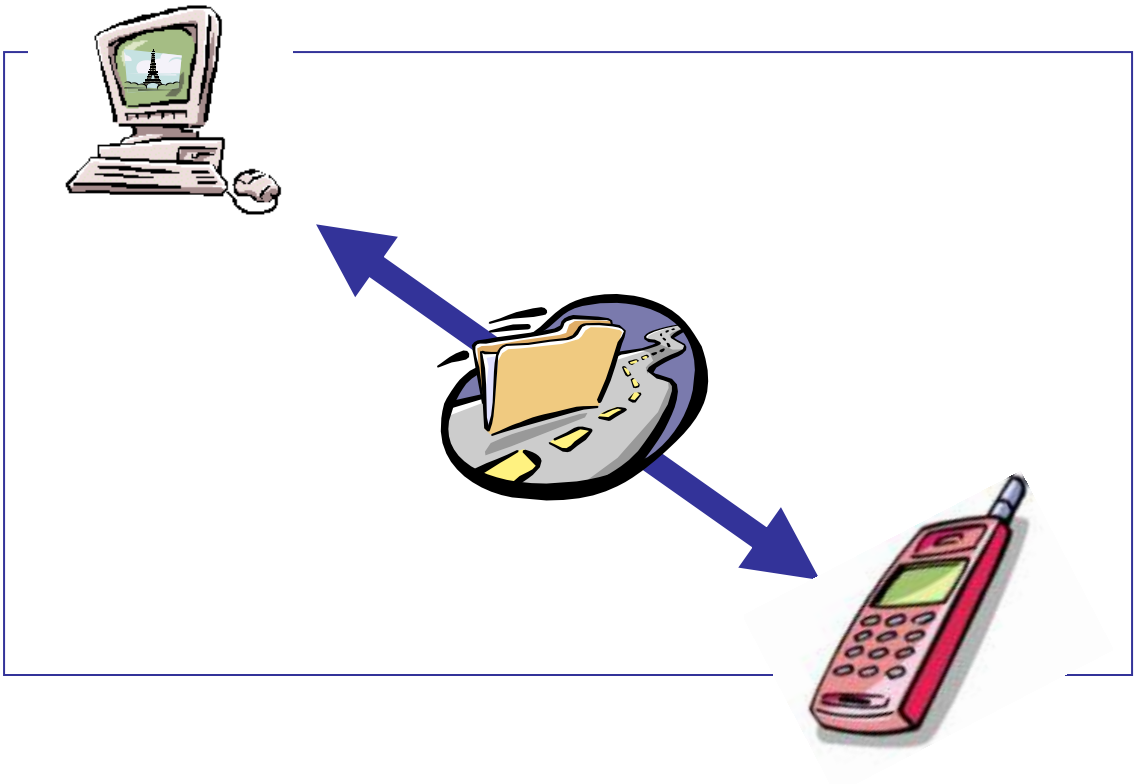
Títol: Disseny i implementació de Web Services mitjançant Java i .NET entre Pocket PC, PC i clients J2ME.

Volum: 1 de 1
Alumne: Ramon Nou Castell

Director: Jordi Torres i Viñals
Departament: Arquitectura de Computadors
Data: 2/Febrer/2004

Disseny i implementació de Web Services mitjançant Java i .NET entre Pocket PC, PC i clients J2ME.



Ramon Nou Castell
Director: Jordi Torres i Viñals

Projecte d'Enginyeria Informàtica (20 crèdits)
Febrer 2004
Departament d'Arquitectura de Computadors
Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

Dedicat a les dues coses que més estimo, la Natàlia i la programació.

Vull agrair al meu director, Jordi Torres, el seu suport per a realitzar el PFC i per deixar-me el mòbil per a realitzar les proves ;). També vull agrair l'ajut de Natàlia Fos en la correcció de la memòria i d'en Salvador Roura, Jordi Petit i la resta dels meus companys del primer Concurs de Programació de l'UPC per fer que el quadrimestre fos 'diferent'.

Resum

Aquest projecte té per objectiu fer una recerca exploratoria per tal de conèixer i avaluar les oportunitats i l'impacte que té la tecnologia emergent dels Web Service i totes les tecnologies associades que estan apareixent. S'estudiarà la viabilitat d'utilitzar els Web Services en dispositius mòbils limitats en molts aspectes com són: capacitat de procés, memòria disponible i la més important, el programari que tenim a l'abast.

Per a realitzar aquesta tasca d'anàlisi de tots els problemes que podem trobar i les seves solucions crearem un sistema d'informació senzill (quant a complexitat i implementació) per tal de poder provar i crear clients amb aplicacions reals.

El sistema d'informació consistirà en un butlletí de notícies en línia, que mitjançant la utilització de clients en els dispositius mòbils basats en Web Services, es podrà actualitzar mitjançant imatges i text. Per a la realització del sistema d'informació s'ha utilitzat el llenguatge PHP per a la façana web i la base de dades mySQL per a emmagatzemar les dades. En la part relativa als Web Services s'ha utilitzat com a servidor la plataforma Apache Axis per la seva facilitat i potència d'ús. Aquest projecte pretén servir de guía per a que després de llegir-lo se sigui capaç d'instal·lar el programari necessari i crear un Web Service i el seu corresponent client mòbil.

En la banda remota, hem utilitzat la plataforma comercial .NET i l'embedded Visual tools amb una llibreria gratuïta com és PocketSOAP. Donada la poca dificultat de la implementació en Pocket PC, es va decidir donar més pes a la implementació sobre J2ME, que ofereix més reptes i dificultats. Sobre J2ME s'ha utilitzat l'única llibreria disponible per a crear Web Services: kSOAP. En el moment de finalització d'aquest projecte Sun encara no havia tret de *public draft* la seva implementació.

Finalment s'analitzen el cost d'implementació i de recursos de totes les alternatives i s'analitza l'interoperativitat del sistema amb .NET.

Paraules Clau: Web Services, J2ME, .NET, Apache Axis, Pocket PC, Windows CE, pocketSOAP, kSOAP, kVM.

Índex

Índex de taules

Índex d'il·lustracions

Índex de continguts

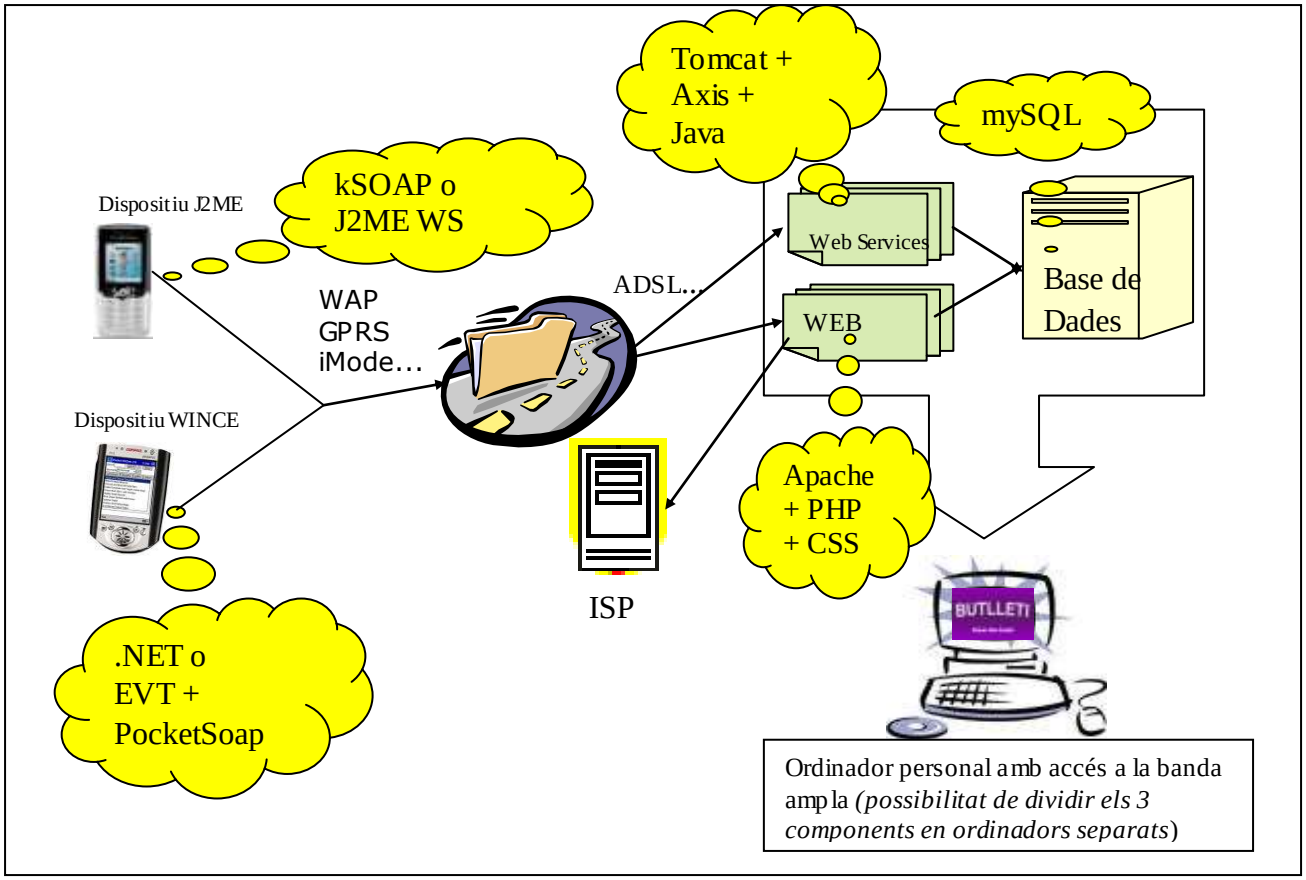
1. Introducció

1.1 Descripció dels objectius del projecte

1.1.1 Introducció

Aquest P.F.C. té com a objectiu estudiar la viabilitat de l'ús de Web Services (WS, a partir d'ara) sobre plataformes limitades, com són telèfons mòbils (**J2ME**) i sistemes basats en Windows CE (**Pocket PC**), així com els problemes que poden aparèixer (com per exemple d'interoperativitat).

La motivació del projecte va sorgir donat el "boom" de les noves tecnologies provocat pels anomenats dispositius mòbils (PDA's, Pocket PC's i telèfons) que cada cop s'assemblen més a ordinadors personals, però tenen una cosa diferenciadora: són portables i mòbils. Per tal de realitzar l'estudi dels problemes que podem trobar a adaptar una tecnologia nova com són els Web Services a un entorn limitat en quant a memòria i capacitat de procés, va sorgir la idea de crear un butlletí de notícies on-line per a un escenari basat en un ordinador personal instal·lat a casa i amb una connexió continua a Internet (ADSL, cable).



Il·lustració 1.1 - Diagrama del sistema

1.1.2 Objectius

Els objectius que volem obtenir amb aquest projecte són els següents:

- Entendre els Web Services, partint d'un coneixement nul.
- Crear alguns Web Services a l'ordinador personal. (Apache AXIS)
- Preparar el sistema d'informació, per tal de poder fer les proves en un entorn més o menys proper a la realitat. En aquest punt no es pretén dissenyar un S.I. en detall, sinó crear un esquelet per a la seva ampliació posterior. (mySQL, Java, PHP...)
- Implementació de Web Services a un Pocket PC, problemes que hi podem trobar, dificultats, programari, rendiment. (PocketSOAP)
- Implementació d'aplicacions per a mòbils J2ME. (SDK's, MMAPI)
- Implementació de Web Services a un mòbil J2ME, dificultats, estàndards, programari, rendiment. (kSOAP)
- Implementació de la façana WEB (PHP) amb la funcionalitat d'actualització remota i la sincronització amb un ISP estàtic.
- Creació d'un paquet instal·lable que ho inclogui tot.
- Funcionament a qualsevol SO (Principalment Windows i Linux)

1.1.3 Camins per assolir els objectius

El butlletí de notícies podria pertànyer a qualsevol organització, però el seu objectiu és aparèixer en un centre excursionista per a poder, mitjançant un dispositiu mòbil únic (com pot ser un telèfon mòbil amb càmera incorporada) i utilitzant Web Services actualitzar el butlletí en temps real des d'un campament per exemple (si tenim cobertura, és clar). Aquest butlletí de notícies, a part d'actualitzar-se via mòbil també inclouria una façana web dinàmica (amb la funcionalitat que disposem al terminal), amb la possibilitat d' actualitzar una pàgina estàtica ofertada per qualsevol ISP de forma gratuïta. Durant la seva creació pretenem assolir els diferents objectius que hem remarcat.

Per tal de provar el programari que creem, tot i que tenim el maquinari necessari, cal destacar que no caldria utilitzar-lo gràcies al grau d'exactitud dels emuladors dels diferents terminals. Per a finalitzar, per poder testejar l'entorn en un entorn real es va realitzar una prova en un viatge a l'estranger, on és van introduir unes quantes notícies i imatges al sistema instal·lat a casa.

1.2 Organització de la memòria

Aquesta memòria pretén explicar les diferents dificultats que hem anat trobant durant la realització del projecte; també pretén donar les pautes que hem seguit per a crear el codi funcional als diferents dispositius. Per això, alguns dels temes que hi podem trobar tenen un elevat contingut amb codi (capítols dedicats a la implementació dels Web Services i del client mòbil sobretot).

Tot i això la majoria del codi es pot trobar al **CD-ROM** que acompanya la memòria per tal d'evitar que la lectura d'aquesta sigui pesada.

Anem a comentar els diferents capítols de la memòria i què podem trobar en ella.

Capítol 2 –

Hi introduirem totes les tecnologies que utilitzarem en aquest projecte. De manera simple explicarem els conceptes de Web Services d'Axis, així com el seu funcionament, donant algunes pautes per a la seva utilització (podem trobar una versió més extensa en l'annex B); mySQL i el programari/maquinari de la part remota tant per Pocket PC com per als dispositius basats en J2ME.

En aquest capítol comentarem també les primeres pautes que hem de seguir per tal de programar aplicacions per a aquests dispositius.

Capítol 3 –

A l'inici del projecte es va realitzar una prova pilot amb un dispositiu Pocket PC, per tal de poder avaluar de forma més acurada el temps que passaríem en les diferents fases del projecte. Aquesta part va servir també per a escollir el sistema d'informació que s'implementaria en la segona part.

En aquest punt, es va decidir de descartar una anàlisi més exhaustiva de les tecnologies basades en Pocket PC, ja que aquestes no aportaven cap dificultat, i ens vam decidir a encaminar el projecte cap als telèfons mòbils ja que donàvem més joc. En aquest capítol trobem també els diferents diagrames (especificació, seqüència) i el disseny de la petita base de dades que necessitàvem per a aquesta prova i les diferents conclusions.

Capítol 4 –

En aquest capítol comença el gruix de la memòria, la construcció del sistema d'informació. En primer lloc realitzem l'especificació global del sistema i n'escollim un subconjunt per a realitzar un prototipus. Vàrem decidir realitzar prototipus per a tenir material tangible ràpidament. Això va provocar també que detectéssim totes les dificultats i problemes abans i així poder-hi trobar una solució.

Aquest capítol avarca la construcció dels WS i la base de dades al PC.

Capítol 5 –

La primera presa de contacte amb la tecnologia J2ME es va realitzar en aquest punt, i vam començar amb una aplicació que enviava les imatges capturades a través del mòbil al sistema d'informació mitjançant WS. L'estructura d'aquest capítol és pràctica i descendent: comencem creant un esquelet bàsic que permet capturar una imatge i anem afegint els elements de kSOAP per la funcionalitat dels WS. En aquest capítol es mostra més codi que altra cosa, però considerem que és una part important per tal d'entendre com es poden realitzar els clients basats en WS (i les aplicacions basades en J2ME en general). En aquest capítol els coneixements adquirits van ser molt elevats.

També vàrem realitzar la primera prova en un terminal real, i la primera decepció al no ser compatible l'aplicació amb el terminal. Finalment aquest problema es va resoldre satisfactòriament.

Capítol 6 –

Per tal de resoldre el problema trobat al Capítol 5, vam dissenyar un procés per a executar en el servidor que processava el correu electrònic cercant les imatges enviades pel mòbil i incorporant-les a la base de dades. Per tant en aquest capítol es comenta la creació del procés i l'elecció del servidor SMTP (Simple Mail Transfer Protocol) escollit.

Capítol 7 –

Una vegada superada la dificultat de la primera aplicació en J2ME, ens disposem a crear la segona, l'aplicació encarregada de crear les notícies i enviar-les al sistema d'informació. En aquest punt es comenta (amb alguns fragments de codi) la creació d'aquesta aplicació, així com els diferents elements que poden fer de la nostra aplicació usable. En aquest capítol trobem també un problema trobat i la seva solució.

Capítol 8 –
Posteriorment al prototipus vam finalitzar les funcionalitats especificades i vam crear el client encarregat de la missatgeria.

Capítol 9 –
En aquest capítol intentem mesurar el rendiment de l'aplicació mitjançant les eines disponibles als SDK dels fabricants. També introduïm el concepte d'ofuscació de codi per a reduir la mida de l'aplicació mitjançant una eina gratuïta.

Capítol 10 –
El client WEB és el protagonista en aquest capítol; comentem els components, creem els diagrames i comentem les funcions utilitzades. Finalment es mostra la creació de CSS per a la WEB i com es pot modificar la web en poc temps.

Capítol 11 –
En aquest capítol es mostra la planificació final del projecte i s'explica la raó de les desviacions que s'hi troben.

Capítol 12 –
Conclusions del projecte i anàlisi de costos, tant monetaris com d'implementació, apareixen en aquest capítol; separem les tres plataformes (Pocket PC, mòbil i PC) i donem les conclusions que en aquest aspecte hem trobat.
Dintre d'aquest capítol fem una menció al rendiment (monetari) que podem treure d'aquest sistema, i es donen algunes idees de possibles continuacions i usos d'aquest PFC.

Capítol 13 –
Finalment donem la llista de referències bibliogràfiques del projecte. Com que és un projecte que utilitza una tecnologia molt nova, la majoria de les referències són documents electrònics. La majoria també són trobats en articles que es poden trobar a les pàgines web dels fabricants dedicades als desenvolupadors. Si més no, les referències més representatives s'inclouen al CD-ROM que acompanya la memòria.

Al final del document podem trobar diferents annexos amb informació específica:

Annex A – Comentaris sobre dispositius J2ME.
Annex B – Manual per a desenvolupar en Axis.
Annex C – Creació del sistema d'instal·lació.
Annex D – Manual d'usuari de l'aplicació.
Annex E – Interoperativitat amb .NET
Annex F – Contingut del CDROM

1.3 Planificació del Projecte

La planificació del projecte inicial va ser de 554 hores. A continuació mostrem els diferents diagrames de planificació que es van crear (el desglossament en detall de cada fase es feia, normalment, en l'iniciar-la).

