ESTUDI COMPARATIU DE TECNOLOGIES PEL DISSENY D'APLICACIONS WEB

TÍTOL DEL TFC: Estudi comparatiu de tecnologies pel disseny d'aplicacions web

TITULACIÓ: Enginyeria Tècnica de Telecomunicació, especialitat Sistemes de Telecomunicació

AUTOR: Joaquim Sardà Carbasse

DIRECTOR: Dolors Royo Vallès

DATA: 10 de juny de 2010

Títol: Estudi Comparatiu de tecnologies pel disseny d'aplicacions web

Autor: Joaquim Sardà Carbasse

Director: Dolors Royo Vallès

Data: 10 de juny de 2010

Resum

La realització d'una aplicació web avui en dia pot resultar una tasca que es pot realitzar de diferents maneres i emprant diferents tecnologies. Escollir la tecnologia més adequada per a les necessitats del nostre projecte pot ser una decisió clau alhora de solucionar els problemes que ens podem trobar al realitzar-lo.

L'objectiu d'aquest treball de final de carrera és realitzar una comparativa de dues de les tecnologies més utilitzades en aquest àmbit com són PHP i Java. D'aquesta manera també definir quan és més adequada la creació d'un Web service segons les necessitats que tinguem en la nostra aplicació i no una aplicació web simple.

S'han creat dos aplicacions web una creada com un servlet PHP i l'altre com a servlet Java que envien un fitxer qualsevol al servidor. També s'ha realitzat un servei web que satisfà el mateix objectiu. Perquè la comparativa fos exacte s'ha intentat executar les diferents aplicacions sobre un mateix entorn.

Per analitzar els resultats s'han fet proves realitzant captures de paquets i comparant els resultats entre les diferents aplicacions. A més a més s'han fer proves de rendiment en el servidor. També s'ha tingut en compte l'experiència al crear una aplicació en cada una de les tecnologies i les dificultats que s'han presentat per a realitzar-les en un mateix entorn.

El treball ha estat realitzat des del punt de vista d'un estudiant de telecomunicacions especialitat en sistemes, per tant un estudiant no familiaritzat en les tecnologies web. Aquest fet s'ha tingut en compte alhora de comparar la facilitat per aprendre i familiaritzar-se amb cadascuna de les tecnologies, així com la facilitat per trobar documentació útil per a aquesta finalitat a la xarxa.

Title: TFC/PFC Model

Author: Joaquim Sardà Carbasse

Director: Dolors Royo Vallès

Date: June, 10th 2010

Overview

Creating a web application nowadays is a task that can be done using diferent ways and diferent technologies. To choose the appropiate technologie to fulfill the requirements of our project it is very important to solve future problems we will probably face when creating the application.

The aim of this project is to compare two of the most used technologies in web applications which are PHP and Java. Another objective is to define when you have to create a web service instead of a simple web aplication, depending on the requirements we have.

Two web applications have been created one of them as a PHP servlet and the other one as a Java servlet. The web application sends a file from a client machine to a server. A web service with the same function have been created too. To make a fair comparison, we've tried to run all the applications in the same enviorenment.

To analize the results some packet captures have been done to compare the results between the applications. Besides some perfromance tests have been done in the server side. The experience of creating an aplication using diferent technologies in the same environment has been an important fact in the comparison.

The project have been done under the point of view of a student of Telecommunication engineering specializing in telecommunication systems, so a student not familiarized with web technologies. It has been an important fact to compare if the technologies are easy to learn or if there is enough useful documentation in the net.

ÍNDEX

INTRODUCCIÓ

En l'entorn actual si volem realitzar una nova aplicació web se'ns obra un ventall de possibilitats. Bàsicament una aplicació web és un programa informàtic al que s'accedeix mitjançant un navegador i que no requereix instal·lar cap software per part del client. Normalment els destinataris d'aquestes aplicacions web són persones humanes però com veurem a continuació en el cas dels web services també poden ser altres màquines i requerir alguna instal·lació en la part del client. Tenint en compte això, s'han escollit dues tecnologies de les més utilitzades per a realitzar aplicacions web, com són Java i PHP, per realitzar una comparativa. També s'ha volgut investigar els web services com a una manera diferent d'entendre les aplicacions web.

L'objectiu d'aquest document és realitzar una comparativa entre dues de les tecnologies més utilitzades en programació web (Java i PHP) i veure quines diferències o avantatges trobem en cadascuna d'elles. D'aquesta manera també definir quan és més adequada la creació d'un Web service segons les necessitats que tinguem en la nostra aplicació i no una aplicació web simple.

En el primer capítol es mostren les característiques de l'arquitectura d'una aplicació web i les seves diferents capes, així com les característiques de l'arquitectura d'un web service.

Al segon capítol es presenten les tecnologies utilitzades en aquest treball i s'explica cadascuna d'elles.

En el tercer capítol es realitza una comparació teòrica entre PHP i Java tenint en compte les particularitats de cada llenguatge i el model d'execució.

En el quart capítol s'explica el treball realitzat i les incidències que s'han hagut d'afrontar en cada una de les tecnologies.

En el capítol 5 es fa un anàlisis dels resultats obtinguts i finalment s'exposen les conclusions en el capítol 6.

CAPÍTOL 1. ARQUITECTURA D'APLICACIONS WEB

Una aplicació web és un programa informàtic que ens permet realitzar un o diversos tipus de treball accedint-hi mitjançant un navegador web dins una xarxa com pot ser Internet. La idea de poder executar una aplicació sense realitzar cap instal·lació a la màquina del client ha fet que la seva popularitat s'hagi vist incrementada de forma espectacular. Per desenvolupar-los s'utilitzen llenguatges que suporten els navegadors com poden ser PHP o Java. Aquestes aplicacions resideixen en el servidor, però estan fetes per ser utilitzades per humans, tot i que més endavant veurem les característiques dels web services. Normalment les interaccions dels usuaris es realitzen mitjançant una pàgina web però en la majoria dels casos les dades son emmagatzemades i manipulades en el servidor.

1.1. Arquitectura d'una aplicació web

Una aplicació web empresarial moderna conté les següents capes que podem observar en la figura 1.1 extreta de la font [3]: la capa client, la capa intermèdia i la capa de dades.

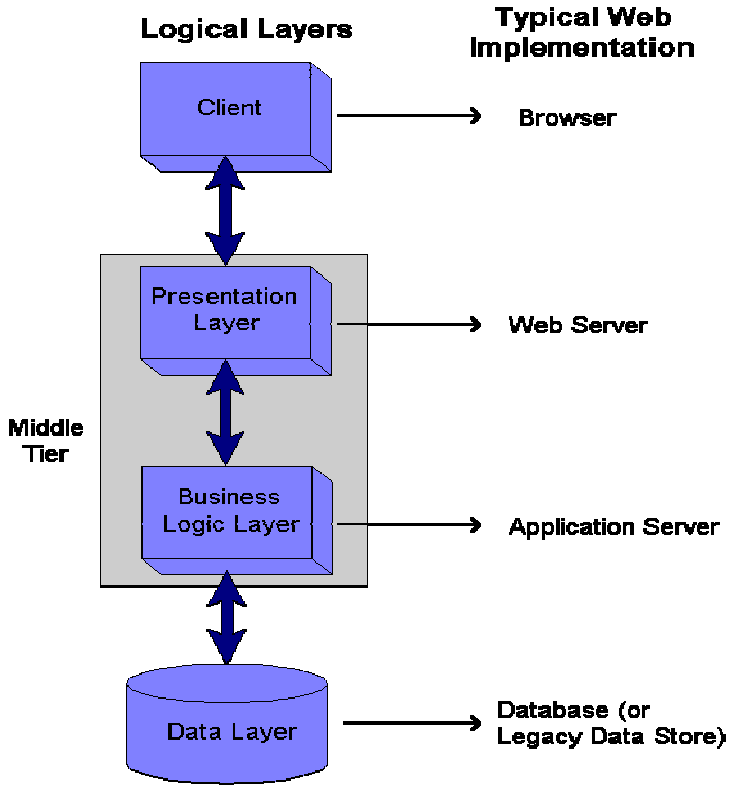

Fig. 1.1 Capes d'una aplicació web

1.1.1. Capa client

Aquesta capa està implementada per una aplicació web que corre sobre un navegador en la màquina del client. La seva feina en una aplicació web es mostrar les dades i permetre que l'usuari pugui accedir o modificar les dades. En línies generals s'utilitza una de les dos propostes següents per construir aquesta capa:

- Un client només HTML: amb aquesta proposta, virtualment tota la intel·ligència està localitzada a la capa intermèdia. Quan a l'usuari se li presenta la pàgina web, tot el procés de validació i els errors que es puguin produir són reenviats al client com a una nova pàgina web.

- Un client semi intel·ligent HTML/HTML dinàmic/JavaScript: amb aquesta proposta s'inclou part de la intel·ligència dins de la pàgina web que corre sobre la capa client. Un exemple simple seria que el client hagi de fer algunes validacions o incloure algunes parts dinàmiques HTML (amagar camps d'un menú que ja no són aplicables, reconstruir llistes de seleccions etc.). Cal dir que aquesta intel·ligència pot ser escrita mitjançant altres llenguatges d'scripting.

En el primer cas tenim l'avantatge que acostuma a funcionar amb versions més antigues dels navegadors. També permet una millor separació de la lògica empresarial i la capa de presentació ja que amb l'altre mètode les seves funcions poden entrellaçar-se.

En el segon cas el tipus de client utilitzat generalment és més fàcil d'utilitzar i requereix menys comunicació amb el servidor en els dos sentits. És cert que s'acostuma a necessitar una versió més recent del navegador però avui en dia això no resulta un problema.

1.1.2. Capa intermèdia

1.1.2.1. Capa de presentació

La capa de presentació genera pàgines web i inclou contingut dinàmic en aquesta. El contingut dinàmic normalment s'origina a partir de la base de dades (una llista de productes, de transaccions etc.). Una altre de les funcions de la capa de presentació is descodificar la pàgina web que retorna des del client (l'usuari introdueix dades i passen a la capa de la lògica del negoci).

Aquesta capa es pot construir amb diferents eines. Originàriament estaven construïdes amb programes CGI (Common Gateway Interface). Actualment podrem trobar diferents eines que ens ajuden a realitzar aquesta tasca de les que destaquen:

- ASP (Active Server Pages) és la opció que ens proporciona Microsoft.

- JSP (Java Server Pages) i Servlets que és la opció que ens proporciona Sun Microsystems amb el llenguatge Java.

- Utilitzant el llenguatge de programació interpretat PHP que és open source.

Aquestes eines ens proporcionen mètodes que ens faciliten allotjar continguts dinàmics en un document HTML estàtic en una pàgina web. També proporcionen eines que fan més fàcil analitzar sintàcticament (parser) una pàgina web que conté la informació que l'usuari ha introduït.

La capa de presentació acostuma ha està implementada en un servidor web (IIS, Apache, Webshere, etc.). El servidor web pot acceptar peticions per a diferents aplicacions i també les peticions per les pàgines web estàtiques.

1.1.2.2. Capa lògica del negoci

La major part de l'aplicació lògica està escrita en la capa lògica del negoci, que inclou:

- Realitzar tots els càlculs i validacions necessàries.

- Controlar el volum de treball (incloent el control de les sessions de dades)

- Controlar tots els accessos de dades a nivell de presentació.

Aquesta capa també pot estar construïda mitjançant la opció que ens proposa Microsoft que seria un objecte COM escrit en Visual Basic C++, o la opció de Sun Microsystems utilitzant Java Beans.

Aquesta capa acostuma ha estar implementada dins un servidor d'aplicacions (com són Microsoft MTS, BEA WebLogic, IBM Web Sphere, etc.) (application server) que generalment automatitza serveis com transaccions, seguretat, missatgeria i nom del servei. Aïllant la lògica del negoci d'aquestes activitats, permet al desenvolupador prestar més atenció en el desenvolupament de la lògica de l'aplicació, mentre que els venedors de servidors d'aplicació diferencien els seus productes per la facilitat de manipulació, seguretat, fiabilitat i eines de suport.

1.1.3. Capa de dades

La capa de dades és la responsable de controlar les dades. En un cas simple estaria composta per una base de dades relacional. Tot i això, pot incloure procediments d'accés a dades a d'altres fonts de dades com bases de dades jeràrquiques, o altres fitxers. La feina de la capa de dades és proporcionar la informació requerida a la capa lògica quan ho necessiti i emmagatzemar dades quan se li demani.

Generalment aquesta arquitectura hauria d'apuntar a no tenir o tenir poca validació/lògica de negoci en aquesta capa però no sempre és recomanable. Restriccions no nul·les i restriccions de claus externes poden ser considerades "regles del negoci" que haurien d'estar incloses només a la capa lògica. A la pràctica és més segur incloure aquestes restriccions simples a la base de dades i realitzar canvis quan les regles del negoci evolucionin.

