CONTENIDOR DE SERVLETS DISTRIBUÏTS

Universitat Oberta de Catalunya

Treball Fi de Carrera dels estudis de l'Enginyeria Informàtica

Josep M. Miró Moreno

Índex

1- Presentació del problema i entorn en el que s'enmarca.

1.1- Introducció:

El desenvolupament i popularització d'internet i dels seus protocols de comunicació, juntament amb la disminució del cost d'accés, ha fet que hi hagi molts ordinadors connectats les 24 hores del dia a la xarxa.

Cada cop hi ha més informació dins la xarxa. Empreses i particulars la utilitzen per facilitar informació a altres persones, o simplement com a medi de transmissió de dades entre diferents ubicacions distants geogràficament.

Un dels mètodes més utilitzats per la presentació de les dades als usuaris dins d'internet son els servidors web.

El funcionament dels servidors web a grans trets és el següent: Un usuari es connecta a una adreça de xarxa en la que hi ha definit el protocol de comunicació acceptat pel servidor web. El servidor, al rebre la petició, busca la informació que se li ha sol·licitat, i retorna a l'usuari la informació demanada. Llavors el navegador interpreta aquesta informació i la mostra per pantalla, per a que sigui comprensible per l' usuari.

En aquest projecte s'estudiarà que succeeix en cas que el servidor no tingui la informació tenint en conte que potser altres servidors que pertanyen a la mateixa empresa, o grup d'usuaris la tenen dins seu, i com resoldre el problema de l'accés a aquesta informació de manera que sigui transparent a l'usuari.

1.2- Entorn del treball

El projecte de final de carrera que presento s'enmarca dins l'àrea de recerca d'aplicacions i sistemes distribuïts. L'aplicació que he construït és totalment descentralitzada.

Per al desenvolupament d'aquesta aplicació serà necessari una aplicació que sigui capaç de transmetre les dades entre els diferents servidors, un servidor, i un entorn de desenvolupament per a la nostra aplicació.

2- Introducció de les diferents tecnologies que intervenen en el projecte

Com s'ha parlat en el capítol anterior, presentaré una introducció de les diferents tecnologies que he utilitzat per al desenvolupament del projecte:

2.1- Introducció: Topologia dels sistemes distribuïts

Les xarxes d'ordinadors com internet poden ésser considerats com a sistema informàtic descentralitzat [Joan M. Marquès].

Els diferents components de les xarxes poden relacionar-se entre sí de diferents maneres [Minar 2001] segons els fluxos d'informació que s'intercanvien.

2.1.1.- Topologies bàsiques:

a) *Centralitzada:* És la topologia més comú, i la que utilitzen les arquitectures client-servidor, que veurem més tard. Hi ha però algunes aplicacions que utilitzen aquest tipus de serveis. Son les anomenades xarxes d'igual a igual híbrides.
Com a avantatges destaquen la simplicitat d'administració, i la consistència de dades, com a desavantatge tenim la vulnerabilitat a les errades, i el coll d'ampolla que es pot formar quant hi ha molts clients que volen accedir al servidor.

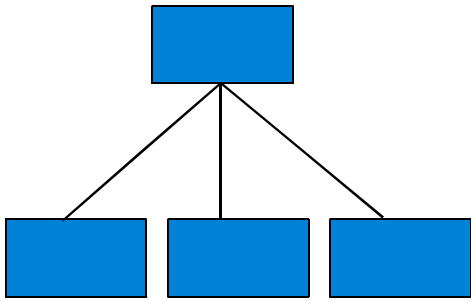

fig. 2.1 Esquema d'una topologia centralitzada

b) *En anell:* Aquesta topologia s'utilitza per solucionar els problemes de saturació del servidor. Consisteix en tenir un conjunt de servidors connectats entre si de manera que es pugui balancejar la càrrega i obtenir un millor comportament davant les errades.

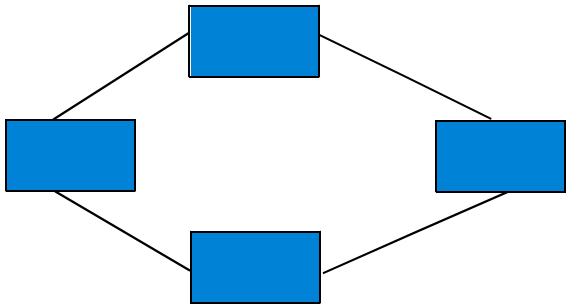

fig. 2.2 Esquema d'una topologia en anell

c) *Jeràrquica:* La topologia jeràrquica es basa en que els nodes estan connectats de manera que el conjunt de nodes superior conté la informació que el node que està sota ha de demanar sempre al superior.

Els sistemes jeràrquics tenen com avantatge l'escalabilitat, ja que es poden afegir nodes a qualsevol punt de la jerarquia per cobrir necessitats de càrrega. Té un comportament més bo davant a fallades, però l'arrel continua sent un punt crític en cas de fallada.

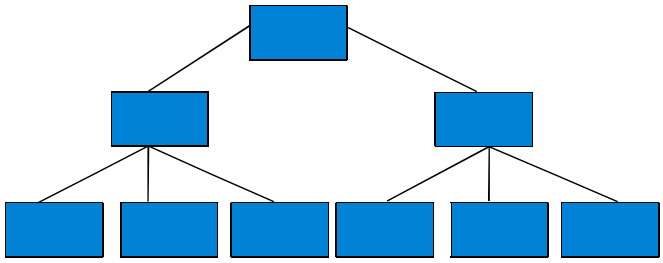

fig. 2.3 Esquema d'una topologia jeràrquica

d) *Descentralitzada:* En aquesta topologia un node qualsevol es connecta a qualsevol altre node. Aquesta topologia la fan servir molts dels sistemes de p2p. Les virtuts d'aquesta arquitectura són la facilitat d'afegir un node al sistema i la tolerància a fallades d'un dels nodes.

Com a dificultat tenim que aquests sistemes acostumen a ser insegurs així com la seva dificultat per administrar-los.

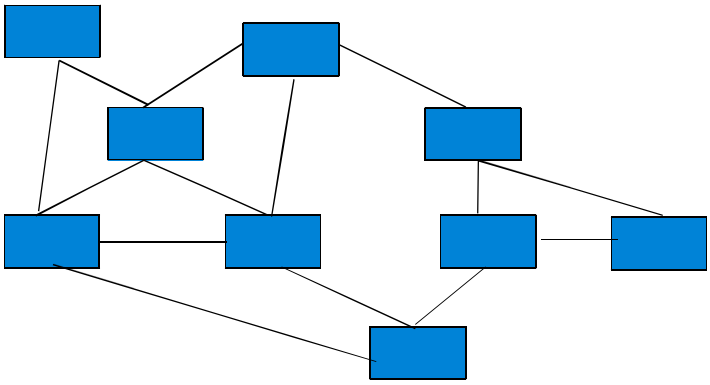

fig. 2.4 Esquema d'una topologia descentralitzada

2.1.2.- Topologies híbrides:

Quant és dissenya un sistema distribuït s'utilitzen combinacions de les topologies que hem vist. Seguidament en veure'm alguna mostra:

a) *Centralitzada + anell:* Aquesta topologia per l'usuari es comporta com si fos un sistema centralitzat, però el servidor no és únic sino que està format per una anella de servidors.

Amb aquesta topologia l'usuari veu un sol servidor, però obtè una major robustesa, escalabilitat i tolerància a fallades.

És molt usada per els servidors web comercials i bases de dades.

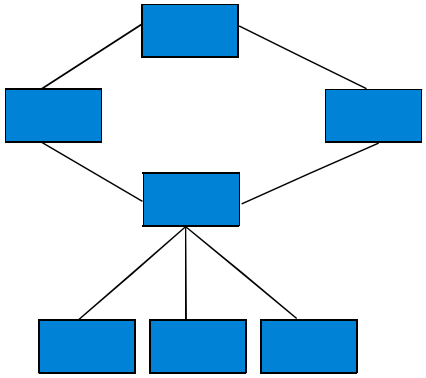

fig. 2.5 Esquema d'una topologia centralitzada+anell

b) *Centralitzada + descentralitzada:* Aquesta topologia híbrida s'utilitza perquè hi ha cops que els sistemes centralitzats purs tenen molta dificultat per a que qualsevol node es pugui relacionar-se amb qualsevol altre node. En aquest cas és crea uns nodes especialitzats que únicament serveixen per interrelacionar grups de nodes que estan agrupats de manera centralitzada. L'exemple més conegut és KaZaA.

