BSCDW : IMPLEMENTACIÓ D'UN ENTORN DE TREBALL COL·LABORATIU I DISTRIBUÏT

per

Josep M Comajuncosas

Memòria proposada com a Treball de Fí de Carrera de

Enginyeria Tècnica en Informàtica de Sstemes

Universitat Oberta de Catalunya

2004

Consultor Joan Manel Marquès Puig

Data d'entrega 28 de juny de 2004

Als meus pares,
pel seu support
i per haver-me animat
a fer aquesta carrera

Universitat Oberta de Catalunya

BSCDW : IMPLEMENTACIÓ D'UN ENTORN DE TREBALL COL·LABORATIU I DISTRIBUÏT

per Josep M Comajuncosas

Consultor: Joan Manel Marquès Puig

Aquest projecte ha suposat la implementació d'un entorn de treball col·laboratiu inspirat en el BSCW però amb una nova arquitectura, basada en Servlets en comptes de CGI.

Les prestacions, per ara reduïdes, de l'aplicatiu són fàcilment expandibles gràcies a patrons avançats de disseny com una redirecció de sol·licituds al servidor inspirada en el model de delegació d'events d'AWT i la incorporació de DAO (Data Access Objects) per proporcionar mecanismes de persistència transparents a l'aplicació.

Al respecte, s'han implementat tres mecanismes de persistència a mesura que el projecte evolucionava i es feien necessaris: persistència local en carpetes del servidor, persistència local en una sola carpeta amb un descriptor Xml de directori virtual, i persistència distribuïda emprant laColla com a middleware.

El projecte ha permès aprofundir en la creació d'aplicacions Web emprant Servlets, en la reutilització de components (com Apache Jakarta Commons Upload per la recepció de fitxers, Jdom pel tractament del descriptor virtual i la mateixa laColla) i en el seu deployment en un servidor on podeu realitzar proves del seu funcionament.

TAULA DE CONTINGUTS

Capítol 1 3
Justificació i context: punt de partida i aportació del TFC 3
Objectius del TFC 3
Enfocament i mètode seguit 4
Planificació 4
Productes obtinguts 5
Capítols de la memòria 6
Capítol 2 8
Entorns col·laboratius 8
El BSCW 9
Arquitectures distribuïdes 10
Tecnologies de distribució d'objectes 11
Aplicacions web 12
Entorns i Comparativa 12
Estructura d'una aplicació web 13
Capítol 3 16
Eines de desenvolupament emprades 16
Prestacions implementades 16
Arquitectura general del projecte 19
Motivacions 19
Capes 19
Redirecció de sol·licituds 21
Gestió de la persistència 23
Cap a una arquitectura distribuïda 26
Treball en local 26
L'API de laColla i la gestió de la persistència 26
Directoris virtuals 27
Incorporant laColla 27
Capítol 4 30
Valoració dels resultats 30
Possibles escenaris de treball 30
BSCDW v BSCW 32
laColla 32
Futures línies 33
Valoració personal 34

LLISTA D'IL·LUSTRACIONS

Fig1. planificació original del projecte 5
Fig2. arquitectura bàsica del BSCW 9
Fig3. arquitectura en mòduls del BSCW 9
Fig4. arquitectura de tres capes 11
Fig5. tres capes i laColla 12
Fig6. servidor amb CGI 13
Fig7. servidor amb Servlets 13
Fig8. screenshot de l'aplicació 17
Fig9. formulari de creació de carpetes 17
Fig10. formulari per enviar fitxers 17
Fig11. identificació tipus MIME 18
Fig12. barra de navegació 18
Fig13.creació d'una carpeta (simple) 20
Fig14. UML estàtic - redirecc. requests 22
Fig15. UML estàtic - DAO 24
Fig16. creació d'una carpeta (complet) 26
Fig17. servidor i múltiples clients 30
Fig18. múltiples servidors i clients 31
Fig19. laColla amb servidor i clients 31

Capítol 1

INTRODUCCIÓ

Justificació i context: punt de partida i aportació del TFC

La realització d'aquest projecte s'emmarca dins el conjunt de proves d'implementació que es duen a terme per comprovar el funcionament dels prototipus de laColla, un middleware que permet la creació d'aplicacions distribuïdes. Des d'aquest punt de partida, la seva aportació ha estat el desenvolupament d'un aplicatiu que emprés laColla per a gestionar la persistència d'objectes i d'aquesta manera comprovar fins a quin punt era possible en l'estat de desenvolupament actual del prototipus.

Malgrat això l'extensió del projecte per un costat, i les limitacions derivades de l'estat actual de desenvolupament de laColla per l'altre, m'han portat a emfatitzar més en aspectes previs a la implementació d'aquest entorn pròpiament dit, posant èmfasi en el disseny d'una aplicació que pugui servir de punt de partida a una futura implementació més completa.

Per manca d'un millor nom l'aplicatiu l'he anomenat BSCDW, de *Basic Support for Collaborative and Distributed Work*.

Objectius del TFC

Aquest Treball de Fí de Carrera ha partit de les funcionalitat més bàsiques d'un entorn col·laboratiu com és el BSCW (Basic Support for Collaborative Work) i les ha reimplementades en un aplicatiu Java basat en Servlets. Es manté per tant una arquitectura C/S però emprant una tecnologia més actual que CGI, com era el cas del BSCW original.

Les funcionalitats implementades són les més simples: creació i navegació en una estructura de carpetes i fitxers, i possibilitat de pujar i baixar fitxers, amb un mecanisme que permet accedir a laColla o altres sistemes de gestió de la persistència de manera transparent a l'usuari.

Enfocament i mètode seguit

El fet de disposar d'un referent funcional com és el BSCW ha simplificat molt l'anàlisi de requisits, de manera que només ha calgut especificar les funcionalitats mínimes del projecte i de bon principi concentrar-se en l'estudi de les tecnologies a emprar (API de Servlets, entorn de treball, reutilització de components externs) i implementar prototipus que s'adequaven a millores graduals en diversos aspectes de disseny.

He donat molta prioritat a aquesta rutina d'elaborar prototipus i instalar-los en un servidor públic per evaluar-ne el comportament, fins el punt que n'he penjat uns deu amb prestacions incrementalment superiors. A més de permetre un millor feedback, he pogut anar comprovant que el seu deployment fos en tot moment simple, a fi d'obtenir un producte portable i fàcilment instalable.

Planificació

La planificació establerta en la primera entrega parcial del projecte, i que adjunto tot seguit, no es correspon del tot al pla de treball finalment adoptat, per diversos motius:

- Vaig sobreestimar el temps dedicat a crear una interfície gràfica pel client, de manera que els primers prototipus van oferir més prestacions de les previstes

- Com he indicat he preferit, també per recomanació del consultor, un desenvolupament basat en prototipatge molt més intensiu a una fase de disseny massa extensa.

- Les limitacions de laColla van exigir implementar un mecanisme de persistència que ha suposat força temps afegit.

- El retràs en l'entrega de prototipus funcionals primer, la successió de diferents versions després i la dificultat per manca de documentació finalment han suposat un retràs molt considerable en l'addició de laColla, repercutint en l'abast final del projecte en relació a la implementació de persistència distribuïda.