Un model més simple per a una aplicació web podria ser el que es mostra en la figura 1.2 extreta de la font [3], on no seria necessària la complexitat de tenir dos capes separades en la capa mitja. En aplicacions més petites la capa lògica i de presentació es troben al mateix servidor web.

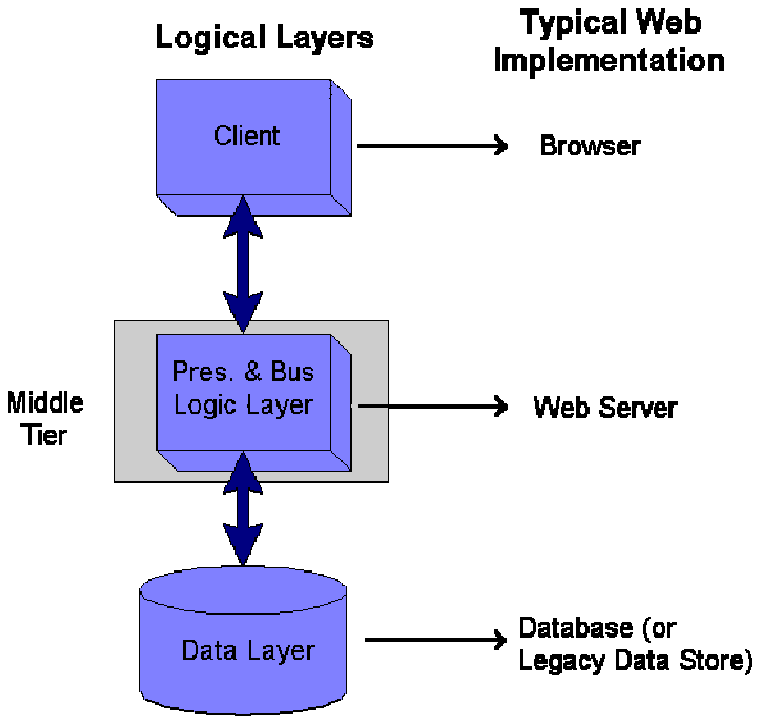

Fig. 1.2 Model d'aplicació web simplificat

1.2. Arquitectura d'un web service

Els web services consisteixen en mètodes que operen en missatges contenint informació orientada a documents i orientada a procediments. L'arquitectura en la que està basada un web service és l'anomenada SOA (Service Oriented Architecture), l'evolució lògica d'un sistema de components distribuïts orientats a objecte en una xarxa de serveis. Els webs services proveeixen en termes generals una infraestructura que permet la integració entre empreses. Per veure més clarament on queden englobats els serveis web observem la figura 1.3 i posteriorment anirem desglossant.

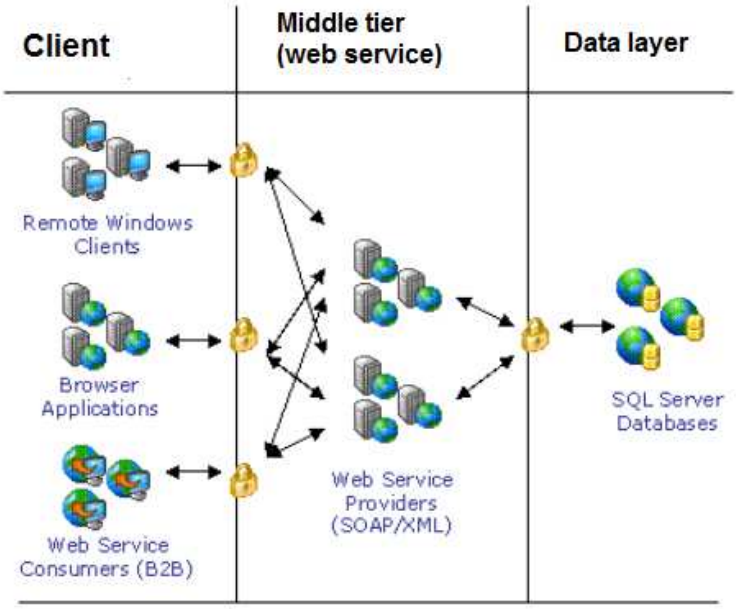

Fig. 1.3 Visió global dels web services

Els web services estan fets perquè la interacció es produeixi entre programes. Aquests han de tenir una interfície de programació d'aplicacions clarament definida que consisteix en proveir respostes a peticions HTTP GET i POST realitzades per a aplicacions remotes. Aquest fet no vol dir que no hi hagin serveis web als quals es pugui accedir mitjançant una interfície gràfica per usuaris, però aquesta no seria necessària. Un dels exemples més comuns seria rebre resultats d'una petició GET o un POST com a strings XML, que requereixen un parser (intèrpret) en el client.

D'aquesta manera una aplicació web és una interfície d'usuari completa, mentre que un web service és una aplicació de la qual es pot accedir des d'una altre aplicació com si fos una classe o una llibreria. La capa intermèdia haurà de ser més complexa.

El disseny d'un web service ha d'estar molt ben organitzat degut a que no necessàriament hi haurà una persona humana a l'altre extrem de la comunicació per a interpretar un possible error en el seu comportament.

Els web services es diferencien d'altres models de components d'objectes existents i dels seus models de protocols d'objecte, com podrien ser COBRA i IIOP, COM i DCOM, i Java i RMI, on els component distribuïts interactuant amb protocols que no estan orientats a objecte específicament. Els webs services poden estar escrits en qualsevol llenguatge i s'hi pot accedir utilitzant HTTP (Hyper Text Transfer Protocol) que és un protocol àmpliament estès.

En la figura 1.4 extreta de la font [4], es mostren les capes que constitueixen un web service. En el seu conjunt formen la base per un mecanisme estàndard per descobrir, descriure i invocar la funcionalitat proveïda per un web service autònom.

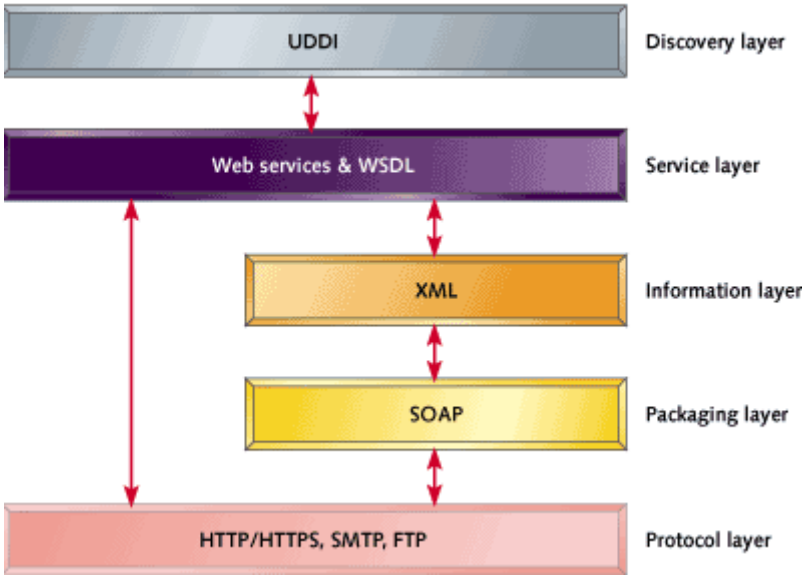

Fig. 1.4 Capes d'un web service

1.2.1. Capa de protocol

Qualsevol dels protocols estàndard d'Internet es pot utilitzar per invocar un web service a través de la xarxa. Per exemple HTTP especifica que un client pot obrir una connexió a un servidor, després enviar una petició molt especifica. El servidor respon, i si és necessari manté la connexió oberta.

Els serveis web no estan pensats només per ser utilitzats amb el protocol HTTP, d'aquesta manera es pot utilitzar SOAP a través d'altres protocols com FTP o SMTP. S'utilitza principalment HTTP per ser un protocol àmpliament difós i que es troba menys restringit per firewalls. El port FTP acostuma a està bloquejat, mentre que el port HTTP és molt probable que no ho estigui.

1.2.2. Capa d'empaquetat

SOAP és un protocol dissenyat per a intercanviar informació. Està centrat en entorns descentralitzats distribuïts. Això proveeix un marc per a invocar serveis a través d'Internet. També proveeix un mecanisme per a fer possible l' integració a través de plataformes independentment del llenguatge de programació i la infraestructura d'objectes distribuïts que utilitzin.

SOAP està basat en text (XML) i no defineix un model de programació o d'implementació. En comptes d'això defineix un model d'agrupació modular i el mecanisme de codificació de dades dins del mòduls. Això permet a SOAP ser utilitzat en un gran àmbit de sistemes, des de sistemes que passen missatges a crides remotes.

Un missatge SOAP està codificat en XML i consta de tres parts:

- Un sobre (envelope), que proveeix el marc per a empaquetar el missatge.

- Les regles de codificació defineixen com s'han de processar els missatges.

- La representació RPC (Remote Procedure Call) que defineix com representar trucades de procediments remots i respostes.

Els missatges SOAP són unidireccionals i es combinen per fer un mecanisme petició/resposta.

Els web services i SOAP són dos coses diferents. SOAP és una manera d'empaquetar informació requerida per invocar web services. És el mètode més relacionat amb els web services, però pot ser invocat utilitzant altres tècniques de codificació, com podrien ser missatges codificats URL per entorns amb ample de banda restringit. SOAP es pot utilitzar com a mecanisme d'accés a objectes remots o procediments o com a simple mecanisme per passar missatges.

1.2.3. Capa d'informació

XML is un metallenguatge que permet l' intercanvi de dades entre plataformes utilitzant un mètode estàndard per codificar i donar format a la informació. XML permet publicar informació no només sobre com estan estructurades les dades sinó sobre que significa (el seu context). Mantenir l'estructura, l'ordre del contingut, la presentació com a tres components diferents en un document XML pot aportar molts beneficis.

HTML bàsicament dona format i exposa en pantalla. XML a part d'això proporciona més informació sobre la descripció del contingut, no només de com es veu per pantalla. Aquest format dona a l' intèrpret del document una visió més fàcil d'entendre del que significa el text. Però tenir compatibilitat en XML no soluciona tots els nostres problemes, de manera que no ens integra en una aplicació host empresarial o funciona com una solució definitiva. XML no proveeix al dispositiu o a la lògica del negoci de la capacitat de decidir què s'ha de fer quan es produeixi un error concret o que aquest es captura de manera corresponent en una o més aplicacions empresarials. Aquest és el motiu pel qual el concepte a alt nivell dels web services, que està basat en XML, és tant important.

1.2.4. Capa de servei

L' interfície dels web services es defineix mitjançant el format WSDL (Web Services Description Language) basat en XML, el qual proporciona tota la informació necessària per a accedir a un servei específic d'una aplicació.

Un document WSDL és un document XML amb la descripció del web service. Elements abstractes i concrets es combinen per a definir la funcionalitat i el mecanisme d'accés d'un web service. Els elements per definir un web service són els següents:

- Definitions: és l'arrel del document XML. Defineix el nom del web service, declara els identificadors que s'utilitzen en el document i conté els elements que hi ha a continuació.

- Type: descriu tots els tipus de dades que s'utilitzen entre el client i el servidor. Normalment utilitza l'esquema W3C XML.

- Message: es una definició abstracte de les dades. Descriu un missatge exclusivament com a petició o com a resposta. Conté el nom del missatge i zero o més parts de l'element del missatge, les quals poden fer referència a paràmetres del missatge o valors de retorn.

- Operation: defineix l'operació per a un Message, com podrien ser un mètode, una cua de Messages o un procés de negoci, que acceptaran i processaran el Message.

- PortType: combina múltiples elements Message per a formar una completa comunicació de només anada (petició o resposta) o d'anada i tornada (petició i resposta en una sola operació).

- Binding: conté les especificacions exactes de com s'implementarà el servei a la xarxa. WSDL incorpora extensions per definir serveis SOAP i per tant la informació especifica de SOAP es troba en aquest element.

- Service: defineix l'adreça per a invocar el servei web. Normalment inclou una URL per a invocar el servei SOAP.

Un web service està definit en un conjunt de ports que a la vegada son una col·lecció d'operacions abstractes i missatges. Guardar les operacions i missatges abstractes permet que estiguin lligats a diferents protocols i formats de dades com SOAP, HTTP, GET/POST o MIME.

1.2.5. Capa de descobriment

Una peça opcional en els web services és el protocol UDDI (Universal Description Discovery and Integration). UDDI ofereix una manera de publicar informació sobre web services i proveeix un mecanisme per descobrir quins web services estan disponibles. És un sistema de registre instanciat com a una serie de documents XML, i un esquema associat que conté una descripció de l'entitat de negocis i el servei que ofereix.

L'especificació UDDI proveeix una interfície de programació que permet als negocis registrà web services i/o buscar a través del registre un web service

concret. Un cop que el web service desitjat es identificat, proveeix un punter a la localització del document WSDL. UDDi és totalment opcional.