En un primer moment es va realitzar la prova pilot per tal de trobar la direcció en que havia d'anar el projecte, i després de realitzar-la es va decidir la resta del projecte.

1.3.1 Tasques a realitzar

Nom	Duració prevista
Cerca d'informació sobre WS (PC)	16 hores
Cerca d'informació sobre WS (Pocket PC)	8 hores
Instal·lació Tomcat/Apache/mySQL/PHP	3 hores
Instal·lació Apache Axis	1 hora
Primer contacte amb Axis (proves, documentació)	8 hores
Prova Pilot	*170 hores*
Especificació de la prova pilot	4 hores
Pilot Part PC	*26 hores*
Disseny de la prova pilot	8 hores
Disseny (BD) prova pilot	1 hora
Implementació BD	1 hora
Implementació WS a AXIS	16 hores
Pilot Façana WEB	*24 hores*
Cerca d'informació sobre PHP	8 hores
Implementació façana WEB	16 hores
Pilot Part Pocket PC	*92 hores*
Instal·lació entorn programació Pocket PC	8 hores
Estudi de la implementació mitjançant .NET	24 hores
Estudi de la implementació mitjançant EVT	24 hores
Especificació/Disseny part Pocket PC	4 hores
Implementació amb .NET	16 hores
Implementació amb EVT	16 hores
Proves i conclusions	8 hores
Documentació.	16 hores
Cerca d'informació sobre mòbils	16 hores

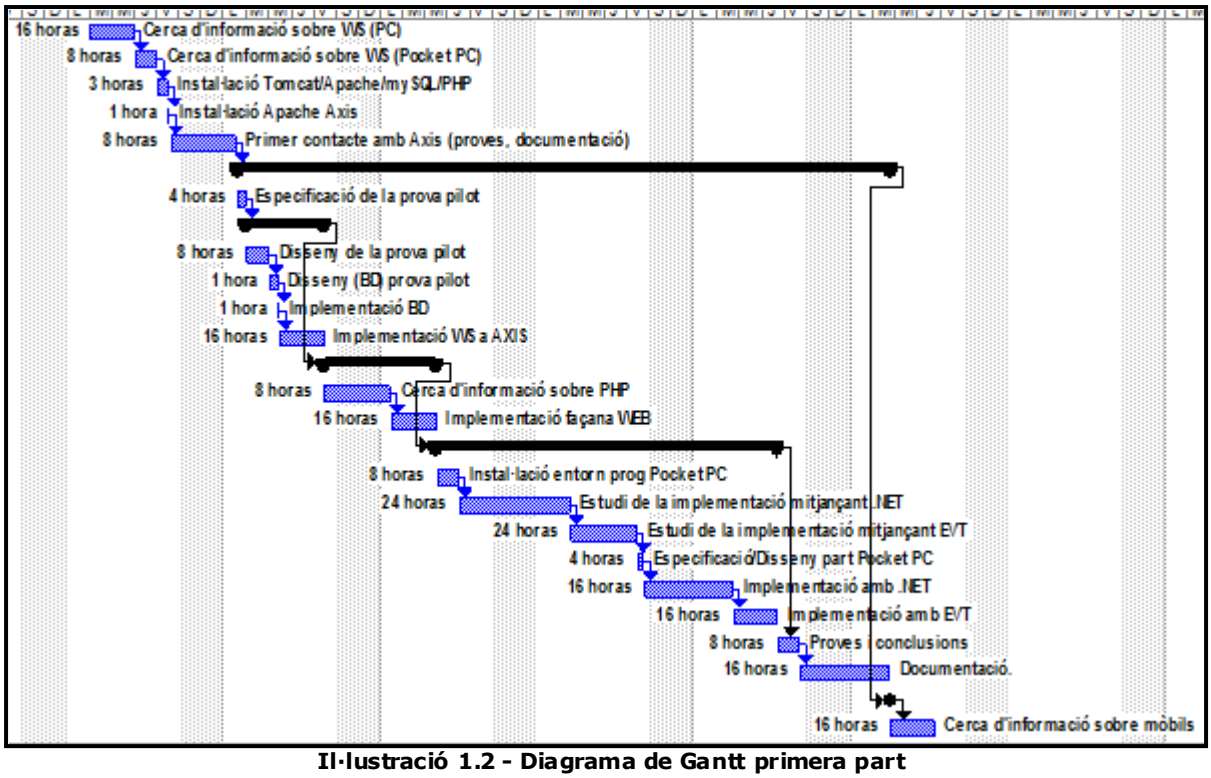
Taula 1.1 -Tasques a realitzar a la primera part

Nom	Duració prevista
Segona Part - Butlletí	*220 hores*
Especificació del sistema d'informació	40 hores
Butlletí part PC	*52 hores*
Disseny del sistema d'informació	30 hores
Disseny de la Base de Dades	10 hores
Implementació BD	4 hores
Implementació dels WS banda PC	8 hores
Butlletí part Mòbil	*96 hores*
Cerca d'informació sobre mòbils (programació)	4 hores
Instal·lació SDK i selecció	8 hores
Especificació/Disseny de la part remota (ABM Imatges)	4 hores
Programació ABM Imatges amb J2ME	24 hores
Proves/Solució problemes	8 hores
Especificació/Disseny de la part remota (ABM Notícies)	8 hores
Programació ABM Notícies amb J2ME	24 hores
Proves/Solució problemes	8 hores
Especificació/Disseny part missatges	2 hores
Programació ABM Missatges amb J2ME	4 hores
Proves/Solució problemes	2 hores
Butlletí part WEB	*32 hores*
Creació de la façana WEB	24 hores
Proves/Solució problemes	8 hores
Preparació de la part d'instal·lació	8 hores
Proves instal·lació	4 hores
Creació de la memòria	100 hores
	554 hores

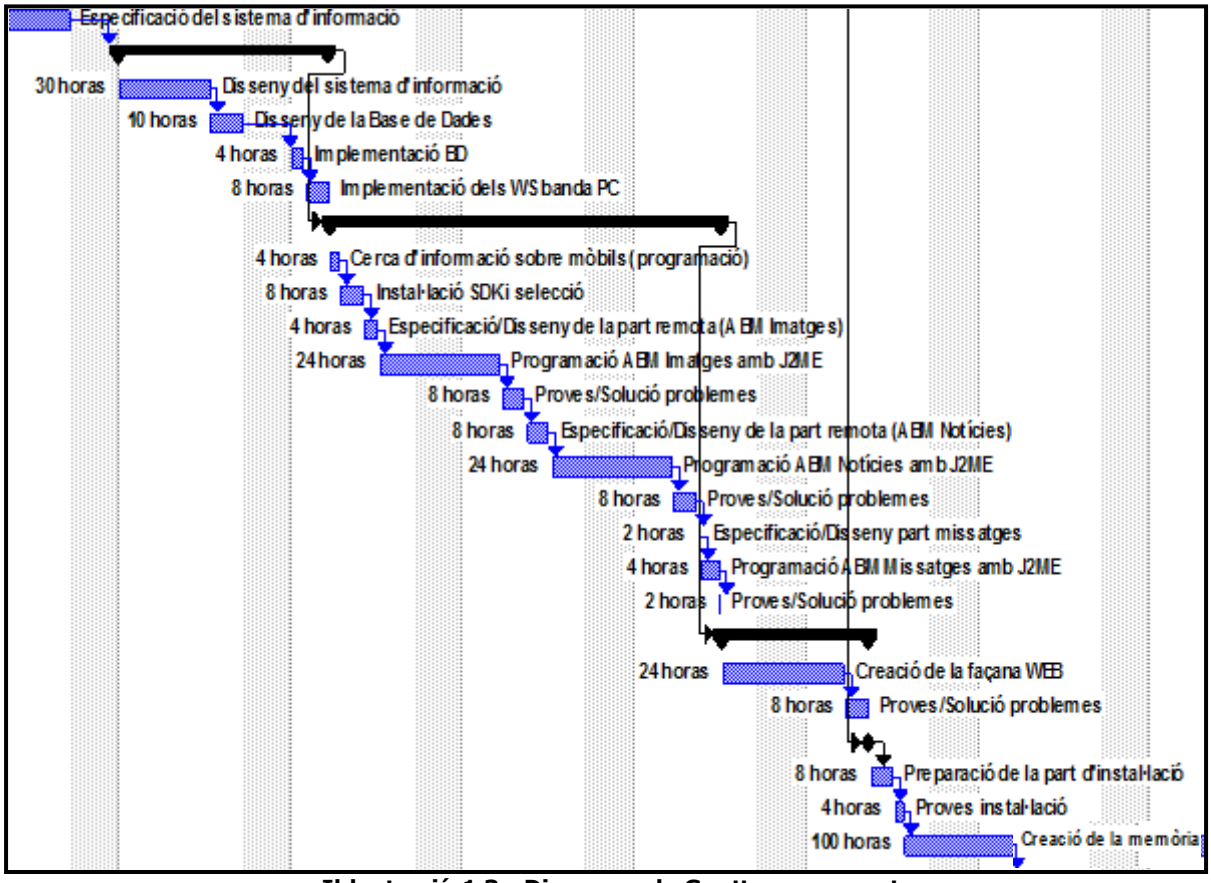
Taula 1.2 -Tasques a realitzar a la segona part

1.3.2 Diagrama de Gantt

Mostrem a continuació el diagrama de Gantt del projecte; tot i que no pretén mostrar paral·lelitzacions de tasques (ja que només hi ha una persona), si que permet veure la dimensió del projecte.



Il·lustració 1.2 - Diagrama de Gantt primera part



Il·lustració 1.3 - Diagrama de Gantt segona part

2. Tecnologia Utilitzada

2.1 Introducció

A continuació mostrarem un quadre amb totes les tecnologies utilitzades durant el desenvolupament d'aquest projecte, juntament amb el percentatge d'importància.

Tecnologia	*Importància*
J2ME + kSOAP	25 %
Web Services	18 %
PHP	15 %
Axis	15 %
J2ME multimèdia	10 %
.NET	5 %
pocketSOAP	5 %
mySQL	5 %
Tomcat	1 %
Apache	1 %

Taula 2.1 - Elements utilitzats.

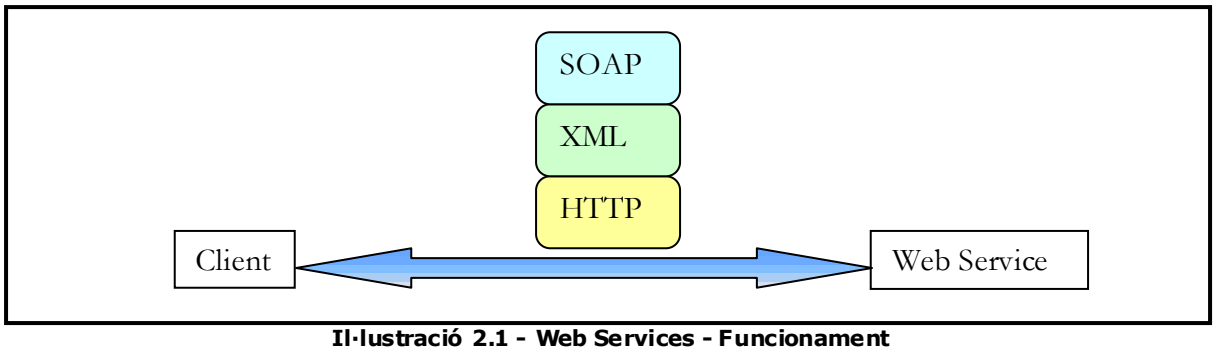
2.2 Web Services

Els Web Services s'han convertit en una eina capaç de facilitar les tasques de comunicació i intercanvi de dades entre dos entorns diferents i llunyans. Si bé és veritat que al final tot es redueix a "*Sockets*[1]", la informàtica intenta sempre fer les tasques més senzilles. El seu funcionament és semblant a altres tecnologies com pot ser CORBA[2], JAVA-RMI[3] o similars, però sense la complexitat del primer ni l'exclusivitat de llenguatge del segon.

Els WS utilitzen el protocol **SOAP**[4], basat en **XML**[5]. SOAP es pot utilitzar amb molts protocols de transport de dades, però el més comú és utilitzar-lo amb **HTTP**[6], que permet evitar tots els problemes associats amb tallafocs.

L'objectiu d'aquest projecte no és estudiar els Web Services en tota la seva amplitud, però si estudiar la viabilitat i tota la problemàtica d'utilitzar una tecnologia emergent com són els telèfons mòbils i els Pocket PC amb la tecnologia dels WS, per tal d'aconseguir accedir a serveis d'una manera senzilla i eficient, tant en implementació com en funcionament.

[1] Socket, canal que s'obre entre dos terminals a través de la xarxa per transmetre dades.
[2] CORBA is the Common Object Request Broker Architecture. This architecture is a collection of objects and libraries that allow the creation of applications containing objects that make and receive requests and responses in a distributed environment
[3] Remote Method Invocation; Java-based technology which allows Java programs to access the objects of another Java program running on a different computer.
[4] *Simple Object Access Protocol (* http://www.w3.org/TR/SOAP/)
SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol.
[5] *Extensible Markup Language, (* http://www.w3.org/XML/)
[6] *Hypertext Transfer Protocol*, (http://www.w3.org/Protocols/)



Il·lustració 2.1 - Web Services - Funcionament

2.2.1 Per què?

Aquesta és una de les preguntes que surten al principi: per quina causa volem utilitzar aquesta tecnologia (per obtenir / enviar dades) quan podem fer el mateix mitjançant un navegador web instal·lat a l'aparell (telèfon mòbil/Pocket PC).

Si realitzem una enquesta i preguntem quantes persones han utilitzat l'explorador web del seu mòbil, hi hauria prou gent que diria que l'ha utilitzat; ara bé, si preguntem quantes vegades l'han utilitzat la majoria de respostes seria una vegada.

La web no està pensada pels terminals esmentats; si bé els Pocket PC han realitzat grans avenços en adaptar la web al dispositiu mitjançant el navegador, en els telèfons mòbils trobem que ens és gairebé impossible accedir a continguts no específics (no-WML). Podem fer la prova realitzant un petit formulari i accedint-hi a través del mòbil, en la gran majoria dels casos no podrem veure res.

Si volem oferir un servei a un dispositiu portàtil tenim diverses alternatives: crear una pàgina web compatible (i intentar que es visualitzi bé en tots els terminals) o realitzar una aplicació a nivell *socket* que es comuniqui directament (servidor – client); aquesta última alternativa, en el cas dels dispositius J2ME no la podrem realitzar ja que només tenim suport pel protocol HTTP (MIDP-1.0).

La primera alternativa, i la que s'estudiarà aquí, és la utilització de WS. Amb els WS nosaltres definim la interfície de sortida/entrada al servidor d'una manera senzilla i posteriorment (i depenent en gran part del programari / maquinari utilitzat en la part del client) podem crear un client en poc temps.

Ja sigui amb .**NET** als Pocket PC o amb **J2ME**[1], els clients es visualitzaran correctament en els aparells (sempre que no utilitzem crides directes a pantalles, etc.), aconseguint una elevada velocitat d'implementació.

A priori els telèfons mòbils tenen més restriccions que els Pocket PC: els Pocket PC tenen una tendència a créixer en memòria i velocitat de CPU i per tant l'ús de XML no suposa gaires problemes; en canvi en els dispositius com els telèfons mòbils es té que jugar amb més paràmetres, com el pes i el consum, i per tant la memòria i la capacitat de procés són més limitades.

El maquinari amb el que ens barallem pot constar de 32 Kb[2] de memòria per la pila del procés (deixant poc espai pel document XML que rebem o enviem) i de no molta més memòria per a l'aplicació (les llibreries que fan de "parser" poden arribar a ocupar tota la memòria).

[1] *Java 2 Micro Edition*
[2] Veure Annex – A per a veure les característiques dels dispositius.

2.2.2 Web Services com funcionen?

L'essència d'un WS és un servidor que ofereix a través del protocol HTTP un servei; al ser HTTP aquesta petició pot travessar proxies i tallafocs fàcilment (i ser compatible amb J2ME), ja que el contingut es HTML (XML). Aquest és un dels principals avantatges sobre una programació client/servidor normal amb sockets, ja que allí on puguem accedir a un protocol HTTP podrem fer servir WS.

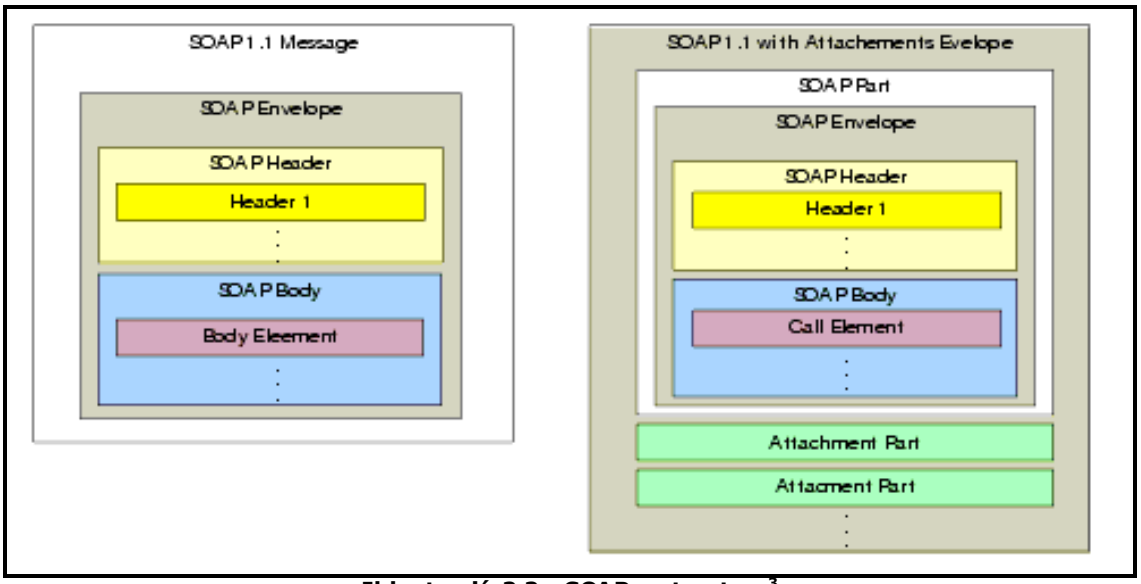
La seva principal virtut és el seu principal *handicap*; les dades s'han de codificar per tal que es puguin transportar correctament a través de la petició i això implica en la majoria dels casos codificar les dades binaries en Base64 (per exemple) amb un augment considerable de l'espai. Aquest *handicap* s'ha resolt en les posteriors revisions amb dos estàndards **DiME(Microsoft)**[1] i **SWA(w3)**[2] (SOAP With Attachments) que donen suport a dades binàries situades després del missatge XML. Aquesta extensió no està suportada en la llibreria utilitzada en el dispositiu J2ME.

