Gestió del parmetratge

Jordi Fernández Moledo

ETIS

Xavier Escudero Sabadell

14 de Gener de 2009

Resum

Aquest treball final de carrera implementa una aplicació web basada en plataforma Java. La implementació d'aquesta aplicació s'ha realitzat incorporant les millors pràctiques a nivell d'arquitectura d'aplicacions pel que fa a l'ús de patrons de disseny ben coneguts. Així mateix, s'han seleccionat un conjunt de tecnologies orientades a aconseguir una implementació amb responsabilitats ben separades i que, al mateix temps, aïllen el desenvolupador de moltes tasques complexes. Aquestes tecnologies (Spring, Hibernate, JPA, Struts 2) incorporen les últimes tendències en inversió de control i programació orientada a l'aspecte, així com persistència transparent. Pel que fa a la gestió del projecte, el desenvolupament s'ha dut a terme amb l'eina de gestió de dependències Maven, tot verificant la seva conveniència per a la gestió de dependències, les quals comencen a ser considerables en projectes mitjans.

L'objectiu funcional de l'aplicació és la gestió del parametratge d'una organització, és a dir, de totes aquelles dades de consum horitzontal per part de qualsevol altra aplicació més especialitzada. Aquesta implementació es realitza posant èmfasi en aspectes molt importants a nivell empresarial: transaccions i rendiment.

Paraules clau

Spring, Hibernate, JPA, Struts2, Maven2, JEE, Java, JQuery.

Nom de l'àrea del TFC

Àrea desenvolupament Java/JEE

Continguts

Index d'Il·lustracions

Introducció

Justificació del TFC

En el contexte de les aplicacions web "empresarials" (en particular en entorns financers) és molt comú que existeixi un conjunt d'informació de parametratge, és a dir, dades de configuració que resideixen en una BBDD relacional i que canvien molt rarament. Exemples d'aquests conjunts de dades poden ser les relatives a poblacions, províncies, codis postals, tipus de productes de l'organització, característiques dels productes, etc, etc. En general hi ha una gran quantitat d'informació de parametratge, la qual és consumida per qualsevol aplicatiu de l'organització.

De la mateixa manera, en aquests entorns és important mantenir el missatges d'internacionalització de manera centralitzada per tal de permetre els agents del negoci modificar-los segons diversos requeriments (a diferència de mantenir-los en arxius de texte -resource bundles- i requerir de nous desplegaments cada cop que es vol canviar algun literal).

Una altre aspecte important de parametratge en entorns financers és el manteniment d'un calendari centralitzat de manera que totes les aplicacions es coordinin d'acord amb aquest calendari.

Atenent aquests requeriments, la intenció d'aquest projecte es desenvolupar una aplicació que gestioni de manera centralitzada aquesta informació de parametratge via web i que, al seu torn, ofereixi la seva lògica de negoci com a llibreria per tal que la resta d'aplicacions de l'organització consumeixin els seus serveis, de tal manera que la lògica de negoci romangui centralitzada.

Objectius

Els objectius que es pretenen assolir són els següents:

1. Obtenir una implementació de referència estable per a la gestió de la parametrització en entorns empresarials.
2. Verificar els beneficis de la inversió de control (IoC) i la programació orientada a l'aspecte (AOP), i la seva aplicació a característiques no funcionals com ara les transaccions.
3. Comprovar l'increment de productivitat quan s'utilitzen productes ORM (Java Persistence API i Hibernate).
4. Avaluar la conveniència de Maven 2 com a eina de gestió de projectes.

Enfocament i mètode seguit

En el desenvolupament d'aquest projecte s'ha seguit una metodologia interativa i incremental, de tal manera que l'addició de característiques era verificada contínuament a mida que avançava el desenvolupament. En primer lloc, es van prioritzar els casos d'ús atenent a criteris de dificultat d'implementació, això és, de més senzills a més complicats per tal d'establi una bona base en la utilització d'eines i tecnologies, i anar incorporant-les progressivament.

Planificació del projecte

La planificació del projecte està basada en el desenvolupament de manera incremental i iteratiu permet realitzar el sistema progressivament, implementant en primer lloc els casos d'ús més significatius. Aquesta estratègia ajuda a reduïr els riscos en les primeres fases del projecte i possibilita la incorporació de canvis i correccions en successives fases, refinant així progressivament la realització del sistema. El desenvolupament de l'aplicació tres iteracions. La següent taula mostra el calendari de cadascuna d'aquestes iteracions:

I	Servei	Data	Entregables
1	Generación de l'arquitectura i disseny de la solució.	5 Novembre de 2008	Anàlisi i casos d'ús (PAC2).
2	Desenvolupament de casos d'ús	19 Novembre de 2008	Casos ús taula parametratge, estructura i dades
3	Desenvolupament de casos d'us	17 de Desembre de 2008	Casos ús internacionalització i calendari

Productes obtinguts

Els productes obtinguts en aquest projecte han sigut els següents:

- Executable de l'aplicació en format WAR
- Codi font de l'aplicació
- Documentació de l'aplicació en format JavaDoc
- Aquesta memòria de projecte
- Documents de casos d'ús
- Realització dels casos d'ús a nivell d'anàlisi
- Realització d'un cas d'ús de referència a nivell de disseny

Breu descripció de la memòria

A fi i efecte de familiaritzar el lector amb la funcionalitat que s'espera del projecte que s'ha desenvolupat, la memòria comença amb una descripció dels casos d'ús. Seguidament es descriu l'arquitectura que s'ha proposat en la implementació, seguint patrons de disseny ben coneguts en l'entorn JEE. Un cop comunicada l'arquitectura, es descriuen les tecnologies escollides per a la seva implementació i el paper que cadascuna d'elles té en el conjunt de l'aplicació. Finalment, es descriu a mode de referència el disseny en detall d'un cas d'ús, i seguidament les conclusions.

Identificació de casos d'ús

Introducció

Aquest capítol exposa els diferents casos d'ús que formen part de l'aplicació. L'objectiu és familiaritzar el lector amb la funcionalitat que s'espera per part de l'aplicatiu.

Gestió de Taula General

Aquest conjunt de casos d'ús permeten gestionar el cicle de vida d'una taula de parametratge. Una taula de parametratge queda completament definida quan s'especifiquen les dades generals de la taula (cosa que fan aquests casos d'ús) i quan s'especifica la seva estructura. Les dades generals d'una taula de parametratge són el seu codi de taula, la seva descripció, si la taula és o no multientitat –és a dir, si les seves dades poden ser diferents depenent del departament o empresa del grup que les demani –. Una taula de parametratge s'ha d'activar abans de poder introduir dades.

Crear taula general

Flux Principal

1. L'usuari sol•licita al sistema crear un nova taula general.
2. El sistema oferix un formulari per a la creació de la taula general.
3. L'Usuari informa el nombre de taula que es va a crear.
4. El Sistema verifica que aquest nombre de taula estigui disponible en BBDD.
5. El Sistema desbloqueja els camps de la pantalla que han de ser informats per a la creació de les dades.
6. L'Usuari informa les dades de la pantalla (Descripció de la taula, Indicador de Multientitat (S/N), Entitat Associada (Només si l'indicador de Multientitat és igual a N)).
7. El Sistema valida la coherència entre l'indicador de Multientitat i Entitat associada, i inicialitza l'indicador d'estat a Inactiu (I).

Flux alternatiu: codi de taula erroni

1. Continúa a partir del punt 4 del flux principal
2. El sistema detecta que el codi de taula no és de 4 caràcters numèrics i informa a l'usuari.

Flux alternatiu: la taula ja existeix

1. Continúa a partir del punt 3 del flux principal
2. El sistema verifica que el nombre de taula informat no està disponible en BBDD (ja existeix una taula general amb aquest nombre) i informa a l'usuari.

Consultar taula general

Flux principal

1. L'Usuari sol•licita consultar una taula en el menú de taules corporatives.
2. L'Usuari informa un nombre de taula o un marge de consulta.
3. El Sistema mostra els detalls de la taula o llistat de taules amb el nombre de taula i descripció.
4. L'Usuari sol•licita veure el detall d'una de les taules.
5. El Sistema mostra la informació de codi de la taula, descripció, estat, indicador multientitat, entitat associada a la taula i les seves dades d'auditoria (alta, baixa i modificació de la taula)

Flux alternatiu: codi de taula erroni

1. Contínua a partir del punt 2 del flux principal
2. El sistema detecta que el codi de taula no és de 4 caràcters numèrics ininforma a l'usuari.

Flux alternatiu: recerca sense resultat

1. Continua a partir del punt 2 del flux principal
2. El sistema informa a l'usuari que no hi ha taules definides per al nombre o marge de consulta indicat.

Modificar taula general

Flux principal

1. (Inclou el cas d'ús Consultar Taula General)
2. L'Usuari selecciona la taula que desitja modificar.
3. L'Usuari modifica la descripció de la taula. No pot modificar-se cap altra dada.
4. El Sistema verifica que la versió del registre de la taula sigui el mateix que el del registre que s'ha consultat anteriorment i modifica la descripció i les dades d'auditoria de modificació de la taula.

Flux alternatiu: taula activa

1. Contínua a partir del punt 2 del flux principal
2. El sistema informa a l'usuari que la taula es troba activa i no pot ser modificada.

Flux alternatiu: error de versió

1. Contínua a partir del punt 3 del flux principal
2. El sistema comprova que la versió de les dades modificades és inferior a la versió en la BBDD i mostra un error a l'usuari.

Activar taula general

1. (Inclou el cas d'ús Consultar Taula General)
2. L'usuari sol•licita activar una taula general.
3. El Sistema valida que la taula té almenys definida una estructura mínima en el sistema (una clau i un detall).
4. El Sistema activa la taula. Des d'aquest moment la taula està disponible per al sistema i es poden agregar dades.

Flux alternatiu: taula sense estructura definida

1. Continúa a partir del punt 3 del flux principal
2. El sistema verifica que la taula no té una estructura mínima definida i informa a l'usuari que no pot ser activada.

Esborrar taula general

Flux principal

1. (Inclou el cas d'ús Consultar Taula General)
2. L'usuari sol•licita esborrar una taula general
3. El sistema valida que la taula general no tingui estructura ni dades associades.
4. El sistema esborra la taula general.