En la figura 1.5 extreta de la font [5], podem observar gràficament el funcionament del protocol UDDI així com el procediment que es segueix per invocar un web service en la arquitectura SOA.

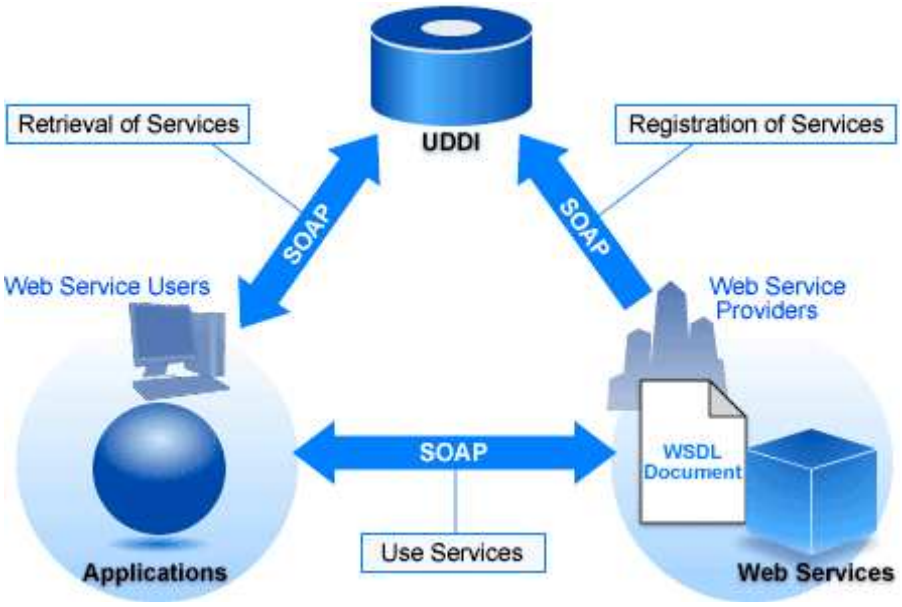

Fig. 1.5 Funcionament d'un web service

CAPÍTOL 2. TECNOLOGIES UTILITZADES

Per a triar les tecnologies a comparar, s'ha realitzat un petit recull de les tecnologies principals de les tecnologies web utilitzades per la banda del servidor, de les quals les més utilitzades són: PHP, Perl, ASP, JSP, Servlets Java, Python, Ruby, entre moltes altres.

Per a escollir les tecnologies s'han tingut en compte els següents paràmetres:

- La varietat de servidors de lliure accés per escollir i que tinguessin compatibilitat amb la tecnologia utilitzada.

- El grau de popularitat de la tecnologia en qüestió entre els usuaris i el grau d'utilització que tenen en l'entorn actual. Per a fer-ho s'ha consultat el "ranking" de la comunitat de programadors TIOBE entre d'altres.

- La disponibilitat de documentació i tutorials de qualitat en la xarxa ha sigut important degut a que el projecte està realitzat per a un estudiant que al començar no posseïa un domini de les tecnologies web.

- Un altre factor clau ha estat que el software associat a la tecnologia fos d'accés lliure i gratuït per a l'ús que li volíem donar en aquest projecte.

Tenint en compte tots aquests paràmetres al final s'han triat les tecnologies PHP i JSP per a realitzar la comparativa. El motiu pel qual s'ha volgut realitzar també un web service ha estat per a comparar quines diferències ens trobem a l'hora de realitzar una aplicació web i un web service, veure quines dificultats podem trobar en aquest procés i tenir-ho en compte quan s'hagi de decidir si realitzar un web service o una aplicació web simple.

Els mateixos paràmetres s'han tingut en compte per a escollir el servidor que hem utilitzat com a entorn per a desenvolupar les aplicacions, motiu pel qual s'ha escollit Apache Tomcat.

2.1. PHP

PHP es un llenguatge de programació interpretat, dissenyat originalment per a la creació de pàgines web dinàmiques. S'utilitza principalment en interpretació en el servidor (server-scripting) però actualment pot ser utilitzat de més maneres.

PHP significa Hypertext Pre-processor. És un llenguatge de propòsit general àmpliament utilitzat i que està dissenyat especialment per al desenvolupament web i pot ser incrustat dins d'HTML. Generalment s'executa en un servidor web, prenent el codi PHP com a entrada i creant pàgines web com a sortida.

Pot ser desplegat en la majoria de servidors web i en casi tots els sistemes operatius i plataformes sense cap cost.

Entre els principals avantatges de PHP trobem que és open source , de manera que és accessible per a tothom. Això significa que pot ser utilitzat de manera gratuïta en el cas que estiguis realitzant una tasca sense ànim de lucre.

Un altre dels importants avantatges que ens ofereix és que com que s'executa en el servidor no necessita que el client hagi de realitzar cap tipus d'instal·lació addicional en el seu equip. Interactua fàcilment amb el servidor Apache/MySQL i la majoria de servidors estan preparats per a utilitzar PHP i configurar les seves opcions. Es una tecnologia de fàcil accés per a qualsevol persona que disposi de connexió a Internet, es troba fàcilment informació a la xarxa, així com, exemples de codi, llibres online o varietat de tutorials per aprendre a utilitzar les seves funcionalitats. Es tracta d'un llenguatge multi-plataforma de manera que es pot utilitzar sense problemes en plataformes Windows, Linux o Mac a part de ser molt escalable.

Alguns desavantatges de utilitzar PHP pot ser que si volem crear alguna cosa més complicada que una pàgina HTML/CSS s'haurà d'utilitzar també java, javascript o altres llenguatges del costat del client.

2.2. Java server pages (JSP) i servlets Java

Un servlet bàsicament és un programa que s'executa en un servidor o un contenidor de servlets i està específicament dissenyat per a oferir contingut dinàmic des d'un servidor web, generalment HTLM. Està escrit en Java com el seu nom indica.

Per altre banda, JSP es una tecnologia similar que també permet generar contingut dinàmic per a webs, en forma de HTML, xml o altres tipus. JSP ofereix una forma d'agregar contingut dinàmic a un arxiu HTML per utilitzar codi escrit en Java dins de l'arxiu utilitzant etiquetes especials que són processades pel servidor Web abans de ser enviades al client.

La diferència entre Servlets i JSP és que els Servlets són programes amb Java purs, classes que han d'implementar la classe abstracte HttpServlet, en especial el mètode doGet() o doPost() i han de ser prèviament compilats, mentre que els arxius JSP contenen codi Java entre el codi HTML utilitzant els símbols <% i %>. Per aquest motiu l'arxiu JSP ha de ser interpretat pel servidor en el moment de la petició del client. JSP facilita la creació de servlets ja que afavoreix la separació entre la part que genera el contingut (Java) i la part que el mostra (JSP).

Quan es crida una pàgina JSP aquesta serà compilada com un Java Servlet (pel motor JSP). A partir d'aquí el servlet és controlat pel motor Servlet, com qualsevol altre servlet. D'aquesta manera es carrega la classe servlet i

s'executa per a crear contingut dinàmic HTML. Aquest procés el podem observar en la figura 2.1 extreta de la font [12]:

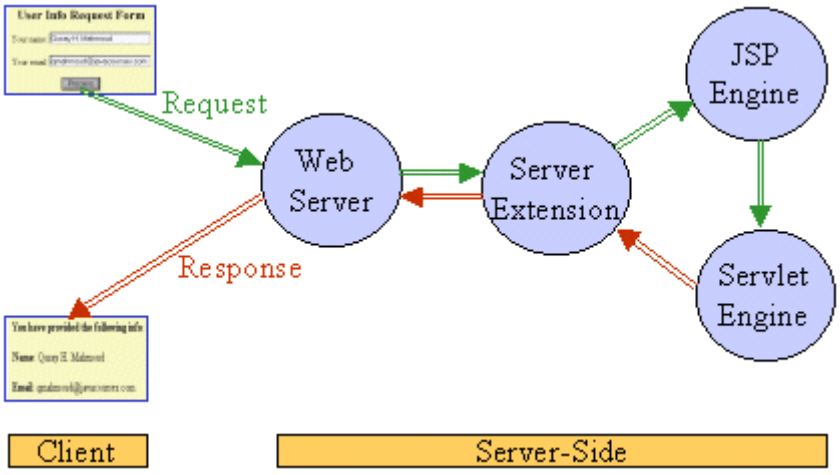

Fig.2.1. Crida d'una pàgina JSP

Un dels seus principals avantatges és que utilitza el llenguatge de programació Java i per tant pot executar les aplicacions en múltiples plataformes i en qualsevol entorn que contingui la màquina virtual de Java sense realitzar canvis. JSP es pot comunicar directament amb el servidor Web de manera que les operacions de localització d'arxius o carpetes es pot realitzar de manera més senzilla. La majoria de servidors tenen suport per a servlets o es poden modificar via "plug-in" per a poder utilitzar-los. En el projecte s'ha utilitzat Tomcat que té suport per a aquest tipus de tecnologia. Al igual que PHP s'executa en el servidor de manera que no necessita de instal·lacions addicionals en el client.

2.3. Web Service

"Tota la informació disponible per a qualsevol persona, a qualsevol lloc, a través de qualsevol dispositiu." Frase extreta de la font [14]

Els web services són components software que permeten als usuaris utilitzar aplicacions de negoci que comparteixin dades amb altres programes modulars, via Internet. Son aplicacions independents de la plataforma que poden ser fàcilment publicades, localitzades i invocades mitjançant protocols web estàndard, com XML,SOAP, UDDI o WSDL. L'objectiu final es la creació d'un directori online de web services, que pugui ser localitzat de manera senzilla i que tingui una alta fiabilitat.

La funcionalitat dels protocols empleats es la següent:

- XML(extensible Markup Language): és un metallenguatge extensible d'etiquetes. No és realment un llenguatge en particular, sinó una manera de definir llenguatges per a diferents necessitats. Es pot utilitzar per a intercanviar informació estructurada entre diferents plataformes. Un servei web es una aplicació creada en XML.

- WSDL (Web Services Definition Service): Aquest protocol s'encarrega de descriure el web service quan es publica. Es el llenguatge XML que els proveïdors utilitzen per a descriure els seus web services. El client pot consultar el fitxer WSDL per determinar les funcions disponibles en el servidor del web service al que s'hagi connectat.

- SOAP (Simple Object Acces Protocol): Permet que programes que corren en diferents sistemes operatius es comuniquin. La comunicació entre les diferents entitats es realitza mitjançant missatges XML que son enviats en un sobre SOAP.

- UDDI (Universal Description Discovery and Integration): Aquest protocol permet la publicació y localització dels serveis. El seu objectiu és que s'hi accedeixi mitjançant missatges SOAP i donar pas a documents WSDL, en els que es descriguin els requisits del protocol i els formats del missatge sol·licitat per a interactuar amb els serveis Web del catàleg de registres. Els directoris UDDI actuen com una guia telefònica de web services. El registre d'un negoci en UDDI té tres parts: les pàgines blanques (direcció, contacte i altres identificadors coneguts), pàgines grogues (categorització industrial basada en taxonomies) i pàgines verdes (informació tècnica sobre els serveis que aporten les pròpies empreses)

La figura 2.2 extreta de la font [15], mostra un exemple de la utilització d'aquests protocols en un web service que proporciona una agència de viatges que interactua amb d'altres webs services.