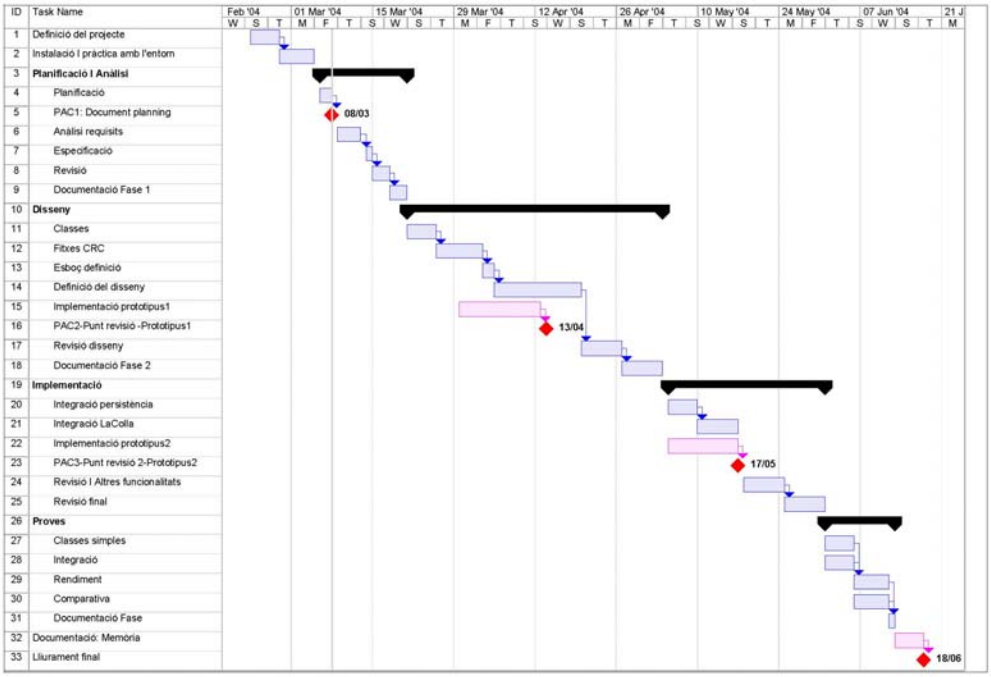

Fig1. planificació original del projecte

Productes obtinguts

El present projecte és fonamentalment d'implementació i el producte obtingut ha estat una aplicació Web que s'entrega comprimit en format WAR i és fàcilment instalable en qualsevol servidor Tomcat, només caldrà editar el fitxer web.xml de configuració, crear la carpeta que indica aquest fitxer al disc dur i configurar laColla.

Igualment per fer proves s'ha anat col·locant cada prototipus de l'aplicatiu en un servidor de prova. He penjat la versió amb persistència local a http://jeppi.net:8080/TDP_09 i la versió amb persistència local i estructura de directoris virtual http://jeppi.net:8080/TDP_10. Aquests tipus de persistència s'expliquen al projecte. No he penjat la versió entregada del projecte, que incorpora persistència amb laColla, perquè no és molt fiable i carrega molt el servidor.

Capítols de la memòria

Aquesta memòria està estructurada en quatre capítols. El capítol 1 és la present Introducció.

El capítol 2 llista breument els fonaments que han servit de punt de partida a la concepció d'aquest projecte, com són els Entorns de Treball Col·laboratius, les diverses possibilitats d'arquitectura distribuïda de tipus Client-Servidor que pot servir per implementar-los, la ubicació d'un middleware com laColla en aquest esquema i els entorns de desenvolupament d'aplicacions Web per implementar aquestes arquitectures.

El capítol 3 versa sobre la implementació pròpiament dita del Projecte. Primer s'esmenta l'entorn de treball i les eines de programari que he utilitzat i per què, per llistar tot seguit quines prestacions s'han implementat. Tot seguit em centro en aspectes de disseny com són l'arquitectura general de l'aplicatiu, la justificació de l'arquitectura escollida, el mètode de delegació d'events per atendre les peticions dels clients i l'ús d'objectes d'accés a dades per aconseguir una gestió de la persistència transparent a l'usuari, per finalitzar parlant de laColla, els passos previs que ha calgut tenir en compte per incloure-la en el projecte (com és la gestió virtual de directoris) i les seves limitacions cara a una implementació més completa.

Finalment el capítol 4 presenta com a conclusió una valoració crítica de les fites assolides en aquest projecte i una disquisició sobre els possibles escenaris d'aplicació de l'eina desenvolupada. També ofereix una comparativa entre el BSCW original i un hipotètic aplicatiu basat en el meu projecte però amb funcionalitats equiparables al BSCW i conclou amb una valoració de les prestacions actuals de laColla en relacío al meu projecte.

Capítol 2

FONAMENTS DEL PROJECTE

Entorns col·laboratius

El treball col·laboratiu, ja sigui per la l'intercanvi d'idees i la planificació de tasques com per l'anàlisi, desenvolupament i revisió de productes, requereix eines programari específiques. Podem classificar-les segons els requisits de nombre d'usuaris (grups tancats de treball o comunicacions puntuals un a un) i temporals (la comunicació pot ser síncrona si es pressuposa la recepció virtualment immediata de la informació o assíncrona si la immediatesa no és essencial) dels usuaris. Per exemple un treball on la sincronia és fonamental pot emprar eines de missatgeria instantània en grups tancats o un a un, o vídeoconferència si és un a un (almenys actualment). Si la comunicacío és assíncrona podem emprar l'email (per comunicacions un a un) o un NewsGroup (per comunicacions en grups tancats). Usualment s'estableix una sinèrgia entre el treball en grup assíncron i síncron.

Els entorns col·laboratius són essencials en el desenvolupament de programari en grup. En aquests casos resluta impressindible mantenir un control de les versions dels documents i productes generats. Eines com CVS (Concurrent Versioning System) estan dissenyades per a aquest fi, i s'empren extensament a nivell industrial i de recerca, però menys a nivell pedagògic. CVS requereix clients específics i és més aviat complexe de gestionar.

Altres requisits desitjables en aquests entorns són la possibilitat de fer un seguiment del desenvolupament del projecte amb un sistema jeràrquic de missatgeria o forums, adjunció de documents i enviament de reports automàtics. Eines com el BSCW són més eficients en aquests aspectes, encara que menys en el control de versions. Un dels seus avantatges és que només cal un navegador Web per accedir-hi. La UOC l'empra com a plataforma de treball col·laboratiu en diverses assignatures.

Un entorn més recent és Twiki, un clon de Wiki Wiki adequat pel treball col·laboratiu i que pot funcionar amb un simple navegador, però garantint el control de versions dels continguts ja que empra CVS internament.. Entre els seus avantatges podem esmentar

que és de codi obert (a diferència d'eines similars com Microsoft Share Point Team Services), altament customitzable i, com funciona sobre una base de dades manipulada amb scripts CGI, ofereix extensibles opcions d'accés a les dades i estadístiques d'ús.

El BSCW

El BSCW (http://bscw.fit.fraunhofer.de/index.html)és l'espai de treball compartit que ha servit de punt de partida del projecte. A nivell d'implementació, originalment el BSCW partia d'una arquitectura C/S simple on el servidor redirigeix les crides del client als scripts CGI adequats, que pròpiament accedeixen al cos de l'aplicació

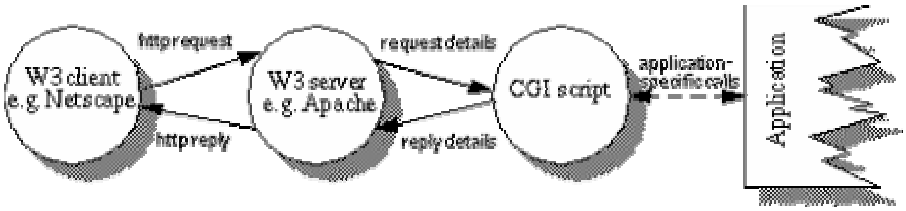

Fig2. arquitectura bàsica del BSCW

Dos aspectes contemplats desde bon principi van ser el control d'accés i el llistat jeràrquic de fitxers (en directoris) amb generació dinàmica de pàgines web.

Aquest esquema internament s'implementa amb tres mòduls clarament definits, el de comunicacions, per enviar i rebre informació de/al client en un protocol determinat, el Servei d'Integració, que redirigeix les sol·licituds del client al mòdul adequat i els Mòduls d'Aplicació,que es comuniquen amb l'aplicació per accedir a dades o efectuar tasques específiques.