Els documents XML que s'utilitzen en la transmissió tenen una estructura d'arbre, per la qual cosa la seva descodificació necessita espai en la pila, ja que s'han de guardar els nivells anteriors (recorreguts en preordre, inordre, postordre). Per això hem de tenir en compte quin tipus de document rebem, ja que si té molts nivells o és molt gran podem exhaurir la memòria del dispositiu.

Aquest *handicap* podem solucionar-lo dividint les dades i fent els documents XML més plans. En compte d'enviar un vector de vectors, enviar els índexs per separat.

2.3 SOAP

El protocol SOAP utilitza **XML** per a demanar i rebre la informació. Tot i que donat Axis, un missatge SOAP té les següents característiques:



Il·lustració 2.2 - SOAP, estructura[3]

[1] Direct Internet Message Encapsulation, (http://msdn.microsoft.com/msdnmag/issues/02/12/DIME/), public draft a http://msdn.microsoft.com/library/en-us/dnglobspec/html/draft-nielsen-dime-02.txt. DiME permet transportar fins i tot missatges SOAP
[2] http://www.w3.org/TR/SOAP-attachments
[3] www.xmlbus.com/docs/5.3/ users_guide/API12.html

SOAP Envelope:

- És el sobre del missatge, expressa el que hi ha dins del missatge, qui té que processar-lo i si el seu procés és opcional o obligatori.

SOAP encoding rules:

- Defineix els mecanismes de serialització que es poden usar per intercanviar dades.
- La representació XML de l'espai de noms per defecte (xsi, http://www.w3.org/1999/XMLSchema-instance) la podem trobar a la següent URL http://schemas.xmlsoap.org/soap/encoding/

SOAP RPC representation:

- És la representació dels RPC i les seves respostes.

Exemple de missatge SOAP:

REQUESTA

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
       <m:GetLastTradePrice xmlns:m="Some-URI">
           <symbol>DIS</symbol>
       </m:GetLastTradePrice>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Observem que executem el procediment remot **GetLastTradePrice** amb el paràmetre **symbol**.

RESPOSTA

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
   <SOAP-ENV:Body>
       <m:GetLastTradePriceResponse xmlns:m="Some-URI">
           <Price>34.5</Price>
       </m:GetLastTradePriceResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

La resposta es col·loca entre un *tag* format pel nom del procediment + *Response*.

La resposta pot ser complicada ja que es poden rebre estructures com per exemple :

```
<m:GetLastTradePriceResponse
         xmlns:m="Some-URI">
           <PriceAndVolume>
               <LastTradePrice>
                   34.5
               </LastTradePrice>
               <DayVolume>
                   10000
               </DayVolume>
           </PriceAndVolume>
</m:GetLastTradePriceResponse>
```

També es poden utilitzar referències a les dades per si es repeteixen en més d'un lloc (una mena d'hiperlinks) :

```
<getCatImaResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
```

```
<getCatImaReturn href="#id0"/>
</getCatImaResponse>

<multiRef id="id0" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="ns1:Vector">
<item href="#id1"/>    <item href="#id2"/>

</multiRef>
```

Com podem observar **id0** és un apuntador a una referència situada més endavant que conté també dues referències a dos ítems més. Els dos mètodes tenen la seva utilitat, però normalment es fa servir el mètode sense referències ja que l'*overhead* és menor.

Una vegada tenim un WS donat d'alta el podríem publicar a una mena de pàgines grogues (UDDI) per tal que qualsevol persona que el vulgui utilitzar el pugui trobar. Donat les característiques dels nostres WS ometrem aquest pas, ja que coneixerem l'URL del servei; si no fos així el nostre client es podria connectar al servidor UDDI i realitzar la cerca de l'URL.

Els WS també disposen d'ajuts per poder veure els mètodes i paràmetres que contenen. Per aquesta tasca s'utilitza un document XML anomenat **WSDL** (*Web Service Description Language*). Observem una part d'un WSDL que descriu un dels serveis d'un Web Service:

```
<wsdl:operation name="getLinks" parameterOrder="sURL">
  <wsdl:input message="impl:getLinksRequest" name="getLinksRequest"/>
  <wsdl:output message="impl:getLinksResponse" name="getLinksResponse"/>
</wsdl:operation>
```

El mètode s'anomena *getLinks* i el paràmetre és *sURL*. En aquest document també s'inclou la definició de tipus no estàndards i les característiques del servei.

2.4 AXIS

Apache Axis[1] és un *servlet*, executat a través de Apache Tomcat, que permet servir Web Services. Concretament és una implementació del protocol SOAP.

Els Web Services es poden implementar de moltes maneres, anant des d'un nivell "*socket*" fins a un nivell en què podem programar una simple aplicació que no conté cap codi que ens doni una pista que és un WS. Aquesta és la funció de Apache AXIS, oferir-nos un entorn controlat on fer "*deployment*[2]" dels nostres WS.

Els WS creats per a AXIS són classes Java, en les que les operacions públiques (per defecte totes) es mostren a l'exterior mitjançant el protocol SOAP. AXIS aporta al programador una API (tant per a Java com recentment per a C++) que permet la flexibilitat d'un entorn basat en classes (podem escriure literalment el missatge SOAP que volem treure, o simplement afegir-ne els paràmetres, etc.). Tot i això nosaltres utilitzarem AXIS per la seva senzillesa en crear WS i així poder-nos centrar en l'objectiu del projecte.

[1] http://ws.apache.org/axis/, Podem descarregar la seva versió 1.1 final de http://ws.apache.org/axis/download.cgi
[2] Procés mitjançant el WS s'instal·la al servidor.

2.4.1 Implementació de Web Services amb AXIS.

A continuació donarem les principals pinzellades de la implementació de WS amb AXIS. Si es vol entrar en detall a l'**annex B** és pot trobar un petit document que explica pas a pas la instal·lació i creació d'un Web Service amb Axis.

Com hem dit els WS d'AXIS són simples classes Java amb els mètodes públics exposats. L'única restricció d'aquests procediments (si no volem especificar les funcions de serialització / deserialització) és la utilització dels següents tipus de dades definits a la versió 1.1. d'AXIS:

xsd:base64Binary	`byte[]`
xsd:boolean	`boolean`
xsd:byte	`byte`
xsd:dateTime	`java.util.`**`Calendar`**
xsd:decimal	`java.math.`**`BigDecimal`**
xsd:double	`double`
xsd:float	`float`
xsd:hexBinary	`byte[]`
xsd:int	`int`
xsd:integer	`java.math.`**`BigInteger`**
xsd:long	`long`
xsd:QName	`javax.xml.namespace.`**`QName`**
xsd:short	`short`
xsd:string	`java.lang.`**`String`**

Taula 2.2 - Tipus de dades d'Axis

A més a més, es recomana utilitzar com a col·leccions d'objectes el tipus Vector. (java.util.vector)

Anem a veure un petit exemple:

```
public class LinkGet {
  final String sVersio = "LinkGet V.1.0";
  public String getVersion ()        { return (sVersio);  }
  public int getSize (String cadena) { return (cadena.length());  }
}
```

Això és un WS d'AXIS; els dos mètodes (*getVersion* i *getSize*) són els que s'exposaran al servidor i els que es podran cridar de manera remota.

Per publicar-lo tenim diverses alternatives:

- Renombrar el fitxer **LinkGet.java** a **LinkGet.jws** i situar-lo al directori **\axis**. En aquesta modalitat el codi Java es compil·larà quan es necessiti.
- Realitzar un deployment més professional amb l'ajut de dos fitxers amb la descripció del **deploy** i del **undeploy** (deploy.wsdd i undeploy.wsdd).

En aquests fitxers es poden introduir les característiques dels WS que volem publicar. Trobem per exemple els mètodes públics que volem utilitzar o mapejar classes

específiques per a serialitzar-les o deserialitzar-les. Amb aquesta alternativa les classes Java han d'estar compilades i en format .**jar** o .**class**.

AXIS té una sèrie de funcionalitats afegides: una d'elles és la creació del **WSDL** de forma automàtica, l'únic que hem de fer és afegir a l'URL del WS ***?wsdl*** i AXIS ens retornarà el document **WSDL**. La següent funcionalitat afegida que disposem és la possibilitat de cridar als mètodes del WS mitjançant l'URL del WS :

URL + ?method=getSize&cadena=Exemple1

Això ens generaria el document XML de resposta al mètode *getSize* i amb el paràmetre cadena *Exemple1*.

```
<soapenv:Envelope>
<soapenv:Body>
<getSizeResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
       <getSizeReturn xsi:type="xsd:int">8</getSizeReturn>
</getSizeResponse>
</soapenv:Body>
</soapenv:Envelope>
```

Aquesta funcionalitat és útil per testejar els WS que construïm sense necessitat de crear un client.

2.5 MySQL

Per tal d'utilitzar persistència, s'ha utilitzat una base de dades. L'elecció de **MySQL** com a Sistema Gestor de Bases de Dades s'ha fet pel tipus de llicencia, no hem de fer cap despesa addicional per a utilitzar-la, i per ser suficientment potent[1] per el sistema que hem de construir.
Tenim una sèrie de desavantatges sobre els altres **SGBD**[2]'s com són la manca de procediments emmagatzemats (s'implementaran en la versió 5.0 de mySQL, actualment en fase de construcció) que podrien ser útils en el nostre projecte ja que els WS els cridarien i no necessitaríem replicar pràcticament codi al client WEB amb PHP i als WS. Tot i això ofereix una bona integració amb el llenguatge PHP, cosa que fa a **mySQL** especialment atractiu.

El mode per defecte de mySQL no permet claus foranes, ni ajudes com "ON DELETE CASCADE" per a mantenir la base de dades neta sense gaires esforços. Per a aconseguir aquesta funcionalitat hem de crear les taules de tipus **INNODB**[3], un nou tipus de taula que s'ha incorporat fa poc a mySQL (versió 4.0).

Tot i que tenim l' "ON DELETE CASCADE", no tenim l' "ON DELETE SET DEFAULT" que apareix en altres bases de dades i que provoca que haguem de realitzar un tractament extra a l'hora d'eliminar alguns elements del sistema d'informació.

Durant el desenvolupament es va trobar un *bug* en el conector Java – MySQL que impedia inserir imatges grans (>300 Kb) en una taula.

[1] http://www.eweek.com/slideshow/0,3018,p=1&a=23120&po=1&i=1,00.asp
[2] Sistema Gestor de Bases de Dades. Oracle, Informix, DB2, SQL-Server són altres SGBD's.
[3] http://www.innodb.com/index.php ,

2.6 PHP

PHP és un llenguatge de programació equiparable amb les **jsp** de Sun i els **asp** de Microsoft.

La seva funció és la mateixa, permetre realitzar pàgines web amb contingut dinàmic mitjançant l'execució d'un **script** PHP al servidor i mostrar el resultat al navegador de l'usuari. Amb PHP podem per exemple mostrar el contingut d'una base de dades al navegador de l'usuari, modificar la base de dades i sense tocar cap línia de codi veure el resultat actualitzat al navegador immediatament. Cal destacar la seva bona integració amb mySQL.

Durant el transcurs del PFC la versió de PHP inicialment utilitzada tenia un *bug* en el tractament d'algunes imatges (s'utilitzen algunes funcions de PHP per a generar una petita imatge a partir d'una de més gran), cosa que es va resoldre en una posterior revisió; això demostra el potencial de PHP, que està en un desenvolupament continuat.

2.7 Part Remota

Els clients utilitzaran els mètodes ofertats pels WS que farem utilitzant diverses implementacions, depenent dels següents factors:

- Maquinari (Pocket PC / Mòbil).
- Velocitat d'implementació.
- Cost.
- Necessitats.

En una primera aproximació a la tecnologia dels WS vàrem realitzar una prova pilot sobre maquinari Pocket PC amb Windows CE.

A continuació comentarem els trets característics de cadascuna de les implementacions.

2.7.1 Web Services amb Pocket PC / Win CE

Els Pocket PC són dispositius que cada cop tenen més velocitat de procés i més memòria disponible, la seva API és suficientment amplia per a poder realitzar qualsevol aplicació. En el nostre cas disposàvem de **64 Mb** de memòria (uns 10 Mb lliures) i una capacitat de procés donada per un processador **StrongARM**[1] a **206Mhz** sobre un Windows CE per a Pocket PC en la seva versió 2002.

El programari que podem fer servir per programar és limitat (en quant a possibilitat d'elecció): per una banda tenim el **Visual Studio 2003**, que inclou el necessari per a programar dispositius amb Windows CE (*SmartPhones*[2] i *Pocket PC*), i per l'altra el paquet gratuït de Microsoft anomenat **Embedded Visual Tools**, que és una suite formada per un Visual Basic i un Visual C++ que permeten generar codi per a dispositius Pocket PC.

Visual Studio 2003 (.NET) permet una programació senzilla i ràpida de clients de WS mitjançant la tecnologia .NET. D'alguna manera la podríem comparar a la que ofereix AXIS, ja que només ens cal assenyalar l'URL on és el WSDL del WS i ens generarà les classes necessàries. Quan es va fer la prova pilot la programació per **SmartPhones** només estava disponible amb els llenguatges de programació Visual Basic i Visual C#. El *handicap* d'aquesta alternativa és el cost del programari, a canvi d'una rapidesa

[1] Desenvolupat per DEC i Intel.
[2] Telèfon mòbil que disposa de Windows CE com a sistema operatiu.

d'implementació molt elevada; per altra banda necessitem instal·lar al Pocket PC el *.NET compact Framework* per a poder executar codi .NET al dispositiu que té una mida gens menyspreable.

Embedded Visual Tools[1] (EVT 3.0) en canvi no té cap llibreria incorporada per a programar clients de WS, ni tant sols per *parsejar* XML. EVT conté els llenguatges Visual Basic i Visual C++. Així doncs per a la implementació de WS necessitem l'ajut de llibreries externes (o bé construir-ne les nostres). Per sort existeix la llibreria anomenada PocketSOAP.[2]

PocketSOAP és una llibreria gratuïta que implementa el protocol SOAP, amb un baix cost de memòria i una API suficientment senzilla per a poder-la utilitzar (tot i que no arriba als nivells de Visual Studio 2003 o AXIS).

Aquesta llibreria, tot i que es pot utilitzar tant en Visual C++ com en Visual Basic, s'ha decidit utilitzar-la en Visual Basic per poder realitzar una millor comparació amb el client de Visual Studio 2003 (realitzat en Visual Basic).

L'API de PocketSOAP consisteix bàsicament en un conector HTTP per a realitzar la petició, i quan tenim el document XML utilitzem una sèrie de funcions per recuperar la informació que ens interessa.

En un altre apartat exposarem amb més detall totes les diferències trobades en aquest dispositiu.

2.7.2 Web Services en dispositiu mòbil J2ME.

En els dispositius mòbils trobem que la memòria és molt més limitada (sent optimistes podríem trobar uns 64 Kb per l'aplicació i 64 Kb de pila). No disposem d'accés al sistema de fitxers del mòbil i tenim al nostre abast les eines justes per a poder realitzar una aplicació. Com a símil, podem fer el que un Applet pot realitzar dins d'un navegador (és a dir no tenim accés, per exemple, al sistema de fitxers del dispositiu).

Per a programar disposem de moltes eines, una per a cada fabricant de dispositiu mòbil, així com les eines pròpies de Java i altres de propietàries (JBuilder o Java Sun One, per exemple). Nosaltres utilitzarem l'SDK[3] que ens dóna Nokia gratuïtament (com tots els altres fabricants) juntament amb emuladors del terminal mòbil.
Una vegada tenim l'SDK instal·lat i mirant l'API de J2ME observem que no tenim disponible cap *parser* de XML ni cap implementació del protocol SOAP, a l'igual que ens havíem trobat amb el Pocket PC.

Una visita a la web de **Sun Microsystems** dedicada a J2ME ens dóna les primeres pistes: Actualment J2ME no té suport per a WS, però s'està estudiant la seva incorporació. De fet el *draft*[4] de la implementació s'ha fet públic per a la seva revisió (tot i que porta molts mesos així).

Donada aquesta situació trobem una llibreria **kSOAP**[5] creada per Enhydra[6] que ofereix el el suport per a realitzar i rebre peticions SOAP a través de la màquina virtual **kVM**[1]

[1] Podem trobar el software de Microsoft a http://www.microsoft.com/windowsmobile/resources/downloads/developer/default.mspx
[2] Podem trobar pocketSOAP i la seva documentació a http://www.pocketsoap.com/
[3] Software Development Kit.
[4] http://www.jcp.org/en/jsr/detail?id=172
[5] http://ksoap.enhydra.org/ , podem llegir algun article sobre aquesta llibreria aquí http://Web Services.xml.com/pub/a/ws/2003/08/19/ksoap.html
[6] http://www.enhydra.org/, organització similar a Apache pero dedicada als servidors de aplicacions i e-Business.

(J2ME). Aquesta mateixa companyia ha realitzat altres llibreries entre les que trobem **kXML**, un parser l'XML amb baix cost de CPU i memòria. Aquestes llibreries també són de lliure utilització, i per tant el seu cost és zero.

Totes les classes *k** s'han d'incorporar al *JAR* que enviarem al mòbil, i com que aquest *JAR* no pot superar els 64 Kb (amb sort) hem d'escollir amb cura quines classes necessitem i quines no. Amb un ús normal la càrrega de la llibreria és d'uns 40 Kb, deixant-nos només amb 24 Kb per a la resta de l'aplicació. Això s'agreuja si necessitem tot el codi per a utilitzar variables en Base64 (per exemple).

La programació amb l'API és suficientment senzilla també en els casos que no necessitem res especial; si li demanem més coses trobarem problemes donada la poca documentació i exemples que hi ha disponibles. L'API s'assembla molt a PocketSOAP i ens permet l'accés tant a baix nivell (accedir al missatge SOAP tal com s'enviarà) com a nivell alt.

Per a poder reduir la mida del JAR existeixen diverses tècniques[2] renyides amb la modularitat i les bones formes; al ser Java un llenguatge semi interpretat els noms de les variables, classes, etc., es mantenen dins del jar. Per tant necessitem:

- Utilitzar el mínim de classes possibles.
- Utilitzar el nom de variables/procediments curts.
- Compil·lar sense informació de debug

Això ho podem aconseguir en la part de disseny, fent servir el mínim de classes possibles, i amb un **ofuscador** de codi Java que ens redueixi el nom de les variables i altres.

Per altra banda el dispositiu mòbil no pot perdre temps creant i destruint objectes, per tant és recomanable tenir els objectes que necessitarem creats (com a variables globals) i reutilitzar-los. Així el **Garbage Collector** de la maquina virtual kVM no es posarà a funcionar mentre estem executant l'aplicació.