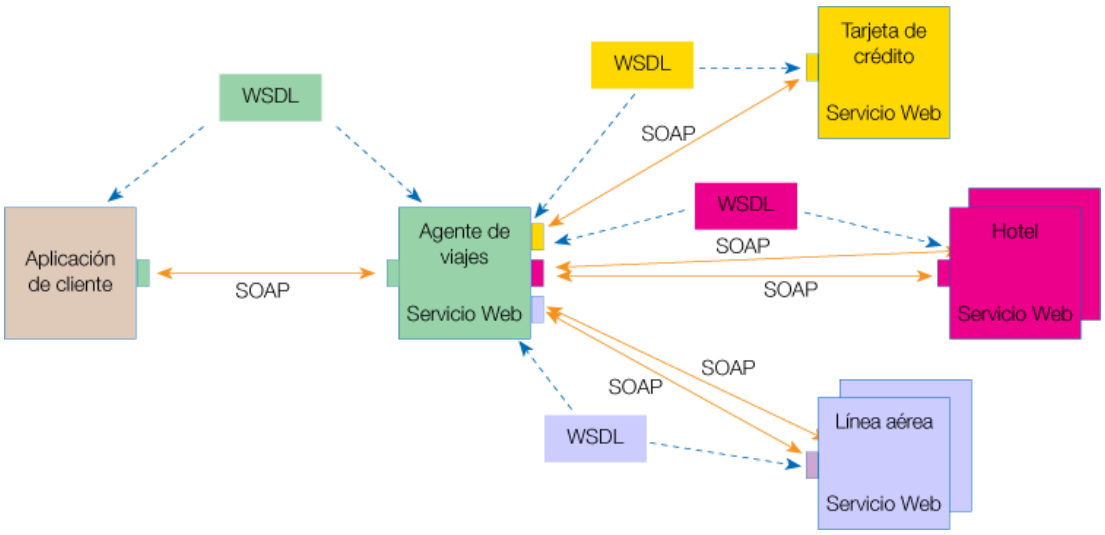

Fig. 2.2 Exemple de servei web

El principal objectiu que s'aconsegueix, es la interoperabilitat i la integració entre aplicacions de software independentment de les seves propietats o de les plataformes sobre les que s'instal·li. Els serveis Web fomenten l'ús d'estàndards i protocols basats en text, que fan més fàcil accedir al seu contingut I entendre el seu funcionament. Al estar recolzat en HTTP, els serveis Web poden aprofitar-se dels sistemes de seguretat firewalls sense necessitat de canviar les regles de filtrat. També permeten que serveis i software de diferents companyies ubicades en diferents llocs geogràfics puguin ser combinats fàcilment per a proveir serveis integrats. Els serveis Web permeten la interoperabilitat entre plataformes de diferents fabricants mitjançant protocols estàndard i oberts. Les especificacions són gestionades per una organització oberta, W3C, per tant no hi ha interessos particulars d'empreses i es garantitza la plena interoperabilitat entre aplicacions.

Un dels principals inconvenients que pot presentar aquesta tecnologia és que entre els objectius del format basat en text que utilitzen els web service, no s'hi troba la concisió ni l'eficàcia de processament. D'aquesta manera podem trobar altres models de computació distribuïda més eficients per a aquest propòsit.

2.4. Tomcat

Apache Tomcat és un servidor web HTTP creat per la "Apache Foundation". Les seves característiques principals són que està escrit en java i té suport per a Servlets i JSP (java server pages). També pot ser utilitzat com a "plug-in" (annexa) de servidors de codi natiu HTTP, com podria ser el servidor web Apache, per proveir-los de suport per a Servlets.

Tomcat és open source de manera que es pot obtenir de manera gratuïta i es pot utilitzar lliurement sempre que sigui sense ànim de lucre.

Les proves realitzades en aquest projecte han estat executades sobre Tomcat, utilitzat com a servidor i contenidor de Servlets, tot i que com s'explica posteriorment en el capítol 4 també s'ha utilitzat el servidor HTTP Apache.

CAPÍTOL 3. COMPARACIÓ DE LES TECNOLOGIES.

3.1. Programació

Una de les principals diferències de programació entre PHP i Java la trobem en la declaració de variables. A diferencia de Java, en PHP no cal declarar el tipus de la variable, simplement es poden utilitzar sobre la marxa i poden contenir qualsevol tipus de dades en moments diferents de l'execució. Java és un llenguatge orientat a objectes, mentre que PHP permet mesclar classes amb funcions de programació estructurada (scripts).

A l'hora d'implementar funcions utilitzant PHP cal tenir en compte que utilitzant aquest llenguatge podem tenir problemes si no som molt ordenats, ja que no apareix el tipus de paràmetre que se li ha de passar a la capçalera i s'ha de mirar la funció amb detall per a saber què retorna o si retorna alguna variable. En aquest sentit Java és més ordenat ja que deixa clar quin tipus de dades espera la funció com a paràmetre i què retorna, sense tenir que mirar detingudament el codi intern de la funció.

Al tenir un grup de desenvolupadors elevat aquest fet podria ser un tema a tenir en compte, tot i que es podria resoldre amb una disciplina estricte i comentaris adequats. El tipus de les variables també pot donar lloc a problemes a l'hora trobar errors en l'etapa de depuració utilitzant PHP, ja que el programa és possible que s'executi i simplement no doni el resultat esperat. Utilitzant Java simplement no compilarà el programa i és més ordenat ja que està tot orientat a objectes.

Per altre banda, tenint nocions de programació en C, començar a programar en PHP és un procés senzill al que l'adaptació és ràpida. Per contra per començar a programar en Java es necessita a priori familiaritzar-se amb la sintaxis de Java i les llibreries per a programar orientat a objectes.

Si el que volem implementar és una aplicació purament web, en la que principalment hi ha presentació en navegador i transaccions amb una base de dades, el nombre de desenvolupadors no és molt elevat i el temps del que disposem és ajustat, PHP serà una opció a tenir en compte.

Per contra, si volem crear una aplicació complexa, en la que pot haver-hi més codi, més algoritmes, a part de la presentació en el navegador i transaccions en base de dades, amb un nombre considerable de desenvolupadors amb un temps força prolongat, llavors la opció a priori més adequada seria JSP/Servlets.

3.2. Els models d'execució

Comparat amb Java, el desenvolupament i hosting dels projectes PHP és molt més simple degut al model d'execució.

En una configuració típica, cada una de les peticions a aplicacions PHP és controlada per un procés d'Apache separat que utilitza la seva pròpia instància de l' intèrpret PHP. Després de processar la petició, el procés s'acaba d'executar i no ocupa recursos en el sistema innecessàriament. Aquest procés sembla ineficient per un ús altament concurrent, però funciona eficientment en un entorn de hosting compartit on diferents aplicacions comparteixen màquina i recursos. Si una aplicació no té cap petició, no utilitza espai en memòria. El temps de desenvolupament per a cada script PHP està escrit com si totes les instàncies d'scripts (procés) fossin les úniques que s'estan executant.

El model d'aplicacions web Java utilitza servlets i múltiples threads per manejar les peticions. Això proporciona una bona escalabilitat, d'aquí el seu èxit en el món empresarial. Això és degut que en la majoria de sistemes operatius els threads consumeixen menys recursos que els processos. D'aquesta manera és més eficient utilitzar múltiples threads que múltiples processos. El problema succeeix quan volem proporcionar moltes aplicacions petites a les que s'accedeix de manera menys freqüent en el mateix contenidor Web, precisament per utilitzar threads. En aquesta situació seria més adequat utilitzar múltiples processos que podem finalitzar independentment. Els desenvolupadors han d'estar molt alerta amb el tema de la concurrència.

El procés de control de mecanismes disponible a nivell de Sistema Operatiu són molt superiors als disponibles pels threads java. El resultat es que un proveïdor de hosting té un control estricte sobre els recursos utilitzats per a cada petició PHP. Per altre banda, un thread Java donant suport a una petició és un objecte que no pots controlar un cop ha començat: es para quan vol, utilitza tants recursos del sistema com necessiti.

El resultat final es que hi han limitacions tècniques en configurar una solució per a un hosting Java amb un preu competitiu i fiable. Java seguirà sobrevivint per a aplicacions web, però si que es veritat que si les noves generacions es senten agust amb l'open-source PHP, és qüestió de temps que aquesta tecnologia s'introdueixi en el món empresarial.

3.3. Comparació d'utilització en la realitat

Tot i les diferències que hem apuntat en els apartats anteriors trobem grans projectes web amb un nombre elevadíssim de visites que estan programats en PHP, en contra del que es podria pensar en principi. Algunes de les webs més populars i visitades realitzades en PHP són Facebook.com, Wikipedia.org i Tuenti.com.

Per altre banda, consultant pàgines molt populars que utilitzen Java com a llenguatge principal de programació trobem pàgines com Amazon.com, Ebay.com, Ingdirect.es, Google.

És inqüestionable que les pàgines esmentades tenen un volum molt important de visites i treballen amb un gran volum de dades. Però s'aprecia que entre les pàgines més populars que utilitzen PHP no s'hi troben empreses bancàries sinó bases de dades grans i xarxes socials. Mentre que les pàgines més populars que utilitzen java tenim empreses bancàries i pàgines que inclouen transaccions. El que ens fa pensar això, és que pàgines que necessiten d'una capa lògica de negoci que hagi de realitzar operacions més complicades, utilitzen java per davant de PHP.

Per comparar la popularitat entre els internautes de les dos tecnologies, podem observar la figura 3.1 extreta de la font [23], que ens mostra una gràfica on podem veure els llenguatges més buscats en els principals buscadors d'Internet (gràfica actualitzada a 29 de desembre de 2009) . Java és una de les tecnologies més populars a dia d'avui però no queda molt clara la tendència que en un futur proper pot tenir la popularitat de PHP.

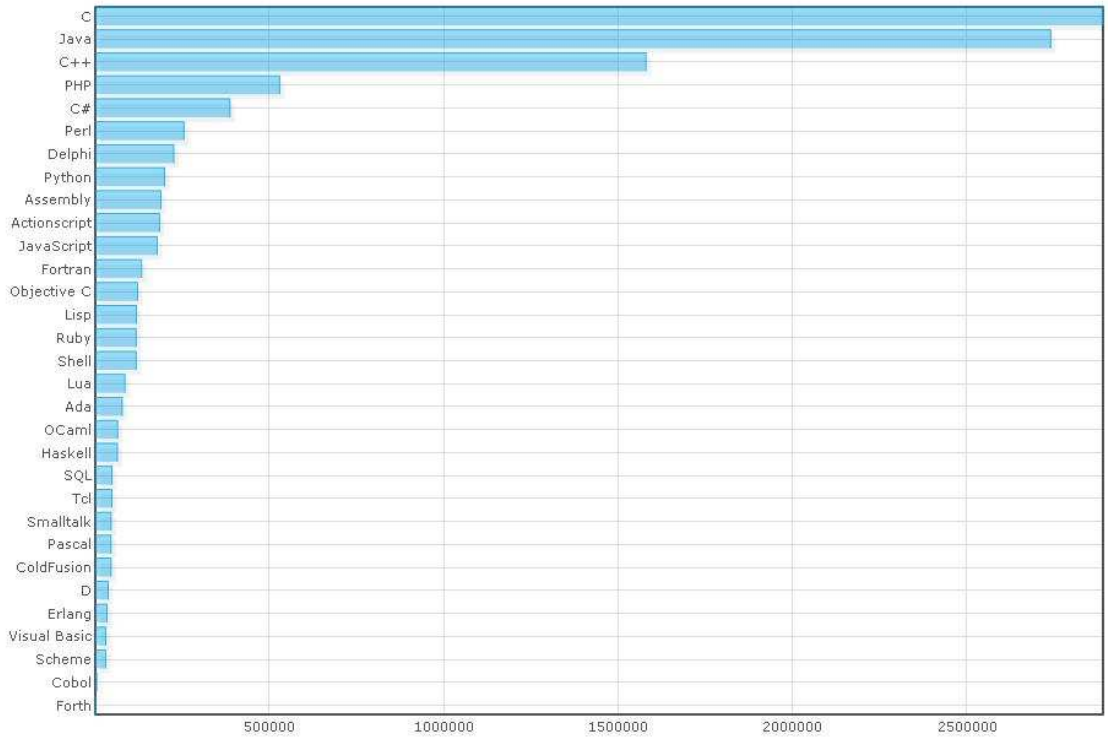

Fig. 3.1 Llenguatges més buscats a la xarxa

CAPÍTOL 4. DESCRIPCIÓ DE LES APLICACIONS

Per poder comparar de manera pràctica el funcionament de les tecnologies esmentades en aquest projecte s'ha procedit a realitzar una aplicació que pujava un fitxer de qualsevol tipus de dades des d'una màquina usuari a una màquina servidor.
Per emmagatzemar les aplicacions i realitzar les proves s'ha utilitzat el servidor Tomcat 4.0 que s'ha establert com a l'entorn per a desenvolupar totes les tecnologies sobre el mateix marc.
A continuació mostrem el detalls del treball realitzat en cada una de les tecnologies.
En primer lloc s'ha instal·lat el servidor Tomcat 4.0 obtingut de la pàgina web http://tomcat.apache.org/ de manera gratuïta.