Flux alternatiu: la taula té estructura o dades associades

1. Continúa a partir del punt 3 del flux
2. El sistema valida que la taula general té estructura i/o dades associades.
3. El sistema mostra un missatge d'error a l'usuari.

Requeriments Especials

1. Per a totes les consultes s'ha de recuperar la versió de les dades.
2. En les modificacions s'ha de verificar que la versió del registre a modificar sigui igual al consultat inicialment. No es realitzar cap modificació quan les versions siguin diferents.

Post-Condicions

1. Les taules noves del sistema s'haurien de crear inactives i romandre amb aquest estat fins a la definició total de la seva estructura

Creació de l'Estructura d'una Taula General

Flux Principal

1. L'Usuari sol•licita crear l'estructura d'una taula general en el menú de taules corporatives.

2. L'Usuari informa el nombre de taula l'estructura de la qual va a crear.
3. El sistema verifica que la taula està inactiva i mostra l'estructura definida fins al moment en la BBDD.
4. L'usuari sol•licita incloure un camp en l'estructura.
5. El sistema mostra una pantalla amb els elements que han de ser informats per a la creació d'un camp d'estructura.
6. L'usuari informa els elements del nou camp (Nom del camp, Descripció del camp, Indicador de Camp Clau (S/N), Tipus de Camp (A//N D), Longitud total del camp, Nombre de Decimals (només per a camps numèrics).
7. El Sistema valida que l'indicador de clau estigui informat amb “” S o “N”.
8. El Sistema valida que el tipus de camp informat sigui alfanumèric (A), Numèric (N) o Data (D).
9. Per a camps Alfanumèrics el Sistema valida que no s'informi decimals.
10. Per a camps Numèrics la longitud informada sempre és la total pel que l'un nombre de decimals no podrà ser major que la longitud total.
11. Per a camps Data es permetran només una longitud. El format que es guardarà en la taula és un alfanumèric de 11 amb el següent format DD-MMM-AAAA.
12. El Sistema calcula la posició del camp dintre de l'estructura de la clau i de l'estructura de dades.
13. El Sistema valida els dades i crea el camp nou en la BBDD.
14. El Sistema mostra l'estructura definida fins al moment per a la taula informada.
15. L'Usuari realitza els passos definits des del punt 5 tantes vegades com camps requereixi la taula general que s'està definint.

Flux alternatiu: error de validació

1. Continúa a partir del punt 12 del flux principal
2. El sistema detecta un error i informa a l'usuari.

Consulta de l'estructura d'una taula general

1. L'Usuari informa el codi d'una taula general i sol•licita consultar la seva estructura.
2. El Sistema mostra els camps de l'estructura en un llistat.
3. L'Usuari pot veure el detall de l'estructura en un llistat. Per cada camp de l'estructura pot observar el nom del camp, la descripció, l'indicador de clau, el tipus de camp, la longitud total del camp i el nombre de decimals.

Flux alternatiu: error en codi de taula

1. L'Usuari informa un codi de taula i sol•licita consultar la seva estructura.
2. El Sistema verfica que el codi de la taula no té quatre caràcters numèrics i detecta un error.
3. El Sistema informa un missatge d'error a l'Usuari.

Modificar estructura d'una taula general

1. (Inclou “Consulta de l'estructura d'una taula general”).
2. L'Usuari selecciona un registre existent en l'estructura de la taula general.
3. El Sistema valida que la taula està Inactiva i consulta les dades del registre.
4. L'Usuari modifica les dades del registre.
5. El Sistema verifica la coherència de dades i realitza la validació corresponent a bloqueig optimista.
6. El Sistema realitza la modificació del registre juntament amb les dades d'auditoria.

Flux alternatiu: la taula està activa

1. (Continúa a partir del punt 2 del flux principal)
2. El sistema comprova que la taula corresponent a aquest nombre ja està activa.
3. El Sistema mostra un missatge informant a l'usuari que la taula no es pot modificar ja que està activa.

Flux alternatiu: registre a modificar actualitzat per altre usuari

1. (Continúa a partir del punt 5 del flux principal)
2. El sistema comprova que la versió de les dades modificades és inferior a la versió en la BBDD i mostra un error a l'usuari.

Esborrar estructura d'una taula general

1. (Inclou “Consulta de l'estructura d'una taula general”).
2. L'Usuari selecciona un registre existent en l'estructura de la taula general i sol•licita la seva esborrat.
3. El Sistema verifica que la taula està Inactiva.
4. El Sistema realitza un esborrat físic del camp de l'estructura.
5. El Sistema recalcula la posició dels camps de la taula en l'estructura de clau o dades en el qual estava definit.

Flux alternatiu: la taula està activa

1. (Continúa a partir del punt 2 del flux principal)
2. El sistema comprova que la taula corresponent a aquest nombre ja està activa.
3. El Sistema mostra un missatge informant a l'usuari que la taula no es pot modificar ja que està activa.

Requeriments Especials

1. Només es permet modificar l'estructura de taules que estan Inactives.
2. AL finalitzar la definició de l'estructura es deu activar la taula. En aquest moment es permetrà donar d'alta dades en la taula definida.

3. Per a totes les consultes s'ha de recuperar la versió de les dades.
4. En les modificacions s'ha de verificar que la versió del registre a modificar sigui igual al consultat inicialment. No es realitzar cap modificació quan les versions siguin diferents.

Pre-Condicions

1. La taula general sobre la qual es desitja especificar la seva estructura ha d'estar creada prèviament.

Post-Condicions

1. Una vegada definida l'estructura i activada la taula no es podrà modificar aquesta estructura

Consultar dades de Taula General

Flux Principal

1. L'Usuari sol•licita consultar les dades d'una taula general.
2. El sistema sol•licita el codi de taula general que es desitja consultar.
3. L'usuari introduïx el codi de taula general a consultar.
4. El Sistema mostra les dades de la taula consultada.

Flux alternatiu: codigo de taula erroneo

1. Continua a partir del punt 3 del flux principal
2. El Sistema valida que el codi de la taula contingui quatre caràcters numèrics i detecta un error.
3. El Sistema informa missatge d'error a l'Usuari.

Crear dades d'una taula general

Flux principal

1. (Inclou "Consultar dades d'una taula general")
2. L'Usuari sol•licita al sistema donar d'alta dades.
3. El sistema presenta un formulari per a donar d'alta una nova dada d'acord a l'estructura definida.
4. L'Usuari introduïx les dades de la clau i els atributs relacionats.
5. L'Usuari prem el botó "Crear".
6. El Sistema valida que els dades informades siguin coherents amb l'estructura definida i que no existeixi altre registre amb la mateixa clau.
7. El Sistema guarda en la BBDD el registre nou conservant l'estructura definida de la clau i del camp dades. S'informen les dades d'auditoria referent a la modificació i l'indicador d'estat igual a "A = Activa".

8. El Sistema mostra un nou llistat a l'Usuari amb les dades definides en la taula.
9. L'Usuari realitza els passos definits des del punt 2 tantes vegades com registres es requereixin.

Modificar dades d'una taula general

Flux principal

1. (Inclou "Consultar dades d'una taula general").
2. L'Usuari selecciona un registre i sol•licita modificar-lo.
3. El Sistema mostra un formulari on es mostren les dades i estan habilitats per a modificar-los (Mai s'han d'habilitar les dades associades a la clau).
4. L'Usuari modifica les dades del registre i prem el Botó "Modificar".
5. El Sistema valida la coherència de dades amb els formats definits en l'estructura i valida la versió de les dades.
6. El Sistema guarda les dades informades per l'Usuari en les BBDD i actualitza les dades d'auditoria referent a la modificació.

Flux alternatiu: registre a modificar actualitzat per altre usuari

1. (Continua a partir del punt 5 del flux principal)
2. El sistema comprova que la versió de les dades modificades és inferior a la versió en la BBDD i mostra un error a l'usuari.

Esborrar dades d'una taula general

1. (Inclou "Consultar dades d'una taula general").
2. L'Usuari selecciona un registre existent en les dades de la taula general i sol•licita la seva esborrat.
3. El Sistema verifica que la versió del registre és correcta (bloqueig optimista).
4. El Sistema realitza un esborrat lògic del camp de l'estructura, marcant el registre com registre donat de baixa.

Requeriments Especials

1. L'esborrat de dades és lògic i no físic
2. En tota modificació ha de verificar-se el bloqueig optimista

Pre-Condicions

1. La taula general sobre la qual es desitja agregar registres ha d'estar activada, és a dir, la seva estructura no pot ser alterada.

Crear data en el calendari

Flux Principal

1. L'usuari sol•licita donar d'alta una data en el sistema
2. El Sistema mostra una pantalla amb les dades necessàries per a l'alta d'aquesta informació.
3. L'Usuari informa les dades de la pantalla associats a la data (data, tipus de data informada (Festiu Nacional, Festiu Autonòmic, Festiu Local, Festiu d'Empresa, Festiu Especial) i els codis de país, comunitat autònoma i localitat si fossin necessaris). Eventualment, l'usuari informa una descripció associada la data.
4. El Sistema valida la coherència de les dades informades i grava la nova data.
5. El Sistema mostra un missatge informant que l'operació s'ha realitzat correctament i mostra el calendari anual reflectint la nova data.

Consultar data en el calendari

Flux principal

1. L'Usuari sol•licita consultar el calendari de cert any.
2. El sistema presenta el calendari de l'any sol•licitat i mostra els festius nacionals per a aquest any.
3. Eventualment, l'usuari sol•licita visualitzar altres tipus de data: festiu autonòmic (haurà d'indicar l'autonomia), festiu local (haurà d'indicar autonomia i localitat), festiu d'empresa o festiu especial.
4. El sistema mostra els festius especificats.

Modificar data del calendari

Flux principal

1. (Inclou "Consultar dades d'una taula general").
2. L'Usuari selecciona una data concreta del calendari.
3. El sistema mostra un formulari d'acord al tipus de data seleccionada però solament permet modificar la descripció o esborrar la data.
4. L'usuari modifica la descripció de la data.
5. El Sistema valida la versió de les dades.
6. El Sistema guarda les dades informades per l'Usuari en les BBDD i actualitza les dades d'auditoria referent a la modificació.