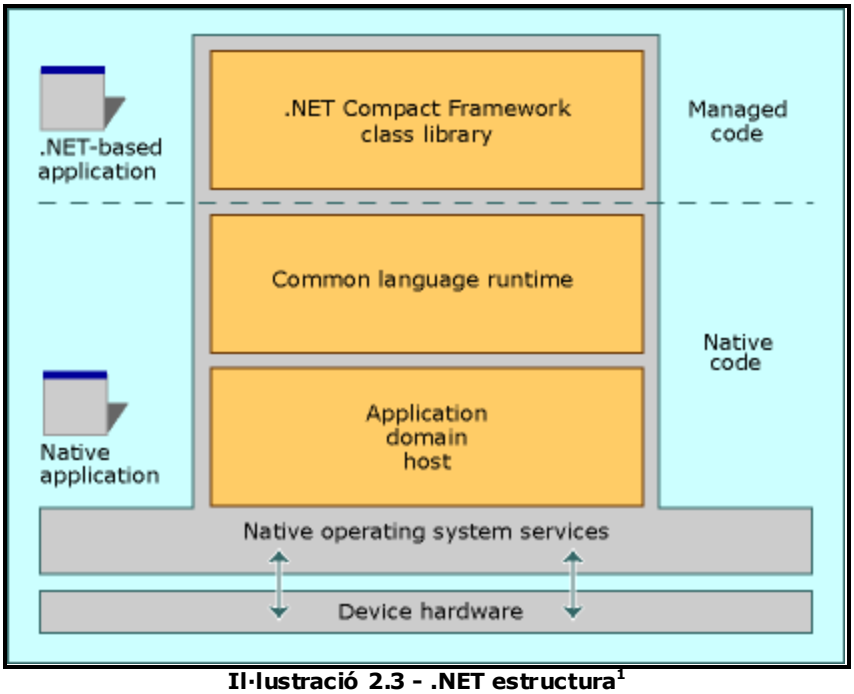
2.8 Programació

2.8.1 Pocket PC.

Els dispositius Pocket PC amb Windows CE com a sistema operatiu poden tenir característiques molt diferents l'un de l'altre. Entre aquestes característiques podem trobar la CPU, que pot tenir diversos models amb jocs d'instruccions diferents; entre elles podem trobar els **ARM** o els **MIPS**.

Aquesta diferència podem solucionar-la gràcies als compiladors, ja que poden generar codi pels diferents sistemes (**Embedded Visual Tools**), o amb el codi mateix, semblant a Java, que genera la plataforma .NET i que és pot executar en totes les plataformes que tinguin el **framework** de .NET instal·lat.

[1] Podem veure l'annex A per trobar més informació sobre kVM.
[2] Algunes les podem trobar aquí : http://java.sun.com/developer/J2METechTips/2002/tt0226.html , http://www.javaperformancetuning.com/tips/j2me.shtml



Il·lustració 2.3 - .NET estructura[1]

Per a poder provar el codi disposem d'emuladors plenament integrats amb el compilador, que ens permeten executar-lo amb les mateixes característiques que un dispositiu real. El nivell d'exactitud de l'emulació és tan elevada que fins i tot no necessitaríem provar-lo en un Pocket PC real; si necessitem depurar aplicacions podem fer-ho tant en l'emulador com en el dispositiu real, ja que els dos entorns de compil·lació ens ofereixen els recursos per fer-ho (execució pas a pas, per exemple).

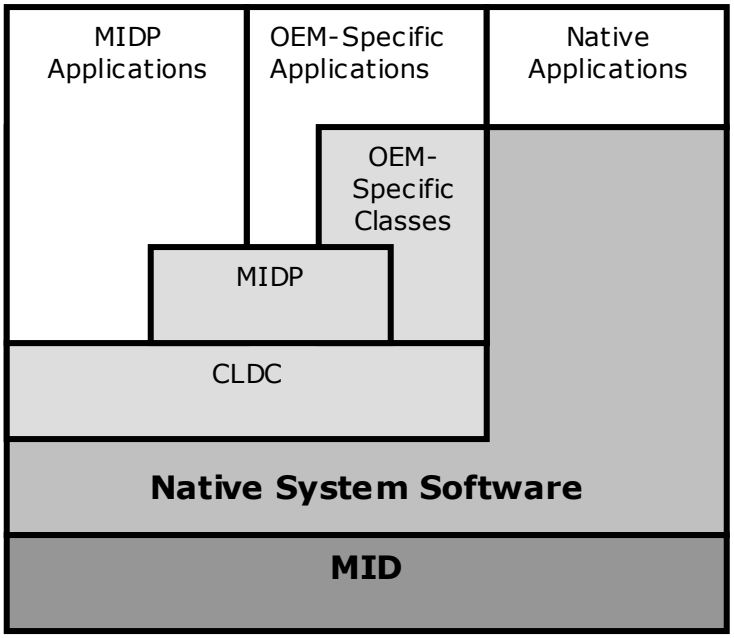
Per a instal·lar aquest entorn de desenvolupament simplement hem de descarregar el EVT de la web de Microsoft i fer la instal·lació de PocketSoap. En el cas de .NET necessitarem també el *.NET Compact Framework* per a poder executar aplicacions (en dispositius amb Pocket PC 2002 o inferior, normalment).

2.8.2 Mòbil J2ME

Per a programar un dispositiu J2ME, únicament necessitem l'SDK de *Sun* de J2ME. Per tal de facilitar la tasca de programació els fabricants de mòbils (*Sony-Ericcson*, *Nokia*, *Siemens*...) ens ofereixen, a través de les seves pàgines dedicades a desenvolupadors, paquets de programari amb tot el necessari: compilador, entorn d'empaquetament i testeig, documentació i emuladors per als diferents dispositius. En el nostre cas i donat que disposàvem d'un terminal Nokia vam escollir la *suite* de **Nokia**.

Nokia inclou diferents perfils per a cada dispositiu, així el **Nokia 6100** té un perfil de la Sèrie 40 amb un MIDP 1.0 (això restringeix les funcions de l'API accessibles, per exemple).

[1] www.yesky.com



Il·lustració 2.4 - J2ME i MIDP, estructura

Els emuladors de dispositius no arriben al nivell de perfecció dels del Pocket PC, ja que s'inclouen 3 emuladors: un per un mòbil estàndard de la *sèrie 40*, un altre pel *Nokia 7210*, i finalment un emulador per la *Sèrie 60*, la més avançada de Nokia. Aquests emuladors inclouen la possibilitat d'escriptura a consola (mitjançant un *System.out* / *System.err*) per a depurar les aplicacions i conèixer el flux de transmissió per **HTTP**, però no disposen de moltes més eines.

En altres SDK's com el de *Sony-Ericsson* disposem d'eines de depuració amb execució pas a pas, des del terminal real, i d'altres utilitats com poden ser activar el **Garbage Collector** a voluntat, veure l'estat de la memòria en qualsevol punt i utilitzar la connexió a Internet de l'ordinador als **Midlets** (nom que reben les aplicacions J2ME) que necessiten una connexió. Això evita fer un consum de **GPRS** en fase de testeig al terminal.

Per a l'execució en un terminal real podem realitzar la transferència mitjançant un cable sèrie, una connexió per infrarojos o fins i tot (Sony-Ericsson T610, Nokia 7650) mitjançant *BlueTooth,* sent aquesta última modalitat la més còmoda.

Il·lustració 2.5 - Adaptador Bluetooth

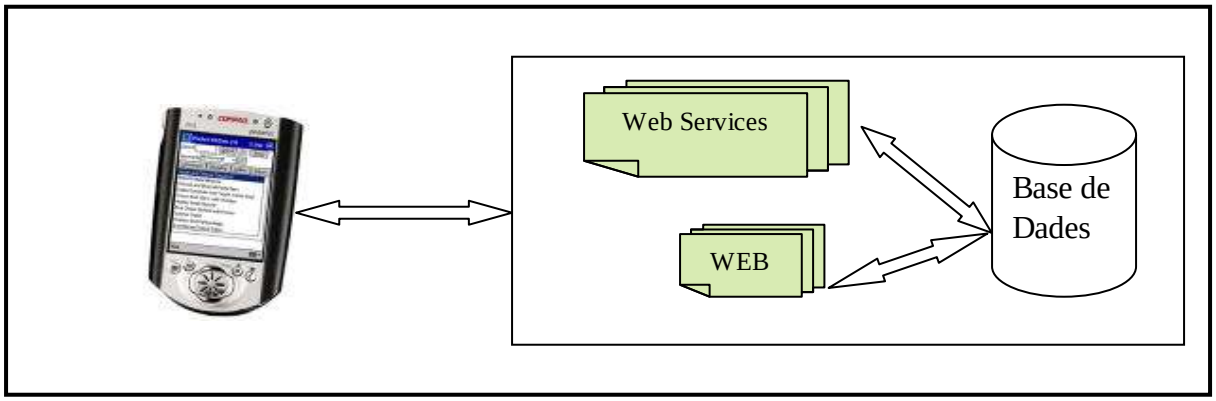
3. Prova Pilot

Per a poder fer una valoració de la viabilitat del projecte, quina profunditat havia de tenir i cap a on havíem de dirigir-nos, es va crear una prova pilot utilitzant un Pocket PC.

3.1 Funcionalitat

Dissenyar i implementar un sistema que permeti a un usuari remot actualitzar una pàgina web amb imatges. La comunicació es realitzarà mitjançant SOAP utilitzant Web Services. Entre les funcionalitats que s'implementaran podem trobar: afegir autors, imatges, eliminar-les...

Disposarem de l'aplicació al **Pocket PC** que demanarà els serveis al Web Service, el Web Service i finalment de la pàgina web, dinàmica, que mostrarà les imatges.



Il·lustració 3.1 - Esquema prova pilot

3.2 Material

Per a la implementació utilitzarem la següent tecnologia.
Banda PC

- **Servidor Web Apache**
- **PHP**
- **MySQL (SGBD)**
- **Apache Tomcat** 4.1
- **Apache Axis**

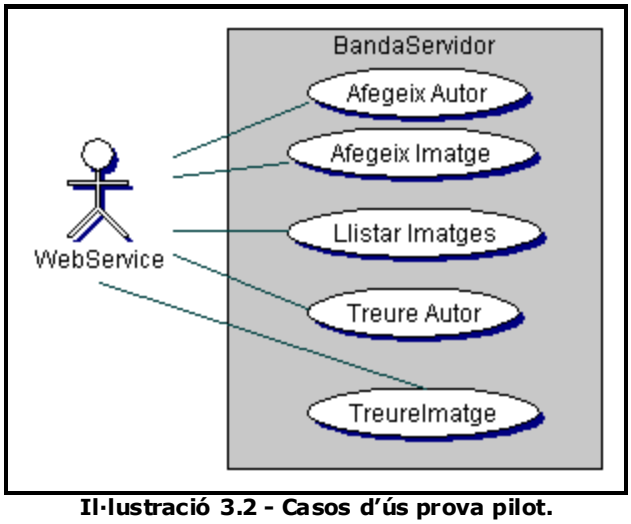
Banda Pocket PC

- **Visual Studio .NET 2003 (Visual Basic)**
- **Embedded Visual Tools 3.0 (eVBasic) + PocketSOAP**

Com podem observar el programari de la part PC és gratuït; en el cas de la banda Pocket PC la versió amb tecnologia .NET no ho és, com hem comentat abans.

3.3 Especificació

A continuació els casos d'ús del sistema que s'implementarà.



Il·lustració 3.2 - Casos d'ús prova pilot.

3.3.1 Contractes:

Cas d'ús: Afegeix Autor

Web Service	Sistema
1.- El *Web Service* vol afegir un nou usuari al *Sistema*.	
2.- Arriba al *Sistema* i li dóna el nom d'usuari desitjat i la contrasenya	**3.-** Es comprova si l'usuari existeix, si existeix retorna error. Si no existeix, afegeix l'usuari al *Sistema* i retorna cert.

Cas d'ús: Afegeix Imatge

Web Service	Sistema
1.- El *Web Service* vol afegir una nova imatge al *Sistema*.	
2.- Arriba al *Sistema* i li dóna el nom d'usuari i la contrasenya, a més tota La informació necessària per tal d'inserir la imatge (nom, descripció i dades)	**3.-** Es comprova si l'usuari existeix (i la contrasenya es valida), si no existeix retorna error.
	4.- Es comprova si la imatge de l'usuari existeix al Sistema (retorna fals). Si no existeix s'insereix al Sistema i retorna cert.

Cas d'ús: Llistar Imatges

Web Service	Sistema
1.- El *Web Service* vol llistar les imatges associades al seu usuari *Sistema*.	
2.- Arriba al *Sistema* i li dóna el nom d'usuari i la contrasenya.	**3.-** Es comprova si l'usuari existeix (i la contrasenya es valida), si no existeix retorna

	error.
	4.- Es retornen el nom de totes les imatges de l'autor.

Cas d'ús: Treure Autor

Web Service	Sistema
1.- El *Web Service* vol donar de baixa al seu usuari del *Sistema*.	
2.- Arriba al *Sistema* i li dóna el nom d'usuari i la contrasenya.	**3.-** Es comprova si l'usuari existeix (i la contrasenya es valida), si no existeix retorna error.
	4.- S'esborren totes les imatges de l'usuari al *Sistema* i finalment s'esborra l'usuari.

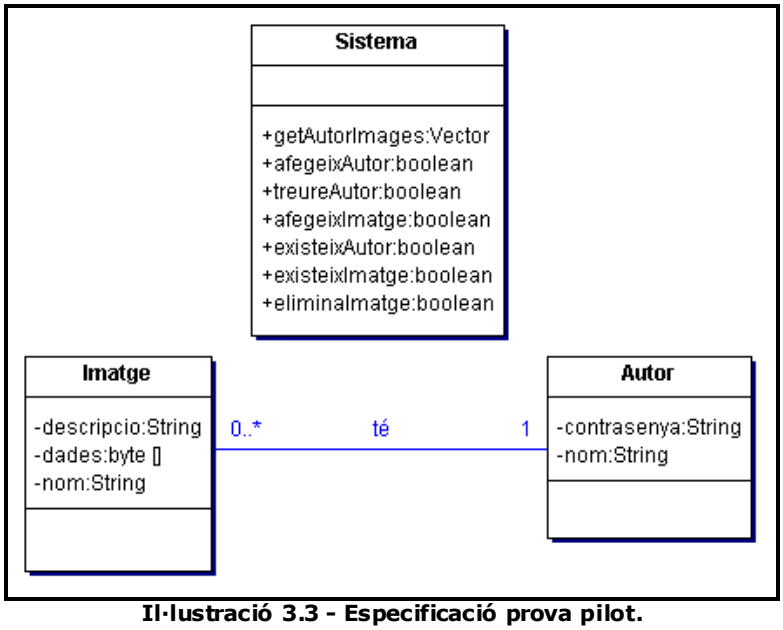
Cas d'ús: Treure Imatge

Web Service	Sistema
1.- El *Web Service* vol donar de baixa al seu usuari del *Sistema*.	
2.- Arriba al *Sistema* i li dóna el nom d'usuari i la contrasenya i el nom de la imatge que vol eliminar.	**3.-** Es comprova si l'usuari existeix (i la contrasenya es valida), si no existeix retorna error.
	4.- S'elimina del *Sistema* la imatge si existeix (retorna Cert)

3.3.2 Diagrama de classes (Especificació)

Podem especificar el sistema amb 3 classes: {**Sistema**}, {**Imatge**}, {**Autor**}
R.I. :

Per cada autor, només hi pot haver una imatge amb el mateix nom.
Només hi ha un autor amb el mateix nom.

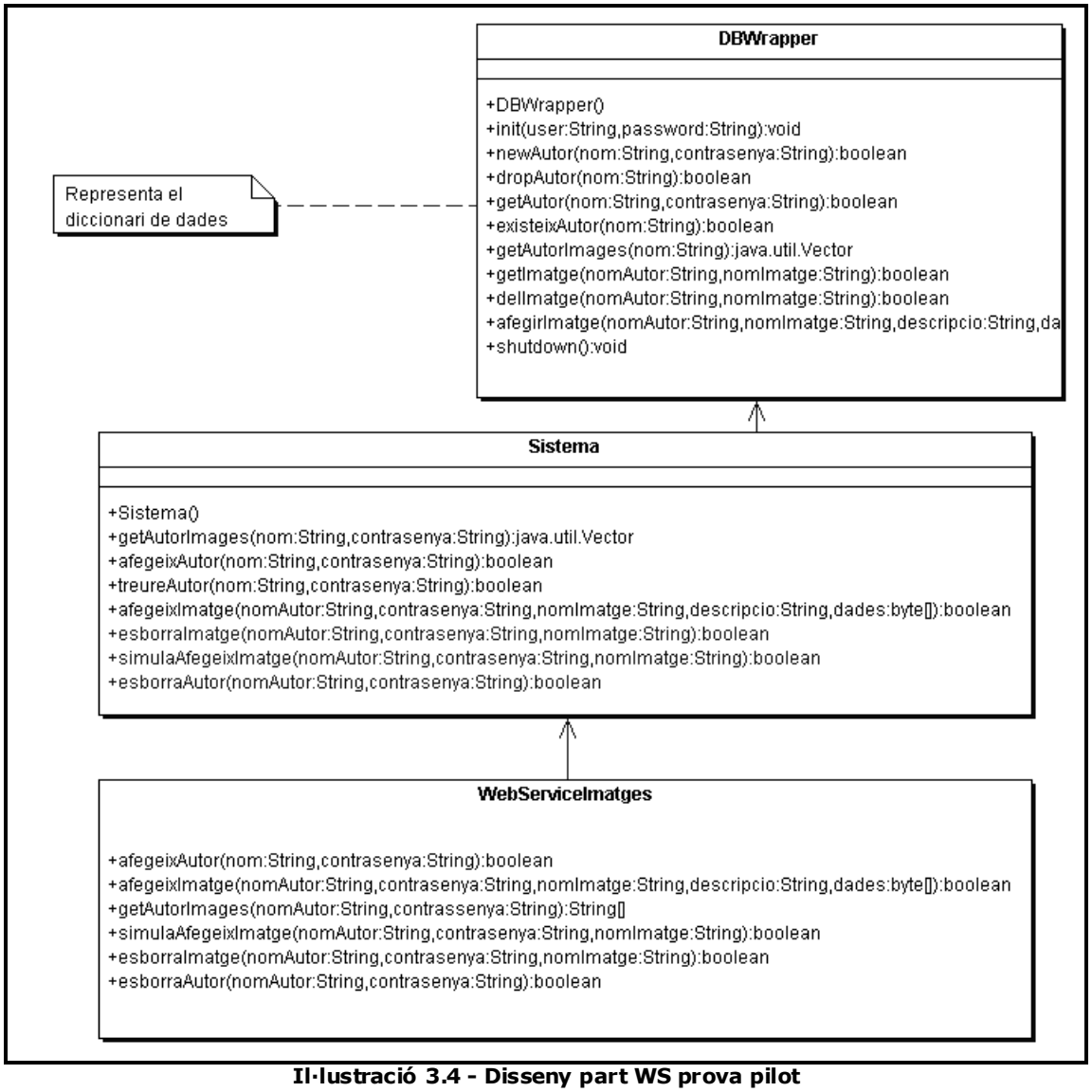


Il·lustració 3.3 - Especificació prova pilot.