Aquests sistemes mantenen els avantatges dels sistemes descentralitzats. Per altra banda segueixen éssent importants les dificultats dels sistemes descentralitzats, com la dificultat d'administració i la seguretat.

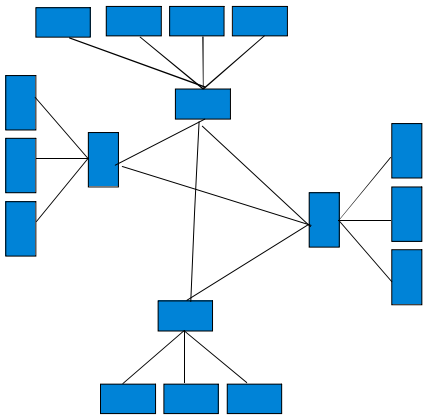

fig. 2.6 Esquema d'una topologia centralitza + descentralitzada

c) *Jerarquica+anell:* En aquest tipus de topologia es poden afegir nodes en qualsevol punt de la jerarquia, i alhora podem balancejar la càrrega amb els avantatges de tenir una anella, de manera que si falla un servidor de l'arrel podem usar un altre servidor sense que hi hagi cap fallada en el sistema.

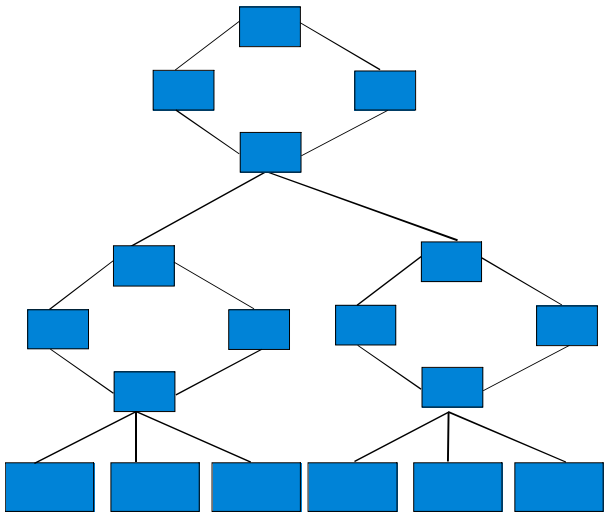

fig. 2.7 Esquema d'una topologia jeràrquica+ anell

2.2– Arquitectura d'igual a igual (peer-to-peer):

2.2.1- Introducció a l'arquitectura d'igual a igual

Internet ha fet que la informació estigui disseminada en la xarxa. L'arquitectura d'igual a igual s'ha popularitzat força sobretot amb el naixement d'aplicacions de compartició de fitxers com Napster [Shirky 2001], i més tard amb l'aparició d'e-mule, KaZa, Gnutella, o Morpheus.

La tecnologia d'igual a igual, es basa en que la informació no està centralitzada. Cada un dels usuaris pot cedir informació a altres participants, o bé rebre informació de la resta.

L'arquitectura d'igual a igual pot localitzar on estan ubicats els altres membres independentment de la seva adreça de xarxa, i és capaç de tornar-los a localitzar fins i tot si canvien d'adreça.

Hi ha dos models clarament diferenciats de xarxes d'igual a igual:

a) *Híbrids:* Alguns nodes de la xarxa són terminals i alhora encaminadors, és a dir faciliten la interconnexió entre els diferents membres de la xarxa.

b) *Pures:* Tots els nodes son usuaris de la xarxa, i cada un d'aquests usuaris pot funcionar com a encaminador, client o servidor de dades.

L'arquitectura d'igual a igual és una estructura auto-organitzativa, és a dir, a l'afegir un node dins la xarxa no s'ha de fer cap modificació en els altres nodes que la integren.

A gran escala una comunitat d'arquitectura d'igual a igual es pot organitzar en milions de xarxes virtuals agrupades en petits grups segons interessos específics.

Una altra de les aportacions de les xarxes d'igual a igual es que minimitzen la congestió ja que les connexions es produeixen punt a punt i no existeixen colls d' ampolla, com pot passar dins l'arquitectura client servidor (que comentarem més tard).

Un dels problemes d'aquesta arquitectura és la seguretat. Ja que al no tenir cap servidor centralitzat resulta molt difícil establir sistemes de seguretat complexos.

2.2.2- El concepte de grup d'usuaris en les xarxes d'igual a igual

Fins ara hem vist el concepte de xarxa peer-to-peer (p2p) o també anomenades d'igual a igual, més popular. Però a partir d'una xarxa p2p la podem dividir en grups d'usuaris tancats. És a dir, podem crear una xarxa d'usuaris que tinguin un interès en comú, i que simplement es vulguin tancar a compartir informació entre ells sense que cap altra persona hi pugui accedir. És aquí on neix el concepte de grups d'usuaris dins una xarxa p2p.

En aquestes xarxes d'usuaris tancats s'ha de tenir molt en compte el concepte de seguretat, si volem que ningú més que un grup d'usuaris determinat hi pugui accedir.

2.2.3- LaColla, una tecnologia que facilita el desenvolupament d'aplicacions d'igual a igual.

Per al desenvolupament del projecte hem utilitzat aquesta eina ja que ens facilita la gestió dels membres que estan dispersos per internet, i alhora ens permet la creació de diferents grups d'usuaris tancats.

LaColla està essent desenvolupada, en java, per en Joan Manuel Marquès i Puig amb col·laboració d'en Xavier Vilajosana.

LaColla és un conjunt de classes que facilita molt la tasca de la creació i gestió d'una xarxa d'igual a igual.

En aquest projecte s'ha usat laColla, perquè sigui l'encarregada de la gestió dels usuaris i per al transport de dades entre aquests.

2.2.4- JXTA, una comunitat oberta i distribuïda.

El projecte JXTA[5], es va començar a desenvolupar per Sun microsystems, i consisteix en realitzar una aplicació capaç d'adreçar diferents pears dins d'un espai d'intercanvi d'igual a igual.

Aquesta tecnologia és un conjunt de protocols p2p que permeten que qualsevol aparell que sigui connectable a la xarxa (teléfon mòbil, pda, pc), sigui capaç de comunicar-se i col·laborar.

En aquest projecte hi col·laboren moltes empreses, tals com: 312 Inc, Alberg, BBN Technologies, Brevient Technologies Inc. Clay International, Codefarm, Digital Dream, Global InfoTek Inc, InfoGlyptic Software, Jet Propulsion Laboratory (JPL) , Morgan Spenser Consulting, National Association of Convenience Stores (NACS), National Association of Realtors, neofonie, Neunet Solutions, Nokia, Oculus Technologies, Peercom, Sankhya, SparkNet S.A., Siemens, Tryllian, VistaPortal Software, Zudha Information Technology

I també universitats tals com: Cardiff University, Canada's Michael Smith Genome Sciences Centre / University of British Columbia, Compute Power Market Project , Ecole Polytechnique Federale de Lausanne, Darmstadt University of Technology, IT Transfer Office (ITO), Hogeschool Limburg, Indiana University - Community Grids Lab, IRISA & ENS, PARIS Research Group - JuxMem Project, National Institute of Advanced Industrial Science and Technology (AIST), Grid Technology Research Center (GTRC), Paderborn University, Germany, Politecnico di Torino, Rutgers University, The Applied Software Systems Laboratory, Stanford University, California,

Stanford University - Edutella project, Trinity College, Dublin, Universidade Federal de Campina Grande, University of Bologna, Italy, University of Bremen, Germany, University of Georgia, University of Glasgow, University of Hamburg, University of Malta, Malta, University of Missouri, Kansas City (UMKC), Università degli Studi di Parma, Universitat Politecnica de Catalunya, University of Southampton (UK), Waseda University, Koyanagi laboratory

I també bastants desenvolupadors individuals .

La idea del JXTA és la mateixa que laColla, que consisteix en dissenyar un sistema d'accés estàndard als pears, de manera que a partir d'aquest sistema es vagin creant diferents aplicacions pel sistema.

2.3– Arquitectura client-servidor

2.3.1- Introducció a l'arquitectura client-servidor.

L'arquitectura client–servidor és un model per al desenvolupament de sistemes d'informació en el que cada un dels usuaris, anomenat client, inicia un procés de diàleg: Produeix una demanda de informació o una sol·licitud de recursos. L'ordinador que respon a la demanda del client se'l coneix amb el nom de servidor.