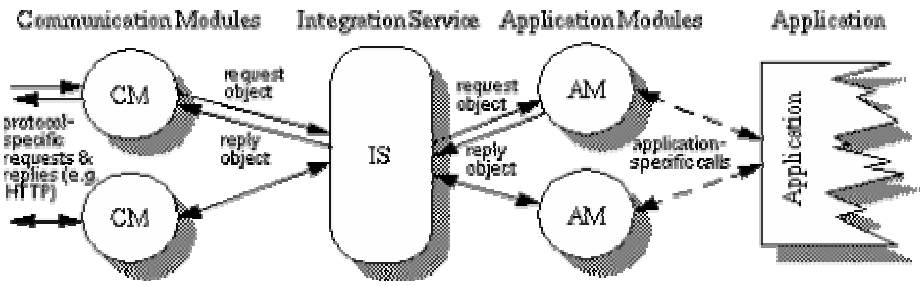

Fig3. arquitectura en mòduls del BSCW

Les darreres versions del BSCW estan escrites íntegrament en Phyton i ofereixen noves prestacions com ara la possibilitat d'establir "comunitats" amb espais públics i privats, sincronització amb "maletí" per treballar offline i disponibilitat d'una API per crear aplicacions que integrin el BSCW com a eina per gestionar el treball col·laboratiu. Totes aquestes prestacions evidentment escapen a l'abast actual del meu projecte.[Que tal afegir-la com a gestor de persistència al meu projecte?:)]

Arquitectures distribuïdes

Aplicacions com el BSCW són de tipus Client/Servidor, amb un client lleuger que només aporta la interfície amb l'usuari (GUI) i un servidor pesant que s'encarrega de tota la resta.

Fins els anys 80 l'arquitectura Client/Servidor va solucionar els problemes de sobrecàrrega de xarxes i simultaneitat d'usuaris amb l'ús de servidors de bases de dades, però deixa de ser òptima per càrregues de més de 100 usuaris.

Actualment els patrons de disseny distingeixen tres capes clarament diferenciades en el servidor, ins i tot si totes les tasques les duu a terme una sola màquina:

- Dades: és el component que s'encarrega d'accedir i gestionar les dades, idealment de manera genèria i independent del mecanisme de persistència finalment emprat, que sol ser una Base de Dades.

- Lògica de l'Aplicació: és el component que realitza pròpiament el conjunt de tasques per a les quals em desenvolupat l'aplicació, també animenada la *lògica de negoci*

- Presentació: és el component que formateja les dades per transmetre-les al client.

En el cas de les aplicacions web com el BSCW la capa de presentació formateja les dades en pàgines HTML amb les quals interacciona el client a través d'un navegador web. I el client en cap moment accedeix directament als gestors de les bases de dades.

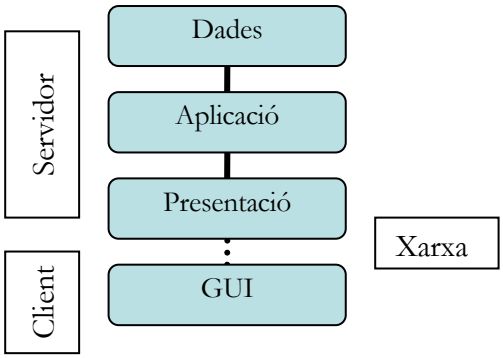

Fig4. arquitectura de tres capes

Aquesta arquitectura en tres capes, que comença a aparèixer als anys 90, permet incrementar l'escalabilitat i disponibilitat optimitzant la reutilització i les despeses de manteniment molt millor que la tradicional estructura en dues capes. Si es vol eventualment distribuir recursos (com ara múltiples bases de dades), aquesta arquitectura, que permet una gran varietat de patrons de disseny, és necessària per amagar al client la complexitat de la distribució de les dades.

Tecnologies de distribució d'objectes

Una millora addicioal a aquesta arquitectura és l'ús d'alguna tecnologia que gestiona les dades de manera distribuïda, com ara ORB o COM/DCOM o, en el cas del nostre projecte, laColla. Aquest component fa d'intermediari (*middleware*) entre el nostre gestor de dades i les dades en sí, que poden estar distribuïdes i replicades en diverses màquines optimitzant-ne per tant la disponibilitat, escalabilitat i robustesa.

En aquest cas l'arquitectura de la nostra aplicació queda modificada com segueix:

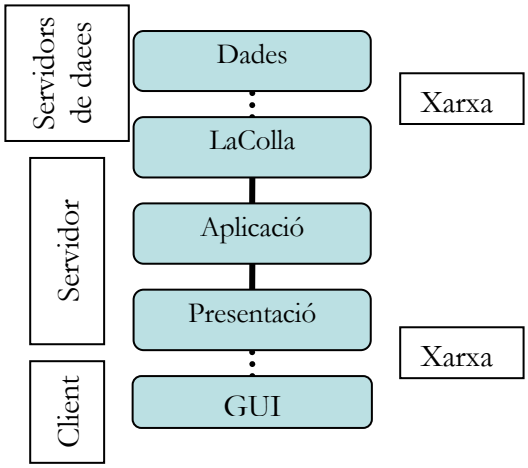

Fig5. tres capes i laColla

Aplicacions web

Entorns i Comparativa

Una aplicació web C/S que segueixi una de les arquitectures anteriorment esmentades s'implementa en el servidor, mentre que el client només requereix un navegador amb el qual efectua *requests* al servidor amb comandes HTML estandaritzades com ara GET, PUT o POST.

L'aplicació en sí es pot implementar emprant diverses tecnologies, com CGI (amb la que tradicionalment ha funcionat el BSCW) o l'API Servlet de Sun, per anomenar dos exemples que crec interessants comentar.

CGI presenta problemes quan a portabilitat i rendiment. Sol·licituds concurrents de diversos clients activen igual nombre de processos al servidor,.

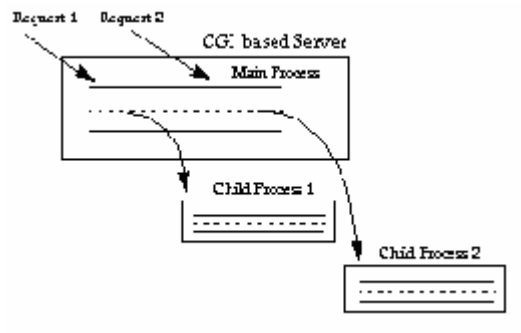

Fig6. servidor amb CGI

Amb la tecnologia de Servlets cada request és un thread diferent que s'executa concurrentment amb una menor càrrega.

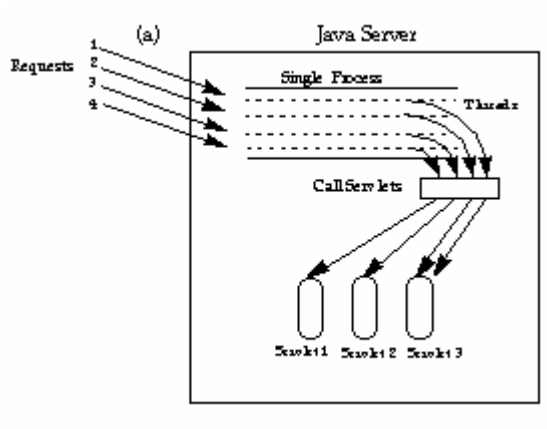

Fig7. servidor amb Servlets

Els Servlets són extensions directes del servidor web que es carreguen dinàmicament i en estar programats en Java permeten accedir amb molta facilitat a Bases de Dades i altres recursos, a més a més d'oferir la possibilitat de *catching* d'accessos a dades i balancejar la càrrega entre servlets..

Estructura d'una aplicació web

Una aplicació web –ja centrant-nos en una aplicació creada amb servlets- conté diversos elements com recursos estàtics (per exemple contingut multimèdia), el codi de l'aplicació i llibreries auxiliars i fitxers de configuració.

Per distribuir i instalar (*deploy* en anglès) una aplicació web en un servidor cal empaquetar-la en un fitxer WAR amb una estructura de directoris determinada que com a mínim ha d'incloure aquests elements:**(ARREL)** - Fitxers visibles ple client, com ara pàgines html o jsp **/WEB-INF/web.xml** - Un descriptor XML amb paràmetres de configuració de la nostra aplicació.**/WEB-INF/classes/** - Classes java no empaquetades, estructurades en *packages* **/WEB-INF/lib/** - classes java empaquetades en fitxers JAR, llibreries i drivers JDBC.