3.3.3 Disseny

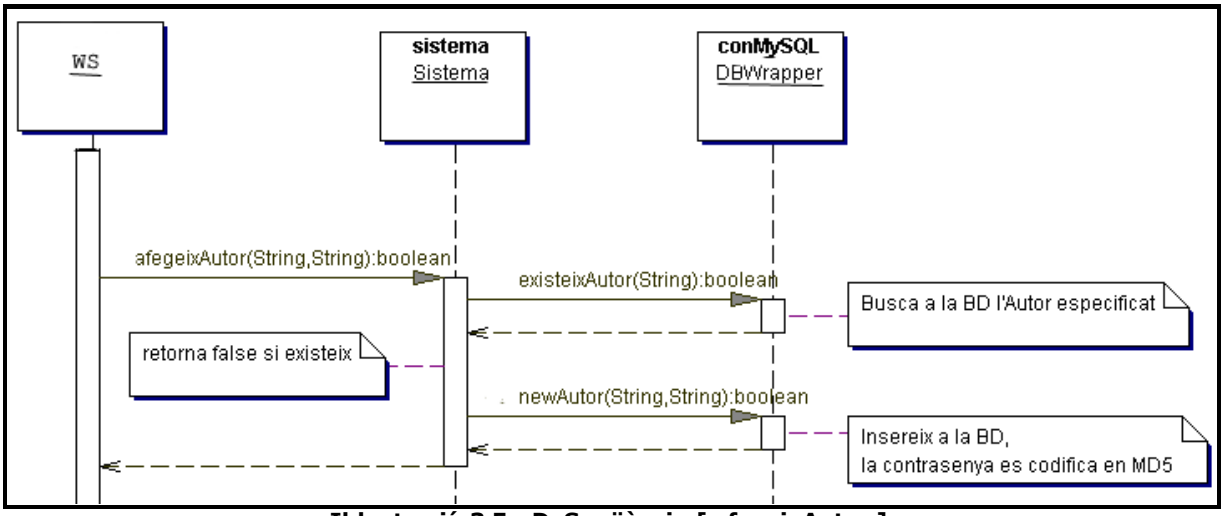
S'han tingut en compte els següents punts:

- Persistència de les classes Autor i Imatge en una base de dades relacional. Això afegeix una capa més al sistema: *DBWrapper*. La definició de la BD es pot trobar després.
- Afegim una nova classe (representant) Web Service que rebrà les peticions del client i les transmetrà al Sistema (i les respostes del Sistema al client). Les seves classes són les que seran visibles pel client (capa externa).
- Les peticions de recuperació/inserció passaran sempre per Sistema (i aquestes les portarà al Wrapper de la Base de Dades).

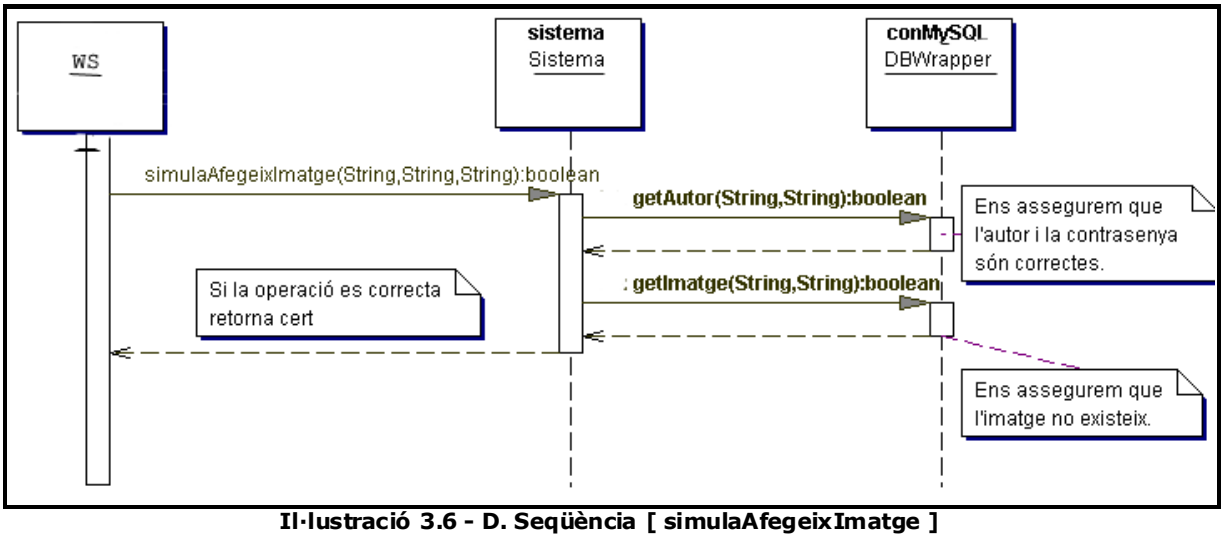


Il·lustració 3.4 - Disseny part WS prova pilot

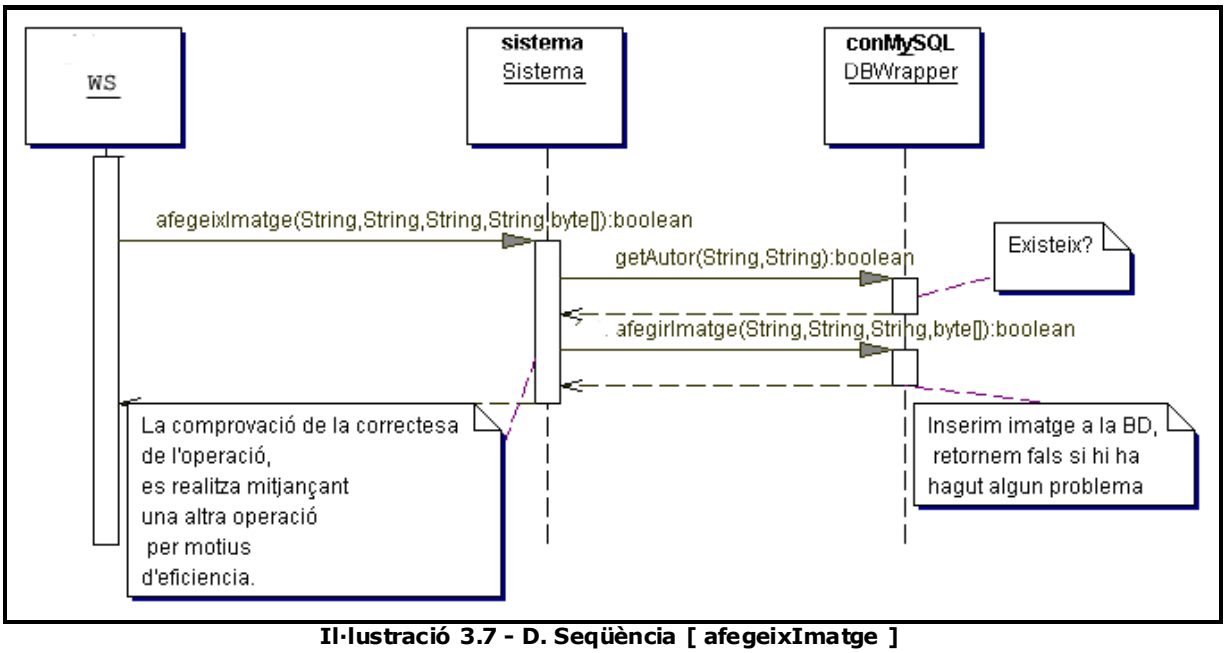
3.3.4 Diagrames de Seqüència



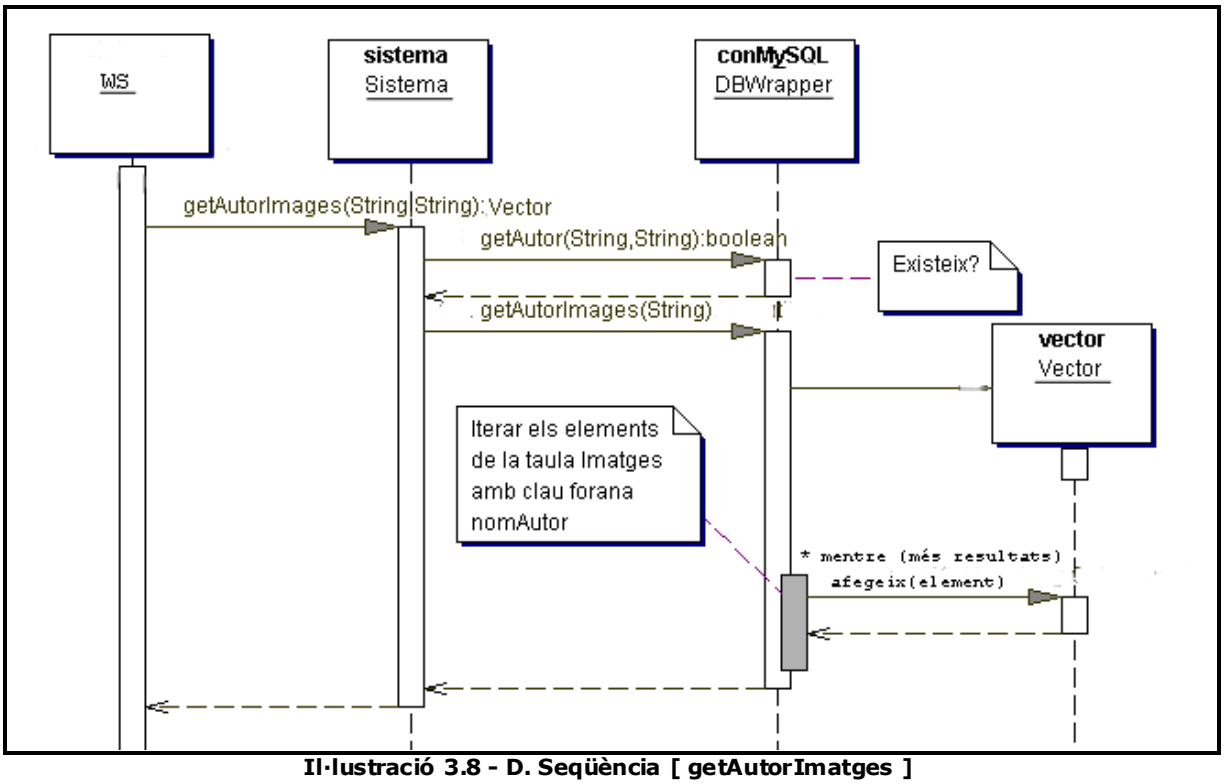
Il·lustració 3.5 - D. Seqüència [afegeixAutor]



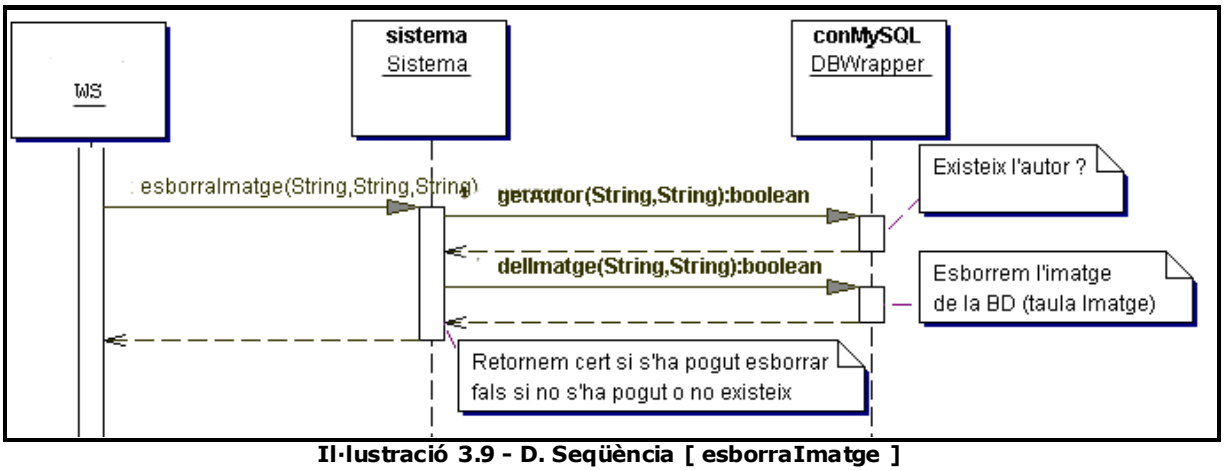
Il·lustració 3.6 - D. Seqüència [simulaAfegeixImatge]



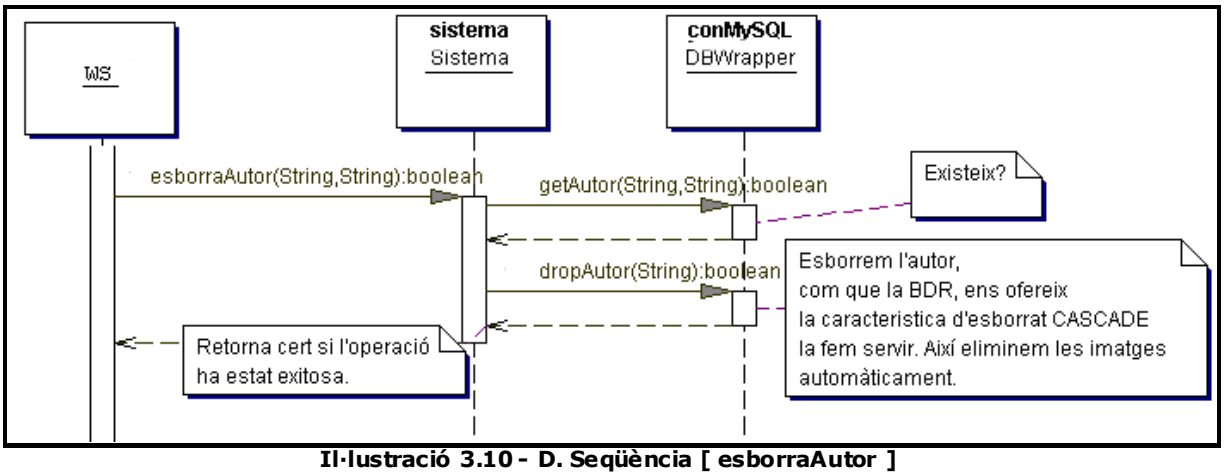
Il·lustració 3.7 - D. Seqüència [afegeixImatge]



Il·lustració 3.8 - D. Seqüència [getAutorImatges]



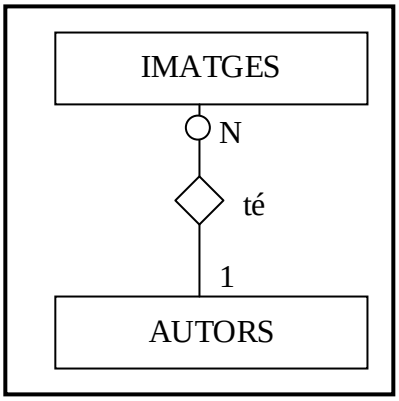
Il·lustració 3.9 - D. Seqüència [esborraImatge]



Il·lustració 3.10 - D. Seqüència [esborraAutor]

3.3.5 Base de Dades

De l'especificació podem treure el disseny de la BD:



Il·lustració 3.11 - Disseny BD prova pilot

Que podem traduir a :

Autors (nom , contrasenya)

Imatges (nom , descripció, dades , nomAutor[not null])

```
CREATE TABLE autors (
  nom char(30) NOT NULL,
  contrasenya char(255) NOT NULL,
  PRIMARY KEY  (nom)
)

CREATE TABLE imatges (
  nom varchar(50) NOT NULL default '',
  descripcio varchar(255) NOT NULL default '',
  dades mediumblob NOT NULL,
  nomAutor varchar(30) NOT NULL default '',
  PRIMARY KEY  (nom,nomAutor),
  FOREIGN KEY (nomAutor) REFERENCES autors (nom) ON DELETE CASCADE
)
```

Utilitzem la característica d'ON DELETE CASCADE per a que quan s'elimini un autor automàticament s'esborrin les seves imatges.

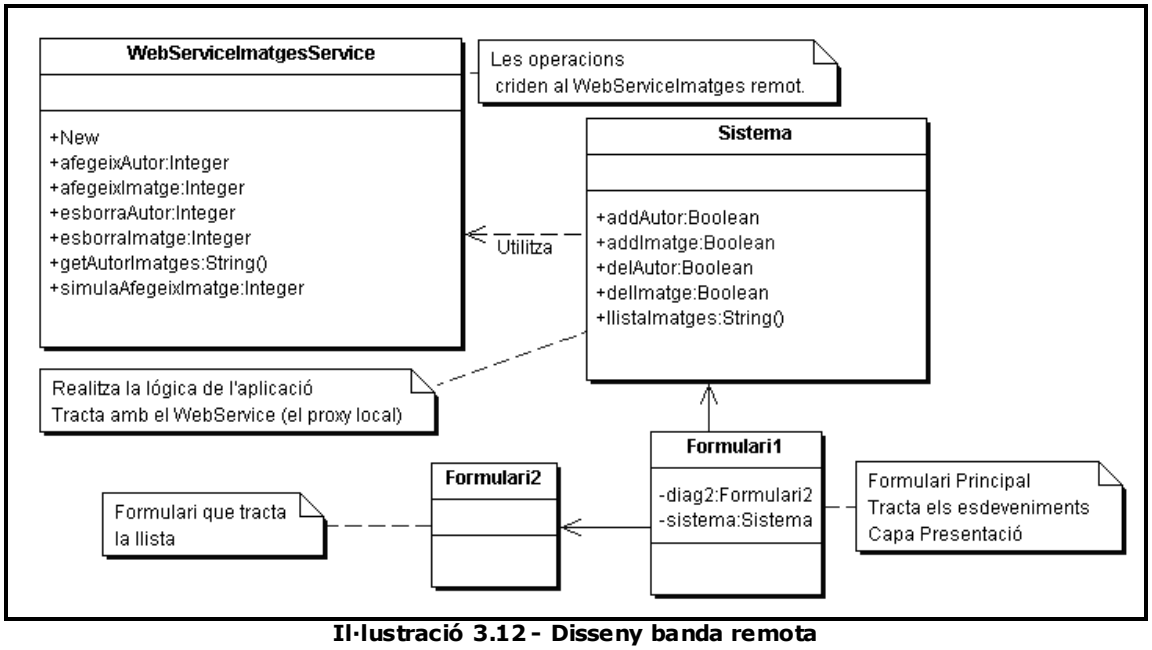
3.4 Banda Remota

En aquest cas la part remota no conté cap lògica important, solament demana/rep les dades i les mostra.