Sota aquest model cada usuari pot obtenir la informació que requereixi en un moment donat, provenint d'una o varies fonts segons convingui.

Els clients i els servidors poden estar connectats a una xarxa local o a una xarxa més àmplia com la que pot tenir una empresa o una xarxa encara més gran com és internet.

El model client/servidor és, encara ara, el model d'interacció més comú en les aplicacions de xarxa.

Entre les principals característiques de l'arquitectura client-servidor, podem destacar les següents:

a) El servidor presenta a tots els clients una interfície única i ben definida.

b) El client no necessita conèixer la lògica del servidor, només la seva interfície externa.

c) El client no depèn de la ubicació física del servidor ni del tipus d'equip físic on es troba, ni del sistema operatiu.

d) Els canvis en el servidor impliquen pocs o cap canvi en el client.

Tots els sistemes desenvolupats en l'arquitectura client-servidor posseeixen les següents característiques distintives d'altres formes d'aplicacions distribuïdes.

a) Serveis. El servidor és un proveïdor de serveis; el client és un consumidor de serveis.
b) Recursos compartits Un servidor pot atendre a molts clients al mateix temps.
c) És un protocol asimètric. Els clients accedeixen als servidors alhora, mentre que el servidor espera de manera passiva la sol·licitud de serveis dels clients.
d) Intercanvi basat amb missatges Els sistemes interactuen a través d'un mecanisme de transcripció de missatges.

En definitiva, l'arquitectura client-servidor és una infrastructura versàtil, modular, i basada en missatges, que permet millorar la portabilitat, la interoperatibilitat y la escalabilitat.

2.3.2- Arquitectura client-servidor dins el món d'internet

Dins del món d'internet l'arquitectura client/servidor és una de les que s'usa més habitualment. En aquest apartat voldria fer simplement un èmfasis especial en algunes de les tasques més habituals que fem i que comporten la utilització d'aquesta arquitectura.

a) Accés al correu electrònic: Demanem al servidor que ens retorni els correus nous, o bé li enviem els correus per a que siguin enviats al seu destí.
b) Aplicacions de transferència de fitxers (ftp): Tenim un servidor que té els fitxers que nosaltres necessitem, i nosaltres els descarreguem d'allí, o bé ens permet deixar els fitxers dins el servidor.

c) Accés a pàgines web, dinàmiques i estàtiques: Cada cop que demanem una pàgina web dins d'internet hem de connectar amb un servidor, que és l'encarregat de buscar la informació dins seu i enviar-nos-la de manera que siguem capaços de presentar-la per pantalla mitjançant una aplicació que sigui capaç d'interpretar-la (navegador).

2.3.3- Contenidors de servlets.

En l'últim punt de l'apartat anterior s'ha parlat sobre l'accés a les pàgines web. Les pàgines web poden ser dinàmiques o estàtiques.

Les pàgines web estàtiques son aquelles pàgines que cada cop que hi accedim no canvien, a menys que el programador de les mateixes les canviï manualment.

Les pàgines web dinàmiques son aquelles pàgines que canvien, per exemple fan una consulta a una base de dades, i ens mostren el resultat per pantalla. Aquest és diferent cada cop que es fa una consulta.

Un cop havent explicat els tipus de pàgines web, veurem què són els servlets.

Els servlets i java server pages (JSPs) són dos mètodes de creació de pàgines web dinàmiques en el servidor usant el llenguatge java. En aquest sentit son similars a altres mètodes o llenguatges (PHP, CGIs...)

Els JSPs i els servlets s'executen en una màquina virtual de java, el qual permet que en principi, es puguin executar en gairebé qualsevol ordinador sempre que hi existeixi una màquina virtual de java.

Cada servlet o JSP s'executa constantment dins el seu propi context, i no és comença a executar cada cop que es rep una petició, sinó que persisteix d'una petició a la següent de manera que no és perd el temps al invocar-lo (Carregar el programa+l'interpret). La seva persistència li permet gestionar d'una manera més eficient la connexió a les bases de dades, i el maneig de sessions, per exemple.

Diferències entre un servlet i un JSP

Els JSPs són en realitat servlets, però en un JSP és fa la compilació d'un programa java el primer cop que s'invoca, i un cop compilat el programa en java crea una classe que és comença a executar en el servidor com un servlet. La principal diferència entre els servlets i els JSPs és la manera en que s'han programat. Un JSP és una pàgina web, amb etiquetes especials i codi java incrustat, mentre que un servlet és un programa que rep peticions i genera a partir d'elles una pàgina web.

Ambdós, servlets i JSPs, necessiten una aplicació que els contingui i sigui el que enviï les pàgines web al servidor, rebi les peticions i les distribueixi entre els servlets i porti a terme totes les tasques de gestió pròpies d'un servidor web.

Mentre que hi ha servidors pensats per a pàgines estàtiques CGIs i programes executats pel servidor, tal com PHP. Hi ha altres servidors específics per a servlets i pàgines JSPs anomenats Contenidors de Servlets (ServletContainers) o servlet engine.

2.4– La tecnologia XML

2.4.1- Introducció a l'xml

El llenguatge de marcatge extensible (xml) proporciona una manera d'escriure dades estructurades.

L'xml utilitza un conjunt d'etiquetes per a definir elements de dades. Cada element encapsula una part de dades, que pot ser molt simple o molt complexa, i permet definir un conjunt il·limitat d'etiquetes xml. Per exemple podria

definir etiquetes xml per declarar parts de dades d'una comanda, com el preu, els impostos o les dades de la comanda.

A mesura que s'utilitzen les etiquetes en tota una organització i també entre diferents organitzacions, totes les dades és poden manipular i intercanviar d'una manera molt senzilla.

L'XML és una plataforma molt senzilla, independent i un estàndard molt usat.

Com hem dit abans l'xml és una forma molt senzilla de transferir dades. A continuació presentem alguns exemples on és pot usar l'xml.

a- Un document normal.

b- Un registre estructurat, com un registre de cites, o comandes.

c- Aplicacions web o internet que mouen dades.

d- Un objecte amb dades .

e- Un registre de dades, com el conjunt de resultats d'una consulta a una base de dades.

f- Codi C# que és pot documentar mitjançant xml.

A continuació enumerarem algunes de les avantatges de l'xml sobre altres formats a l'hora d'emmagatzemar informació:

a- El format xml és una base amb text, que ho fa més llegible, més fàcil de documentar i en alguns casos més fàcil de depurar.

b- Els documents xml poden utilitzar l'infrastructura ja creada per l'html, incluït el protocol HTTP, i alguns servidors HTTP permeten que es transmeti XML a través de servidors de seguretat.

c- L'anàlisi XML està perfectament definit i àmpliament implementat, el que possibilita la recuperació d'informació dels documents xml en diversos entorns.

d- Les aplicacions poden confiar en els analitzadors XML per a realitzar alguna validació estructural, així com en la comprovació de tipus de dades (Quan utilitzen esquemes).

e- L'XML és crea en una base unicode, fet que facilita la creació de documents internacionals.

No obstant, l'xml no és adequat per a totes les situacions, el format xml tendeix a ser més detallat que el format binari que substitueixen. Consumeixen més ample de banda de xarxa, i espai d'emmagatzemament, o requereixen més

temps per la comprensió. L'anàlisi de l'xml pot ser més lent que l'anàlisi molt optimitzat per a formats binaris i pot requerir més memòria.

Validar documents XML:
Per a validar els documents XML que contenen dades i estructures no desitjades hem d'associar un esquema XML al document XML. Els esquemes XML son regles que defineixen com s'estructuren els elements i atributs per a formar documents XML. És poden compartir esquemes entre organitzacions per a fer més simple la transferència i el procés de dades compartit.

És pot obtenir més informació i la seva especificació al consorci World Wide Web (W3C) a: http://www.w3.org/XML/.

3- Requeriments del nostre projecte "El ServletContainer".

3.1.- Introducció als servidors Web i el paradigma de l'arquitectura client-Servidor.

El paradigma client-servidor [2] és un dels més extensos dins dels serveis de la xarxa.
La idea bàsica i general que hi ha darrera d'aquest model és que hi ha algú que ofereix un servei (El servidor) i algú que demanda aquest servei (el client).

En el cas de les pàgines web tenim un servidor web que és aquell qui té la informació i un client (navegador) que és qui demana la informació.

Generalment, quan naveguem per Internet ens trobem, per tant, en el cantó del client. En la part del servidor haurem de gestionar convenientment els continguts i recursos per oferir-los als clients que els sol·liciten.