4.1. Java servlet / JSP

En la figura 3.1 es mostra el funcionament que es vol obtenir de l'aplicació:

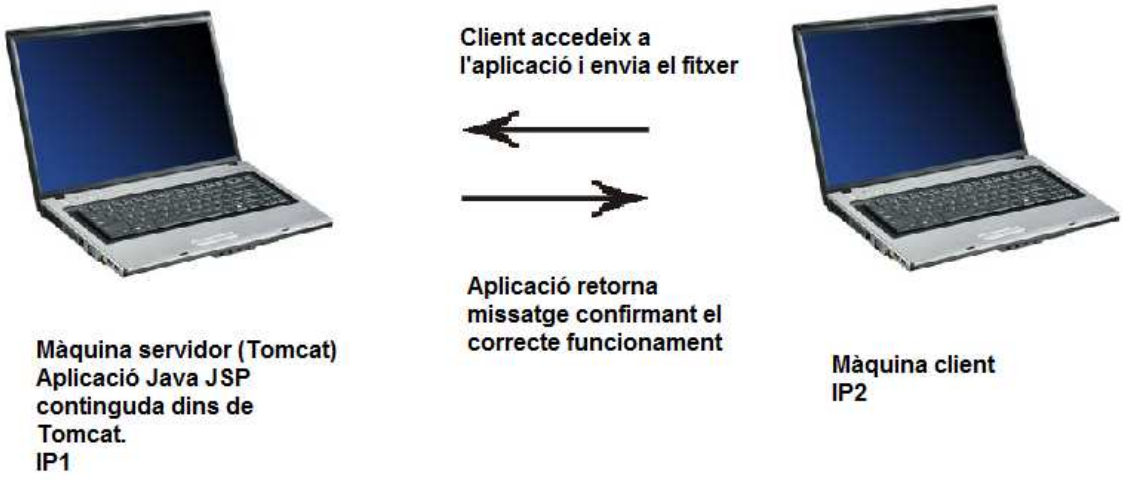

Fig. 3.1 Model de funcionament utilitzant Java

Per a realitzar l'aplicació web amb el llenguatge Java, en primer lloc s'ha de descarregar de la pàgina oficial de Sun Microsystems la versió més recent de la màquina virtual java. Com he esmentat anteriorment una de les principals funcions de Tomcat és que s'utilitzi com a contenidor de servlets Java. D'aquesta manera no he trobat problemes de compatibilitat de la màquina virtual Java amb el servidor que utilitzo. Després d'instal·lar els dos components s'ha de modificar la variable d'entorn Classpath per a incloure les

llibreries Java i que siguin detectades per a l'aplicació al ser executada en el servidor.

El primer pas realitzat ha estat escriure un formulari JSP especificant que volem utilitzar el mètode POST, l'encriptació multi part/form-data i el nom del servlet (.java) que s'ha d'executar quan confirmem la pujada.

En segon lloc s'ha hagut d'escriure una aplicació o servlet Java, que a partir d'un formulari al qual introduirem la ruta del fitxer a enviar, enviarà el fitxer al servidor utilitzant el mètode POST.

El problema que en principi trobem és que JSP i els servlets java no tenen mètodes per a poder pujar un fitxer al servidor a partir d'un formulari. Això converteix en un problema poder extreure el contingut des d'una petició HTTP. Per a solucionar aquest inconvenient s'ha trobat que existeix una llibreria robusta de codi obert dissenyada per a aquest propòsit del paquet "Apache Jakarta Commons Fileupload". Obtenim aquest paquet descarregant-lo de manera gratuïta de la pàgina http://commons.apache.org/fileupload/ . D'aquest paquet obtenim la llibreria commons-fileupload-1.2.1.jar que conté les funcions necessàries per a fer possible la pujada del fitxer. Per a fer funcionar la llibreria esmentada necessitem una altre llibreria anomenada commons-io-1.4.jar que s'utilitza com a complement intern.

Posteriorment s'ha creat el servlet, que envia el fitxer al servidor. Podem graduar alguns paràmetres d'aquesta pujada com són el mida màxim del fitxer o tenir en compte que hi pot haver un fitxer amb el mateix nom que el que nosaltres volem pujar a la carpeta on es guarda, entre d'altres coses. Cal esmentar que el temps màxim de pujada d'un fitxer es configura en el server.xml que es troba dins la carpeta de configuració del Tomcat. Aquest paràmetre pot ser que s'hagi de modificar per a fitxers pesats. L'aplicació retorna una pàgina HTML per a confirmar que el fitxer s'ha rebut correctament.
Per a què l'aplicació pugui funcionar sobre el servidor Tomcat es necessita crear el fitxer web.xml que proporciona informació sobre els components web que formen part d'una aplicació web. Els exemples dels components web són els paràmetres del servlet, les definicions de servlets i JSP i la correlació d'URLs. En aquest fitxer hi podem especificar paràmetres que volem poder modificar sense tenir que modificar el codi del nostre servlet. En aquest cas el paràmetre de mida màxima del fitxer a pujar, es pot modificar des del web.xml, així com el directori dins de la màquina servidor on es vol que es guardin els fitxers que pugem. D'aquesta manera no cal que entrem en la nostre aplicació i modifiquem el codi per a realitzar aquests tipus de canvis.

Hem de guardar el formulari JSP, el fitxer web.xml i el propi servlet compilat (.class) dins d'una carpeta que posarem dins la carpeta "webapps" del directori del Tomcat.

Per provar el correcte funcionament de l'aplicació cal executar el Tomcat i accedir al port local on estiguem executant el Tomcat, en aquest cas el port 8080 que és el que el Tomcat utilitza per defecte. Obrim el navegador i accedim al port esmentat i en concret a la nostre aplicació escrivint la URL corresponent,

en aquest cas: http://localhost:8080/web/upload.jsp. Veurem el que es mostra en la figura 3.2.

Fig. 3.2 Finestra inicial de l'aplicació Java

Quan se'ns obri la pàgina només hem d'introduir el fitxer que volem pujar al formulari i fer un clic sobre "Tramet la consulta". Si tot funciona correctament ens sortirà una pàgina web que confirma el correcte funcionament com podem observar a la figura 3.3.

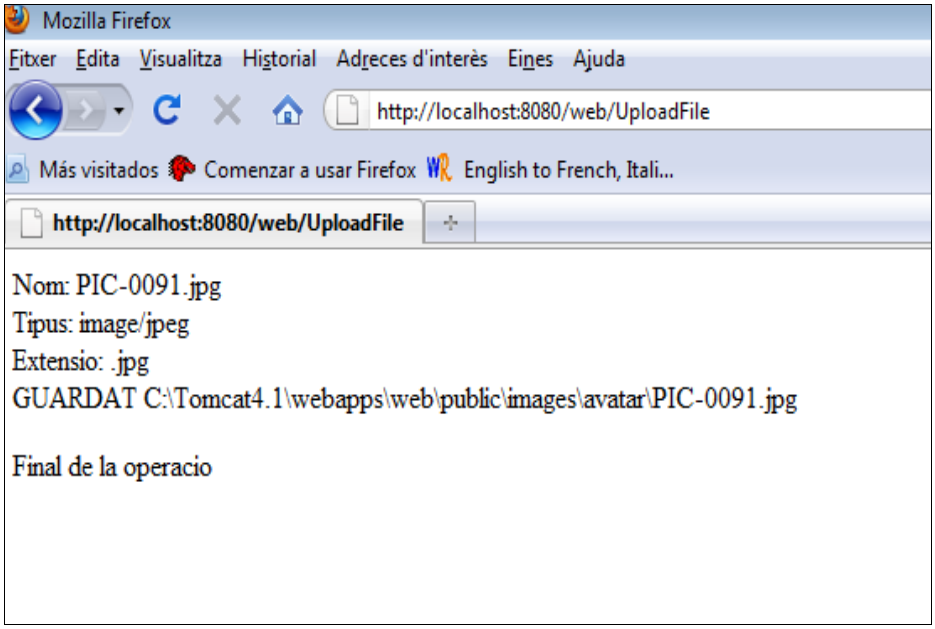

Fig. 3.3 Finestra amb el missatge de confirmació

4.2. PHP.

En la figura 3.4 es mostra el funcionament que es vol obtenir de l'aplicació:

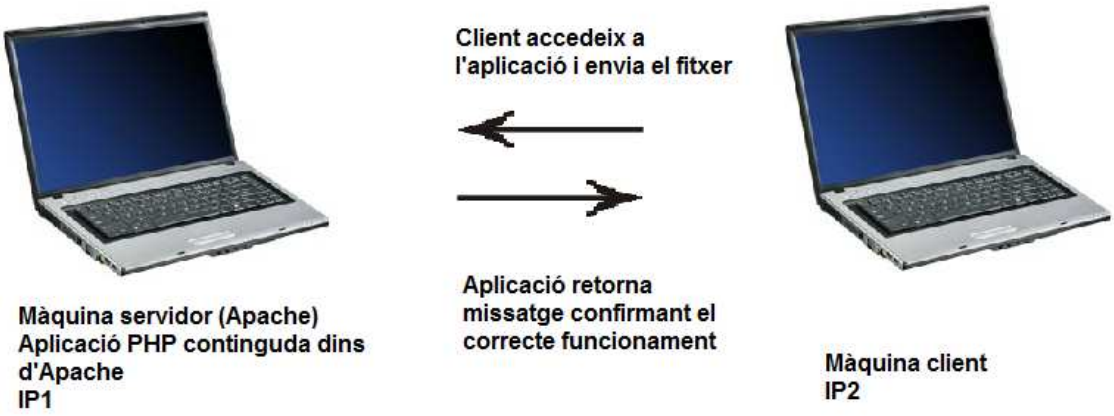

Fig. 3.4 Model de funcionament utilitzant PHP

En primer lloc he descarregat la versió més recent de PHP des de www.phpw.net . El problema principal a abordar en realitzar una aplicació PHP per a pujar un fitxer des d'una màquina client al servidor ha estat que Tomcat no té suport per a PHP de manera que s'ha hagut de realitzar una recerca en aquest aspecte per a aconseguir que l'aplicació es pogués executar sobre Tomcat.

En primer lloc s'ha intentat fer el que proposava en la pàgina web http://nexus.zteo.com/2008/02/15/how-to-setup-php5-with-tomcat-5/ on bàsicament es crea una aplicació web característica on el seu fitxer web.xml , en principi, li proporciona les característiques necessàries per a executar fitxers en format PHP. Al dur-ho a la pràctica es pot observar que es poden utilitzar variables d'entorn PHP i variables d'entorn Windows però les variables HTTP_ no es poden utilitzar de cap manera. Per a realitzar l'aplicació es necessiten aquestes variables ja que les funcions que utilitzem per a pujar un fitxer al servidor són variables HTTP_.

En segon lloc s'ha provat d'utilitzar un servlet PHP extret de la pàgina http://tools.herberlin.de/phpservlet/index.shtml . La opció que proposa la pàgina és un servlet PHP que és una implementació pura amb Java que no utilitza llibreries natives. Aquesta opció també ha estat descartada degut a que ens interessa realitzar les proves en un entorn PHP com a mòdul d'un servidor o com a CGI.

Finalment s'ha optat per a instal·lar el servidor HTTP Apache i utilitzar PHP com a mòdul d'Apache ja que aquest servidor si que té suport per a PHP.

S'ha de tenir en compte que per a poder pujar fitxers més pesats, s'han de configurar les variables "post_max_size" i "max_execution_time", en el fitxer php.ini, a valors més elevats que ens puguin interessar. El fitxer php.ini es troba en la carpeta on tinguem instal·lada la nostra versió de PHP.

L'aplicació creada bàsicament consisteix en una pàgina HTML amb un formulari on especificarem la ruta del fitxer que vulguem pujar al servidor i que executa el mètode POST per a realitzar l'enviament. Quan fem clic sobre el botó "Submit" s'executa el POST i l'aplicació que hem creat, en aquest cas d'extensió .php.

Per a escriure l'aplicació que ens permet pujar el fitxer al servidor no cal descarregar cap llibreria addicional ja que la versió més recent de PHP ja ens proporciona les eines necessàries per a realitzar aquesta tasca. Les funcions que utilitza són les funcions "file" . Aquestes ens faciliten molt la feina a l'hora de a pujar el fitxer i controlar els paràmetres d'aquesta pujada com són: la mida màxima del fitxer, ubicació dins la màquina servidor on s'ha de descarregar el fitxer o programar restriccions en el tipus de fitxer que volem que es puguin pujar (.jpeg, .gif, .mp3) etc.

Per provar el correcte funcionament de l'aplicació cal executar Apache i accedir al port local on estiguem executant Apache, en aquest cas el port 8080. Obrim el navegador i accedim al port esmentat i en concret a la nostre aplicació escrivint la URL corresponent, en aquest cas: http://localhost:8080/phpservlet/upload.html . Al obrir aquesta URL veurem el que es mostra en la figura 3.5.

Fig. 3.5 Finestra inicial de l'aplicació PHP

Quan se'ns obri la pàgina només hem d'introduir el fitxer que volem pujar al formulari i fer un clic sobre "Tramet la consulta". Si tot funciona correctament ens sortirà una pàgina web que confirma el correcte funcionament, com es mostra en la figura 3.6.