Flux alternatiu: més d'un festiu definit per a una data

1. (Continua a partir del punt 2 del flux principal)
2. El sistema informa que hi ha més d'un festiu definit per a aquesta data i sol•licita a l'usuari que seleccioni el festiu que desitja modificar o esborrar.

3. L'usuari selecciona el festiu a modifica
4. (continúa en el punt 4 del flux principal)

Esborrar data del calendari

1. (Inclou "Consultar data en el calendari").
2. L'Usuari selecciona una data concreta del calendari.
3. El sistema mostra un formulari d'acord al tipus de data seleccionada però solament permet modificar la descripció o esborrar la data.
4. L'usuari sol•licita esborrar la data.
5. El Sistema valida la versió de les dades.
6. El Sistema esborra la data de la BBDD.

Flux alternatiu: més d'un festiu definit per a una data

1. (Continúa a partir del punt 2 del flux principal)
2. El sistema informa que hi ha més d'un festiu definit per a aquesta data i sol•licita a l'usuari que seleccioni el festiu que desitja modificar o esborrar.
3. L'usuari selecciona el festiu a modifica
4. (continúa en el punt 4 del flux principal)

Requeriments Especials

1. L'esborrat de les dades és físic
2. Solament es permet esborrar dates a futur
3. En tota modificació ha de verificar-se el bloqueig optimista

Pre-Condicions

1. La taula general sobre la qual es desitja agregar registres ha d'estar activada, és a dir, la seva estructura no pot ser alterada.

Crear missatge

Flux Principal

1. L'usuari sol•licita donar d'alta un nou missatge en el sistema
2. El Sistema mostra una pantalla amb les dades necessàries per a l'alta d'aquesta informació.
3. L'Usuari informa les dades de la pantalla associats al missatge: el codi de l'aplicació, el codi del missatge (caràcters numèrics) i l'idioma del missatge.
4. El Sistema valida la coherència de les dades informades (no existeix el mateix codi de missatge per al mateix codi d'aplicació) i grava el missatge.
5. El Sistema mostra un missatge informant que l'operació s'ha realitzat correctament.

Consultar missatge

Flux principal

1. L'Usuari sol•licita consultar un missatge o un conjunt de missatges.
2. El sistema presenta un formulari en el qual és obligatori informar el codi d'aplicació i l'idioma. Opcionalment s'informarà el codi de missatge concret que es desitja consultar.
3. El sistema mostra el missatge concret sol•licitat (si es va informar codi de missatge) o un llistat amb els missatges per a certa aplicació i idioma.

Modificar missatge

Flux principal

1. (Inclou "Consultar dades d'una taula general").
2. L'Usuari selecciona un missatge per a modificar.
3. El sistema mostra els detalls del missatge i permet modificar només el text del missatge.
4. L'usuari modifica el text del missatge i sol•licita al sistema el seu enregistrament.
5. El Sistema valida la versió de les dades.
6. El sistema verfica que la versió de les dades sigui correcta (bloqueig optimista).
7. El Sistema guarda les dades informades per l'Usuari en les BBDD i actualitza les dades d'auditoria referent a la modificació.

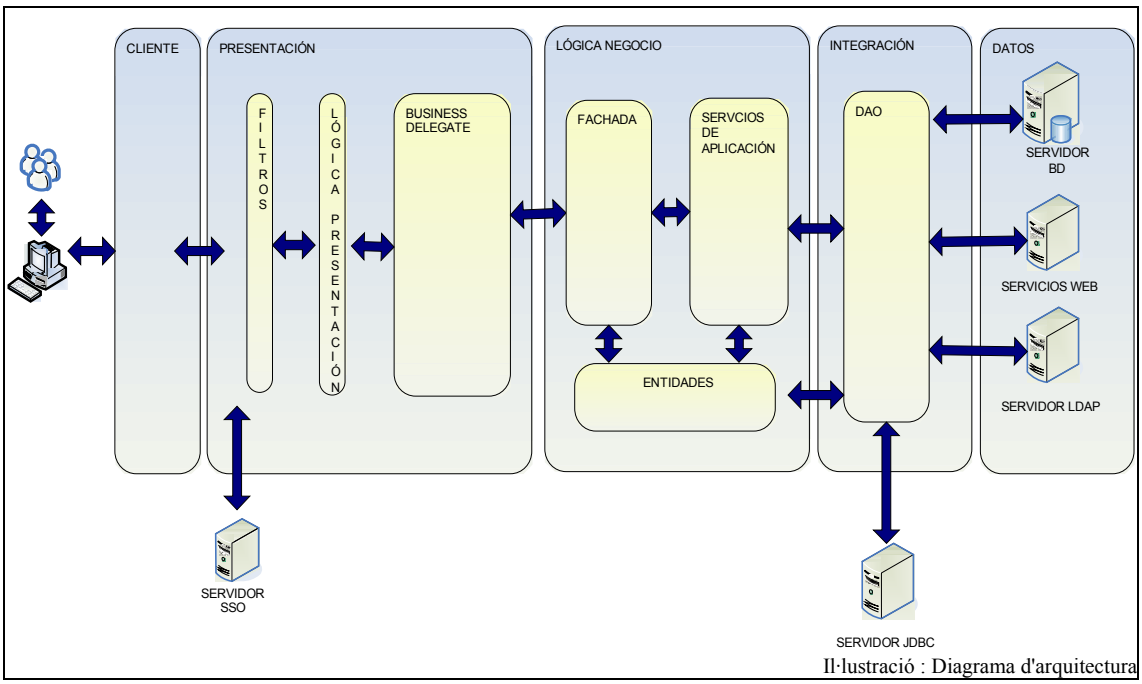
Esborrar missatge

1. (Inclou "Consultar data en el calendari").
2. L'Usuari selecciona un missatge per a esborrar.
3. El sistema demana confirmació abans d'esborrar el missatge.
4. L'usuari confirma l'esborrat del missatge.
5. El Sistema valida la versió de les dades.
6. El Sistema esborra el missatge de la BBDD.

Requeriments Especials

1. L'esborrat de les dades és físic
2. En tota modificació ha de verificar-se el bloqueig optimista.

Arquitectura



Il·lustració : Diagrama d'arquitectura

Aquesta capa està té com objectiu la interfície d'usuari, dedicant-se en exclusiva a la presentació i recollida de dades. Per a això es basa en el Framework MVC Jakarta Struts 2.0 [Struts2]. Exemples d'aquests tipus de components són navegadors web, agendes personals, etc. Els principals components d'aquesta capa seran els següents:

- Pàgines html, aquestes pàgines poden estar codificades directament en l'aplicació si és necessari o ser el resultat de la transformació del codi de Servlets o pàgines
- JSP (sense codi scriptlet) provinents de la capa de presentació (Web layer) de l'aplicació. Les pàgines JSP s'encarreguen de presentar les dades provinents de negoci obtingudes des dels Actions que al seu torn invoquen el servei a través de classes que implementen el patró Business Delegate.
- Fulles d'estil, en aquests fitxers es definiran els estils a utilitzar en l'aplicació.
- Scripts javascript en fitxers externs. Aquests fitxers d'extensió .js implementen comportament dinàmic en l'aplicació web a nivell de capa client. Tindrem almenys dos tipus de fitxers javascript : Fitxers amb codi genèric per a realitzar tasques comunes com obrir finestres, carregar de forma dinàmica, etc.
- Imatges. Les imatges seran utilitzades en el programa per a millorar l'aspecte visual de l'aplicació. Aquestes haurien de tenir format .gif o .jpg i s'intentarà que ocupin el menys possible.

Capa de presentació

La responsabilitats d'aquesta capa són, fonamentalment:

- Transformar la petició que arriba a l'aplicació en objectes comprensibles per la lògica de negoci
- Invocar la lògica de negoci adequada per a la petició rebuda

- Seleccionar la vista que ha de ser presentada a l'usuari final.

Aquesta capa ha d'estar allotjada en un contenidor de Servlets. Aquesta capa està formada pels següents elements d'arquitectura:

Filtres És el patró de disseny intercepting filter. Té com objectiu manipular la petició i la resposta abans i després del processat per part de la lògica de presentació. D'aquesta manera podem encadenar diversos filtres amb la finalitat de portar a terme tasques comunes a totes o un subconjunt de les peticions. Les tasques típiques que es porten a terme en aquest punt són:

1. Verificar l'autenticació del client
2. Verificar la validesa de la sessió del client
3. Realitzar labors de seguretat addicionals (IP des de la qual s'accedeix, horari d'accés, domini DNS d'accés, etc)
4. Filtrar caràcters no desitjats de la petició
5. Determinar/transformar la codificació de caràcters
6. Determinar si el dispositiu client està suportat Etc.

Alguns dels beneficis d'implementar aquesta estratègia són:

1. Centralitzar el control de certes parts del processat de manera desacoblada
2. Promoure la reutilización sobre la base de l'ús i combinació dels mateixos filtres en diverses aplicacions. Típicament això es realitza de forma declarativa. Per exemple, el filtre que correspon a Struts 2.0, el qual s'encarregarà de processar la petició.

Lògica de presentació Aquests components implementen la part Controlador del patró MVC. Aquestes accions o controladors són vinculades amb les pàgines JSP de l'aplicació, el model d'entitats per a recollir les dades dinàmiques a mostrar i la capa de serveis de l'aplicació per a accedir a la lògica de negoci i disparar les accions pertinents. Sobre la base de la informació rebuda en la petició, la lògica de presentació:

1. Valida els dades rebudes
2. Realitza les conversions necessàries per a acomodar les dades rebudes en els objectes de destinació (típicament entitats)
3. Invoca els serveis de negoci adequats
4. Determina la vista que ha de ser presentada com a resultat de la lògica de negoci.

Alguns dels beneficis d'implementar aquesta estratègia són:

1. Permet identificar fàcilment el processat de presentació associat a certa petició
2. S'ocupa d'acomodar dades que no estan sota el paradigma de l'orientació a objectes (dades que arriben en la petició) en objectes del domini sota el paradigma de l'orientació a objectes. En aquest sentit, adapta la informació que arriba sota les condicions del protocol entrant als objectes del domini.