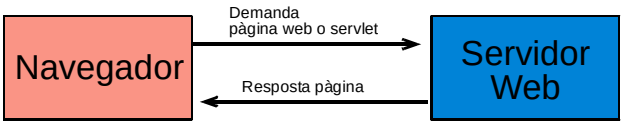

fig. 3.1 diagrama de blocs d'un accés a un servidor web centralitzat

3.2.- Servidors web, i els entorns distribuïts, un concepte diferent.

A partir de l'arquitectura clàssica client-servidor, un servidor web, i una adreça de xarxa, podem servir pàgines en servidors web que estiguin en altres ubicacions, simplement fent links als diferents servidors.

Aquests servidors han d'estar localitzats mitjançant un camí determinat.

Podríem fer un pas més enllà i aconseguir que per accedir als diferents objectes situats en diferents servidors no s'hagi de conèixer la seva ubicació dins la xarxa, d'aquesta manera quan a un servidor li falta un objecte, sigui capaç de localitzar-lo dins la xarxa, i servir la pàgina estigui on estigui. És en aquest punt en què sorgeix el projecte de final de carrera que s'ha proposat.

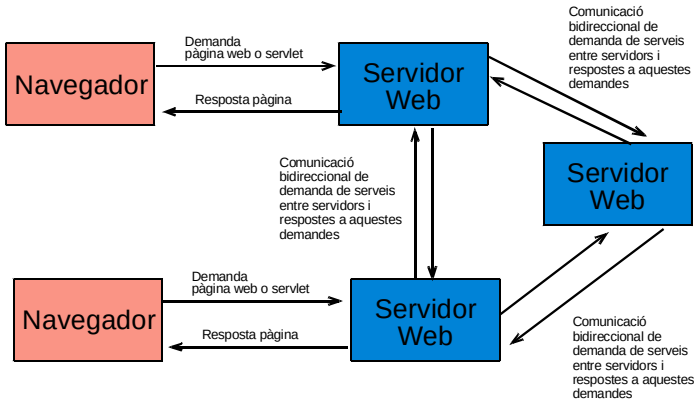

fig. 3.2 diagrama de blocs d'un accés a un servidor web distribuït

Avantatges del servidor web distribuït respecte al servidor clàssic:

- Podem afegir continguts en altres servidors sense necessitat de fer cap modificació en els altres servidors.

- Podem canviar d'ubicació els servidors de manera totalment transparent, als altres servidors.

- Podríem crear servidors web replicats, de manera que si un servidor cau o es desconnecta de la xarxa, els altres el supleixen.

Desavantatges del servidor web distribuït respecte al servidor clàssic:

- Els temps de resposta en alguns casos és superior degut a que s'ha de cercar per internet en altres servidors i en cas de que sigui un servlet s'ha d'executar, i enviar la resposta al primer servidor que el retorna al client.

- S'ha de portar un catàleg distribuït exhaustiu d'objectes que circulen dins els diversos contenidors, perque no puguin existir dos objectes amb el mateix nom i diferent comportament.

3.3.- Requeriments del ServletContainer.

El projecte que volem desenvolupar consisteix en dissenyar un servidor de pàgines web, servlets i JSPs distribuït.

En l' apartat anterior, hem introduït el concepte de Servidor web distribuït.

Volem que el servidor web distribuït compleixi els següents requeriments:

- Sigui capaç de servir tant pàgines web estàtiques com dinàmiques.

- Que localitzi altres servidors web dins la xarxa sense cap necessitat de saber la seva ubicació física.

- Es pugui utilitzar en diferents sistemes operatius, sense necessitat de realitzar cap modificació, es a dir, que sigui multiplataforma.

- Que un servidor web es pugui connectar al sistema de servidors distribuïts sense cap necessitat de fer modificacions en la xarxa de servidors distribuïts.

- Donat que en una xarxa extensa es puguin utilitzar diferents servidors distribuïts que no tinguin res a veure entre ells, hem d'aconseguir crear grups de servidors independents, amb un mínim de seguretat.

- La informació continguda dins els servidors web estigui dispersa dins la xarxa, i que no hi hagi cap punt que la centralitzi.

- Incorporar la possibilitat que la informació estigui replicada, en diferents servidors web.

- Sigui totalment transparent a l'usuari, és a dir que un cop l'usuari s'hagi connectat a un servidor web, el servidor sigui l'encarregat de buscar la informació entre els altres contenidors, sense que l'usuari hagi de fer cap acció.

- Pugui ser embegut per a ser utilitzat per altres apliacacions.

3.4.- Programes usats per al desenvolupament del ServletContainer.

Per a programar el projecte necessitem desenvolupar, o cercar alguna aplicació que puguem aprofitar per a desenvolupar-lo:

- Servidor de pàgines web estàtiques

- Contenidor de servlets.

- Un entorn per a gestionar els diferents contenidors de servlets que estiguin dispersos dins d'internet.

- Entorn de programació multiplataforma.

3.4.1- Contenidors de servlets i pàgines web.

Per al desenvolupament del contenidor de Servlets necessitem dissenyar o localitzar algun contenidor de servlets.

Anem a veure alguns dels contenidors de servlets i contenidors de pàgines estàtiques que hi ha al mercat.

Actualment un dels contenidors de servlets més usat és l'apache tomcat.

- **Apache Tomcat:**

  L'Apache tomcat Està composat per una banda d'un servidor de pàgines estàtiques, l'Apache i un contenidors de servlets, el Tomcat.

  En principi l'apache està disponible en diferents sistemes operatius, però el seu codi no és portable del tot, almenys hauríem que fer una recopilació de part del codi font i crear diferents distribucions del nostre projecte.

  Una altra dels desavantatges de l' Apache Tomcat és que hauríem d'inserir el codi dins el codi ja desenvolupat de l'apache, ja que és una aplicació que no es pot estendre fàcilment.

  Com avantatge tenim que és un dels servidors web més usat. I que és totalment opensource, cosa que ens permet no haver de comprar cap llicencia, i tenim tot el codi font disponible.

  És pot obtenir més informació sobre l'apache a la seva plana web: http://jakarta.apache.org/

- **Jetty:**

  Com una altra opció presentem un contenidor de servlets no tant conegut, anomenat Jetty.

  Jetty: El jetty és un servidor web, i contenidor de servlets que no s'ha de configurar ni executar de manera separada, com és el cas de l'apache, i que serveix tant per a pàgines estàtiques com dinàmiques.

  Aquest servidor està desenvolupat íntegrament en java, i funciona en totes les plataformes que es suporti java.

  Una altre avantatge del jetty és que és pot incloure dins de qualsevol aplicació i es pot incloure dins de qualsevol aplicació java, de forma embeguda.

Es pot més obtenir informació sobre el servidor web jetty a: http://www.mortbay.org/jetty/index.html.

Com que desenvoluparem el projecte a partir de laColla, que està desenvolupada amb java, no ens hauria de suposar un esforç molt gran utilitzar el jetty per a desenvolupar el nostre projecte, fent una extensió d'aquest.

4- Funcionalitat del Servlet Container.

Seguidament presento a grans trets el funcionament del ServletContainer que he dissenyat mitjançant un diagrama de blocs.

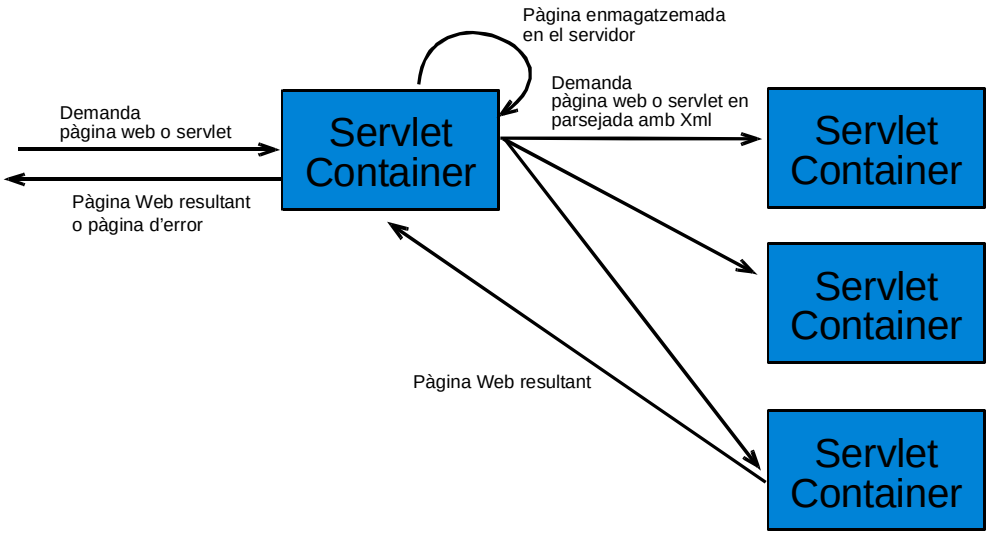

fig. 4.1 diagrama de blocs de connexió entre diferents servletContainer

El navegador accedeix a un dels servletContainer que hi ha dins la xarxa.