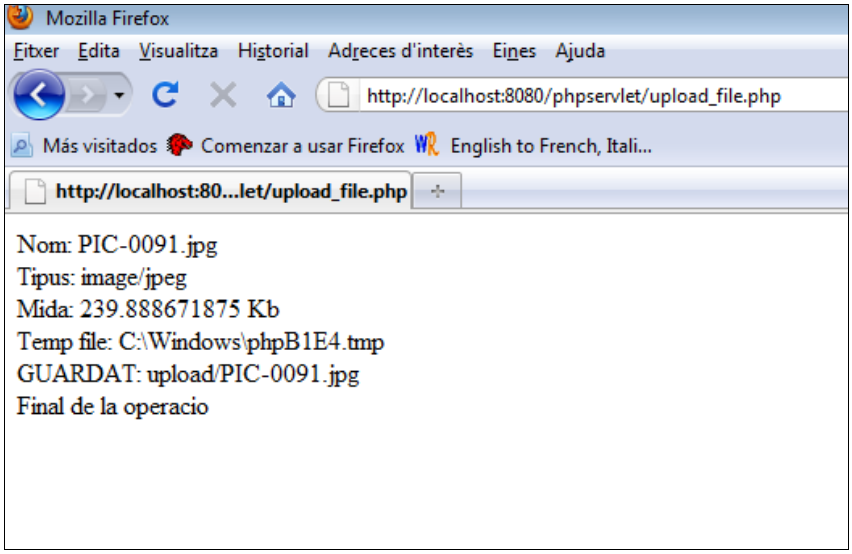

Fig. 3.6 Finestra amb el missatge de confirmació

4.3. Web service

Abans d'explicar com s'ha procedit a realitzar un web service per pujar un fitxer al servidor faré un parèntesis per parlar de les eines utilitzades per a dur a terme aquesta tasca.

4.3.1. Eines utilitzades

4.3.1.1. Apache Axis

Com a client d'un web service, codificar la teva petició per al web service i descodificar les respostes que t'arribin d'aquests podria arribar a ser un procés problemàtic, per no anomenar la implementació de la lògica per acceptar peticions i enviar respostes. Ens trobem amb el mateix problema si som nosaltres els que volem escriure el web service. Axis bàsicament realitza aquests processos per a nosaltres. Es podrien escriure web services sense Axis però seria un procés molt més complicat.

Axis és una implementació del protocol SOAP i ens ajuda a l'hora de manejar-nos amb els protocols SOAP i amb WSDL. Axis està desenvolupat per la "Apache Software Foundation", és open source i es pot obtenir de la pàgina http://ws.apache.org/axis/. S'utilitza en el costat del servidor per a escriure el web service i desenvolupar-lo com a aplicació web dins del Tomcat. Genera

automàticament l'WSDL per al web service i per aquest motiu clients escrits en qualsevol llenguatge poden consumir aquest web service. En el costat del client ens facilita molt la feina a l'hora d'escriure el client. Tot el que has de fer és realitzar les crides a mètodes a l'objecte web service com si fossin objectes locals.

Llavors el que has de fer bàsicament per a desenvolupar un web service es instal·lar el teu web service com si fos una aplicació web del Tomcat i el client hi pot accedir mitjançant una línia de comandes de programació Java.
En aquest cas, s'ha utilitzat Axis 2 per a desenvolupar i executar el web service.

4.3.1.2. Apache Ant.

Ant és una eina de open source, està desenvolupat per la "Apache Software Foundation" i es pot obtenir de la pàgina http://ant.apache.org/. S'utilitza en la compilació i creació de programes Java. Està escrit en Java i està basat en arxius de configuració XML per a realitzar les diferents funcions.

Altres eines de les mateixes característiques com podrien ser make (Linux), gnumake, jam i altres, presenten algunes desavantatges degut a que estan configurades en base al sistema operatiu de l'entorn, a més que tenen arxius de configuració poc descriptius.

El fet de que ant estigui escrit en Java ens proporciona interoperabilitat a nivell de sistema operatiu. La utilització de XML ens proporciona configuracions més descriptives.

Una eina de construcció (build) d'aquestes característiques permet al desenvolupador descriure el procés de construcció. Aquest fet és important en la mesura que ens interessa realitzar alguns passos abans que d'altres o inclús executar-ne d'altres independentment.

En resum ant és una eina de construcció que proveeix un mecanisme per a descriure i estructurar les operacions en el procés de construcció (build). Quan s'invoca utilitza aquesta descripció, examina l'estat actual del cas per determinar els passos que s'han d'executar i en quin ordre, i per suposat manejar l'execució d'aquests passos.

En aquest cas s'ha utilitzat ant per a construir el web service i per a executar el client.

4.3.2. Descripció de l'aplicació

Un cop introduïdes aquestes eines, anem a veure el diagrama de l'aplicació que volem fer, observant la figura 3.7:

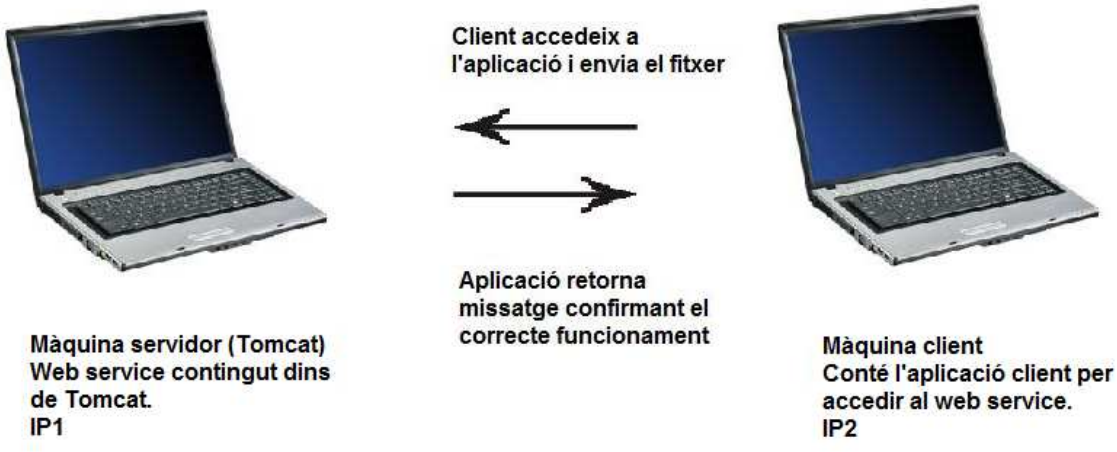

Fig. 3.7 Model de funcionament del Web service

En primer lloc s'ha descarregat l'arxiu axis2.war de la pàgina web oficial http://ws.apache.org/axis2/ . Es posa aquest arxiu en la carpeta webapps del Tomcat. Per comprovar si s'ha fet el pas correctament, s'executa el Tomcat, obrim el navegador i comprovem la URL següent: http://localhost:8080/axis2/. Si el procés s'ha realitzat correctament veurem la figura 3.8.

Fig. 3.8 Pàgina principal de Axis 2

Per a desenvolupar el codi del costat del servidor hem de fer clic sobre l'apartat "Administration" de la pàgina mostrada. Axis2 ens demana un nom d'usuari i una contrasenya per a accedir a aquest apartat. Per defecte el nom d'usuari és "admin" i la contrasenya "axis2". Un cop dins del mòdul d'administració d'Axis2 podem desenvolupar el nostre servei utilitzant l'apartat "Upload Service". Es necessita el fitxer .class generat a partir del codi de la part del servidor, i el fitxer services.xml empaquetats dins un fitxer .aar . El fitxer services.xml bàsicament conté el nom del servei, una petita descripció d'aquest, paràmetres definits i els noms de les operacions que realitza el servei.

Per a crear el fitxer d'extenció .aar hem de comprimir els fitxers esmentats services.xml i el .class com si volguéssim crear un fitxer .jar utilitzant la línia de comandes (jar –cvf). Un cop creat el fitxer .jar només tenim que canviar l'extensió a .aar de manualment. Després de fer aquestes passes ja podem carregar el nostre servei utilitzant la opció "Upload Service" d'Axis 2.

Per comprovar que el servei s'hagi desenvolupat correctament es pot fer clic sobre l'apartat Services de la pàgina principal d'Axis2. Fent això veurem la llista de serveis desenvolupats com podem observar en la figura 3.9. En la llista de serveis desenvolupats s'hi hauria de trobar el nostre servei sense cap error. Fent Clic sobre el nostre servei, veurem el codi del fitxer .wsdl generat per Axis 2 corresponent, com podem observar a la figura 3.10.

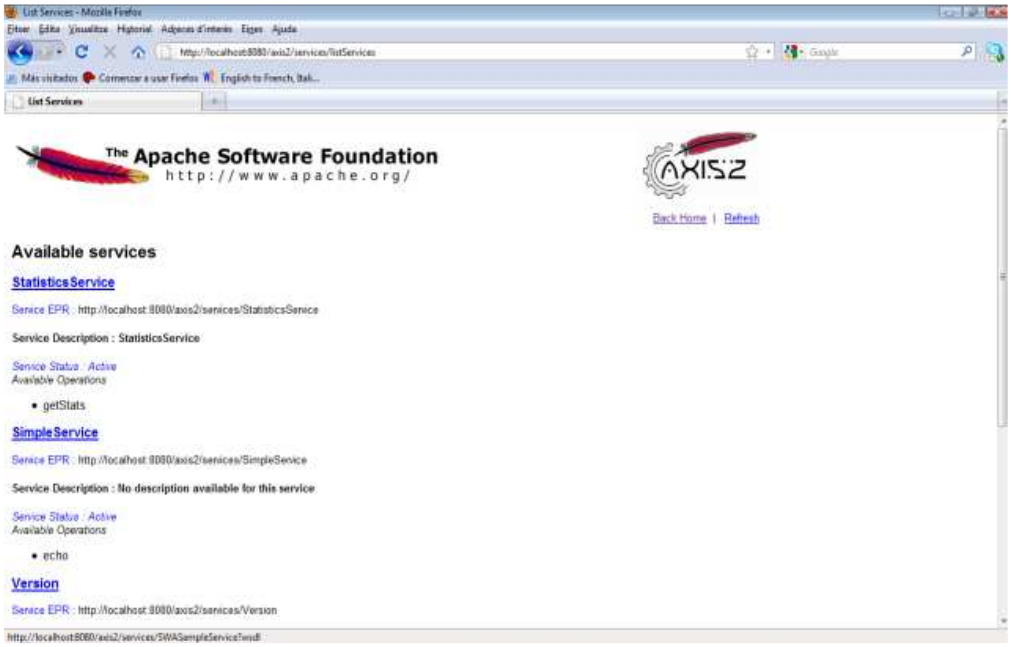

Fig. 3.9 Llista de serveis continguts a Axis 2

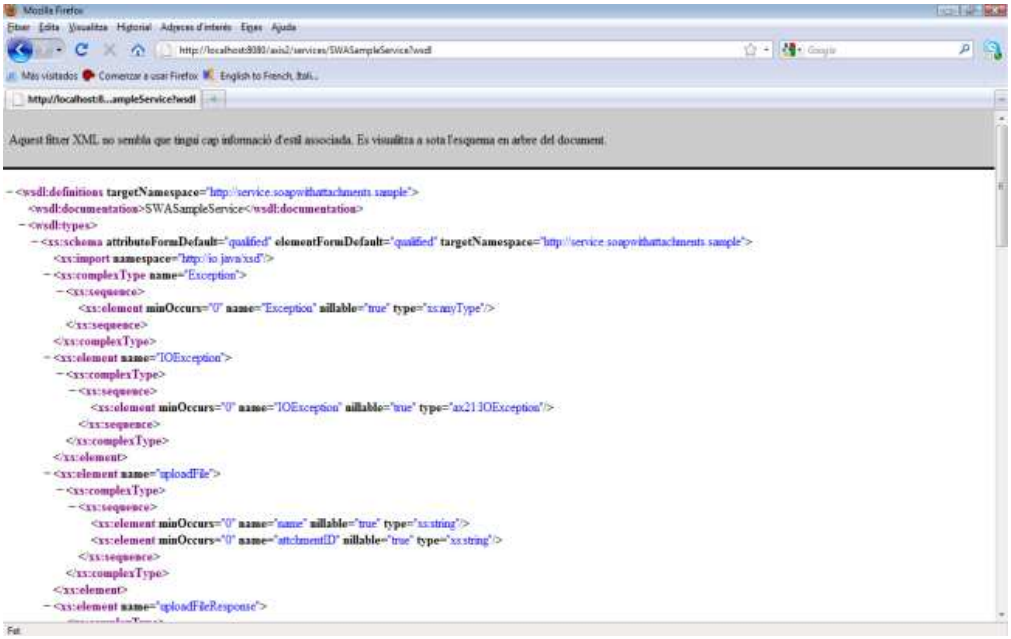

Fig. 3.10 Fitxer .wsdl generat per Axis 2

En el costat del client, s'ha hagut de descarregar la versió binària de Axis2 ja que es necessiten llibreries que conté aquest paquet. S'han d'afegir les llibreries corresponents la variable d'entorn Classpath per a poder compilar el nostre codi correctament i Java ens reconegui les llibreries corresponents.

També es necessita descarregar la versió binària d' Apache Ant i descomprimir el contingut en la ubicació més adequada en la nostra màquina. Per a què es pugui executar Ant des de qualsevol carpeta del sistema haurem d'afegir la

carpeta bin en la variable d'entorn Path del nostre sistema. Per comprovar que el nostre sistema detecta la presència d'Ant podem teclejar "ant -version" en la línia de comandes i veure la versió instal·lada.