Si hem seguit aquesta estructura l'aplicació Web resultant és totalment portable: només cal desempaquetar el fitxer WAR que la conté al servidor i estarà immediatament disponible.

Capítol 3

IMPLEMENTACIÓ

Eines de desenvolupament emprades

D'acord amb les línies de treball suggerides, tot el projecte s'ha desenvolupant emprant programari lliure: Tomcat com a servidor d'aplicacions, Eclipse com a editor i debugger amb el plugin Sysdeo per gestionar Tomcat des d'Eclipse i Open Office per a la documentació.

També ha calgut MySql i el connector JDBC per les proves amb laColla, així com el paquet Jakarta Commons Upload per la gestió de la recepció de fitxers. La versió del SDK de Java utilitzada ha estat j2sdk1.4.2_03

Puntualment s'ha recorregut a eines comercials com ara Microsoft Visio, Microsoft Project o Microsoft Word.

Prestacions implementades

Les prestacions que ofereix a hores d'ara l'aplicatiu, des del punt de vista d'un client web, són les següents:

- Oferir un entorn web similar al BSCW amb creació dinàmica de pàgines web per navegar en una estructura de fitxers i carpetes. Com podeu veure en aquesta captura de pantalla l'entorn de treball és força similar al BSCW, per bé que s'han eliminat les *comboboxes* relatives a prestacions no implementades.

Fig8. screenshot de l'aplicació

- Crear noves carpetes

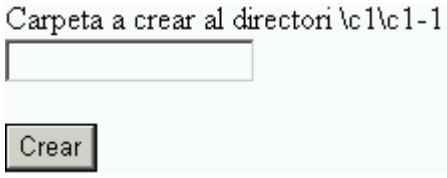

Fig9. formulari de creació de carpetes

- Pujar nous fitxers sense preidentificació del tipus MIME i amb limitació del tamany

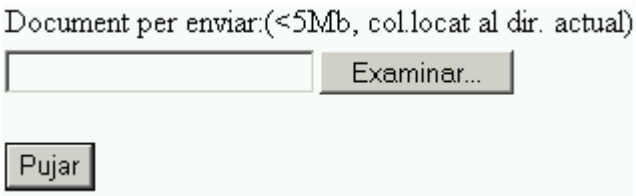

Fig10. formulari per enviar fitxers

- Baixar fitxers amb identificació del contingut MIME i del tamany del fitxer per mostrar la barra de progrés

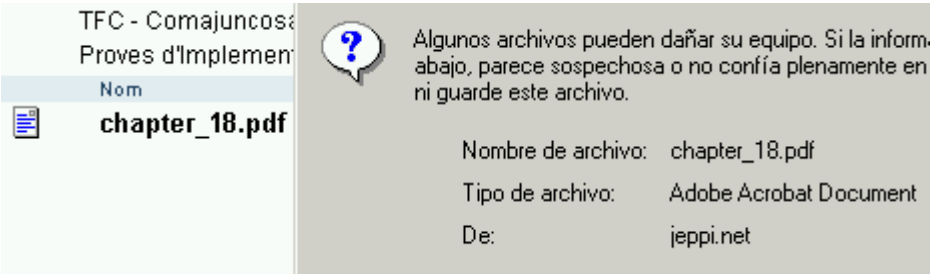

Fig11. identificació tipus MIME

- Navegació amb una barra de directori com al BSCW original (es recomana utilitzar aquesta barra més que no pas els botons del vostre navegador).

Fig12. barra de navegació

A més s'han implementat, entre d'altres, les següents prestacions que redunden en una millor navegació

- Control i estadístiques de sessió amb Cookies, que es mostren en tot moment al capdamunt de la pàgina de moment. Si l'usuari no té les cookies habilitades no es fa reescriptira de URL i per tant ara per ara són necessàries.

- Detecció del tipus de fitxer segons el seu contingut (i no segons l'extensió)

- Codificació de URL's per no tenir problemes amb noms de carpetes i fitxers amb espais

- Altres aspectes de disseny de la web:

    - Icones i fulls d'estil css entregats pel servidor

- Inclusió de documents estàtics a les pàgines ("Copyright" etc, eliminat en les darreres versions)
- refresc automàtic de pàgines

L'eliminació de fitxers i carpetes, una addició trivial, podia presentar problemes de concurrència en entorns distribuïts i per tant s'ha deixat per més endavant quan la implementació de laColla estigui del tot operativa.

Igualment cal advertir que no s'ha previst què fer amb fitxers repetits.

Com veieu només hi ha control de sessió, no autentificació d'usuaris, per tant tothom té lliure accés a l'entorn ara per ara.

Arquitectura general del projecte

Motivacions

Un cop establert que el pla de treball s'encaminaria a crear un *clon* del BSCW que emprés laColla, calia decidir com implementar-ho.

L'opció triada d'emprar un servidor que proveeixi l'entorn gràfic als navegadors dels clients és la més similar al BSCW original, afavoreix una implementació més ràpida i permet aprofundir en la tecnologia de creació d'aplicacions web. Com a contrapartida és segurament menys flexible que crear una aplicació amb un client amb interfície gràfica pròpia, una solució però més costosa i que requereix que els clients tinguin un petit aplicatiu instalat.

La tecnologia emprada per implementar el servidor Web ha estat fonamentalment l'API Servlet de Java.

Capes

L'abast del projecte és prou simple com per permetre una implementació completa del servidor en una sola màquina. Tot i així s'ha dissenyat respectant el model d'una arquitectura de tres capes.

La capa de Dades correspon als mòduls encarregats d'accedir a les dades persistents: fitxers, carpetes i descriptors inicialment.

La capa d'Aplicació és el mòdul que atén les col·licituds del client, que ens arriben com requests GET i POST, i les redirecciona als gestors adequats.

La capa de Presentació és el conjunt de mòduls que generen la interfície d'usuari (pàgines Web).

En rigor, desenvolupar aplicacions web amb servlets no separa prou bé el codi d'interfície (de generació de pàgines) de la lògica de negoci. Una solució millor seria emprar Jsp, que a més permet utilitzar entorns WYSIWYG de disseny de pàgines Web, però no ha semblat necessari per l'abast limitat del projecte.

De totes maneres he intentat separar clarament ambdues funcionalitats, de manera que hi ha servlets especialitzats en la creació de contingut web i servlets que s'encarreguen de dur a terme les tasques pròpiament dites, per exemple en el cas d'ús de creació d'una carpeta el diagrama de col·laboració simplificat seria

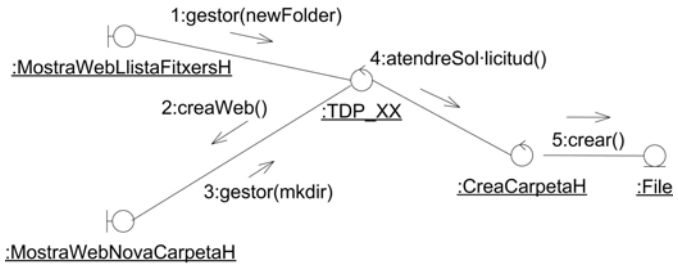

Fig13.creació d'una carpeta (simple)

on TDP_XX és el punt d'entrada a l'aplicatiu (cada versió prototipus s'anomena TDP_00, TDP_01 etc).

Com es pot veure els mòduls que gestionen una sol·licitud (*request handlers*) de generació de GUI, que corresponen a la capa de presentació, serien MostraWebLlistaFitxersH que és la que genera la pàgina de treball amb la relació de fitxers i carpetes i MostraWebNovaCarpetaH que genera una pàgina amb un formulari per enviar al servidor la informació relativa a la carpeta a crear.