El servletContainer mira si la pàgina la té local. Si és local, executa el servlet, o l' html i retorna el resultat.

Si la pàgina no està en el servidor local, és parseja en un xml la petició de la pàgina i envia a tots els membres del mateix grup de servletContainer, la petició.

Si un dels membres del grup te el servlet o pàgina web demanada, llavors el retorna al servidor primer. En cas que cap tingui la pàgina, no es rebrà res en el primer servletContainer, i llavors hem fet un timeout, que explicarem més endavant.

Presentem de forma més formal un diagrama de casos d'us del ServletContainer:

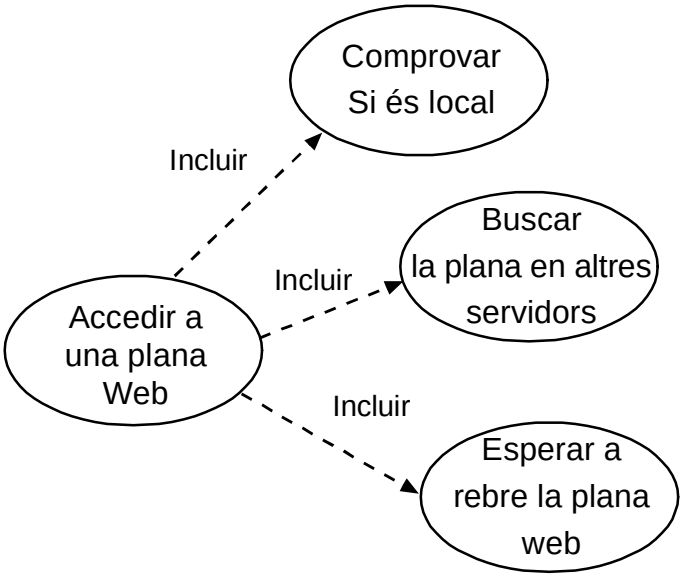

fig. 4.2 diagrama de casos d'un del projecte.

5- Arquitectura i Disseny del ServletContainer.

5.1- Introducció a l'Arquitectura del Contenidor de servlets distribuïts

El Contenidor de servlets distribuïts està format per tres grans blocs:

- LaColla: entorn de sistema col·laboratiu que ens permet fer la distribució dels objectes entre els diferents servidors web.
  LaColla està éssent desenvolupada per en Joan Manuel Marquès i Puig amb col·laboració de Xavier Vilajosana, i no és una aplicació totalment desenvolupada, és pot trobar més informació sobre laColla a: http://personals.ac.upc.es/marques/LaCOLLA-tesiJM.pdf

- Jetty: El jetty és un servidor web, i contenidor de servlets que no s'ha de configurar ni executar de manera separada, com és el cas de l'apache, per a pàgines estàtiques com dinàmiques.

  Aquest servidor està desenvolupat íntegrament en java, i funciona en totes les plataformes que es suporti java i es pot incloure dins de qualsevol aplicació java.
  Es pot més obtenir informació sobre el servidor web jetty a: http://www.mortbay.org/jetty/index.html

- Servlet Container: El servlet Container, és una extensió del servidor web jetty que utilitza els mètodes de laColla, que he desenvolupat per a poder dissenyar un conjunt de servidors web distribuïts.
  El que fa és unir els métodes de laColla i el Jetty per així poder crear el servidor web distribuït.

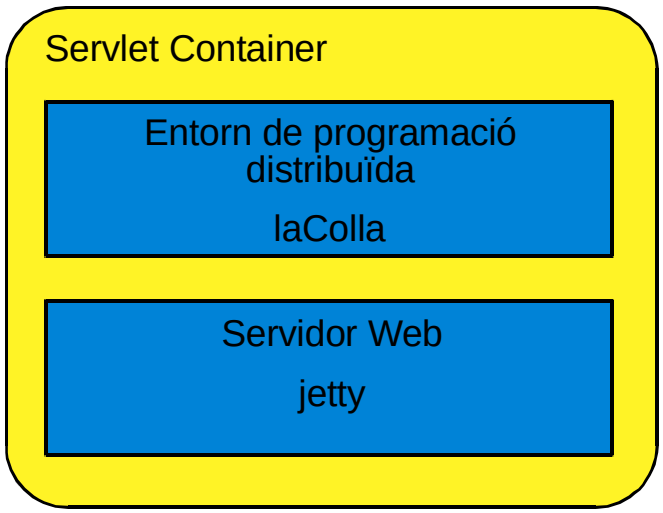

fig. 5.1 Components bàsics de l'arquitectura del servletContainer.

5.2- Diagrama de blocs del ServletContainer.

Com hem vist en el punt anterior, el Contenidor de Servlets, està compossat per tres grans grups. En la següent figura, podem veure com interactuen els tres blocs.

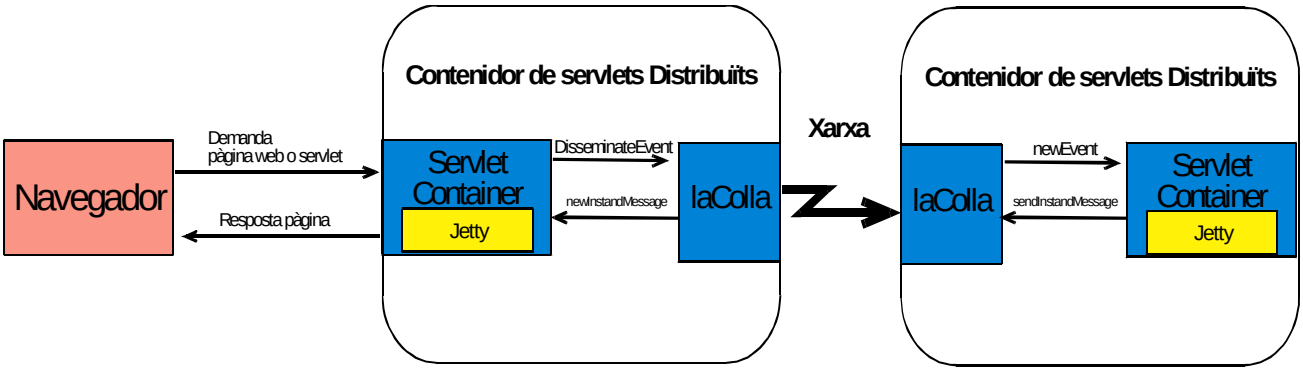

fig. 5.2 diagrama de blocs del servletContainer.

En aquesta figura es pot veure com el ServletContainer és una extensió del servidor web jetty, i utilitza els serveis de laColla, que li proporciona els serveis de middleware, per poder accedir als contenidors de servlets que estiguin en el mateix grup.
S'ha de tenir en compte que hem simplificat, en la figura 5.2 i només hem tingut en compte que hi ha dos contenidors de servlets distribuïts.

5.3- Diagrama de seqüència del funcionament del ServletContainer.

Per il·lustrar el funcionament i introduïr l'arquitectura a grans trets del contenidor de servlets, presentem un diagrama de seqüència.