Business Delegate Adaptador entre la capa de presentació i la lògica de negoci. Reduïx l'acoblament entre els components de la capa de presentació i els serveis de negoci que utilitzen. El patró de dissenyo Business Delegate tal com apareix en el catàleg de patrons J2EE està motivat per aplicacions en les quals hi ha separació física entre les capes de presentació i de negoci. Les classes Business Delegate no han de contenir lògica de negoci, simplement ha d'actuar com invocador de l'objecte que implementi el servei de negoci que es va a utilitzar. En aquest context, el patró de disseny Business Delegate realitza labors tals com la localització del servei remot o la invocació de serveis remots. Aquests beneficis no són particularment rellevants en el context que ens ocupa. Segons el catàleg de patrons J2EE, una de les

possibles estratègies d'implementació d'aquest patró és la Delegate Adapter Strategy. Aquesta estratègia és el benefici que es pretén destacar aquí, per aquest motiu es mantingui el nomeni Business Delegate para referir-nos a aquest patró.

En ocasions, és convenient : adaptar la interfície que exposa la lògica de negoci a les necessitats de la capa de presentació, per exemple, transformant objectes retornats per la lògica de negoci en objectes més simples i fàcils de gestionar en la capa de presentació i/o proporcionar paràmetres addicionals a mètodes presents en les façades en favor de la capa de presentació. Alguns dels beneficis d'implementar aquesta estratègia són:

1. Permet identificar un punt únic d'adaptació d'interfícies entre la capa de presentació i la de negoci
2. S'encarrega d'oferir valors per omissió als serveis façade quan sigui necessari.

Capa de Lògica de Negoci

En aquesta capa s'implementa les regles de negoci de l'aplicació i s'exposen de la manera més senzill possible per a ser consumida per la capa o capes de presentació. Les responsabilitats d'aquesta capa són, fonamentalment:

1. Agrupar les regles de negoci de manera que siguin fàcilment reutilizables
2. Exposar les regles de negoci mitjançant interfícies gruixudes
3. Gestionar les transaccions

Per a la separació de responsabilitats en la capa de lògica de negoci, s'han utilitzat els patrons de disseny que es detallen a continuació. Aquesta capa està formada pels següents elements d'arquitectura:

Façana És el patró de dissenyo Facade. Amaga els detalls d'un conjunt de components de negoci complexos mitjançant una senzilla interfície orientada a serveis, simplificant així l'accés als components de la capa de negoci. Exposa els serveis de negoci a la capa de presentació amb un nivell de granularitat adequat. En general, aquests serveis tenen les següents característiques: Posseïxen un nivell de granularitat més "gruixut" que els serveis en els quals es donarà suport. Aporten un valor a nivell de negoci, per exemple, crear un nou client en una organització, el qual es recolza en serveis més granulars com: persistir en BBDD les dades del client, enviar correu postal al client, enviar SMS al telèfon mòbil del client, generar un model de contracte ... Són transaccionals (s'inicia i finalitza la transacció en el servei de façana).

Serveis d'Aplicació És el patró de dissenyo Application Service. Aquesta capa agrupa les regles de negoci amb el nivell de granularitat necessari. Oferixen un lloc centralitzat en el qual situar la lògica de negoci entre les façades de servei i els objectes de negoci, afavorint la re-utilització de lògica de negoci. Requereix ser coordinat per un servei a nivell de façana ja que els serveis d'aplicacions: Implementen una regla de negoci que no pot ser dividida. No tenen valor de negoci en si mateix. NO tenen el control de la transacció.

Entitats En la capa de lògica de negoci s'implementen les classes que representen a les entitats de l'aplicació. Habitualment, les entitats contenen l'estat que es persisteix en la capa de dades.

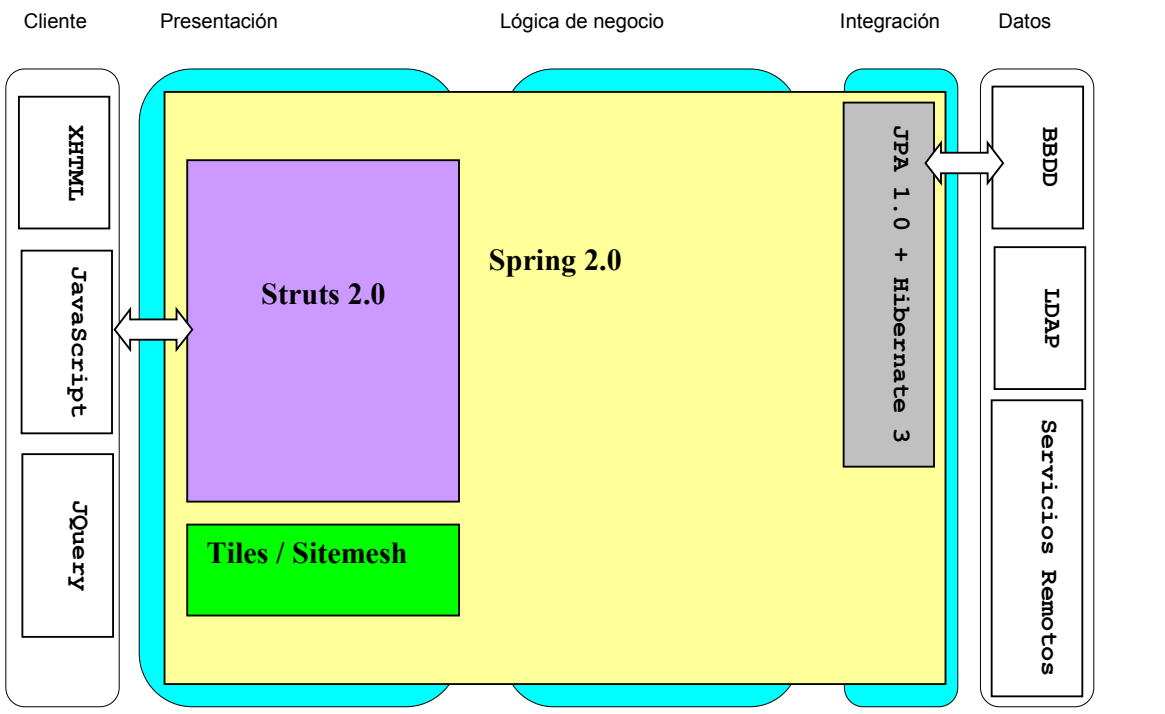
Capa d'accés a dades Aquesta capa s'encarrega d'accedir a les dades del sistema i persistir-los, obtenint la informació independentment del seu origen. Les responsabilitats d'aquesta capa són, fonamentalment:

1. Mostrar una interfície senzilla per a persistir l'estat de l'aplicació independentment de la tecnologia d'emmagatzematge implementada
2. Emmagatzemar de forma persistent l'estat de l'aplicació sol•licitat per la capa de lògica de negoci.

Aquesta capa està formada per l'element d'arquitectura DAO, el qual realitza la següents funcionalitats: Encapsula lògica d'accés a dades i simplifica l'accés a fonts de dades dispars, retirant aquesta responsabilitat a les entitats. Aquesta capa independitza l'accés a dades de la tecnologia utilitzada per a implementar-lo podent ser base de dades mitjançant JPA, LDAP o web services de forma totalment transparent a la capa de negoci.

Tecnologia

L'arquitectura de referència proposada en l'apartat anterior està basada en un model multicapa bé conegut en el context de les aplicacions JEE. Aquest model realitza una divisió de responsabilitats en 5 capes. Per a les capes de presentació, lògica de negoci i integració, es mostren els frameworks principals a utilitzar en el següent diagrama:



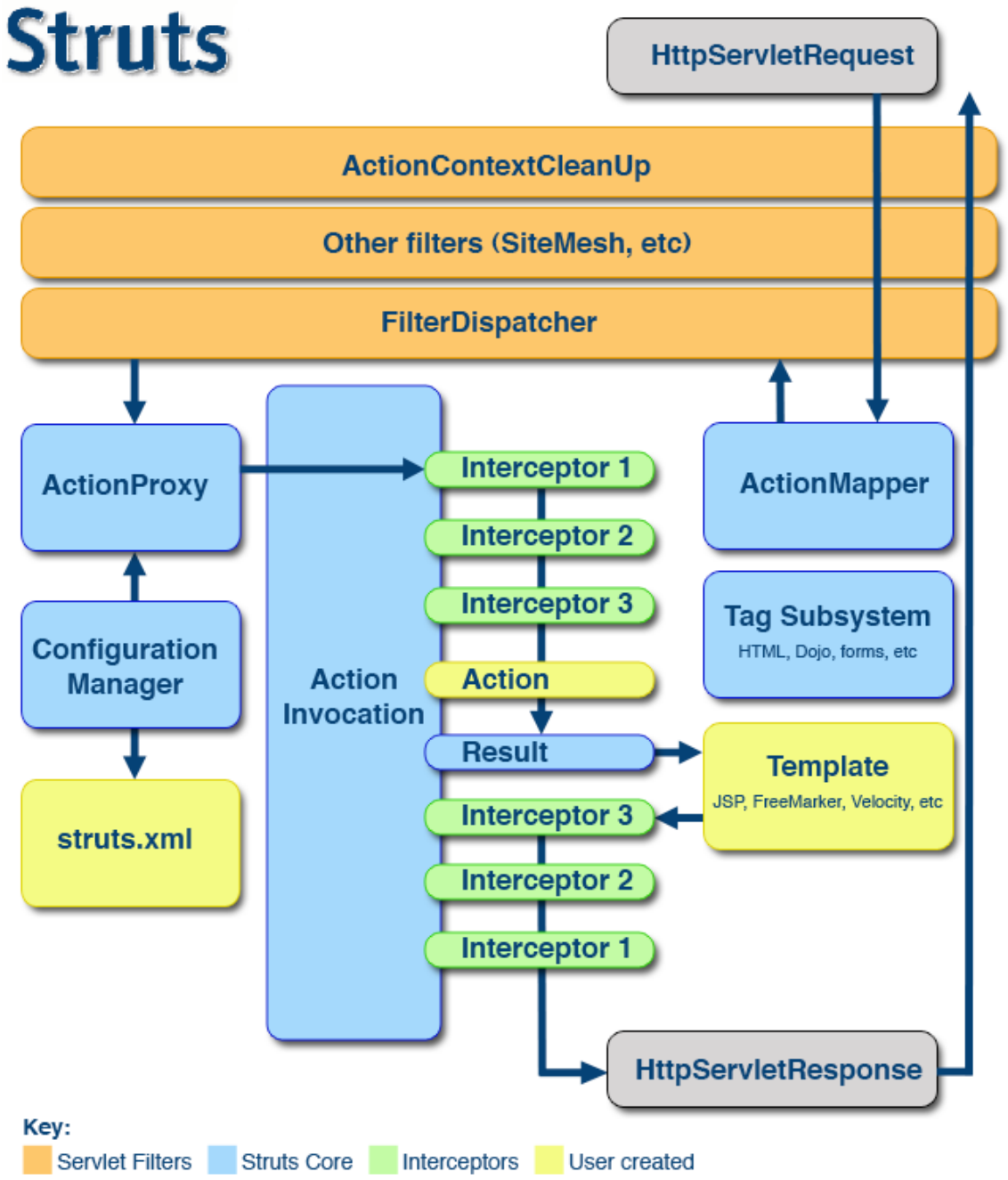
Il·lustració : Diagrama de Tecnologia

Capa de Presentació

Amb la finalitat de facilitar la implementació de les directives d'arquitectura expressades en l'apartat anterior, s'utilitzarà el framework Struts 2 per a la implementació de la capa de presentació. En concret, Struts 2 proporciona les següents facilitats:

1. Bona part de l'arquitectura de Struts 2 està basada en filtres, pel que facilita la implementació del patró Intercepting Filter, a més de les facilitats quant a filtres que proporciona l'especificació web de JEE.
2. Implementa la lògica de Front Controller + Dispatcher, pel que de forma declarativa pot seleccionar la lògica de presentació (Action) que serà executada per a cada petició.
3. Permet integrar diverses tecnologies de presentació de vistes, tals com Tiles, SiteMesh, etc.
4. Incorpora bona part de la funcionalitat necessària en la capa de presentació (validacions, conversions de tipus de dades, etc.)

El lloc web principal es troba en http://struts.apatxe.org/2.x/. Vegem l'arquitectura general de Struts 2 i el flux que seguiria una petició qualsevol:



Il·lustració : Arquitectura d'Struts 2

1. En el diagrama pot veure's com una petició inicial que arriba al contenidor web passarà a través d'una cadena de filtres (definits en web.xml) fins a arribar al FilterDispatcher.
2. Aquest consultarà al ActionMapper para determinar si la petició hagués d'invocar cert action.
3. Si el resultat és afirmatiu, FilterDispatcher delegarà el control en el ActionProxy.

4. ActionProxy consultarà el gestor d'arxius de configuració (inicialitzat des de l'arxiu struts.xml).
5. ActionProxy crea un ActionInvocation, el qual implementa un patró de dissenyo Command.
6. Aquest invocarà els interceptores configurats declarativament i el Action que el programador haurà proporcionat. Des del codi del Action es procedirà a consumir la lògica de negoci adequada mitjançant el « Business Delegate » corresponent.
7. El codi de resultat retornat pel action serà resolt pel ActionInvocation consultant el codi retornat en l'arxiu struts.xml. Sovint això implica la generació d'una vista.
8. Després de la generació de la vista es torna a passar per la cadena de interceptores en ordre invers a l'inicial, i finalment es torna a passar per la cadena de filtres definits en web.xml.

Tots els objectes d'aquesta arquitectura (Actions, Results, Interceptors, etc) són creats per una factoria d'objectes. En el nostre cas, utilitzarem la integració Struts 2/Spring perquè Spring sigui la factoria d'objectes de Struts 2.

Guió de desenvolupament

Aquest apartat pretèn explicar el desenvolupament d'un cas d'ús típic en la capa de presentació, en el sentit d'oferir perspectiva sobre quins arxius han de ser manipulats típicament per a implementar una nova fucionalitat:

1. Elaborar la vista/s que invocaran la funcionalitat a crear (típicament des d'un formulari)
2. Definir les validacions corresponents a aquest formulari
3. Definir la nova funcionalitat (típicament un nou mètode en un Action) en l'arxiu de configuració struts.xml.
4. Codificar la lògica de presentació en la classe Action corresponent.
5. Configurar Spring para injectar les noves dependències en la classe Action.

Generació de Vistes

Encara que el projecte utilitza Tiles (http://tiles.apatxe.org/) per a la generació de vistes, SiteMesh (http://www.opensymphony.com/sitemesh/) s'està imposant com l'alternativa més simple d'utilitzar.

Capa de Lògica de Negoci

Spring

Si bé el framework Spring [spring] (http://www.springframework.org) és d'aplicabilitat en tota l'arquitectura proposada, la seva presència es fa molt més evident al programador en la capa de lògica de negoci. En particular, les característiques de Spring de les quals es farà un ús més intensiu són les següents:

1. Inversió de control per a la injecció de dependències ens les classes de l'aplicació (de fet, aquesta norma aplica a totes les capes),

2. Programació orientada a l'aspecte per a la implementació de diverses responsabilitats no funcionals, tals com seguretat, auditoria i transaccions.
3. Ús del template JPA proporcionat per Spring en la capa d'integració.

Les següents seccions descriuen les pràctiques que s'han aplicat a nivell de desenvolupament en la lògica de negoci.

Separar la interfície respecte de la implementació

1. Per a totes les classes, en particular per a aquelles que implementin patrons: Business delegate
2. Façade Application
3. Service
4. Data Access Object

S'ha separat la definició de la interfície mitjançant una Interface Java respecte de la implementació en la classe corresponent. Aquest és un aspecte fonamental per a poder aprofitar les característiques AOP de Spring.

Especificar característiques transaccionals de forma declarativa

Les classes Façade són el punt d'inici i finalització de les transaccions. La declaració d'aquestes transaccions s'ha fet de forma declarativa aprofitant les característiques AOP de Spring. És molt important qualificar la tipologia de les transaccions (per exemple, qualificar-les com a "read-only" implica una millora de rendiment molt important en la persistència amb Hibernate [hibernate]).

Establir relacions mitjançant inversió de control

L'obtenció de referències per part dels objectes respecte a altres objectes es realitzarà mitjançant el mecanisme d'inversió de control, això és, Spring s'encarregarà d'establir les dependències entre els objectes de forma declarativa.

Capa d'Integració

El conjunt de tecnologies principals a utilitzar en la capa d'integració són (a més de Spring) Java Persistence API i Hibernate.

Java Persistence API

JPA [jpa] és la API estàndard de Java para interactuar amb productes de persistència objecte/relacional, elaborada juntament amb l'especificació de EJB 3.0. D'aquesta manera, el desenvolupador codifica sobre la API JPA, la qual interactua amb un producte objecte/relacional concret. En el nostre cas, aquest producte serà Hibernate. Spring aporta una capa d'abstracció addicional que simplifica molt l'ús de JPA en les aplicacions reestructurant la jerarquia d'excepcions i simplificant la API (vegi's http://static.springframework.org/spring/docs/2.0.x/reference/orm.html#orm-jpa-template) .

Hibernate

Hibernate és el producte objecte/relacional a s'ha utilitzat en l'aplicacó. S'ha configurat JPA declarativament per a utilitzar Hibernate com a proveïdor objecte/relacional.