Els gestor que accedeix al sistema de fitxers CreaCarpetaH és qui crearà un fitxer de tipus directori en principi (més endavant veurem com aquesta arquitectura és realment més complexa) i és part de la capa de Dades.

Finalment TDP_XX és el gestor que captura les sol·licituds del client i crida a la resta de gestors, i seria de fet qui porta a terme la lògica de negoci.

Redirecció de sol·licituds

L'aplicatiu té un únic punt d'entrada, el servlet TdpXX a on Tomcat dirigeix totes les crides. Aquest servlet primer oferia totes les funcionalitats, després el vaig reescriure perquè, segons els paràmetres de la sol·licitud, la redirigís al servlet adequat.

Actualment implementa un model inspirat en el mecanisme de delegació d'events de AWT que permet ampliar la funcionalitat d'una manera molt més elegant i modular, sense necessitat d'omplir el servlet d'entrada de cadenes de condicionals. L'esquema seria com segueix:

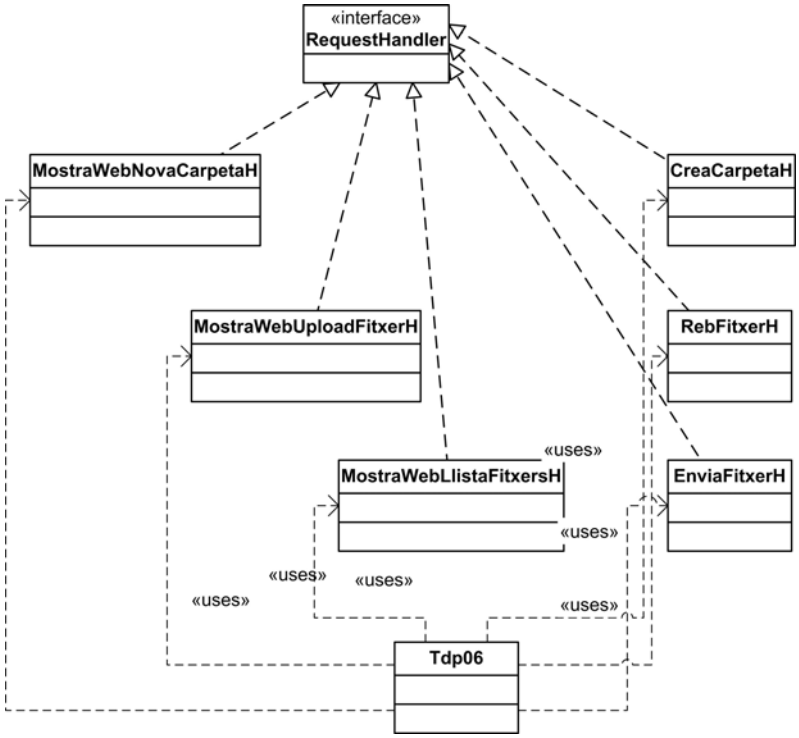

Fig14. UML estàtic - redirecc. requests

És a dir, tots els gestors implementen la interfície RequestHandler que només declara un mètode abstracte, processRequest(), que s'encarrega de gestionar la sol·licitud.

```
public interface RequestHandler
{
      public void processRequest(HttpServletRequest req,
            HttpServletResponse res, Object[] services)
            throws ServletException, IOException;

}
```

El servlet d'entrada TdpXX manté una taula de dispersió amb les referències a instàncies de cadascun d'aquests gestors. Les crides GET o POST dels clients passen la clau que indexa la taula com el valor del paràmeter *handler*, i per polimorfisme (tots els gestors són *RequestHandler*'s) es crida al gestor escaient en cada cas :

```
String handlerName = request.getParameter("handler");
(…)
  RequestHandler handler = getGetRequestHandler(handlerName);
  handler.processRequest(request, response, _services);
```

Gestió de la persistència

Com he esmentat anteriorment l'esquema de funcionament intern d'accés a les dades és realment més complex que l'indicat. El motiu és que des de bon principi em va interessar implementar diversos esquemes de persistència de les dades, tan per poder provar prototips funcionals sense laColla com per, un cop afegida al projecte, comparar el rendiment d'un i altre.

Quan l'emmagatzematge de les dades és heterogeni o, com és el cas, es pretén accedir-hi de formes diverses, és essencial una capa intermèdia que en permeti un accés uniforme i transparent desde l'aplicació.

Hi ha un patró (*pattern*) de disseny específicament concebut per aquesta tasca anomenat Data Access Object (DAO). Un DAO és un component que incorpora una capa intermèdia entre la lògica d'accés a les dades de l'aplicació i els diversos mecanismes de persistència implementats.

D'aquesta manera podem incorporar, provar i migrar a diferents mecanismes de persistència sense haver de modificar el codi que accedeix a les dades. L'avantatge que obtenim minimitzant la transició a un nou esquema de persistència té com a contrapartida la necessitat de reescriure codi repetitiu però m'ha semblat plenament justificat.

La implementació en Java d'aquest patró l'he trobada a javapractices i empra exhaustivament la delegació i el polimorfisme. Aquest seria un esquema de la relació entre les classes involucrades en la *recepció d'un fitxer*

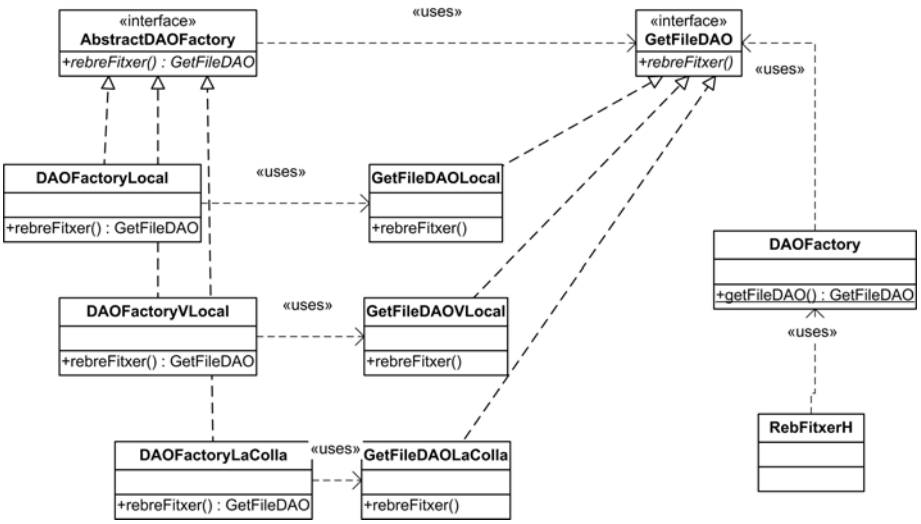

Fig15. UML estàtic - DAO

Podem veure com una interfície declara els mètodes que cal implementar en totes elles, actualment ben senzilla

```
public interface AbstractDAOFactory {
public ListDAO llistarCarpeta();
public MkDirDAO crearCarpeta();
public SendFileDAO enviarFitxer();
public GetFileDAO rebreFitxer();
public GetDirSizeDAO tamanyDir();
public GetFileSizeDAO tamanyFitxer();
}
```

els tres mecanismes de permanència implementats, "local","vlocal" i "lacolla" els explicaré en la secció següent. Per ara mostro com la classe DAOFactory instancia la que cal

```
  private static void initFactory(){
      if ( isStore(fLOCAL) ) {
        fFactory = new DAOFactoryLocal();
      }

      else if ( isStore(fVLOCAL) ){
        fFactory = new DAOFactoryVLocal();
      }

      else if ( isStore(fLACOLLA) ){
        fFactory = new DAOFactoryLaColla();
      }
```

```
  }
```

un exemple d'implementació mostra com la crida a aquests mètodes ens retorna per polimorfisme un objecte que serà pròpiament el que implementa el mètode d'accés a les dades

```
public final class DAOFactoryLocal
implements AbstractDAOFactory{

public ListDAO llistarCarpeta(){
return new ListDAOLocal();
}
(...)
```

per tant des dels servlets una crida a, posem per exemple, la recepció d'un fitxer, requereix instanciar primer l'objecte a través del DAOFactory i tot seguit cridar el mètode que implementa:

```
DAOFactory.sendFileDAO().enviarFitxer(fitxer,out);
```

La implementació dels mètodes d'accés de les classes "vlocal" en molts casos deleguen a les classes "local" la permanència. Els mètodes d'accés emprant laColla deleguen bona part de la gestió a les classes "vlocal" i alguna empra auxiliarment mètodes de "local", com explicaré en la secció següent.

