes (web, ejb, negoci) i és una forma senzilla de poder compartir objectes que no cal que siguin passats com paràmetres dels mètodes invocats.

5.2.9 Log

El sistema de traces de l'aplicació s'ha desenvolupat a partir de la llibreria log4j i utilitzant la configuració del servidor d'aplicacions.

5.2.10 Internacionalització

Dins el codi JAVA s'ha minimitzat la presència de cadenes de caràcters, de forma que tots els missatges es troben emmagatzemats en un fitxer de propietats dependent del "Locale" del servidor d'aplicacions: TextMessages_es.properties. Aquest fitxer també és el que utilitza Struts per emmagatzemar els literals dels missatges.

Les excepcions definides en l'aplicació inclouen un codi d'error que es tradueix a missatge en el moment de presentar l'error a l'usuari. Aquesta traducció també es realitza d'acord al contingut del fitxer TextMessages_es.properties.

5.2.11 Struts

Les dades introduïdes en els formularis de creació de tasques i planificacions es validen mitjançant Struts Validator.

Les classes de model utilitzades per la capa web (*Struts*) estan basades en ActionForms, en canvi la lògica de negoci utilitza com a model els POJOs resultants de l'accés a les dades mitjançant *Hibernate*. Per evitar haver de convertir unes classes en les altres, s'ha optat per incloure dins dels ActionForms atributs corresponents als POJOs de Hibernate, publicant com a mètodes de l'ActionForm els mètodes del POJO, és a dir, els ActionForms actuen com un recobriment web dels POJOs.

```
public class TascaValidatorForm extends ValidatorForm {

      private Tasca tasca = new Tasca();
      private int tipus = -1;
      private int[] destinataris = null;
      private Parametre[] parametres;

      public Tasca getTasca() {
            return tasca;
      }

      public void setTasca(Tasca tasca) {
            this.tasca = tasca;
      }

      public Integer getCodi() {
            return this.tasca.getCodi();
      }

      public void setCodi(Integer codi) {
            this.tasca.setCodi(codi);
      }
```

A més un cop s'ha fet el submit, l'Action (o DispatchAction) corresponent pot recuperar directament la instancia de POJO:

```
public ActionForward insereix(ActionMapping mapping, ActionForm form,
HttpServletRequest request, HttpServletResponse response) {

TascaValidatorForm theForm = (TascaValidatorForm)form;
Tasca tasca = theForm.getTasca();
[...]
```

5.2.12 JSF

Dins l'aplicació s'utilitzen components de *RichFaces* per a mostrar de forma pagina els resultats de les cerques. Aquesta paginació es realitza a nivell de component gràfic, és a dir, en la capa de lògica de negoci s'obtenen totes les dades i posteriorment en la capa web (servidora) qui realitza la paginació.

L'aplicació disposa dels dos *frameworks* configurats (*JSF* i *Struts*), però degut a un *bug* en el component de *Struts-Faces*, no s'han pogut integrar totalment: la idea inicial de tenir codi *JSF* en la vista (*JSP*) però utilitzant el controlador de *Struts* (ActionServlet, Actions, DispatchActions) no ha pogut portar-se a la pràctica, en el seu lloc s'ha optat per integrar-los mitjançant URLs: des d'un Action de *Struts* es fan *forwards* a pàgines *JSF*, afegint prèviament a la HTTPRequest o HTTPSEssion tots els *beans* que la pàgina *JSF* necessita.

Un exemple d'aquesta integració el tenim en els Actions de *Struts* que processen les cerques lliures, ja que un cop han obtingut les dades fan el *forward* a la pàgina *JSF* amb el component de paginació:

```
[...]

TascaForm theForm = (TascaForm)form;
ITascaGestor bc = (ITascaGestor)BCProxy.wrap(ITascaGestor.IMPLEMENTOR,
ITascaGestor.class);
Tipustasca tipustasca = new Tipustasca();
tipustasca.setCodi(theForm.getTipus());
Tasca tascacamps = theForm.getTasca();
tascacamps.setTipustasca(tipustasca);

List<Tasca> llistaTasques = bc.cercaTasques(tascacamps);
request.setAttribute("llistatasques", llistaTasques);

ScrollerBean jsfBean = new ScrollerBean();
jsfBean.setLlista(llistaTasques);
jsfBean.setUrl("gestionaTasques.do");
jsfBean.setUrlParam("do=detall&coditasca");

request.getSession().setAttribute("scrollerBean", jsfBean);

[...]
```

I en el fitxer struts-config.xml hi ha la definició:

```
<action path="/gestionaTasques"
    type="edu.uoc.tfc.tasca.controller.DispatchTasquesAction"
    name="tascaForm"
    scope="request"
    parameter="do">

  [...]
  <forward name="cerca"       path="/jsp/comu/cerca-ok.jsf" />
  [...]

</action>
```

6 Conclusions

Abans de tot cal recordar els objectius que ens havíem marcat a l'inici del TFC:

- Aplicar de forma pràctica els coneixements de programació orientada a objectes per tal de resoldre un problema real, i de forma anàloga en l'ús de la notació *UML* per especificar i dissenyar la proposta d'una solució.