L'especificació consistiria en la classe Sistema, que conté les operacions a realitzar; el model conceptual és el mateix que la banda PC, ja que es pot veure el client com si fos local al Sistema.

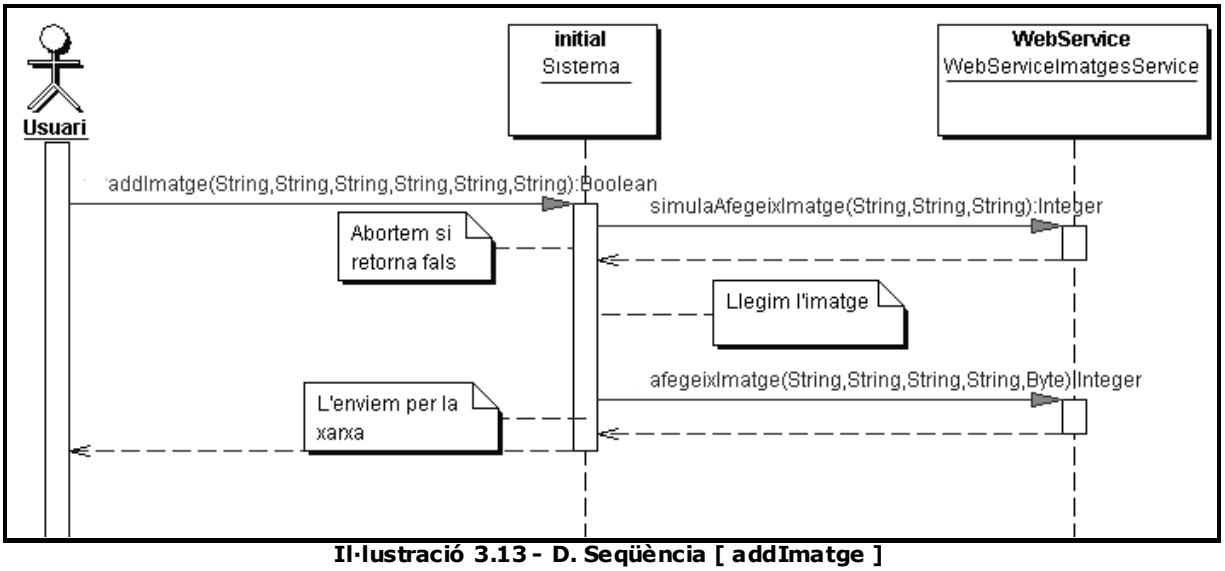
El diagrama de classes de disseny és el següent:

S'ha dissenyat un *proxy* per tal d'amagar les crides per la xarxa al Web Service (generació de la petició, etc...). Al ser una aplicació gràfica, tenim dos formularis que ens generen events i criden a les operacions de la classe Sistema (encarregada de cridar al proxy del Web Service).



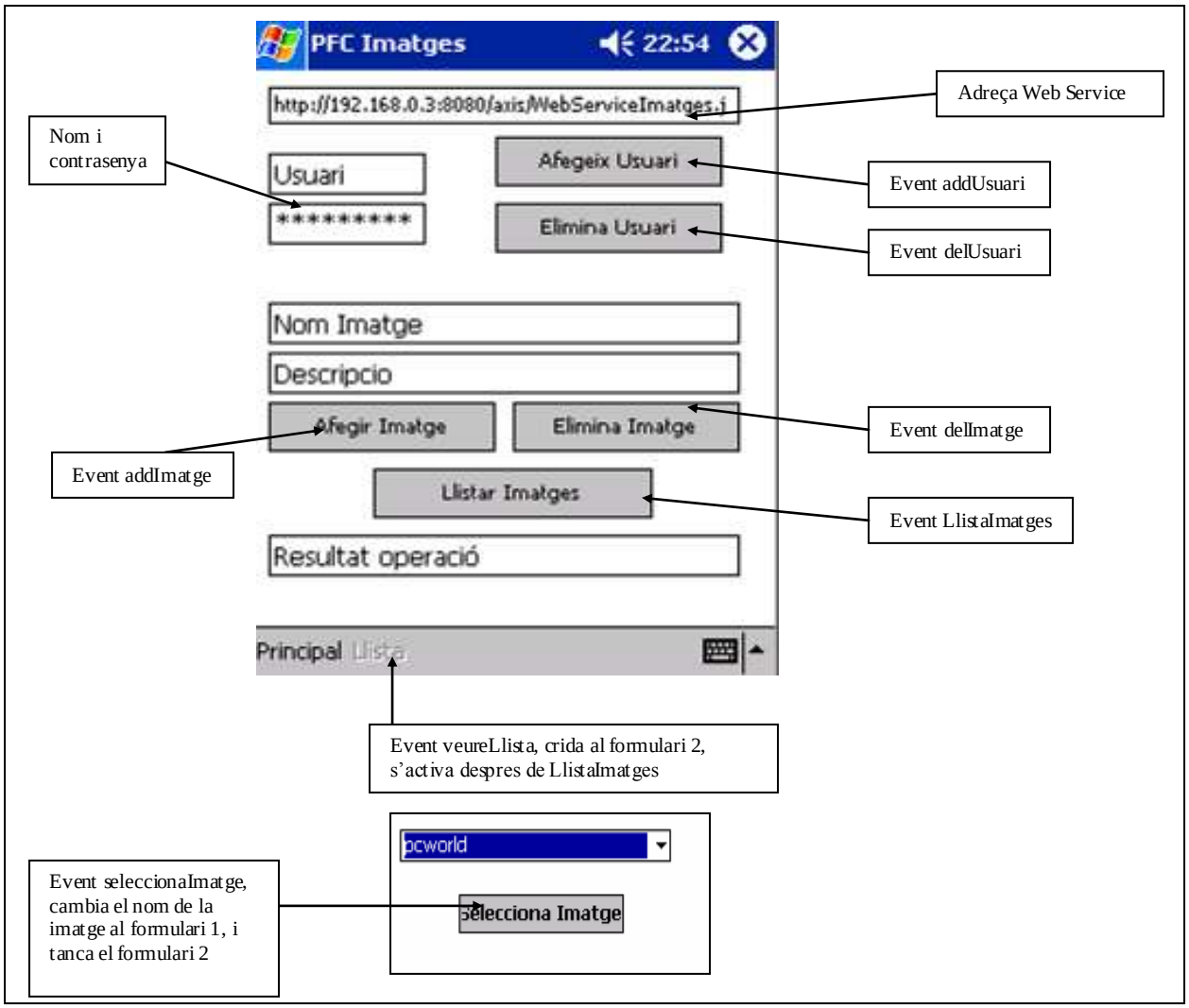
Il·lustració 3.12 - Disseny banda remota

Les operacions són simples crides al proxy amb els paràmetres capturats del formulari. L'única operació que té una mica de lògica al darrere és ***addImatge,*** ja que comprova abans d'enviar si es pot realitzar l'operació.



Il·lustració 3.13 - D. Seqüència [addImatge]

La interfície i els esdeveniments que es capturen són els següents:



Il·lustració 3.14 - Interfície gràfica prova pilot

3.5 Client Web

S'ha implementat mitjançant PHP una pàgina simple per tal de poder veure els resultats de les operacions realitzades remotament. Es mostren les imatges de cada autor agafant-les de la base de dades.

La principal operació és la següent: la seva funció és retornar el flux de dades des de la base de dades al navegador com si fos una imatge real.

mostraImatge.php
Entrada: $nomImatge, $nomAutor
Sortida: Resposta HTTP (tipus image/jpeg)

Hem de reconstruir les imatges a partir del flux de bytes que hem guardat a la base de dades, recuperable mitjançant la següent sentència SQL :

```
SELECT dades FROM imatges WHERE nom=$nomImatge AND NomAutor=$nomAutor
```

El contingut l'enviarem directament al navegador, utilitzant la capçalera següent: "**Content-type: image/jpeg**" i enviant el flux de bytes a continuació.

Cridant aquest fitxer php des de qualsevol navegador ens retornarà la imatge.

Les altres sentències SQL necessàries són:

```
Recuperar nom dels autors:
SELECT * FROM autors ORDER BY nom ASC

Recuperar imatges de l'autor:
SELECT * FROM imatges WHERE nomAutor=$autor
```

Podem trobar el codi d'aquesta web al CD-ROM que acompanya aquest document.

3.6 Parts representatives del codi

Exposarem a continuació algunes parts representatives del codi que s'ha implementat. La resta es pot trobar en el CD-ROM que acompanya aquest document.

3.6.1 .NET

Aquesta part del codi correspon a la classe Sistema (en Visual Basic .NET sí que podem tenir classes).

```
Public Function addImatge(ByVal remot As String, ByVal nomAutor As String, ByVal contrasenya
As String, ByVal nomImatge As String, ByVal descripcio As String, ByVal path As String) As
Boolean
    Dim Web Service As New PFCImatges.Web ServiceImatgesService
    Dim resultat As Boolean
    Web Service.Url = remot

    resultat = Web Service.simulaAfegeixImatge(nomAutor, contrasenya, nomImatge)
    If (resultat = False) Then
         ' La operació no serà exitosa, la abortem
         Return False
    Else
         'La operació pot anar bé, obrim la imatge i agafem les dades
          Dim fsstream As New System.IO.FileStream(path, System.IO.FileMode.Open)
          Dim dades(fsstream.Length - 1) As Byte
          Dim readbtes As Integer = fsstream.Read(dades, 0, fsstream.Length)
          fsstream.Close()

          If (readbtes <= 0) Then
              Return False ' L'operació crearía un fitxer buit.. abortem
          End If
          Return Web Service.afegeixImatge(nomAutor, contrasenya, nomImatge, descripcio,
dades)
     End If
End Function
```

Observem com s'encapsula la funcionalitat remota dins la classe Web Service; aquesta classe es realitza automàticament per .NET

Finalment mostrem el procediment que crida a aquesta funció per tal d'incorporar una imatge seleccionada al sistema, per a seleccionar-la s'utilitza un diàleg predefinit a .NET per a obrir fitxers.

```
Private Sub BOpen_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
BOpen.Click
    Dim result As New Boolean
    OpenFileDialog1.Filter = "Imatges (*.JPG;*.GIF)|*.JPG;*.GIF|Tots (*.*)|*.*"
    If (OpenFileDialog1.ShowDialog() <> DialogResult.OK) Then
        TResult.Text = "No has escollit cap fitxer"
    Else
        result = sistema.addImatge(TRemot.Text(), Tuser.Text(), Tcontrasenya.Text(),
Tname.Text(), TDesc.Text, OpenFileDialog1.FileName())
        If (result = True) Then
            TResult.Text = "Correcte"
        Else
            TResult.Text = "Usuari incorrecte o nom existent"
        End If
    End If
End Sub
```

3.6.2 EVT + PocketSOAP

Mostrem les mateixes funcions, aquest cop creades per pocketSOAP i EVT.

```
Public Function addImatge(ByVal remot As String, ByVal nomAutor As String, ByVal contrasenya
As String, ByVal nomImatge As String, ByVal descripcio As String, ByVal path As String) As
Boolean
   Const GENERIC_READ = &H80000000
   Const GENERIC_WRITE = &H40000000
   Const OPEN_EXISTING = 3
   Const FILE_SHARE_READ = &H1
   Const FILE_SHARE_WRITE = &H2

   Dim resultat As Boolean

   resultat = r_simulaAfegeixImatge (remot, nomAutor, contrasenya, nomImatge)
   If (resultat = False) Then
      ' La operació no serà exitosa, la abortem
      addImatge = False
      Return
   Else
      Dim FileSize As Long
      Dim hFile As Long
      Dim dwRead As Long
```

```
      hFile = CreateFile(path, GENERIC_READ + GENERIC_WRITE, FILE_SHARE_READ +
FILE_SHARE_WRITE, 0, OPEN_EXISTING, 0, 0)


      FileSize = GetFileSize(hFile, 0)
      CloseHandle (hFile)
      Dim Dades


      Form1.File1.Open path, fsModeBinary
      Dades = Form1.File1.InputB(FileSize)


      Form1.File1.Close
      addImatge = r_AfegeixImatge (remot, nomAutor, contrasenya, nomImatge, descripcio, dades)
   End If
End Function
```

Observem que l'API de winCE és més complicada que la de .NET, a més com que *Visual Basic* no té un suport molt natural s'han de definir les constants.

Tot seguit, en el procediment que crida a l'anterior observarem que no hi ha gaires diferències amb .NET:

```
Private Sub BAddImatge_Click()
Dim result As Boolean
    Cdialog1.Filter = "Imatges (*.JPG;*.GIF)|*.JPG;*.GIF|Tots (*.*)|*.*"
    If (Cdialog1.ShowOpen <> 0) Then
      TResult.Text = "No has escollit cap fitxer"
    Else
      result = addImatge(TRemot.Text, Tuser.Text, Tcontrasenya.Text, Tname.Text, Tdesc.Text,
Cdialog1.FileName)
      If (result = True) Then
         TResult.Text = "Correcte"
      Else
         TResult.Text = "Usuari incorrecte o nom existent"
      End If
    End If
End Sub
```

I per acabar els dos mètodes que criden a pocketSOAP:

```
Public Function r_simulaAfegeixImatge(ByVal remot As String, ByVal nomAutor As String, ByVal
contrasenya As String, ByVal nomImatge As String) As Boolean
    SOAPInit


    pEnvelope.SetMethod "simulaAfegeixImatge", ""
    pEnvelope.Parameters.Create "nomAutor", nomAutor
    pEnvelope.Parameters.Create "contrasenya", contrasenya
    pEnvelope.Parameters.Create "nomImatge", nomImatge


    pHTTP.Send remot, pEnvelope.serialize
```

```
    pEnvelope.parse pHTTP
    r_simulaAfegeixImatge = pEnvelope.Parameters.ItemByName("simulaAfegeixImatgeReturn").Value
    Return
End Function
```

L'ordre natural d'una petició a un WS és: s'inicialitza el missatge SOAP (*SOAPInit)*, s'introdueix el mètode a cridar (*SetMethod)* i finalment s'introdueixen els paràmetres. Una vegada fet això ja tenim el missatge preparat, i el que realitzem a continuació és enviar-lo (serialitzat) utilitzant el transport HTTP (*Send*) i processem la resposta (*parse)*; per acabar agafem l'element de la resposta que ens interessa mitjançant *ItemByName.*

```
    Public Function r_AfegeixImatge(ByVal remot As String, ByVal nomAutor As String, ByVal
contrasenya As String, ByVal nomImatge As String, ByVal descripcio As String, ByRef dades As
Variant) As Boolean
    SOAPInit
    pEnvelope.SetMethod "afegeixImatge", ""
    pEnvelope.Parameters.Create "nomAutor", nomAutor
    pEnvelope.Parameters.Create "contrasenya", contrasenya
    pEnvelope.Parameters.Create "nomImatge", nomImatge
    pEnvelope.Parameters.Create "descripcio", descripcio
    pEnvelope.Parameters.Create "dades", Dades
    pHTTP.Send remot, pEnvelope.serialize
    pEnvelope.parse pHTTP
    r_AfegeixImatge = pEnvelope.Parameters.ItemByName("afegeixImatgeReturn").Value
    Return

End Function
```

Observem que tot i no ser gaire complicat d'utilitzar **pocketSOAP**, la dificultat apareix en la utilització de Visual Basic, amb el qual sense la funcionalitat de **.NET** hem de donar moltes voltes per a poder obrir un fitxer, obtenir la seva mida i llegir el contingut. Per fer-nos una idea, s'ha trigat més en aconseguir obrir/llegir el fitxer que en crear pràcticament tota l'aplicació en .NET.

3.7 Conclusions de la prova Pilot

Si comparem les dues implementacions a nivell de codi observarem la claredat del codi que s'aconsegueix amb poc esforç en la implementació amb .NET. Amb .NET només cal que assenyalem al WSDL i ens generarà tot el necessari per a poder invocar els procediments remots; a més a més, els demés aspectes de la implementació, com poden ser tractaments de fitxers, són una mica més fàcils amb l'API de .NET que amb la de *Win CE*, però no suposa cap problema.

En l'apartat d'interfície gràfica en les dues variants és similar ja que tots dos són un llenguatge de programació visual que facilita enormement la seva creació.

Com a punts febles en la implementació amb .NET i la implementació amb PocketSOAP i EVT trobem la velocitat d'inici de l'aplicació, que és més ràpida amb PocketSOAP, i el consum de memòria, ja que al cost de l'aplicació hem de sumar-li el cost del ***Compact FrameWork***[1] de .NET.

[1] http://msdn.microsoft.com/mobility/prodtechinfo/devtools/netcf/

Donat l'elevat preu de l'entorn **VS .NET,** si el temps d'implementació no és crític és recomanable l'altra alternativa, ja que ofereix pràcticament el mateix de manera gratuïta. A més **PocketSOAP** té l'avantatge de ser un producte en contínua evolució i de codi obert, per tant la seva implementació millora i va incorporant els últims estàndards.

En l'ultima versió de **PocketSOAP** (1.4.3) s'inclou suport per a tots els tipus de dades més utilitzats (**Base64**[1] per exemple), així com arrays multidimensionals i tipus complexos. Aquesta llibreria conté suport per a **DiME** i per a Soap with Attachments (**SWA**), cosa que en el cas de J2ME i kSOAP no ho trobem; també inclou suport per a serialitzadors de dades per a poder construir els que necessitem.

PocketSOAP encara no compleix el necessari per a complir amb la versió 1.2 de SOAP. [2]

Sobre la capacitat de procés d'arxius XML grans, no hem tingut cap problema en transmetre fitxers de 2 Mbyte utilitzant Base 64 dins d'un missatge SOAP tant amb PocketSOAP com amb .NET; per tant podríem acceptar que totes dues implementacions són igual de vàlides.

L'elecció d'una o altra plataforma depèn del nostre entorn de desenvolupament i necessitats d'implementació; .NET també té un punt a favor, ja que PocketSOAP no té una documentació i una col·lecció d'exemples gaire extensa, fet que provoca que el seu ús sobre Visual C++ sigui difícil. Com a punt en contra hi ha la impossibilitat de desenvolupar aplicacions per a Pocket PC en Win CE amb un altre sistema operatiu que no sigui Windows.

[1] Base64 s'utilitza per enviar dades binàries a través de text ASCII, requereix una conversió i un augment de mida al transmetre les dades-
[2] Podem llegir les característiques de PocketSOAP aquí http://www.pocketsoap.com/pocketsoap/

4. Butlletí de Notícies amb actualització remota

4.1 Introducció

Crearem un sistema d'informació per tal de trobar les dificultats de la utilització de WS en un terminal J2ME, per a que el seu ús sigui el més proper a la realitat possible.