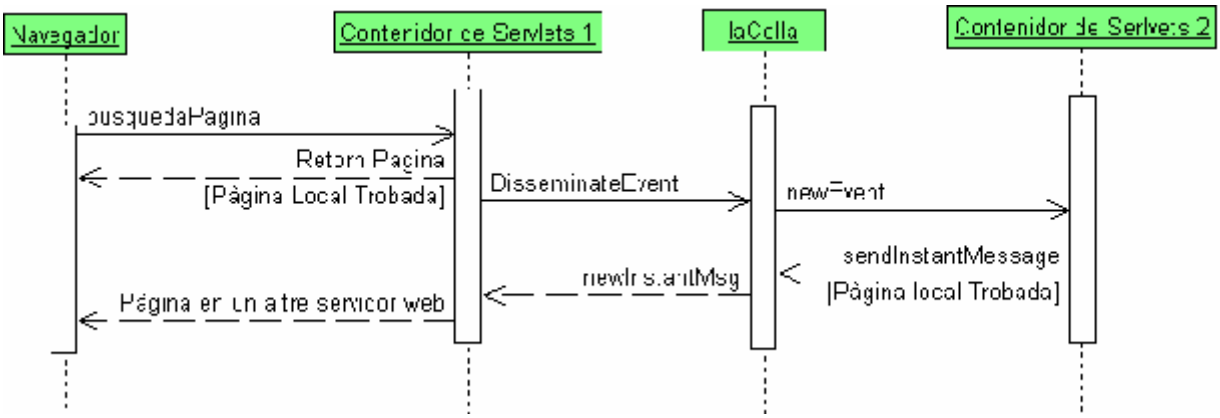

fig. 5.3 diagrama de seqüència del servletContainer.

Aquest diagrama de seqüència mostra l'exemple de funcionament entre un grup format per dos contenidors de servlets, i la seva comunicació amb laColla.

Per a simplificar el diagrama s'ha col·locat un sol objecte anomenat LaColla, però en la implementació el fet és que laColla és un objecte distribuït que te dues instàncies, una en cada un dels contenidors de servlets. Aquí la presentem com una caixa negra, que no ens importa la seva implementació ja que no forma part del projecte, però si l'hem usat per a poder desenvolupar l'aplicació.

En aquest diagrama l'usuari demana mitjançant un navegador web una plana web, o servlet, a un dels Contenidors de servlets, en el nostre cas al contenidor de Servlets 1.

En cas de que la pàgina estigui disponible de forma local, i com que el Contenidor de servlets és una extensió del servidor web jetty llavors retornem el codi html de la pàgina.

En cas de que el Contenidor de servlets no tingui la pàgina local disponible, llavors llancem una petició de DisseminateEvent, amb els paràmetres corresponents per accedir al servlet o pàgina web.

LaColla envia a tots els membres del grup un missatge anomenat NewEvent. Aquest missatge és recollit per tots els servidors de servlets, menys el qui l'envia.
Els servidors de servlets que tenen la pàgina local, envien un sendInstandMessege, a el servidor que ha fet la demanda de servei. El servidor que ha fet la demanda rep un NewInstantMsg amb el resultat, que és mostrat pel navegador.

5.4- Arquitectura del Contenidor de servlets distribuïts

Per crear el ServletContainer ho hem fet a partir de la classe Server definida per org.mortbay.jetty.Server.

Anem a veure quins d'aquests mètodes hem afegit o rescrit a aquesta classe.

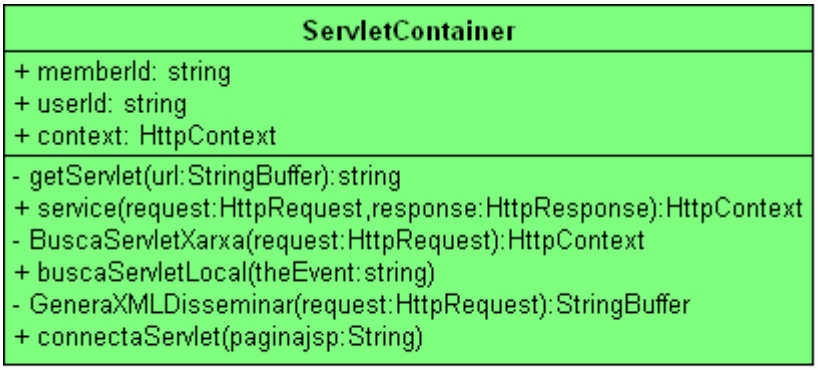

fig. 5.4 La classe principal del projecte

Per a programar el ServletContainer hem re-escrit la classe definida per org.mortbay.jetty.Server.

Anem a veure quins mètodes que hem re-escrit per aquesta classe que serà la central de la nostra aplicació:

buscaServletLocal: Aquest mètode és l'encarregat de buscar un servlet o pàgina estàtica dins el servidor local. Com a paràmetre d'entrada te l'event que hem rebut d'un servidor extern. Un cop hem rebut l'event el mètode l'analitza, ja que està en format xml, i si hi ha la pàgina web o servlet dins el servidor local l'envia al Servidor que ha fet la demanda de la plana web resultant.

buscaServletXarxa: Aquest mètode el que fa és que, donada una petició d'un servlet o pàgina estàtica, utilitzant el mètode ***GeneraXMLDisseminar***, parseja la petició i llança un missatge a tots els altres servidors del grup fent una demanda d'una pàgina web o d'un servlet.

connectaServlet: Aquest mètode simplement mostra al navegador la pagina jsp, que hem rebut de manera externa d'un altre servidor.

service: Anem a analitzar d'una manera més detallada el mètode service, ja que aquest mètode es un dels més importants del projecte.

Introducció al mètode service:
Quant un navegador accedeix al servidor, s'executa el mètode service. Un cop s'ha executat aquest mètode ja s'ha d'haver retornat mitjançant response_ la pàgina web que s'ha de mostrar.
Per tant el que fem en aquest mètode és primer comprovar si és local la pàgina. Si és local, simplement fem un this.super i ens oblidem.

El problema rau quan la pàgina web no està en el servidor local i està en un servidor extern.

Que fem per a solucionar aquest problema? Si es surt del mètode service, ja no podrem escriure la plana perquè es produeix un commit de la plana web.

Per a solucionar aquest problema de sincronisme, hem tingut que crear la classe SemaphoreTimeOut, que fa un wait en l'execució de l'aplicació uns segons determinats.

Durant aquests segons s'ha de buscar la classe de forma global i rebre els resultats. Finalment quan es rep la pàgina es fa un signal per a que pugui continuar l'execució. Si passat un temps determinat no hi ha cap resultat s'entén que no s'ha trobat la pàgina i és mostra la pàgina d'error.

5.5- Disseny del contenidor de Servlets

Finalment presentem el diagrama estàtic que hem dissenyat al projete.