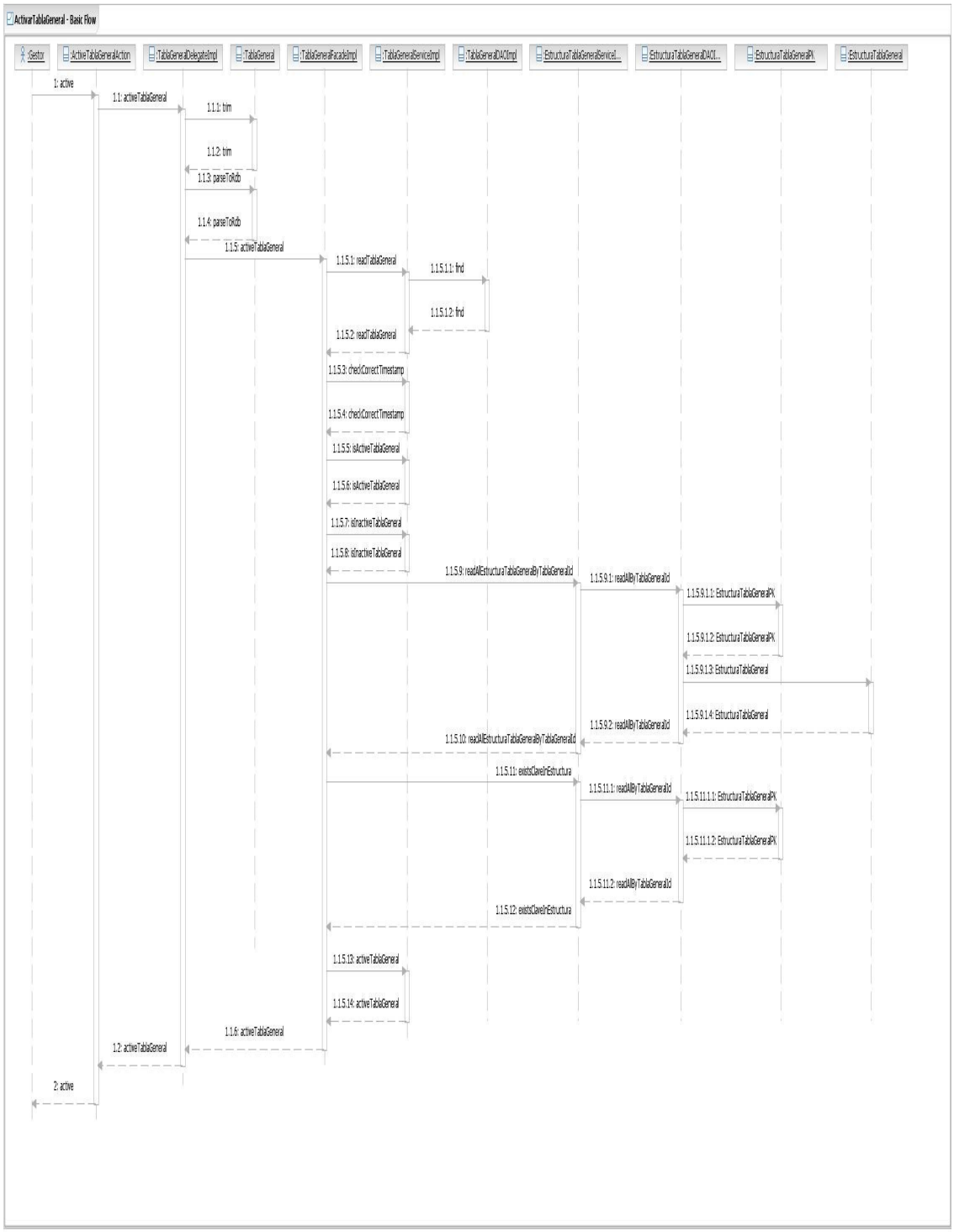
Seqüència d'exemple d'una petició

L'objectiu d'aquest apartat és oferir un exemple complet sobre "camí" d'execució d'una petició HTTP sobre l'aplicació implementada amb la tecnologia descrita en aquest document. Com ajuda a aquesta descripció s'adjunta el diagrama de seqüència del cas d'ús "Activar Taula General" de l'aplicació. Aquest cas d'ús permet activar una taula general una vegada aquesta taula s'ha definit i s'ha creat una estructura associada. En l'activar-la ja és possible afegir dades a aquesta taula. Per a poder activar una taula general, han d'observar-se diverses regles de negoci:

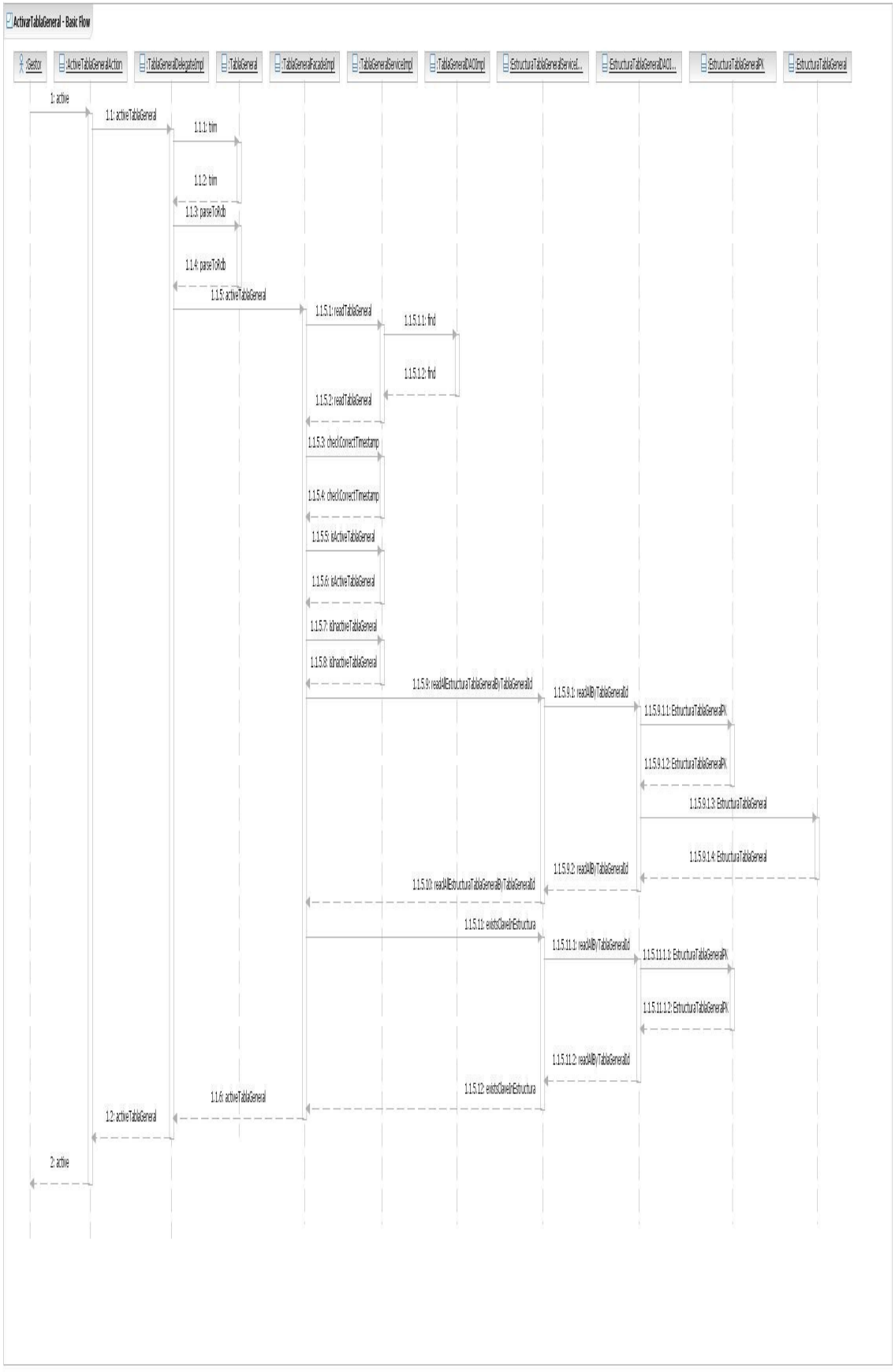
1. verificar que no va haver canvis en la base de dades entre l'instant que es van sol•licitar les dades de la taula a activar i l'instant és que se sol•licita gravar les noves dades
2. verificar que la taula no estigui ja activada
3. verificar que la taula tingui una estructura associada i que aquesta estructura tingui almenys una clau

Vegem com s'implementa aquest cas d'ús seguint el diagrama de seqüència adjunt. En primer lloc l'usuari sol•licita consultar les taules generals per a trobar aquella que desitja activar. Aquesta petició es rep en la classe ManageTablaGeneralAction, la qual retornarà la vista adequada a l'usuari per a poder realitzar la consulta. Una vegada l'usuari rep la nova vista, selecciona activar una de les taules generals. De nou, aquesta petició és tractada per ManageTablaGeneralAction, que presentarà a l'usuari una vista per a confirmar l'activació de la taula. Aquesta nova vista conté els detalls de la taula que es desitja activar. Pot observar-se com en aquest cas el Action de Struts ActiveTablaGeneralAction es recolza en TablaGeneralDelegateImpl (el qual implementa la interfície TablaGeneralDelegate, separant així la interfície de la implementació). El Delegate, com frontera entre la capa de presentació i lògica de negoci, realitzarà les adaptacions d'interfície i/o d'entitats de negoci que correspongui. El Action posseeix una referència al Delegate gràcies a que Spring la va injectar. D'aquesta manera Spring actua de factoria d'objectes i ens permetria provar diverses implementacions tan sols amb un canvi en la configuració declarativa. El Delegate sol•licita a TablaGeneralFacadeImpl (que implementa la interfície TablaGeneralFacade i la referència de la qual li ha proporcionat Spring) els detalls de la taula general que es desitja activar. En aquest instant s'inicia la transacció que es va indicar de forma declarativa en el descriptor de Spring. El façade coordinarà els serveis de negoci adequats. Aquest cas és senzill i tan sols ha de consumir el servei d'obtenció dels detalls d'una entitat que li ofereix TablaGeneralServiceImpl (que implementa la interfície TablaGeneralService). Al seu torn, aquest es recolza en el DAO TablaGeneralDAOImpl (que implementa la interfície TablaGeneralDAO) que s'encarrega d'interactuar amb la BBDD via JPA i Hibernate. Tots els objectes han aconseguit referències entre si gràcies a la característica IoC de Spring). Al final de procés s'haurà obtingut una entitat TablaGeneral de forma transaccional que serà lliurada a la vista per a la seva visualització. Una vegada l'usuari confirma que desitja activar aquesta taula general, arribarà la petició al Action ActiveTablaGeneralAction, el qual invoca sobre el TablaGeneralDelegateImpl el servei "activeTablaGeneral". Abans de consumir aquest servei en la capa de lògica de negoci, el delegate realitza adequacions en l'entitat TablaGeneral i invoca el servei "activeTablaGeneral"

en el façade, el qual s'executarà de forma transaccional. En aquest punt, el façade coordina diverses regles de negoci presents en TablaGeneralServiceImpl i EstructuraTablaGeneralServiceImpl amb la finalitat de verificar les restriccions exposades anteriorment. Aquests serveis, al seu torn, es recolzaran en els DAOs corresponents a les entitats amb les quals tracten: TablaGeneralDAOImpl i EstructuraTablaGeneralDAOImpl. AL final del procés s'haurà marcat una taula general com a activa.



Il·lustració : Activar Taula General I



Il·lustració : Activar Taula General II

Implementació a nivell de disseny

Aquest capítol mostra, a nivell de referència, un exemple d'implementació d'un cas d'ús a nivell de disseny. La resta d'implementacions són anàlogues a aquesta. Aquest exemple es basa en el cas d'ús *modificar dades d'una taula general*.

Introducció

Sobre la base de l'arquitectura de referència abans definida, a continuació s'explicarà la Capa Lògica de Negoci detallant les regles de negoci implementades en aquest Diagrama de seqüència. Recordem que la Façade coordina tots els serveis d'aplicacions de les regles de negoci, i serveix per a proveir una interfície senzilla cap a la Capa de Client, encapsulant les dades i lògica de negoci i amb ells la complexitat dels serveis implicats.

Descripció del flux

La transacció del servei Facade s'inicia quan el Delegate *DatosTablaGeneralDelegateImp* sol•licita a la Façade *DatosTablaGeneralFacadeImp* l'actualització de les dades d'una taula mitjançant el mètode (*updateDatosTablaGeneral*).

La Façade sol•licita al Servei *TablaGeneralServiceImp* la lectura de la taula mitjançant el mètode (*readTablaGeneral*), el Servei li enviarà el codi de taula (codtabla) al DAO *TablaGeneralDAOImp* a través del mètode (find) perquè aquest realitzi l'acció, retornant-li el resultat. També verifica l'estat actiu de la taula mitjançant el mètode (*isActiveTablaGeneral*).