El resultat final és que només modificant un paràmetre del fitxer de configuració web.xml del servidor, aquest funcionarà en el mode de gestió desitjat, sense haver tenir en compte aquest fer en cap lloc del codi, tot es gestiona automàticament.

Ara podem completar el diagrama de col·laboració anterior pel cas d'ús de creació d'una carpeta, emprant per exemple persistència "local"

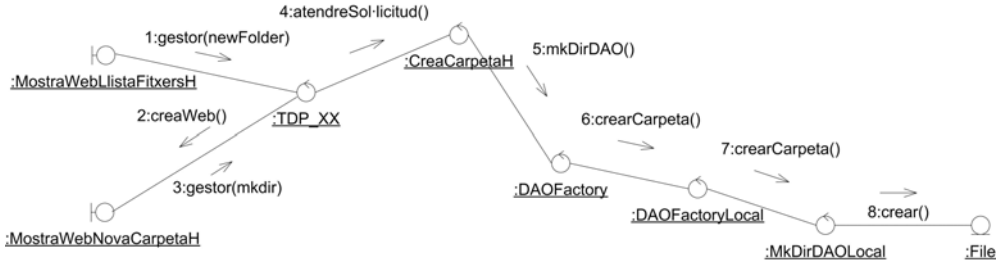

Fig16. creació d'una carpeta (complet)

Cap a una arquitectura distribuïda

Treball en local

Els primers prototipus del projecte implementaven una gestió de persistència simple, emmagatzemant els fitxers i l'estructura de directoris directament en una carpeta del servidor. Aquest és el mecanisme de persistència "local".

En aquest cas tasques com llistar les carpetes, obtenir el tamany d'un fitxer, crear una nova carpeta o un nou fitxer es poden implementar simplement a partir dels mètodes oferts per la classe File de Java.

L'API de laColla i la gestió de la persistència

Actualment l'API de laColla [doc 24.3] ofereix els següents mètodes relacionats amb la gestió de la persistència:

- *putObject()* per col·locar un objecte en laColla
- *getObject()* per obtenir la referència a un objecte de laColla
- *removeObject()* que no està implementat

Com es pot veure les prestacions que en local són trivials d'implementar no estan contemplades en laColla: no podem crear carpetes, només podem enviar i rebre fitxers. No podem reemplaçar fitxers (però sí d'alguna manera actualitzant-ne les referències) ni eliminar-los. I no podem llistar objectes.

Directoris virtuals

Una possible solució a alguna d'aquestes limitacions va ser emprant un descriptor que indiqués "on" està situat cada fitxer dins una estructura jeràrquica de directoris, encara que des del punt de vista de laColla tots els objectes estiguin al mateix nivell.

Aquest mecanisme de persistència pot implementar-se per separat, sense recórrer a laColla de moment: és el mecanisme "vlocal" i el que fem en aquest cas és desar tots els fitxers en una carpeta, i la informació de subcarpetes i fitxers continguts desar-la en un descriptor XML, vdir.xml que es gestiona amb DOM (amb la llibreria jdom.jar que ens proveeix Sun).

En aquest cas hi ha mètodes que només operen sobre el descriptor, com ara crear o anar a una carpeta o obtenir el nombre d'elements que conté, mentre que d'altres com enviar i rebre fitxers requereixen accedir al descriptor i també al sistema de fitxers.

Aquest mecanisme comporta que no poden haver fitxers repetits enlloc, fins i tot si pertanyen a carpetes diferents. Una possible solució seria reanomenar-los amb identificadors únics i mantenir el nom real al descriptor, però per ara no s'ha fet.

Aquest és l'aspecte que té el descriptor amb un fitxer afegit dins la carpeta1

```
<?xml version="1.0" encoding="UTF-8"?>
<virtualDir>
<!--Descriptor estructura fitxers-->
<folder name="carpeta1">
<folder name="carpeta3" />
<file name="100_0231.JPG" />
</folder>
<folder name="carpeta2" />
</virtualDir>
```

Incorporant laColla

Finalment es va procedir a incorporar laColla a l'aplicatiu.

El que fem és implementar un nou mecanisme de persistència "lacolla" que fonamentalment delega tota la gestió en local als serveis de "vlocal". La recepció de fitxers delega en "local" per crear un fitxer temporal amb el qual construïm un ObjecteLaColla que enviem a laColla, i tot seguit l'esborrem. Per localitzar-lo de nou i conèixer el seu tamany sense haver-lo de rebre de laColla, he modificat el descriptor de fitxers afegint dos atributs més a l'element fitxer: l'identificador de laColla i el seu tamany.

L'enviament de fitxers podria fer-se simplement copiant l'inputStream del socket a l'outputStream cap al navegador, o bé passant per memòria o bé passant per fitxer (delegant a "local"). He provat els tres casos, tots ometen part de les dades excepte en fitxers molt petits. Aquest problema se'm presenta de vegades en la recepció també.

En previsió d'una configuració de l'aplicació basada en més d'un servidor simultani, els mètodes que modifiquen el descriptor virtual de directori (creació de carpeta i addició de fitxer) disseminen un esdeveniment a la resta d'aplicacions sol·licitant la recàrrega del descriptor. La classe remota d'implementació, a través del mètode newEvent(), s'encarrega de fer-ho en rebre un esdeveniment apropiat.

Malauradament el meu aplicatiu no sembla rebre esdeveniments externs, o no ho mostra per consola, per tant no s'ha provat aquesta darrera implementació. Igualment, aquest problema de sincronia només és un dels molts possibles escenaris on poden haver conflictes de concurrència, que per manca de temps no s'han pogut provar.

Capítol 4

CONCLUSIONS

Valoració dels resultats

Possibles escenaris de treball

L'aplicatiu obtingut permet diverses possibilitats d'ús, especialment si contemplem la gestió distribuïda amb laColla.

- Servidor centralitzat i múltiples clients Web : segueix el paradigma d'un servidor de recursos on el mateix servidor proveeix la interfície gràfica i els clients poden accedir als seus serveis amb un simple navegador. El servidor en sí pot implementar qualsevol dels mecanismes de persistència analitzats.

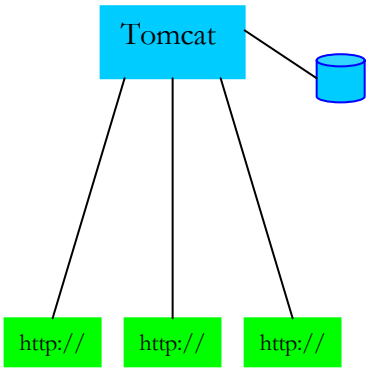

Fig17. servidor i múltiples clients

- Múltiples servidors amb laColla i múltiples clients Web : en aquest cas el client es connecta a algun dels servidors que té instalat l'aplicatiu. Cada aplicatiu instancia un RA, una UA i un GAPA de laColla que es coordinen entre ells oferint una persistència distribuïda transparent al client. Els aplicatius *han de mantenir la sincronia entre descriptors virtuals*, a base de disseminació de missatges a nivell d'aplicació. No s'ha pogut provar però està implementat.