4.2 Funcionalitats i característiques

Aquest sistema d'informació tindrà les següents característiques:

- Incorporació d'imatges obtingudes a través de la càmera d'un mòbil, per exemple, o des d'un Pocket PC.
- Creació de notícies formades per imatges i text seguint una plantilla creada que indicarà les posicions on es col·loquen els elements. Aquestes notícies es construiran utilitzant la façana web o a través d'un mòbil (o Pocket PC) mitjançant WS.
- Possibilitat de deixar missatges per a altres usuaris.
- Normalment els elements disposaran d'altes, baixes i modificacions.
- Tindrem diversos rols per a controlar cada un dels aspectes (administrador, creador de notícies, creador d'imatges, ISP, plantilles, usuari...)
- Una façana web donarà accés a totes les facilitats per a realitzar les funcions a través d'un navegador, d'un ordinador estàndard i per les tasques de manteniment (actualització pàgines estàtiques etc...).
- Aquesta façana web serà dinàmica i situada a un ordinador personal amb accés a Internet, per això s'implementarà una funcionalitat d'actualització de pàgines estàtiques amb l'últim contingut per tal de mantenir a la xarxa (a un ISP gratuït, per exemple) una versió estàtica de la pàgina.

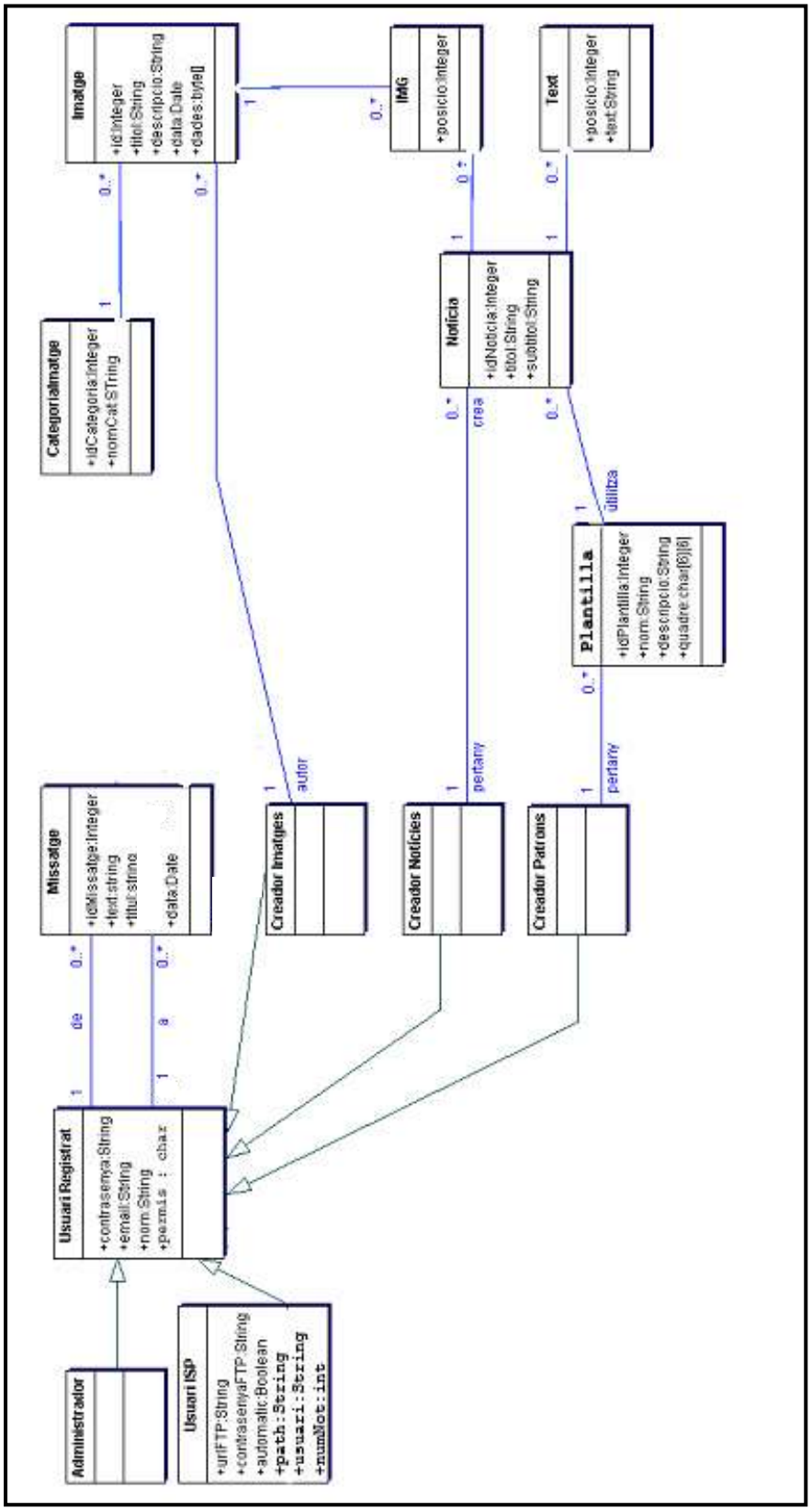
Començarem a crear el sistema amb un prototipus que incorporarà la funcionalitat bàsica per finalitzar-lo posteriorment.

Donada la utilitat final del sistema, treure fotografies, escriure la noticia i enviar-la, és adient per a una aplicació basada en un telèfon mòbil amb càmera, ja que ens permet total independència (amb un sol dispositiu ho podem fer tot): podem fer la fotografia (depenent de les característiques del terminal tindrà més o menys qualitat)[1], escriure el missatge (com si fos un SMS) i finalment activar la connexió a Internet (GPRS o GSM) i enviar les dades al WS del servidor.

[1] La qualitat de la càmera del Nokia 6100 arriba a una resolució de 640x480. La d'un Sony Ericsson T610 a 352x288. Actualment ha sortit el primer mòbil amb una qualitat de 2 Megapixels http://www.vodafone.jp/english/products/kisyu/v601sh/index.html

4.3 Especificació del sistema

Exposarem a continuació l'especificació del sistema a construir.



Il·lustració 4.1 - Especificació del sistema

Les claus de les classes són tots els atributs id, i per la classe Usuari Registrat el nom.
Per cada notícia només podem tenir una classe text a una posició determinada. El mateix ocorre amb IMG, que solament pot tenir una a cada posició.

Comentem algunes de les parts de l'especificació:

4.3.1 Jerarquia d'usuaris:

- Administrador: Té el control total de tot el sistema
  - Accés directe a la base de dades
  - Afegir / Eliminar usuaris
  - Donar / Revocar permisos
  - Afegir / Eliminar elements del sistema (notícies, plantilles, imatges)
  - Actualitzar totes les pàgines estàtiques

- Creador de notícies:
  - Afegir / Modificar / Eliminar les seves notícies

- Creador d'imatges
  - Afegir / Modificar / Eliminar imatges
  - Afegir Categories

- Creador de patrons
  - Permet dissenyar els patrons de notícies que es faran servir per donar format a les notícies.

- Usuari simple (no registrat)
  - Veure el butlletí a través de la façana Web, les imatges etc...

- Usuari 'ISP'
  - L'usuari podrà indicar en el seu perfil les dades necessàries perquè quan l'administrador actualitzi els ISP es generi una pàgina web amb HTML pla, i es transmeti al seu servidor Web mitjançant FTP. També pot realitzar ell mateix l'actualització manual.

- Usuaris registrats, podran llegir o crear missatges amb destinatari un altre usuari registrat.

4.3.2 Construcció de notícies:

Les notícies estaran formades per tres elements:
- El patró o plantilla amb la qual es visualitzarà a la façana Web
- Els elements de text, que constaran de text i de la posició que han d'ocupar en la plantilla.
- Els elements d'imatge, que consten de la imatge a inserir i de la posició que ocuparan en la plantilla.

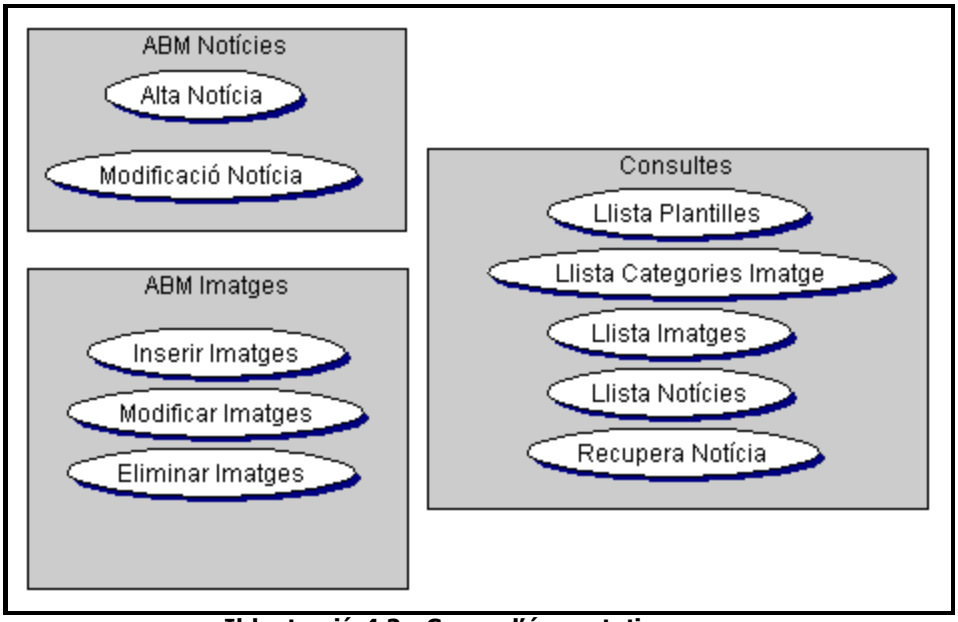
4.3.3 Imatges:

Les imatges constaran d'una categoria per tal de classificar-les i de l'autor.

Dit això escollirem un subconjunt de casos d'ús del sistema per tal d'implementar un primer prototipus.

4.4 Subconjunt de casos d'ús a implementar

- No s'implementarà la lògica de permisos (Diferents rols), tot i això només els usuaris registrats a la base de dades tindran accés (per motius de seguretat).



Il·lustració 4.2 - Casos d'ús prototipus

Dintre de l'ABM usuari, s'implementarà Alta Usuari.

4.4.1 Contractes

ABM Usuari

Alta Usuari:
Usuari no registrat

1. L'usuari s'identifica	*2. Si la identificació és correcta es continua, sinó es demanen les dades de registre.*
3. L'usuari envia les dades de registre	*4. L'usuari s'insereix al sistema si tot és correcte (nom correcte).*

ABM Notícies

Alta Notícia:
Usuari: Creador de Notícies i Administrador

1. L'usuari s'identifica	*2. Si la identificació és correcta es continua*
3. L'usuari envia la Notícia	*4. S'insereix la notícia al sistema i es retorna afirmatiu a l'usuari.*

Modificació Notícia:
Usuari: Creador de Notícies i Administrador

1. L'usuari s'identifica	*2. Si la identificació és correcta es continua*
3. L'usuari demana una notícia (idNotícia)	*4. El sistema retorna a l'usuari les dades de la notícia.*
5. Dóna les dades al sistema	*6. El sistema comprova que l'usuari pot modificar la notícia, i procedeix. Retorna el resultat de l'operació a l'usuari.*

Esborrar Notícia (No pertany al prototipus):
Usuari: Creador de Notícies i Administrador

1. L'usuari s'identifica	*2. Si la identificació és correcta es continua*
3. L'usuari especifica l'idNotícia que vol esborrar.	*4. El sistema esborra la notícia si l'usuari té permís per a fer-ho.*

ABM Imatges

Inserir Imatges:
Usuari: Creador d'imatges i Administrador

1. L'usuari s'identifica	*2. Si la identificació és correcta es continua*
3. L'usuari envia la imatge (i les dades)	*4. S'insereix la imatge al sistema i s'informa a l'usuari del resultat*

Modificar Imatge:
Usuari: Creador d'imatges i Administrador

1. L'usuari s'identifica	*2. Si la identificació és correcta es continua*
3. L'usuari demana una imatge (idImatge)	*4. El sistema retorna a l'usuari les dades de la imatge..*
5. Retorna les dades al sistema	*6. El sistema comprova que l'usuari pot modificar la imatge, i procedeix. Retorna el resultat de l'operació a l'usuari.*

Esborrar Imatge:
Usuari: Creador d'imatges i Administrador

1. L'usuari s'identifica	*2. Si la identificació és correcta es continua*
3. L'usuari demana una imatge (idImatge)	*4. El sistema comprova si té permisos per esborrar la imatge, i procedeix. Retorna el resultat de l'operació a l'usuari.*

Consultes

Llista Plantilles:
Usuari: Qualsevol

1. L'usuari demana la llista de Plantilles que hi ha al sistema	*2. El sistema prepara la llista i li dóna a l'usuari (idPlantilla, nom i descripció)*

Llista Categories Imatge:
Usuari: Qualsevol

1. L'usuari demana la llista de Categories que hi ha al sistema.	*2. El sistema prepara la llista i li dóna a l'usuari (idCategoria, nom)*

Llista Imatges:
Usuari: Qualsevol

1. L'usuari demana la llista d'imatges que hi ha al sistema d'una categoria determinada..	*2. El sistema prepara la llista i li dóna a l'usuari (idImatge, nom, descripció)*

Llista Notícies:
Usuari: Qualsevol

1. L'usuari demana la llista de Notícies que hi ha al sistema.	*2. El sistema prepara la llista i li dóna a l'usuari (idNoticia, títol) (Ordenada per data)*

Recupera Notícia:
Usuari: Qualsevol

1. L'usuari demana una notícia amb l'idNoticia	*2. El sistema prepara la noticia (si existeix) i li retorna a l'usuari.*

Llista Usuaris:
Usuari: Qualsevol

1. L'usuari demana la llista d'usuaris del sistema.	*2. El sistema prepara la llista i li retorna l'usuari.*

ABM Missatge (no pertany al prototipus)

Inserir Missatge:
Usuari: Usuari Registrat

1. L'usuari s'identifica	*2. Si la identificació és correcta es continua.*
3. L'usuari envia el missatge (amb el destinatari).	*4. S'insereix el missatge al sistema i s'envia a l'usuari del resultat.*

Modificar Missatge:
Usuari: Usuari registrat (creador o administrador)

1. L'usuari s'identifica.	*2. Si la identificació és correcta es continua.*
3. L'usuari envia l'identificador del missatge que vol modificar (idMissatge).	*4. El sistema retorna a l'usuari les dades del missatge, si l'usuari pot veure el missatge.*
5. Retorna les dades al sistema.	*6. El sistema comprova que l'usuari pot modificar el missatge, i procedeix.*

	Retorna el resultat de l'operació a l'usuari.

Esborrar Missatge:
Usuari: Usuari Registrat (creador, administrador o destinatari)

1. L'usuari s'identifica	*2. Si la identificació és correcta es continua*
3 L'usuari dóna al sistema l'identificador del missatge que vol esborrar (idMissatge)	*4. El sistema comprova si pot esborrar el missatge i retorna el resultat de l'operació.*

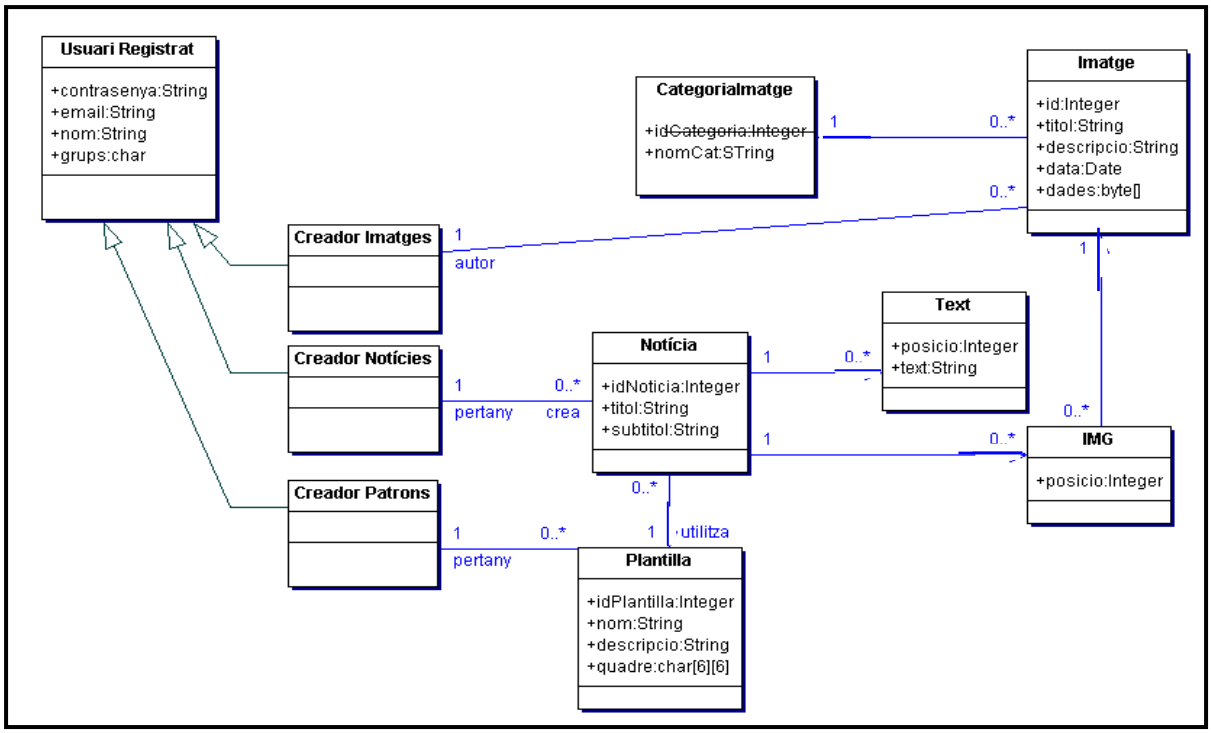
Llista Missatges:
Usuari: Usuari registrat (creador o administrador)

1. L'usuari s'identifica	*2. Si la identificació és correcta es continua*
3. L'usuari demana la llista de missatges.	*4. El sistema retorna a l'usuari les dades dels missatges que pot llegir.*
5. L'usuari rep el missatge	

Llegir Missatge:
Usuari: Usuari registrat (creador o destinatari)

1. L'usuari s'identifica.	*2. Si la identificació és correcta es continua.*
3. L'usuari demana el missatge amb l'identificador (idMissatge).	*4. El sistema retorna a l'usuari les dades del missatge si el pot llegir.*
5. L'usuari rep el missatge i el llegeix.	

4.4.2 Especificació de la part del prototipus:



Il·lustració 4.3 - Especificació prototipus

L'usuari registrat és l'usuari en la prova pilot que resideix a la base de dades, i que obtindrà, pel fet de ser registrat, els permisos necessaris per ser 'creador imatges', 'creador notícies' i 'creador patrons'.