Una vegada rebuda la verificació correcta, la Façade invoca al Servei *EstructuraTablaGeneralServiceImp* pel mètode (*readAllEstructuraTablaGeneralByTablaGeneralId*) enviant-li el codi de la taula (codtabla) per a obtenir l'estructura de la mateixa, aquest últim li sol•licita l'operació al DAO *EstructuraTablaGeneralDAOImpl* mitjançant el mètode (*readAllByTablaGeneralId*), passant-li el mateix paràmetre, el qual realitza l'acció sobre l'Entitat EstructuraTablaGeneral a través del mètode (*estucturaTablaGeneral*).

Seguidament la Façade sol•licita pel mètode (*parseDatosLogicosTablaGeneralToDatosTablaGeneral* els ajustaments necessaris al Servei *DatosTablaGeneralServiceImp* qui realitza l'acció sobre l'Entitat *DatosTablaGeneral* pel mètode (*DatosTablaGeneral*). Després executa el mètode (*generateSpaces*). El flux continua quan la Facade crida al servei el mètode (*readDatosTablaGeneral*) per a la lectura de les dades de la taula i aquest li passa al DAO, *DatosTablaGeneralDAOImp* el mètode (*findById*) passant la petició a l'Entitat *DatosTablaGeneral* a través del mètode (*DatosTablaGeneral*).

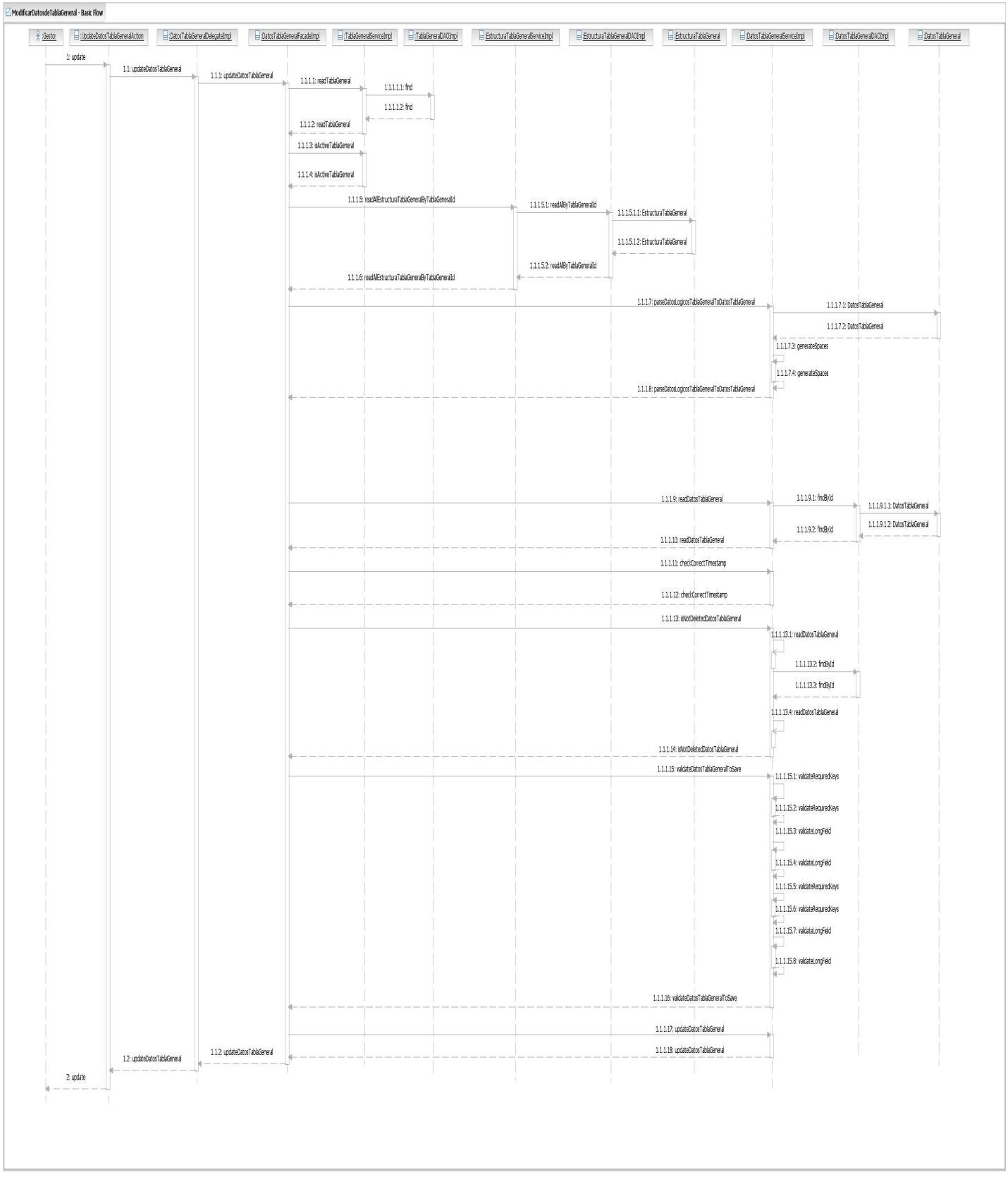
Després la Facade invoca al Servei per a la verificació de la data i hora de l'última modificació mitjançant el mètode (*checkCorrectTimeStamp*).

Posteriorment li sol•licita al Servei la verificació de l'existència de les dades a modificar, mitjançant el mètode (*isNotDeletedDatosTablaGeneral*) aquest servei executa el mètode (*readTablaGeneral*) i li envia la petició al DAO, *DatosTablaGeneralDAOImp* a través del mètode (*findById*), qui s'encarrega de retornar les dades sol•licitades.

La Facade invoca al Servei per a la validació de les noves dades de la taula mitjançant el mètode (*validateDatosTablaGeneralToSave*), aquest servei executa els mètodes (*validateRequiredKeys*), (*validateLongField*), consecutivament. Finalment la Facade li sol•licita

al Servei el update de les dades de la mateixa a través del mètode (*updateDatosTablaGeneral*) La Facade finalitza la transacció una vegada rep del Servei les dades de la taula modificada.

Diagrama de seqüència



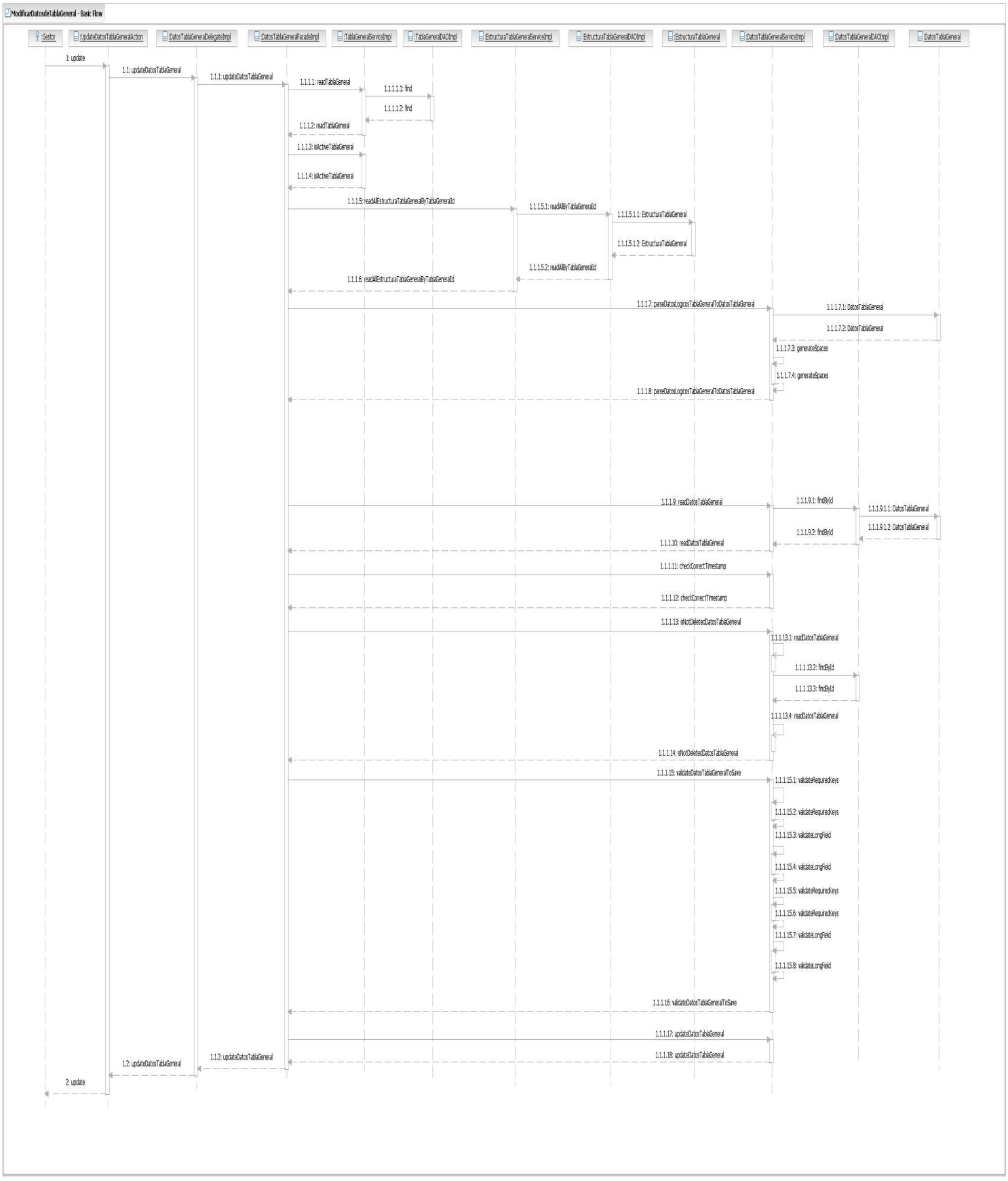
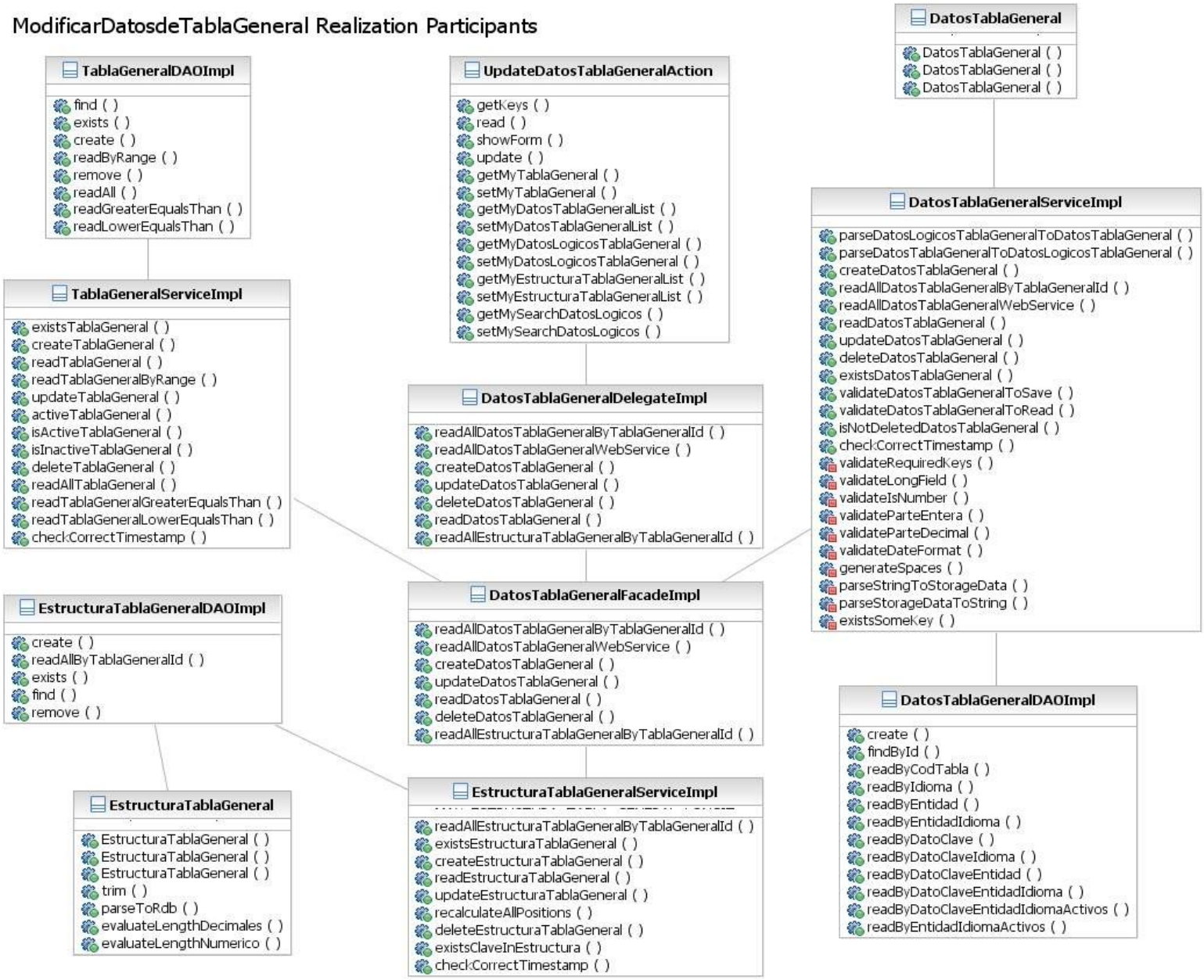