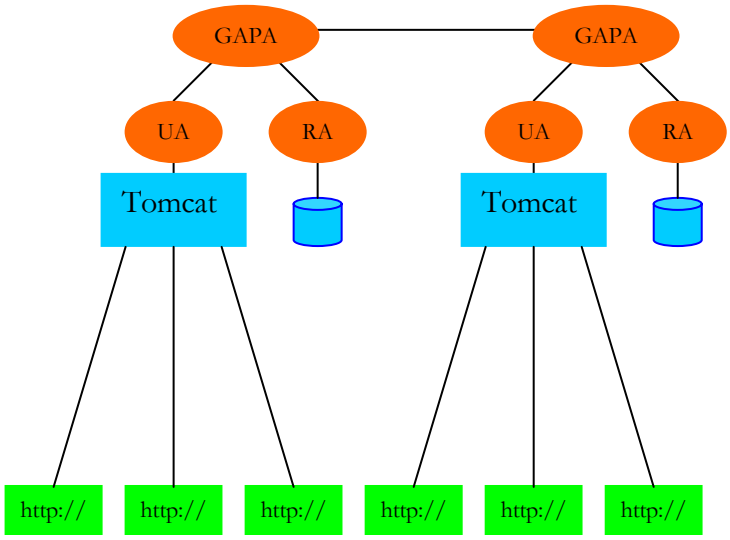

Fig18. múltiples servidors i clients

- Un servidor amb l'aplicatiu i múltiples màquines amb laColla, i múltiples clients Web : en aquest cas tots els clients es connecten a un únic servidor però la persistència està igualment distribuïda amb els serveis oferts per laColla en diferents màquines. Com tenim un únim descriptor virtual no hi ha problemes de sincronia. Aquesta és la versió provada que s'entrega.

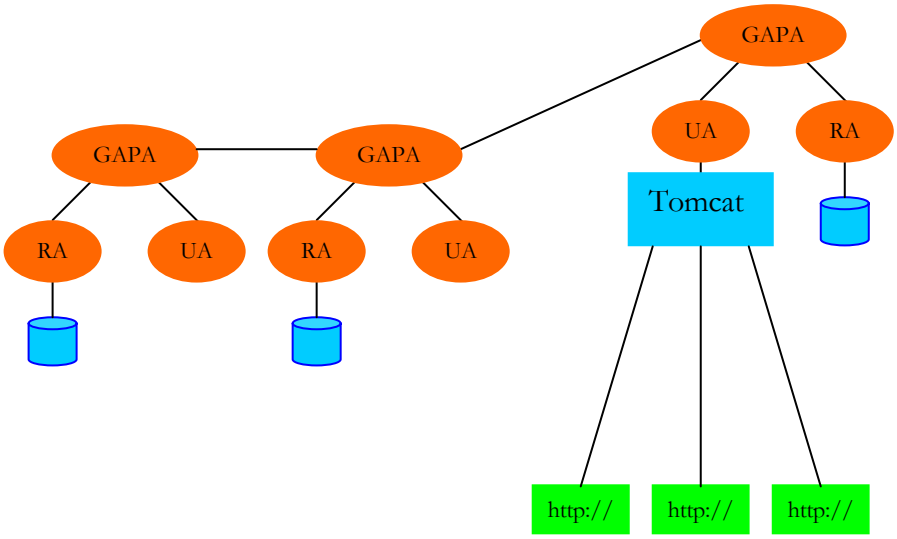

Fig19. laColla amb servidor i clients

BSCDW v BSCW

Evidentment en l'estat actual no és possible cap comparació amb el BSCW original, ja que les prestacions són molt reduïdes.

De totes maneres aventuro que una implementació més completa i que empri laColla pot ser equiparable en prestacions a l'original i més robust a fallades i sobrecàrrega.

Fins i tot ara ja hi ha incorporades millores que en el BSCW que he emprat de model no hi són: codificació de URL's per evitar que noms amb caràcters no anglòfons quedin alterats en enviar-se a través de HTML i reconeixement automàtic d'alguns dels formats de fitxers més coneguts (el BSCW requereix indicar-ho manualment i sino ho delega al navegador).

laColla

Alguns comentaris respecte al desenvolupament d'aplicacions amb laColla

Operativa

- Les darreres versions en jar són molt manejables ja, i la seva instal.lació molt més simple. Tot i així és la única part del meu aplicatiu que requereix instal.lar-se per separat de la resta ja que cal MySql, crear la BBDD lacolla i instanciar una o més lacolla manualment en consoles separades.
- Quan es van realitzant proves cal anar buidant o recreant la BBDD (és ràpid recrear-la encara que no sigui necessari). És engorrós però segurament molt més centralitzat i còmode que si calgués reemplaçar o esborrar fitxers de text arreu.
- Finalment, les consoles de laColla mostren tal quantitat d'informació que cal molta pràctica per anar seguint el que succeeix. Però de fet el mateix succeeix amb la meva aplicació, és molt informatiu pel desenvolupador.

Xarxa

- Un middleware hauria d'amagar a l'aplicació la complexitat de la xarxa. Els fitxers de configuració i l'ús extensiu de ports vulnera en part aquest requisit desitjable. A més no està documentat.

- Una eina per crear fitxers de configuració seria bona idea, però idealment no s'hauria de demanar explícitament la configuració de ports de cada UA, RA o GAPA.

API

- Poc documentada: es llisten els atributs dels mètodes però no s'expliquen i en algun cas seria desitjable. A aquest nivell de prototipus fins i tot seria interessant afegir comentaris de funcionament intern del tipus "aquest Event només requereix tenir instanciats els atributs…" o "per localitzar un Objecte, laColla empra l'atribut…".
- Algun mètode documentat no existeix: removeObject()
- Requereix accedir a estructures de dades internes (Event, Timestamp, ObjectLaColla) i per tant al codi font. Seria preferible prescindir-ne i treballar només indirectament amb identificadors.
- Implementació de la recepció d'events: no documentat. Enlloc s'explica quins atributs d'un Event són modificats per laColla durant el trànsit entre UA,RA i GAPA's i per tant no podem emprar com identificadors.

Latència

- LaColla ralentitza enormement l'accés a recursos, crec que perquè molts threads interns inclouen tems d'espera d'alguns segons per assegurar-se la sincronització i disponibilitat. Per exemple si intentem rebre un fitxer que acabem d'enviar cal esperar força o es reb incomplet. Potser els mecanismes interns de transferència de laColla no són molt eficients.

Futures línies

Sens dubte el projecte d'un clon del BSCW és molt ambiciós i aquest projecte només en suposa l'inici. He procurat que el disseny de les prestacions implementades faciliti ulteriors millores que podrien anar encaminades a implementar més prestacions "bàsiques" del BSCW o una millor robustesa de l'aplicació:

- implementació completa de les prestacions bàsiques amb laColla
- gestió de fitxers repetits
- prestacions addicionals de fàcil implementació:
    - esborrar carpetes i fitxers
    - treball sobre múltiples carpetes o fitxers
    - pujar més d'un fitxer a la vegada
    - estadístiques de sessió més complertes
    - tancar i obrir manualment una sessió
    - codificar URL's si l'usuari té les cookies desactivades
    - millorar una mica la navegació i el look d'alguna pàgina
- Prestacions més complexes:
    - gestió d'usuaris
    - navegació segura
    - funcions que l'API de laColla ja proveeix: conèixer els usuaris connectats i afegir funcions bàsiques de missatgeria instantània

I ja queda per Projectes amb un abast molt superior prestacions del BSCW com els repports automàtics per e-mail.

Valoració personal

El desenvolupament d'aquest Projecte ha estat per mi molt fructífer encara que sovint en aspectes que no preveia. Realment l'aspecte que ha quedat més a darrer terme tant pels problemes esmentats com pel volum de feina prèvia que calia és el referent als sistemes distribuïts en sí.

En canvi he pogut aprofundir força en el desenvolupament d'aplicacions web, ara només em falta familiaritzar-me amb Jsp. Aquestes tecnologies són immediatament aplicables en molts entorns professionals, de fet possiblement les empraré gràcies a haver-les emprat en el projecte.