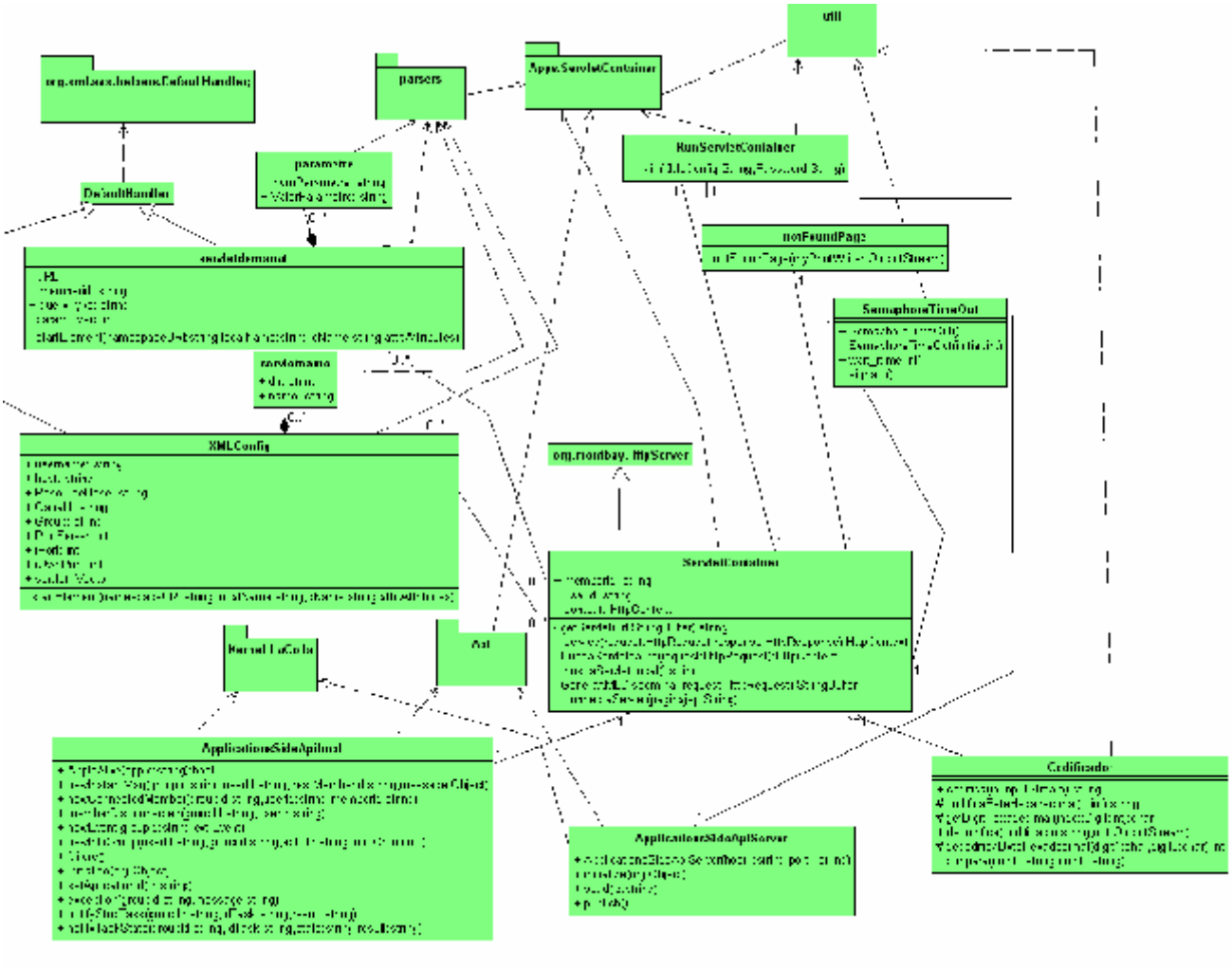

fig. 5.5 El diagrama estàtic del ServletContainer

5.5.1.- Detalls d'implementació.

Com ja s'ha comentat anteriorment, la classe principal és el ServletContainer,

Seguidament anem a veure una descripció de les classes que hem dissenyat per al servletContainer:

a) servletDemanat:Aquesta classe és l'encarregada de llegir els paràmetres que es passen entre els servletsContainers, agafar el xml i passar-ho a una estructura per a que així el servletContainer pugui utilitzar-la.

b) paràmetre: Classe depenent de servletDemanat que simplement emmagatzema un dels paràmetres que hi ha entre les consultes. S'ha de tenir en conte que un servlet pot tenir varis paràmetres d'entrada.

c) XMLConfig: Llegeix el fitxer de configuració i l'emmagatzema dins d'una estructura per a la seva posterior utilització. Dins d'aquesta estructura tenim un vector on hi ha la classe servletname, que conté tots els servlets que hi ha dins el contenidor de servlets.

d) servletname: Classe on enmagatzemem tots els servlets i la seva ubicació dins el disc dur del contenidor de servlets.

e) ApplicationsSideApiImpl: Classe depenent de laColla, on rebem les notificacions dels altres membres connectats a laCola, tal com els membres que es connecten a laColla, com els missatges que van dirigits a tot els grup o missatges que van dirigits únicament al membre.

f) ApplicationsSideApiServer: Aquesta classe és l'encarregada de publicar a laColla el nostre servidor de servlets.

g) Codificador: Per a transportar entre dos servletsContainers un objecte que té elements binaris hem hagut de crear aquesta classe per a codificar els objectes amb hexadecimal, ja que amb una string no podem representar els elements.

h) notFoundPage: Aquesta classe simplement genera una pàgina web d'error, en cas de que no es trobi cap pàgina web o servlet que correspongui.

i) semaphoreTimeOut: Classe usada per el servletContainer per a gestionar el sincronisme.

j) RunServletContainer: Classe que conté el main per a configurar i executar el contenidor de servlets.

5.6- Pas de paràmetres entre servletsContainers.

Quant un ServletContainer no te la pàgina web fa una demanda de pàgina als altres ServletContainer.

El servletContainer ho fa de la següent manera:

Crea un xml exemple de la següent manera:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Servlet>
 <Url servlet="HelloWorldServlet"/>
 <memberid idmember="member#27a8980cbaef1004868a3d1f6ebae727#"/>
```

```
 <Params> <Method type="GET"/>
 <data name="texto2" value="El segon parametre"/>
 <data name="texto1" value=" El primer parametre "/>
 </Params>
</Servlet>
```

On l'explicació dels paràmetres és la següent:
`Url servlet` -> És el servlet o pàgina html que demana el ServletContainer.
`memberid idmember` -> Identificador del membre de laColla que demana la pàgina.
`Params` -> Secció que correspon als paràmetres.
`Method type` -> Mètode de pas de parametres, GET o POST.
`<data name="texto2" value="El segon parametre"/>` -> En aquest tag tenim la descripció dels paràmetres, en primer lloc el nom del paràmetre i després el seu valor.

Tenim que el fitxer xml ha de poder-se validar amb el següent DTD:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT Method EMPTY>
<!ATTLIST Method CDATA #REQUIRED>
<!ELEMENT Params (Method, data+)>
<!ELEMENT Servlet (Url, memberid, Params)>
<!ELEMENT Url EMPTY>
<!ATTLIST Url CDATA #REQUIRED>
<!ELEMENT data EMPTY>
<!ATTLIST data CDATA #REQUIRED CDATA #REQUIRED>
<!ELEMENT memberid EMPTY>
<!ATTLIST memberid CDATA #REQUIRED>
```

5.7- Format de retorn de pàgines entre ServletsContainers

Quan el servidor web te una pàgina demandada per un altre servletContainer, el que fem és codificar la pàgina en hexadecimal i enviar-la unicament al servletContainer demandant. Com que hem rebut el seu identificador de laColla sabem a qui li hem d'enviar, evitem el fet d'enviar a tothom el missatge.
El que sí fem es codificar en hexadecimal, perquè ens hem trobat que hi ha caràcters que no és poden representar si no els passem a algun altre tipus de representació.

6- Instal·lació del contenidor de servlets.

Un cop tenim correctament configurada laColla, hem d'instal·lar el ServletContainer de la següent manera:

El contenidor de servlets s'ha d'instal·lar de la següent manera.
Per executar s'ha d'executar la classe:
Apps.servletContainer.RunServletContainer

I els arguments han de ser:
Argument 1: Ruta d'accés on és troba el fitxer de configuració xml del Servlet Container.

Argument 2: Password de laColla.

Consideracions:
S'ha de tenir en compte si s'executa en la mateixa màquina dos contenidors de servlets de no publicar el servidor web en el mateix port.
De manera idèntica s'ha de tenir en compte de no usar els ports de laColla en la mateixa màquina.

S'ha de tenir cura que l'aplicació laColla estigui correctament instal·lada, per a que el contenidor de servlets funcioni correctament.

El Fitxer de configuració hauria de complir les regles del següent DTD:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT LaColla EMPTY>
<!ATTLIST LaColla
        CDATA #REQUIRED
        CDATA #REQUIRED
        CDATA #REQUIRED
        CDATA #REQUIRED
>
<!ELEMENT Servlet EMPTY>
        <!ATTLIST Servlet
        CDATA #REQUIRED
        CDATA #REQUIRED
        >
<!ELEMENT ServletContainer EMPTY>
        <!ATTLIST ServletContainer
        CDATA #REQUIRED
        CDATA #REQUIRED
        CDATA #REQUIRED
>
<!ELEMENT ServletContainerConfig (LaColla, ServletContainer, User, Servlet)>
<!ELEMENT User EMPTY>
<!ATTLIST User
CDATA #REQUIRED
>
```

Un exemple de fitxer de configuració seria el següent:

```
        <?xml version="1.0" encoding="UTF-8"?>
        <ServletContainerConfig>
```