Ant compila tot el codi client i servidor, i crea l'estructura de fitxers necessària per a crear el web service en una sola execució. Això és possible mitjançant l'execució d'un fitxer build.xml que indica els passos a seguir per a construir el nostre web service. Aquest fitxer és molt útil per a desenvolupar el servei de manera ràpida en una altre màquina. En el fitxer build.xml també hi ha un mòdul creat per a executar el client especificant la ruta del fitxer a pujar al servidor i el nom que volem que tingui un cop guardat en el servidor.

Després d'haver fet els passos indicats i compilar el codi de la part del client podem passar a comprovar el correcte funcionament del web service. Per a comprovar el correcte funcionament del web service s'ha d'executar Tomcat. Seguidament obrir la línia de comandes de windows i executar el mòdul run.client amb ant. S'haurà d'especificar la ruta del fitxer que volem pujar i el nom que volem que tingui un cop es trobi en el servidor. Si la operació s'ha realitzat correctament rebrem un missatge escrit per la línia de comandes que ens confirmarà que tot ha anat correctament com es mostra a la figura 3.11.

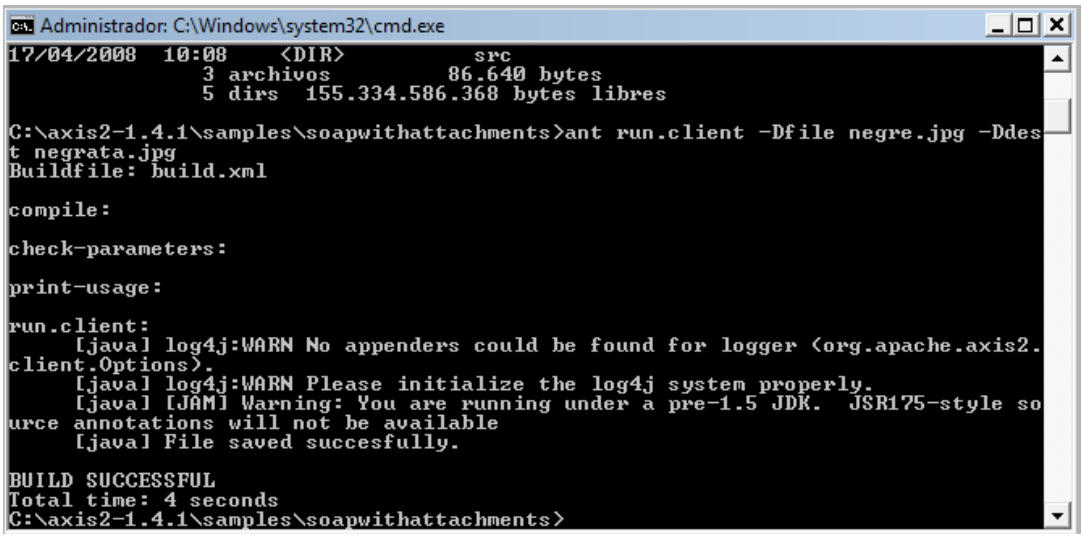

Fig. 3.11 Mostra d'execució en la línia de comandes del web service

CAPÍTOL 5. PROVES I RESULTATS

Per a comparar i veure les diferències entre les tres aplicacions realitzades s'ha utilitzat l'analitzador de protocols "Wireshark" per a fer captures de paquets.
Les proves s'han realitzat utilitzant un ordinador com a màquina servidor i un altre com a màquina client. El funcionament de les aplicacions ja s'havia comprovat utilitzant un sol ordinador de manera local, d'aquesta manera no hi ha hagut cap problema al accedir al servidor des d'una màquina client connectada a la mateixa xarxa.

La única diferència bàsica és que en accedir des de la màquina client tenim que obrir al navegador i accedir a la IP de la màquina servidor i al port 8080 d'aquesta mateixa màquina. D'aquesta manera en comptes de http://localhost:8080/... Haurem de canviar "localhost" per el número de la IP de la màquina servidor. Això ens funcionarà per a les aplicacions PHP i Java.

Per al web service s'haurà d'instal·lar l'aplicació client a la màquina client. Perquè l'aplicació es pugui executar també necessitarem instal·lar tant "Apache Ant" com "Apache Axis" en la màquina client. Ant es necessita per a construir i executar el client, mentre que necessitem llibreries que ens venen amb Axis per a què l'aplicació funcioni de manera correcte. D'aquesta manera també s'haurà de modificar les variables d'entorn Classpath i Path per a poder executar l'aplicació correctament. A part d'això també s'ha hagut de modificar una petita part del codi que corresponia a la URL on accedia el client. Bàsicament el que s'ha fet és que el client no accedeixi a la màquina de forma local, sinó que accedeixi a una altre màquina servidor amb la IP corresponent i la ruta i nom del web service que es correspon.

Evidentment abans per poder accedir al servidor tenim que executar el Tomcat, en cas contrari no ens podríem connectar a la pàgina desitjada ja que no la trobaria.

5.1. Comparació de la capa d'aplicació (Wireshark)

A l'estar treballant amb protocols basats en TCP l'opció "Follow TCP" que ens proporciona el Wireshark ens permet veure l' intercanvi de dades que es produeix vista des de la capa d'aplicació.

Observant les captures no s'ha observat cap diferència significativa entre Java i PHP. Les dos tecnologies bàsicament utilitzen un mètode POST i això és el que es veu en la capa d'aplicació. No s'aprecien diferències en el sistema de capçaleres i l'estructura de paquets com s'observa en la figura 5.1 i 5.2 de manera que en aquest aspecte les dos tecnologies es poden considerar pràcticament iguals.

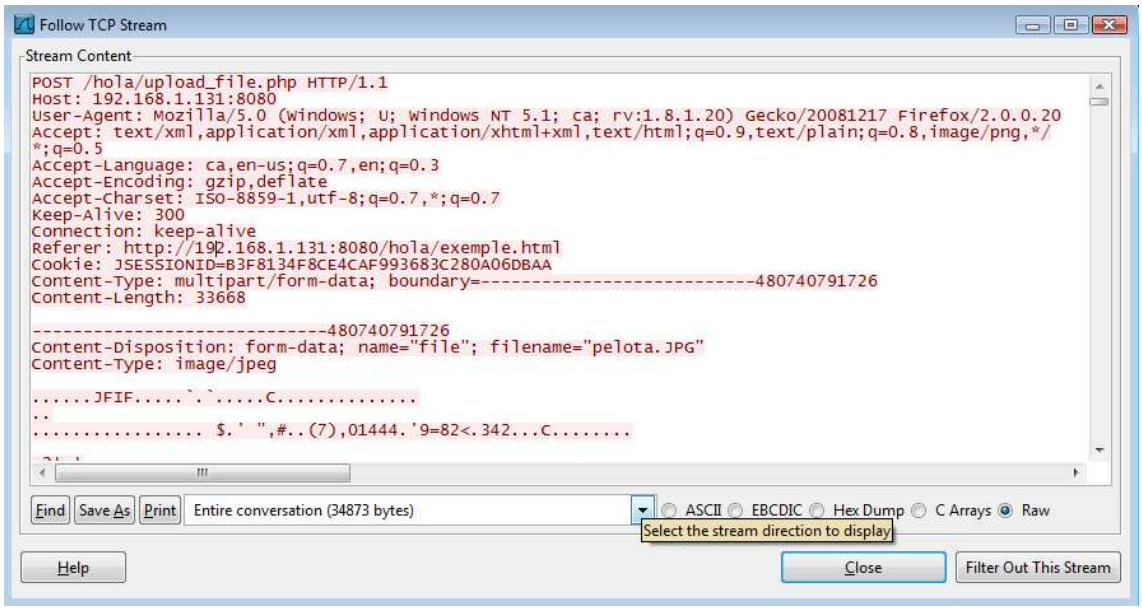

Fig. 5.1 Capçalera PHP

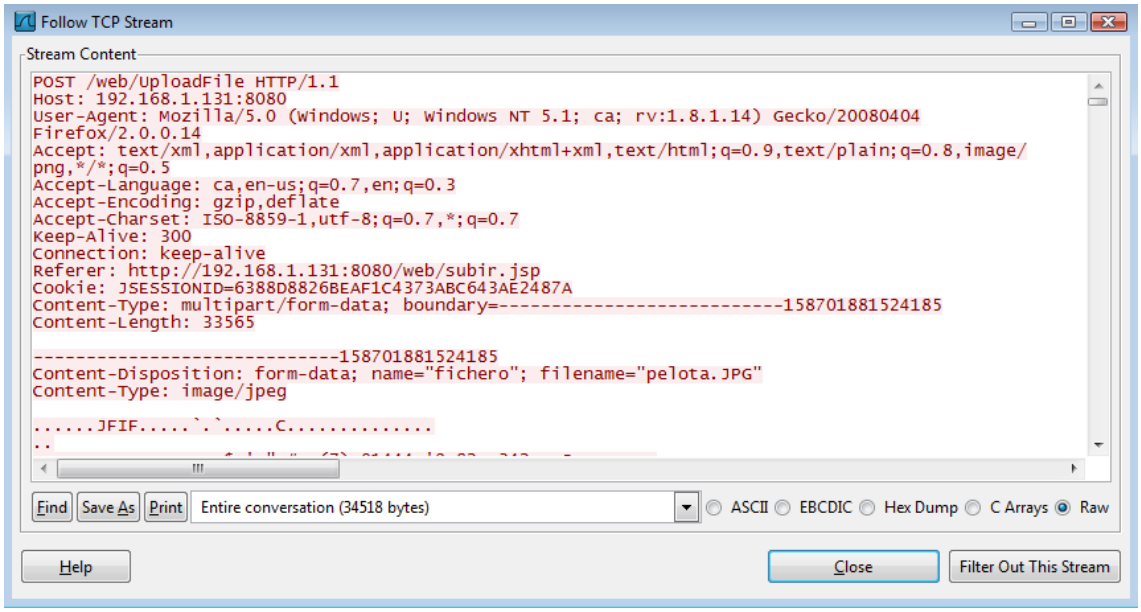

Fig. 5.2 Capçalera Java

En el web service es pot apreciar en la capçalera que s'utilitzen missatges SOAP i els elements característics d'aquest tipus de missatge que defineixen el fitxer XML com a SOAP com podem veure en la figura 5.3.

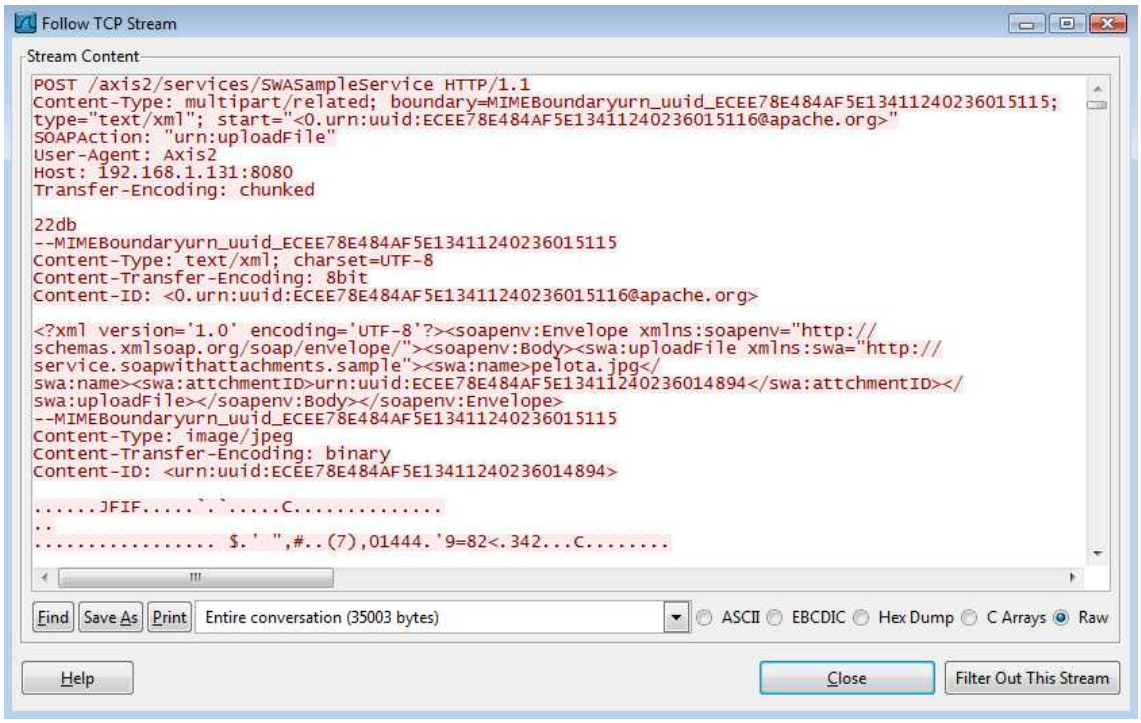

Fig. 5.3 Capçalera web service

5.2. Comparació del rendiment en el servidor

Per a realitzar aquesta comparació bàsicament s'ha fet servir el mateix mètode que l'apartat anterior, utilitzant dos màquines una com a client i l'altre com a servidor. Aquest cop s'ha buscat analitzar si en el costat del servidor s'aprecien diferències quan es realitzen enviaments de fitxers utilitzant les diferents tecnologies.