ModificarDatosdeTablaGeneral - Basic Flow
:Gestor
:UpdateDatosTablaGeneralAction
:IDatosTablaGeneralDelegate3Impl
:IDatosTablaGeneralFacade3Impl
:ITablaGeneralService3Impl
:ITablaGeneralDAO3Impl
:IEstructuraTablaGeneralService3Impl
:IEstructuraTablaGeneralDAO3Impl
:IEstructuraTablaGeneral
:IDatosTablaGeneralService3Impl
:IDatosTablaGeneralDAO3Impl
:IDatosTablaGeneral
1: update
1.1: updateDatosTablaGeneral
1.1.1: updateDatosTablaGeneral
1.1.1.1: readTablaGeneral
1.1.1.1.1: find
1.1.1.1.2: find
1.1.1.2: readTablaGeneral
1.1.1.3: isActiveTablaGeneral
1.1.1.4: isActiveTablaGeneral
1.1.1.5: readAllEstructuraTablaGeneralByTablaGeneralId
1.1.1.5.1: readAllByTablaGeneralId
1.1.1.5.1.1: EstructuraTablaGeneral
1.1.1.5.1.2: EstructuraTablaGeneral
1.1.1.5.2: readAllByTablaGeneralId
1.1.1.6: readAllEstructuraTablaGeneralByTablaGeneralId
1.1.1.7: parseDatosLogicosTablaGeneralToDatosTablaGeneral
1.1.1.7.1: DatosTablaGeneral
1.1.1.7.2: DatosTablaGeneral
1.1.1.7.3: generateSpaces
1.1.1.7.4: generateSpaces
1.1.1.8: parseDatosLogicosTablaGeneralToDatosTablaGeneral
1.1.1.9: readDatosTablaGeneral
1.1.1.9.1: findById
1.1.1.9.1.1: DatosTablaGeneral
1.1.1.9.1.2: DatosTablaGeneral
1.1.1.9.2: findById
1.1.1.10: readDatosTablaGeneral
1.1.1.11: checkCorrectTimestamp
1.1.1.12: checkCorrectTimestamp
1.1.1.13: isNotDeletedDatosTablaGeneral
1.1.1.13.1: readDatosTablaGeneral
1.1.1.13.2: findById
1.1.1.13.3: findById
1.1.1.13.4: readDatosTablaGeneral
1.1.1.14: isNotDeletedDatosTablaGeneral
1.1.1.15: validateDatosTablaGeneralToSave
1.1.1.15.1: validateRequiredKeys
1.1.1.15.2: validateRequiredKeys
1.1.1.15.3: validateLongField
1.1.1.15.4: validateLongField
1.1.1.15.5: validateRequiredKeys
1.1.1.15.6: validateRequiredKeys
1.1.1.15.7: validateLongField
1.1.1.15.8: validateLongField
1.1.1.16: validateDatosTablaGeneralToSave
1.1.1.17: updateDatosTablaGeneral
1.1.1.18: updateDatosTablaGeneral
1.1.2: updateDatosTablaGeneral
1.2: updateDatosTablaGeneral
2: update

Diagrama de clases participants



Conclusions

Aquest projecte ha servit per a obtenir una implementació sòlida d'un sistema que gestiona els paràmetres d'una organització de manera flexible i ràpida. Crear noves taules de parametratge, actualitzar el calendari corporatiu i gestionar els missatges de text de les aplicacions són operacions senzilles i ràpides.

La lògica de negoci d'aquesta aplicació podria ser incorporada com a llibreria Java en d'altres aplicacions ja que, amb l'ajuda de Maven, el projecte s'ha estructurat en mòduls. Un d'aquests mòduls (tco-core) genera una llibreria JAR que pot ser incorporada per altres aplicacions, de manera que la consulta de la parametrització d'una organització es faria de manera consistent i centralitzada.

La utilització d'Spring ha facilitat molt la correcta estructuració de l'aplicació en capes que poden ser configurades de forma senzilla mitjançant la injecció de dependències. Aquesta tècnica ha resultat fonamental per maximitzar el desacoblament entre capes i provar implementacions alternatives en diversos serveis només realitzant canvis de configuració.

Les facilitats de programació orientada a l'aspecte oferides per Spring han permès una separació clara de diverses responsabilitats, com ara:

1. La gestió de les transaccions, la qual ha esdevingut completament declarativa, cosa que representa un guany molt significatiu,
2. La gestió de les traces (logs) s'ha pogut realitzar en part mitjançant programació orientada a l'aspecte, cosa que ha suposat una simplificació,
3. La gestió de caches de dades. Aquest també ha sigut un punt que s'ha vist molt beneficiat. L'aplicació utilitza la llibreria *ehcache* per a la gestió de cache mitjançant AOP. En concret, es fa cache de les dades llegides de la taula de missatges de internacionalització.

Tot i que no s'han fet servir en aquest projecte, altres característiques com ara la gestió de la seguretat també s'haurien vist molt simplificades amb aquesta tècnica.

La integració que Spring ofereix amb JPA i Hibernate ha incrementat significativament la productivitat. Les abstraccions que ofereix Spring a aquest respecte ofereixen un excel•lent nivell d'aïllament de les APIs de JPA i Hibernate, de tal manera que pràcticament totes les línies de codi del programador realitzen alguna tasca productiva. En canvi, si haguéssim d'accedir directament a les APIs de Hibernate o, fins i tot, de JDBC, la quantitat de codi necessari només per a configuracions i tasques repetitives implicarien una inversió de temps extra molt considerable.

Finalment, Maven2 ha ajudat molt en la gestió del projecte. Si bé imposa una corba d'aprenentatge molt important en els primers moments, un cop dominada allibera el programador de moltes tasques administratives, com ara la gestió de dependències. Cal tenir en compte que en aquest projecte el nombre de llibreries ja era prou significatiu, de l'ordre de desenes. Però Maven no és només una eina de gestió de dependències. La gran quantitat d'extensions disponibles poden realitzar, per exemple, nombrosos informes respecte de la qualitat del codi, cosa que es por verificar en aquest projecte si s'executa el *target* "site" de Maven.

Per últim, cal destacar la brillant i compacta sintaxi de JQuery, cosa que permet aconseguir molts efectes a la capa de presentació amb poc esforç. JQuery compta amb un conjunt de components atractius, com ara el calendari que s'ha utilitzat en aquest projecte.

Bibliografia

Struts2: Donald Brown, Chad Michael Davis, Scott Stanlick, Struts 2 in Action, 2008
spring: Craig Walls, Spring in Action, 2008
hibernate: Christian Bauer, Gavin King, Java Persistence with Hibernate, 2007
jpa: Mike Keith, Merrick Schincariol, Pro EJB 3 Java Persistence API, 2006