```
<LaColla port="2222" ownport="2226"
GAPAID="GAPA#22b340ere9ec10048e4342c0ecd2cd56#"
group="group#60d15a84b981100486235ffd6c5acde9#default"/>
<ServletContainer Port="81" host="localhost"
resourcebase="E:/UOC/Projecte/jetty-4.2.22/webapps"/>
<User name="Josep" />
<Servlet dir="/HelloWorldServlet/*"  name="HelloWorldServlet" />
<Servlet dir="/HelloWorldServlet/*"  name="ElMeuServlet" />
</ServletContainerConfig>
```

On:

Configuració LaColla:

* port i ownport: son els ports de comunicació de laColla.
* GAPAID: GAPA ussat per laColla.
* Group: grup d'usuaris de laColla.

* Per a més informació sobre aquestes configuracions consultar la documentació de l'aplicació laColla.

Configuració ServletContainer:

port: Port on es publica el servidor ServletContainer.
host: Adreça IP del servidor ServletContainer.
Resourcebase: Directori local on s'hi ha deixat les aplicacions Web que es volen publicar a la xarxa.

Configuració User:

Name: Nom de l'usuari que es vol connectar a laColla.

Configuració Servlet:

Dir: Directori on és troba el servlet.
Name: Nom del servlet, a carregar dins el servidor de Servlets.

NOTA: ES MOLT IMPORTANT QUE ELS SERVLETS ESTIGUIN DINS EL CLASPATH, ja que si no no és podran carregar a memoria pel contenidor de Servlets.

7- . Conclusions

7.1.- Conclusions sobre l'arquitectura p2p.

L'arquitectura d'igual a igual és una arquitectura molt nova, que poc a poc s'anirà implantant. Sobretot hem de tenir en compte que cada cop van apareixent al mercat nous aparells (telèfons mòbils de 3G, Pdas,....) que s'aniran interconnectant entre ells, i que cadascun d'ells tindrà la possibilitat d'accedir a la xarxa, o fer treball cooperatiu.
Aquests aparells, en molts casos accediran mitjançant IP dinàmica, ja que la gran proliferació de màquines connectades a internet i el cost que suposa el mantenir una IP fixa, començarà a fer escassa la possibilitat d'usar una IP fixa per cada aparell[6].

Algunes aplicacions que accediran a comunitats necessitaran buscar-se dins la xarxa, i d'aquesta manera s'evitarà que hagi d'haver un servidor central.

També hi ha usos de l'arquitectura p2p que ajudaran a cedir temps de processador, que no utilitzem per a projectes de recerca, com pot ser el Saving lives with p2p [7].

7.2.- Altres projectes similars al ServletContainer.

Un projecte similar al contenidor de servlets distribuïts és el JXTA-HTTPD, dins el projecte jxta [5]

Dins el jxta s'ha creat un contenidor que permet publicar pàgines web, mitjançant aquest mòdul. Per oferir el servei web, s'utilitza també el servidor web jetty.
Sembla ser que aquest projecte únicament, és un contenidor de pàgines web, el que no aporta és un contenidor de servlets, Tot i que jetty ens ho permet fer.
També té una eina automàtica per transferir un lloc web standard a un lloc JXTA-HTTPD.

Dins també de jxta hi ha un projecte que consisteix en crear grups de discussió dins de pàgines web, s'anomena juxtaprose.

7.3.- Conclusions sobre l'aplicació de contenidor de servlets.

S'ha assolit l'objectiu principal del treball.

- A partir del servidor web jetty i l'aplicació laColla, hem aconseguit estendre el servidor, per a crear un servidor web capaç de buscar dins el grup de servidors els components que li falten, per a completar qualsevol pàgina web.

- L'eina que hem creat es capaç de buscar i executar qualsevol servlet i pàgina html, que estigui dins un grup de servidors, de forma totalment transparent per l'usuari que es connecta a la pàgina web.

- Hem usat per a la transmissió de dades entre els servidors web XML, de manera que s'ha estudiat el funcionament d'aquesta tecnologia i alhora s'ha dissenyat un sistema de pas de paràmetres totalment standard.

- A nivell personal el treball ha estat força enriquidor. Les tecnologies de l'xml, el java i els servidors web amb que hem hagut de treballar estan a primer ordre del dia en l'actualitat.

7.4.- Utilitats del contenidor de servlets.

La principal utilitat és utilitzar-lo de servidor web descentralitzat, és a dir, si tenim una empresa situada geogràficament en diferents llocs del món, on cada un dels centres de treball han de fer servir diferents bases de dades, el contenidor de servlets ens ofereix un accés a aquestes pàgines i permet poder unir totes les pàgines web, sense necessitat de saber ni configurar les adreces físiques de cada un dels centres de treball. Així podem fins i tot usar adreces IP, que no siguin fixes.

Una altra de les utilitats del contenidor de servlets pot ser la creació de llocs web, que tenen ubicacions diferents, i es poden provar els servlets si funcionen correctament, sense haver de definir una adreça fixa entre els diferents servidors.

També una de les utilitats del contenidor de servlets pot ser que a partir d'un grup de contenidors de servlets, dissenyar servlets que siguin capaços de realitzar qualsevol tipus de tasques, com pot ser la d'aprofitar els cicles de rellotges de les màquines que estan connectades al contenidor de servlets, i que tenen cicles de rellotges sobrants.

També es poden crear aplicacions basades en servlets, que les pot cedir al grup del contenidor de servlets, per a poder ésser usades per els altres membres.

Un altra de les utilitats del servletContainer, pot ser la de dissenyar a partir d'ell un sistema d'objectes distribuïts. És a dir crear un servlet dins el propi contenidor de servlets, que sigui capaç d'agafar objectes locals i traslladar-los sencers a altres contenidors de servlets, de manera de que si algun dels contenidors de servlets cau, llavors és pugui accedir a aquest objecte de manera local o bé a traves d'un altre contenidor de servlet que ja el tingui.

Per altra banda podríem utilitzar el servletcontainer, per a replicar servidors i d'aquesta manera si cau un dels servidors assegurar-nos que hi ha un altre servidor que el supleix. El Apache Tomcat a partir de la versió 5.0, és capaç de fer-ho. Però aquesta possibilitat s'hauria d'estudiar en profunditat, i veure si s'ha de realitzar algun canvi al servletContainer.

7.5.- Línies Futures de Treball

Les millores que es proposen per al Contenidor de Servlets son les següents:

- Creació d'un diccionari d'objectes (Servlets, i pàgines html) distribuït, de manera que no calgui enviar a tots els membres les demandes de serveis, i així també podrem saber si hi ha objectes amb el mateix nom i amb una funcionalitat diferent.

- Realitzar una aplicació que generi el fitxer xml de configuració de manera que sigui més fàcil a l'usuari realitzar la configuració.

- Al realitzar les consultes es fa amb fitxer xml que viatja a través de la xarxa. Una bona idea, per a la seguretat de les dades que es transporten seria el encriptar les dades que es transmeten per la xarxa.

Finalment una altra línia futura podria ser el crear un Container compatible J2EE distribuït.

La fundació apache està desenvolupant el projecte Apache Geronimo[8], per a dissenyar un contenidor J2EE. Aquest projecte s'està desenvolupant íntegrament en java. I com a nucli de contenidor web està usant el jetty.

Geronimo, en aquests moments encara no te el certificat J2EE, però sembla que aviat l'aconseguirà.
La idea seria substituir el servidor jetty que conté el projecte Geronimo, per el nostre contenidor de servlets distribuït, i realitzar els ajustos pertinents a l'aplicació per a que així es pugui crear un servidor totalment distribuït, que suporti la tecnologia j2EE.

Altres projectes per a crear servidors web J2EE, són el JonAS[9], que també pot usar com a contenidor web el jetty (a part de l'apache), i per tant s'hauria de canviar el jetty, per la nostra aplicació servletContainer.

Tant un com l'altre projecte JOnAS, com Geronimo, s'hauria d'estudiar, si simplement substituint el nostre servidor funcionaria correctament, o bé s'hauria de retocar el codi del servidor J2EE.

8.- Bibliografia

[1] Joan Manuel Marquès Puig. “Dissertació per l'obtenció del títol de doctor en informàtica pel Departament d'Arquitectura de Computadors de la Universitat Politècnica de Catalunya” Setembre de 2003
http://personals.ac.upc.es/marques/LaCOLLA-tesiJM.pdf

[2] Robert Orfali, Dan Harkey, Jery Eduards . “Essential Client/Server Survival Guide”. ISBN 0-442-01941-6

[3] http://www.mortbay.org/jetty/index.html Informació sobre el servidor web jetty.

[4]Minar, N. (2001). “Peer-to-Peer is Not Always Decentralized: When Centralization is Good”.
Comunicació presentada a la The O'Reilly Peer-to-Peer and Web Services Conference. Washington, D.C. Novembre 2001.
[5] http://www.jxta.org

[6] http://www.mouse.cl/antes/Nro.042-1996.15.08/Nro.042D.html

[7] http://www.openp2p.com/pub/a/p2p/2001/03/01/medical.html

[8] http://geronimo.apache.org/

[9] http://jonas.objectweb.org/

Agree, Philip E. *P2P and the promise of internet equality.* pàgines: 39 – 42
Communications of the ACM Volume 46 , Issue 2 (February 2003).
www.sindominio.net/suburbia/article.php3?id_article=68
www.lavanguardia.es
www.zonap2p.com
www.mexicoextremo.com.mx/noticias/p2p.php
www.barcelnanetactiva.com/aplic/bd/noticies.nsf
http://www.w3.org/XML/.