- Aprofundir en el coneixement de la tecnologia *J2EE* i en l'ús alguns dels *frameworks* i tecnologies*: Struts, EJB, JSP, JMS*

- Aprendre l'ús de *frameworks* i tecnologies: *Hibernate, Quartz, JSF, RichFaces*

- Estudiar la possible integració de dos dels *frameworks* de presentació més utilitzats en l'actualitat: *Struts i JSF*.

- Aplicar de forma pràctica molts dels patrons de disseny més *populars*.

Un cop acabada la feina, estem en condicions de dir que tots ells han estat aconseguits i per tant en aquest sentit podem considerar el TFC com un èxit.

Ara bé, més enllà d'aquests objectius m'agradaria destacar dues lliçons més apreses durant l'elaboració del TFC:

- La dificultat de realitzar una planificació realista de la feina, tant pel que fa a la identificació de tasques com a l'esforç a dedicar a cadascuna d'elles. Segurament aquest és un aspecte que està en funció de l'experiència de la persona que fa la planificació en temes semblants i que per tant, en el meu cas, és quelcom a treballar.

- Encara que la comunitat JAVA està fent un esforç molt gran en especificar uns estàndards, la gran diversitat de versions de les pròpies especificacions, així com la gran quantitat de diferents implementacions amb incompatibilitats entre elles, provoca que la integració en una mateixa aplicació de diferents components resulti impossible, a pesar que "tots ells compleixen l'estàndard".

7 Glossari

A continuació es presenta el glossari del model de negoci:

- **API**: de l'anglès Application Programming Interface, és un conjunt de classes i mètodes que ofereix una biblioteca de software per a ser utilitzat per un altre component de software com una capa d'abstracció.

- **Avortar procés** acció per la que un operador por aturar definitivament l'execució d'una tasca.

- **Consultor de negoci**: tipus d'usuari del sistema, encarregat de definir les tasques i els seus paràmetres d'execució.

- **Estat**: informació que reflecteix la situació d'una tasca dins del sistema. Inicialment només es consideraran els següents estats: planificada, en execució, finalitzada correctament, finalitzada amb error, avortada, pausada.

- **Execució**: procés mitjançant el qual una tasca realitza unes accions d'acord a uns paràmetres i en un instant de temps determinat per una planificació temporal prèvia.

- **Notificació de canvi d'estat**: en produir-se un canvi d'estat en l'execució d'una tasca, el sistema enviarà automàticament un missatge electrònic a l'adreça corresponent, informant de l'esdeveniment.

- **Traça de log**: conjunt d'informació generada per les accions que s'executen.

- **Operador**: tipus d'usuari del sistema encarregat de la planificació i gestió de les execucions de les tasques.

- **Paràmetre d'execució**: per tal d'afavorir la reutilització de components, la codificació de les accions ha de permetre canviar el seu comportament en base a uns paràmetres definits pel consultor de negoci en el moment de definició d'una tasca.

- **Pausar procés** acció per la que un operador por aturar temporalment l'execució d'una tasca. Posteriorment aquesta tasca es podrà tornar a posar en marxa.

- **Planificació**: procés mitjançant el qual un operador defineix les dates en que una tasca s'ha d'executar.

- **Procés de llarga durada**: veure Tasca de llarga durada.

- **Tasca de llarga durada**: conjunt d'accions que per la seva complexitat o bé no poden executar-se en certs moments del dia, o bé la seva durada és tan llarga que fa inviable la obtenció síncrona dels seus resultats.

8 Bibliografia

CAMPDERRICH, BENET. *Enginyeria del programari; Anàlisi orientada a objectes* . UOC. (material de l'assignatura).

FATOS XHAFA. *Tècniques de desenvolupament de programari*. UOC (material de l'assignatura).

BRIAN GOETZ. *Threading lightly, Part 3: Sometimes it's best not to share*. IBM, 2001 http://www-128.ibm.com/developerworks/java/library/j-threads3.html

HIBERNATE Reference Documentation. *HIBERNATE - Relational Persistence for Idiomatic Java*. http://www.hibernate.org/hib_docs/reference/en/html/index.html

JEAN-MICHEL GARNIER. Struts 1.1 Controller UML diagrams. The Apache Software Foundation, 2002. http://rollerjm.free.fr/pro/Struts11.html