De totes maneres l'aspecte més satisfactori per mi ha estat poder agafar soltura en la programació desde zero d'un projecte relativament extens, emprant i reutilitzant llibreries i patrons de disseny de terceres parts, sovint més d'un a mesura que els requisits canviaven (com la llibreria per rebre fitxers, primer de O'Reilly i després de Jakarta). A més d'adquirir molta seguretat com a programador hi ha l'oportunitat d'aprendre tecnologies noves (com XML/DOM, que ha estat molt engrescador) i aspectes de programació avançada (com els DAO).

L'únic aspecte que considero millorable és l'eficiència en la implementació de laColla. M'ha suposat molta feina en configuració i proves que probablement no es vegi reflectida en els resultats d'aquest projecte.

Sincerament per mi ha estat una bona síntesi d'assignatures com Enginyeria de Programari i Tècniques de Desenvolupament de Programari, però amb l'estímul de ser més lliure i variat.

Finalment, agraeixo a en Joan Manuel Marquès el seguiment i l'optimisme que ha mantingut al llarg de tot el quatrimestre, i a en Xavier Vilajoana la paciència i dedicació amb les qüestions que anava formulant relatives a aspectes d'implementació de laColla.

BIBLIOGRAFÍA

Instal·lació

- Setup de Tomcat: http://mdsh.com/notes/tomcat5OnWindowsNT.jsp
- Setup de l'Eclipse: http://www.purposesolutions.com/Resources/EclipseJ2EE.html
- Setup de Sysdeo: http://www.eclipsetotale.com/articles/tomcat/tomcatPluginDocFR.html
- Tomcat amb Eclipse i Mysql: http://confluence.atlassian.com/display/DOC/Configuring+a+MySQL+Datasource+in+Tomcat

Reutilització de components

Upload de fitxers

- Relació de solucions per pujar fitxer: http://www.magelang.com/faq/view.jsp?EID=160&page=2
- Solució avançada per pujar fitxers i enviar reports per MAIL: http://www.oop-reserch.com/mime_prop.html
- Altres possibilitats (no s'han provat): http://java.freehosting.co.kr/opensrc/upload/
- Identificació de formats de fitxers pel contingut: http://www.geocities.com/marcoschmidt.geo/java-file-format-identification.html
- http://hul.harvard.edu/jhove/ (no s'ha emprat)

Patrons de disseny

- Patrons de disseny per Java (inclou el patró DAO) : http://www.javapractices.com/index.cjp
- Patró inspirat en delegació d'events d'AWT per atendre sol·licituds : http://www.javaworld.com/javaworld/jw-09-1999/jw-09-servlet.html

Aplicacions Web

Arquitectura

- MVC (no s'ha emprat): http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e/web-tier/web-tier5.html
- Com customitzar el descriptor web.xml: http://e-docs.bea.com/wls/docs61/webapp/web_xml.html
- Informació diversa, i sobre streams d'entrada i de sortida en Java: http://members.chello.nl/karin2/index.htm

Documentació referent a XML/DOM

Tutorials d'XML

- http://java.sun.com/xml/jaxp/dist/1.1/docs/tutorial/TOC.html
- http://www.ibiblio.org/xml/slides/

Tutorials de DOM i JDOM

- http://www.devguru.com/Technologies/xmldom/quickref/xmldom_intro.html
- http://www.w3schools.com/dom/default.asp
- James Britt, *Interactive Java & JDOM online tutorial*, disponible a http://www.topxml.com/tutorials

Servlets

- Documentació oficial a J2EE Java Servlet Technology http://java.sun.com/products/servlet/

Aplicacions web i Servlets

- http://www.acm.org/crossroads/xrds8-2/servletsProgramming.html
- http://java.sun.com/products/servlet/whitepaper.html
- *deployment*:http://jakarta.apache.org/tomcat/tomcat-4.1-doc/appdev/deployment.html

Més documentació:

- Prentice Hall and Sun Microsystems: *Servlet and JSP Quick Referencedisponible a www.coreservlets.com (no és gratuït)*
- Bruce W. Perry: *Java Servlet & JSP Cookbook*, O'Reilly, 2004
- The O'Reilly Java Authors : *Enterprise Best Practice,* O'Reilly , 2002
- Bruce W. Perry: *Java Servlet & JSP Cookbook,* O'Reilly , 2004

Tutorials

- http://java.sun.com/webservices/docs/1.0/tutorial/doc/JavaWSTutorialTOC.html
- http://www.javaskyline.com/learnservlets.html
- Alguns aspectes bàsics com el pas de paràmetres des d'una URL
- http://stage.itp.nyu.edu/danclass/ICM/FILES/Servlets/SevletsFromBrowser/icmhandout7.html
- Pràctica sobre servlets a la FIB de la UPC: http://docencia.ac.upc.es/FIB/AAD/lab/index.html

Entorns col·laboratius

- http://www-clips.imag.fr/calie04/actes/Buffa_final.pdf
- http://www.site.uottawa.ca/~adler/publications/2004/adler-nash-noel-2004-challenges-in-collaborative.pdf
- Un projecte molt similar al meu en alguns aspectes: http://motu.sourceforge.net/

BSCW

- *Jonathan Trevor, Richard Bentley and Gerrit Wildgruber* Exorcising daemons: A modular and lightweight approach to deploying applications on the Web de *disponible a:*

http://bscw.gmd.de/Papers/WWW5-Paris/WWW5-Paris.html

Arquitectures C/S de tres capes

- http://www-db.stanford.edu/~burback/dadl/node1.html
- http://www.sei.cmu.edu/str/descriptions/threetier.html
- http://members.aol.com/kgb1001001/Articles/threetier/threetier.htm

Tecnologies de distribució d'objectes

- Marquès i Puig , Joan Manuel: *LaCOLLA: una infraestructura autònoma i autoorganitzada per facilitar la col·laboració* disponible a http://people.ac.upc.es/marques/LaCOLLA/
- http://www.sei.cmu.edu/str/descriptions/clientserver_body.html
- http://www.cs.ucl.ac.uk/staff/W.Emmerich/lectures/3C05-03-04/Middleware.pdf

ÍNDEX

aplicació web
  estructura, 13
assíncrona
  comunicació en entorns col·laboratius, 8
AWT, 21
BSCDW, 3
BSCW, 3, 9
  arquitectura en capes, 10
  avantatges en entorns de treball col·laboratiu, 8
  barra de navegació, 18
  i el meu projecte, 32
  i l'entorn del projecte, 16
  referent del projecte, 4
C/S
  definició projecte, 3
  en aplicacions web, 12
capes
  arquitectura de tres capes, 10
CGI
  definició projecte, 3
  desavantatges, 12
  dins el BSCW, 9
  en aplicacions web, 12
  front-end de twiki, 9
Client/Servidor
  definició, 10
Cookies, 18
CVS, 8
  motor intern de Twiki, 8
DAO
  definició, 23
DOM, 27
Eclipse
  programari del projecte, 16
entorns col·laboratius
  requisits de programari, 8
HTML
  capa de presentació del BSCW, 10
  codificació de URL's, 32
  en aplicacions web, 12
Jakarta Commons Upload
  programari del projecte, 16
JDBC
  programari del projecte, 16
Jsp, 20
laColla
  anàlisi critica, 32
  definició projecte, 3
  delegació de la persistència, 25
  escenaris, 30
  implementació, 27
  implicacions en la temporització del projecte, 4
  mètodes rellevants de l'API, 26
  programari del projecte, 16
  projecte i configuració, 5
  tecnologia de tres capes, 11
local
  persistència, 26
MIME
  baixada de fitxers, 17
  pujada de fitxers, 17
MySql
  i instal·lació de l'aplicatiu del projecte, 32
  programari del projecte, 16
O'Reilly, 34
Open Office
  programari del projecte, 16
Phyton
  dins el BSCW, 10
prototipus
  disponibilitat, 5
Servlet
  en aplicacions web, 12
  enfront de GUI de client, 19
Servlets
  avantatges, 13
  projecte, 3
síncrona
  comunicació en entorns col·laboratius, 8
Sysdeo
  programari del projecte, 16
Tomcat
  programari del projecte, 16
Twiki, 8
vlocal
  persistència, 27
WAR
  i aplicacions web, 14
XML, 27