Per a fer-ho s'ha utilitzat el "Monitor de confiabilitat i rendiment " de Windows. En aquest programa podem veure els canvis que es produeixen en la CPU, en el disc dur, en la xarxa i en memòria durant la transferència dels fitxers, en temps real. Els resultats es poden visualitzar en gràfics separats a la vegada que es poden veure les aplicacions que s'estan executant en el mateix moment.

Bàsicament s'han realitzat dos proves. Una enviant un fitxer i una altre enviant tres fitxers seguits utilitzant la mateixa aplicació però canviant el nom d'aquesta.

5.2.1. Resultats obtinguts en Java

A l'activar el servidor Tomcat observem que apareix un procés java.exe en memòria degut a que la instància del servlet és carregada en temps

d'arrencada del contenidor. Aquest procés es queda en memòria mentre el contenidor de servlets segueixi actiu.

Quan accedim al nostre servlet i realitzem l'enviament de dades del client al servidor s'inicialitza el servlet per a servir la invocació pel seu propi fil d'execució (Thread). D'aquesta manera apareix un procés java.exe en CPU mentre s'executa el servlet i no veiem el nom del servlet en execució com cabria esperar.

En les figures 5.4 i 5.5 veiem las gràfiques obtingudes en l'enviament d'un fitxer i diferents fitxers seguits respectivament.

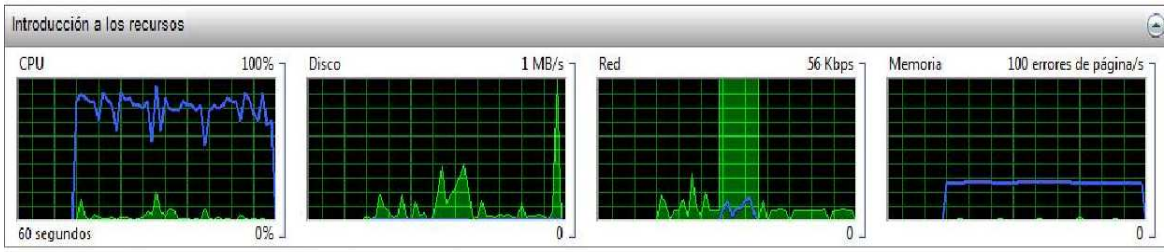

Fig. 5.4 Enviament d'un fitxer

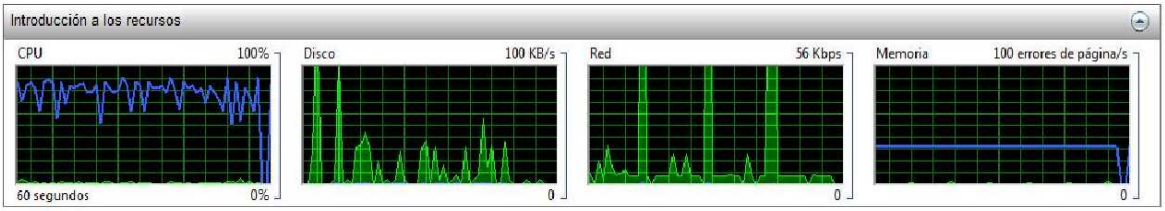

Fig. 5.5 Enviament de tres fitxers consecutius

En la gràfica CPU les línies verdes representen el percentatge total de la capacitat de la CPU que està en ús, mentre que les blaves mostren la freqüència màxima de CPU en percentatge.

La gràfica "Disco" mostra en verd l'entrada/sortida de dades total actual i en blau el màxim percentatge de temps actiu.

La gràfica "Red" mostra en verd el tràfic total de la xarxa actual (Kbps) i en blau el percentatge de la xarxa utilitzada.

Per últim la gràfica memòria mostra en verd els errors severs per segon actuals i en blau el percentatge de memòria física en ús.

5.2.2. Resultats obtinguts en PHP

Al activar el servidor Apache apareix un procés httpd.exe (Hypertext transfer protocol daemon) que es queda en memòria. Un cop fetes algunes activitats preliminars per activar el servidor, apareix un procés fill amb el mateix nom, que té la funció d'escoltar i respondre les peticions dels clients. El primer procés httpd.exe es segueix executant com a usuari principal i el procés fill com a un usuari menys privilegiat.

Quan accedim al nostre servlet i realitzem l'enviament de dades del client al servidor s'inicialitza el servlet per a servir la invocació pel seu propi fil d'execució (Threat). D'aquesta manera apareix un procés httpd.exe en CPU mentre s'executa el servlet i no veiem el nom del servlet en execució com cabria esperar.

En les figures 5.6 i 5.7 veiem las gràfiques obtingudes en l'enviament d'un fitxer i diferents fitxers seguits respectivament.

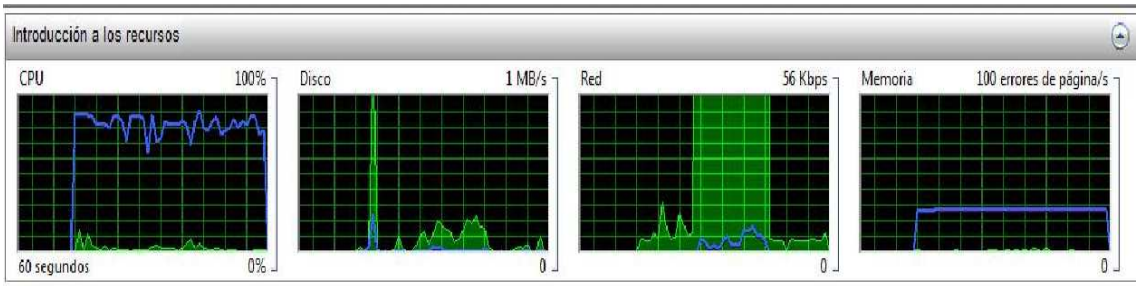

Fig. 5.6 Enviament d'un fitxer

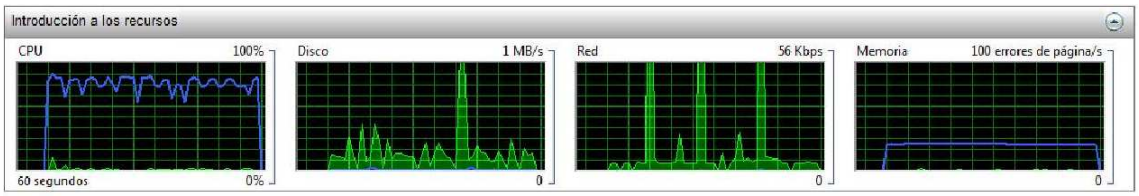

Fig. 5.7 Enviament de tres fitxers consecutius

Podem observar clarament quan arriba cada un dels fitxers en el servidor a la gràfica de xarxa ("Red" en el gràfic) . Una vegada més els resultats son molt semblants amb alguna lleugera diferència. S'ha comprovat fent la mateixa prova repetides vegades, que les diferències que es poden apreciar entre les gràfiques "Disco " i "Red" no es deuen al funcionament de l'aplicació sinó al comportament de la xarxa en el moment de l'enviament. D'aquesta manera hi pot haver diferències en els pics d'utilització del disc i en el temps més prolongat de l' utilització de la xarxa. En les gràfiques CPU i Memoria no s'aprecia cap diferència significativa.

On si que s'ha apreciat diferència és en l'ús de la CPU un cop es produeix l'enviament, quan mirem la llista de processos actius. Aquesta diferència s'ha observat en els dos casos però més clarament en la segona prova realitzada. Quan executem la prova en java veiem en CPU el procés java.exe que s'executa quan es produeix l' intercanvi de dades. Quan executem la prova en PHP veiem el procés httpd.exe que tot i ser un procés associat al servidor Apache, ens indica quan la nostra aplicació ocupa espai en CPU. D'aquesta manera, un cop realitzat l' intercanvi de fitxers, java.exe segueix utilitzant recursos durant un temps lleugerament superior al que utilitza httpd.exe però no d'una manera realment significativa.

CAPÍTOL 6. CONCLUSIONS

L'experiència de realitzar una aplicació web per a un alumne sense experiència en aquests camps, ha esta enriquidora i penso que els coneixements obtinguts em poden ser útils en el futur. Al principi s'ha hagut de dedicar les primeres setmanes a aprendre el nivell necessari per a realitzar les aplicacions corresponents amb Java i PHP. S'ha comprovat directament que l'aprenentatge de PHP és més ràpid per a realitzar una aplicació web simple que no pas amb Java, tot i que hi ha àmplia documentació sobre els dos llenguatges en la xarxa.

Respecte a la instal·lació dels servidors Tomcat i Apache, ha estat força intuïtiva. M'ha semblat molt positiu que aquest software sigui open source i disponible de forma gratuïta per a realitzar proves amb aplicacions web.

Els objectius que es van marcar al principi d'aquest treball de fi de carrera de realitzar una comparativa s'ha dut a terme. Cal matisar que en la part pràctica no s'ha trobat diferències rellevants en l'ús de les tecnologies Java i PHP degut a que les màquines d'avui en dia tenen una potència que fa que aquestes siguin insignificants per a una aplicació simple. Una possible línia futura podria ser realitzar proves amb una aplicació web molt més complexa, amb volums de dades molt grans i amb concurrències molt elevades per veure si a aquest nivell trobem diferències en el rendiment.

Per altre banda s'ha deixat una mica desplaçat l'objectiu inicial de profunditzar més en l' utilització de web service ja que no era una tecnologia directament comparable amb l'ús de PHP i Java en aplicacions web simples. En tot cas s'ha comprovat que per a realitzar una aplicació senzilla no caldria realitzar un web service que ens suposaria una inversió de temps i treball més importants. Com a línia futura es podria realitzar un web service amb PHP i amb Java i realitzar una comparativa.

D'es del punt de vista d'ambientalització es pot afirmar que el treball realitzat no ha tingut un impacte rellevant. Això es degut a que tot el que s'ha fet ha estat utilitzant l'ordinador i cap altre material o escenari addicional.

CAPÍTOL 7. BIBLIOGRAFIA

[1] http://en.wikipedia.org/wiki/Web_application

[2] http://en.wikipedia.org/wiki/Web_service

[3] http://www.woodger.ca/archweb.htm

[4] http://www.embedded.com/story/OEG20020125S0103

[5] http://www.esi.uem.es/jccortizo/temasConcu/soa.pdf

[6] http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

[7] http://jeez.eu/2009/10/15/programming-languages-rank-with-a-rather-strange-way/

[8]http://wiki.answers.com/Q/Advantages_and_disadvantages_of_php_hypertext_preprocessor

[9] http://es.wikipedia.org/wiki/.php

[10] http://www.apl.jhu.edu/~hall/java/Servlet-Tutorial/Servlet-Tutorial-Overview.html

[11] http://es.wikipedia.org/wiki/JavaServer_Pages

[12]http://java.sun.com/developer/technicalArticles/javaserverpages/servlets_jsp/

[13] http://www.dcc.uchile.cl/~jbarrios/servlets/index.html

[14]http://articulosinformativos.com/Introducion_a_los_Servicios_Web_Fort_Smith_AR-r1106612-Fort_Smith_AR.html

[15] http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb

[16] http://es.wikipedia.org/wiki/XML

[17] http://es.wikipedia.org/wiki/SOAP

[18] http://es.wikipedia.org/wiki/UDDI

[19] http://tomcat.apache.org/

[20] http://blogs.sun.com/robogeek/entry/php_versus_java_jsp_j2ee

[21] http://rajcheram.wordpress.com/2009/04/22/open-source-web-applications-php-vs-java/

[22] http://blog.chuidiang.com/2009/04/23/%C2%BFjsp-o-php/

[23]http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=PHPVsJava

[24] http://commons.apache.org/fileupload/

[25] http://php.net/index.php

[26] http://ws.apache.org/axis/

[27] http://codefeed.com/tutorial/ant_intro.html

[28] http://ant.apache.org/

[29] http://httpd.apache.org/docs/2.2/invoking.html

[30] http://ws.apache.org/axis2/1_2/mtom-guide.html

[31]http://trevinca.ei.uvigo.es/~txapi/espanol/proyecto/superior/memoria/node76.html

[32] http://httpd.apache.org/docs/2.2/invoking.html

[33]http://trevinca.ei.uvigo.es/~txapi/espanol/proyecto/superior/memoria/node76.htm