La funcionalitat "*Alta Usuari*" només es donarà via web.

Tant les plantilles com les categories de les imatges estan solament de suport i per fer el prototipus més complet i pròxim al producte final. Per això les modificacions o altes d'aquests es faran directament a la base de dades.

Les notícies estan formades per elements de text (Text), i per elements d'imatge (IMG) que fan referència a imatges. Aquests elements tenen una posició lògica assignada (posició).

R.I. :

```
Claus:
        Usuari Registrat        { nom }
        Notícia                 { idNotícia }
        Text                    { idNotícia, posició }
        IMG                     { idNotícia, posició }
        Imatge                  { idImatge }
        Plantilla               { idPlantilla}
        Categoria               { idCategoria }
        Missatges               { idMissatge } (no pertany al prototipus)
```

4.4.3 Operacions a implementar

Alta Usuari (nom, contrasenya) : Booleà
Donades les dades de l'usuari donem d'alta un usuari.
getUsuari (nom, contrasenya) : Usuari
Si l'usuari existeix ens retorna les dades de l'usuari
insImatge (Usuari, nom, desc, categoria, dades) : Booleà
L'usuari, si té els permisos necessaris, dóna d'alta la imatge.
getCatIma () : Vector
Retorna la llista de categories d'imatges.
getImaCat (idCategoria) : Vector
Donada una categoria, retorna la llista d'imatges de la categoria.
getPlantilla() : Vector
Dóna la llista de plantilles disponibles.
insNoticia (noticia, Usuari, idNoticia): boolean
Insereix la notícia al sistema, (o l'actualitza si idNoticia és vàlid)
getNotDesc (String autor) : Vector
Retorna la llista de notícies de l'autor especificat (o tots si no s'especifica cap)
recNoticia (idNoticia) : Noticia
Retorna la noticia amb l'idNoticia especificat.
També crearem la funció **tstImatge** amb similars característiques que insImatge, però sense transmetre la imatge (per comprovar que tot ha anat bé)

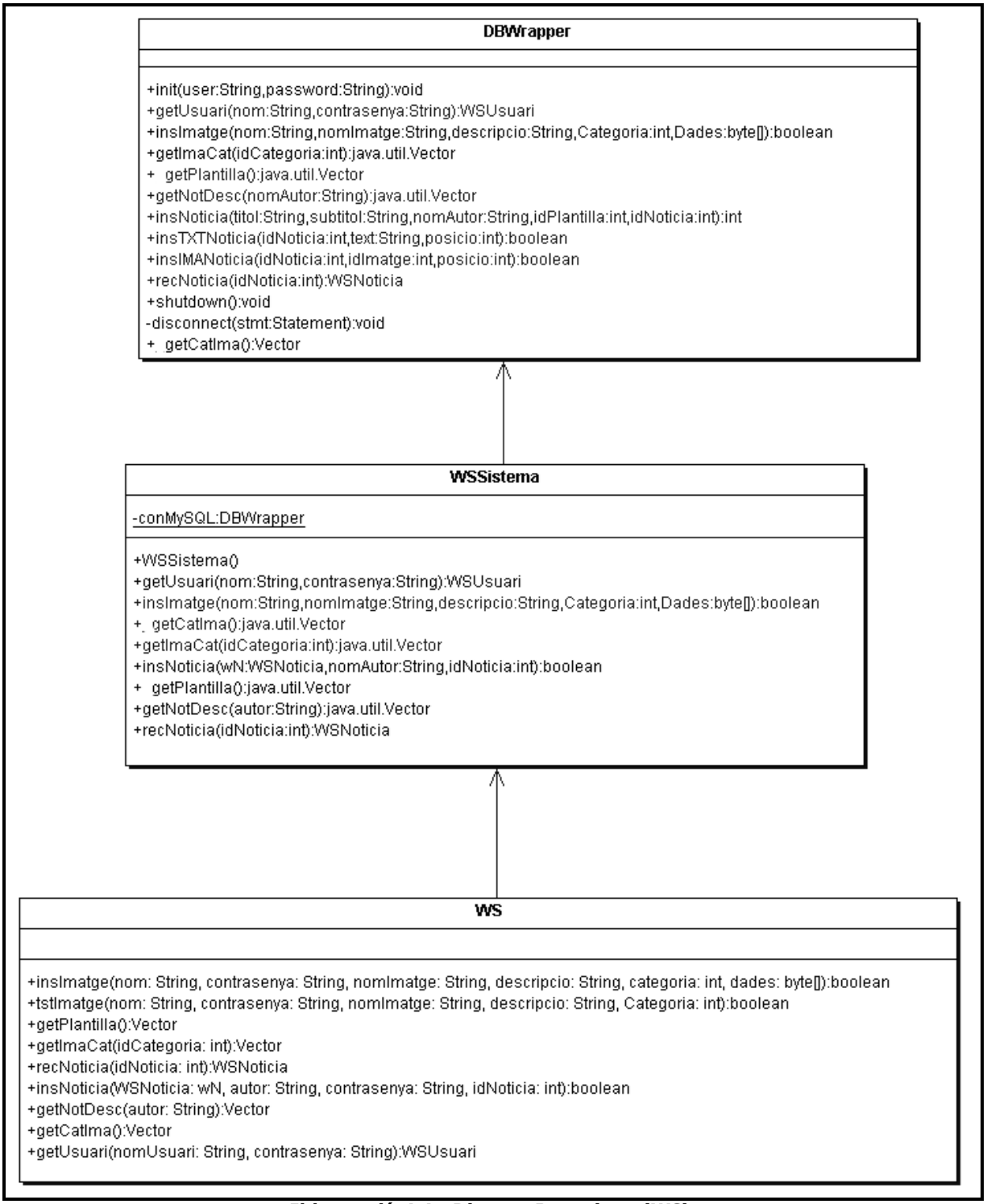
Fora del prototipus:
getUsers () : Vector
Retorna el nom dels usuaris (String).
insMissatge (Missatge, contrasenya) : Boolean
Insereix el missatge al sistema.
getMissLl (usuari, contrasenya) : Vector
Retorna la llista de missatges per a llegir de l'usuari especificat.
recMissatge (usuari, contrasenya, idMissatge) : Missatge
Retorna el missatge especificat als paràmetres.
esbMissatge (usuari, contrasenya, idMissatge) : Boolean
Esborra el missatge especificat als paràmetres.
esbNotícia (usuari, contrasenya, idNoticia) : Boolean
Esborra la notícia especificada als paràmetres.

4.4.4 Disseny prototipus

S'han tingut en compte els següents punts (semblants al de la prova pilot realitzada anteriorment) :

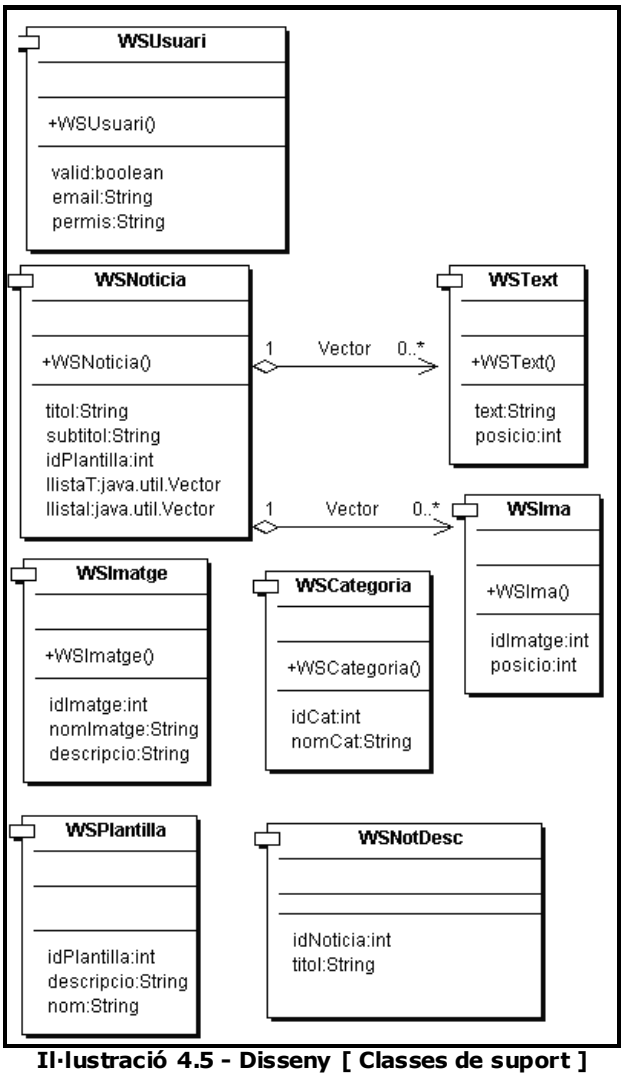
- Persistència de les classes **Usuari**, **Notícia**, **IMG**, **Text**, **Categories**, **Plantilla** i **Imatge** en una base de dades relacional; això afegeix una capa més al sistema que la tractarà la classe **DBWrapper** (per tal d'amagar la base de dades). La definició de la BD es pot trobar després.
- Afegim una nova classe (*representant*) **WS** que rebrà les peticions del client i les transmetrà al **Sistema** (i les respostes del Sistema al client). Les seves classes són les visibles pel client (capa Externa).
- El client WEB s'implementarà amb PHP, directament sobre la BB.DD. (ja que no disposem de procediments emmagatzemats).
- Les classes que s'utilitzaran de magatzem temporal pel transport de dades entre el client i el WS, són classes reduïdes (*representant*), les prefixarem amb **WS**.

El diagrama de classes de disseny és per tant el següent:



Il·lustració 4.4 - Disseny Prototipus (WS)

Hem d'afegir a aquestes classes els representants dels elements persistents i les classes (*Java Beans*) que serviran per transportar la informació necessària entre el client i el servidor, els mostrem a continuació:

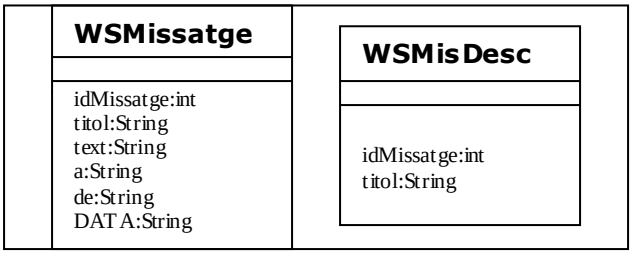


Il·lustració 4.5 - Disseny [Classes de suport]

4.4.5 Disseny parts que no pertanyen al prototipus

Hem afegit les operacions a les classes **WSDBWrapper**, **WSImatge** i **WSSistema**. A WSDBWrapper se li passa la classe Missatge desglossada per a la seva inserció a la base de dades:

insMissatge (títol, text, de, a, idMissatge):enter que indica el número de missatge inserit.
Finalment creem les classes de suport (**WSMissatge** i **WSMisDesc**):

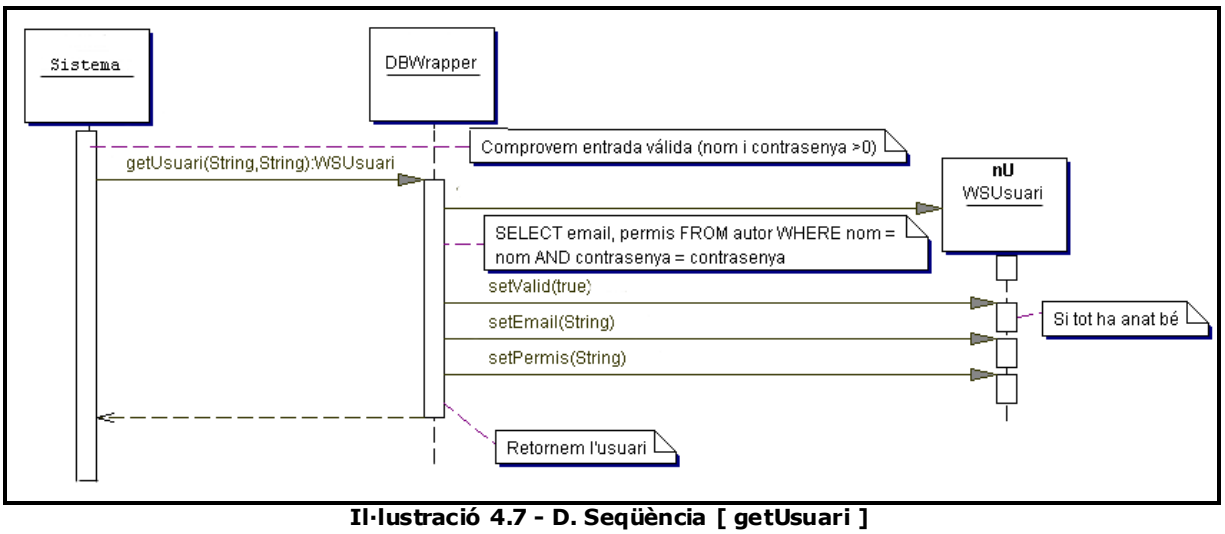


Il·lustració 4.6 - Classes de suport - Part Missatges

4.4.6 Diagrames de seqüència

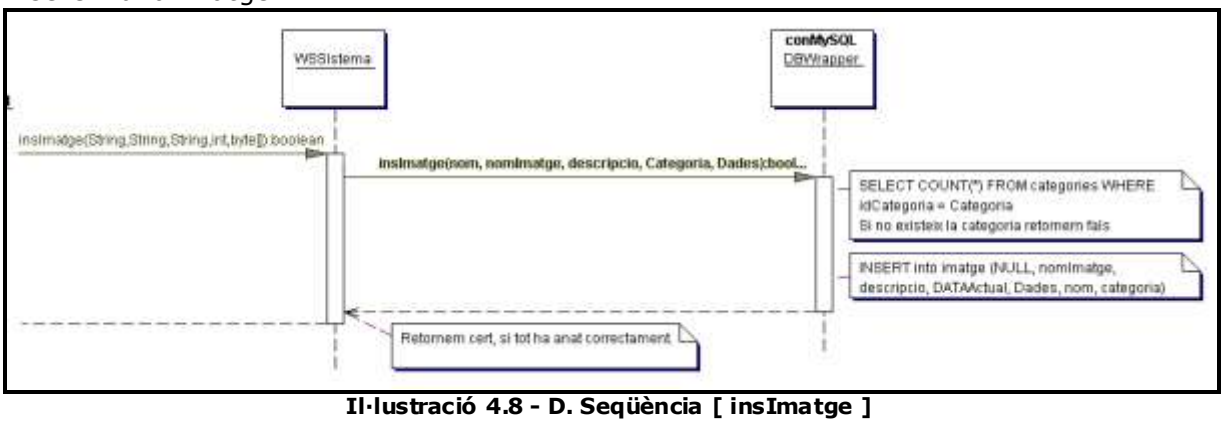
Els diagrames de seqüència de les operacions anteriorment esmentades són els següents.

Recuperar la informació d'un usuari.



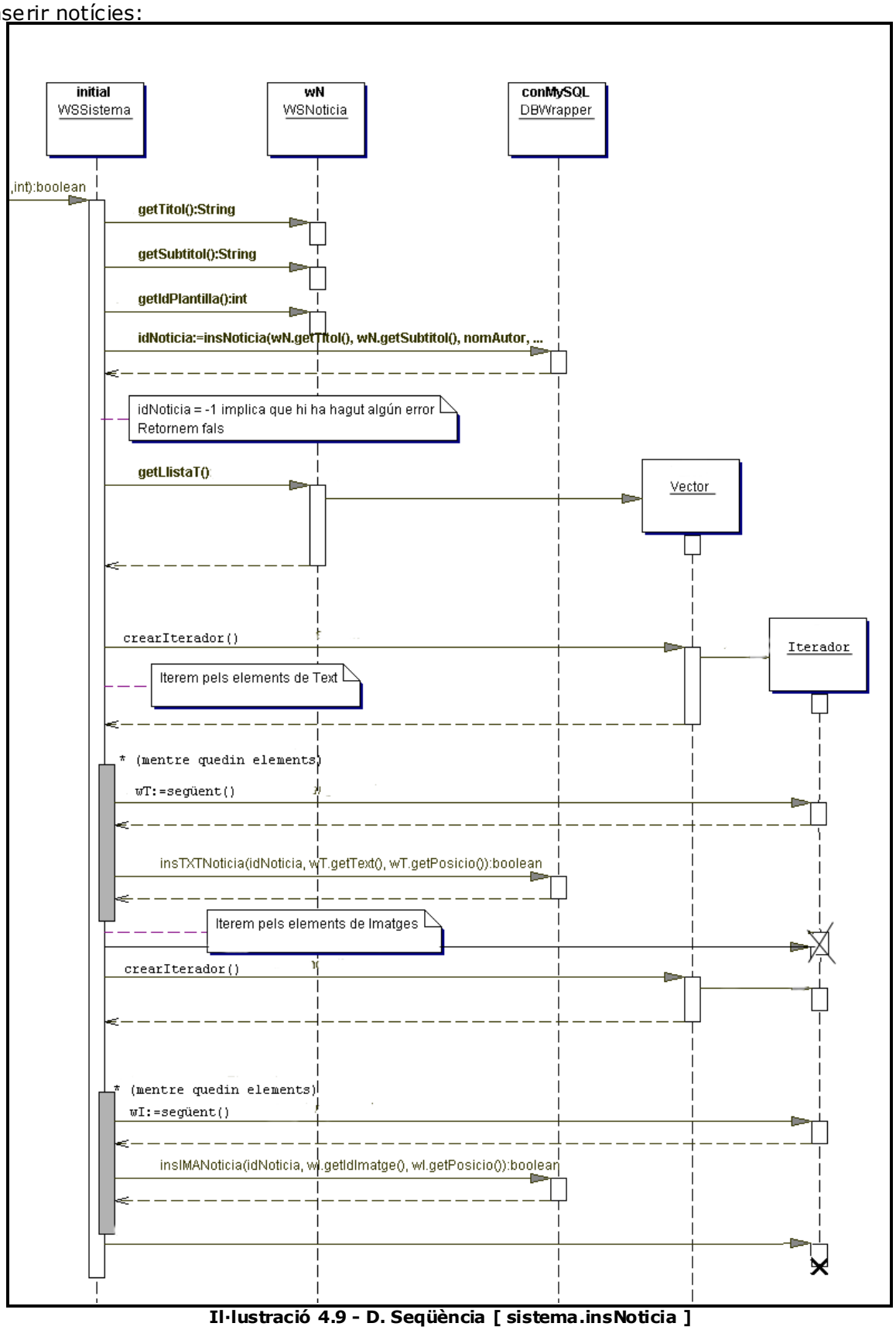
Il·lustració 4.7 - D. Seqüència [getUsuari]

Insereix una imatge.

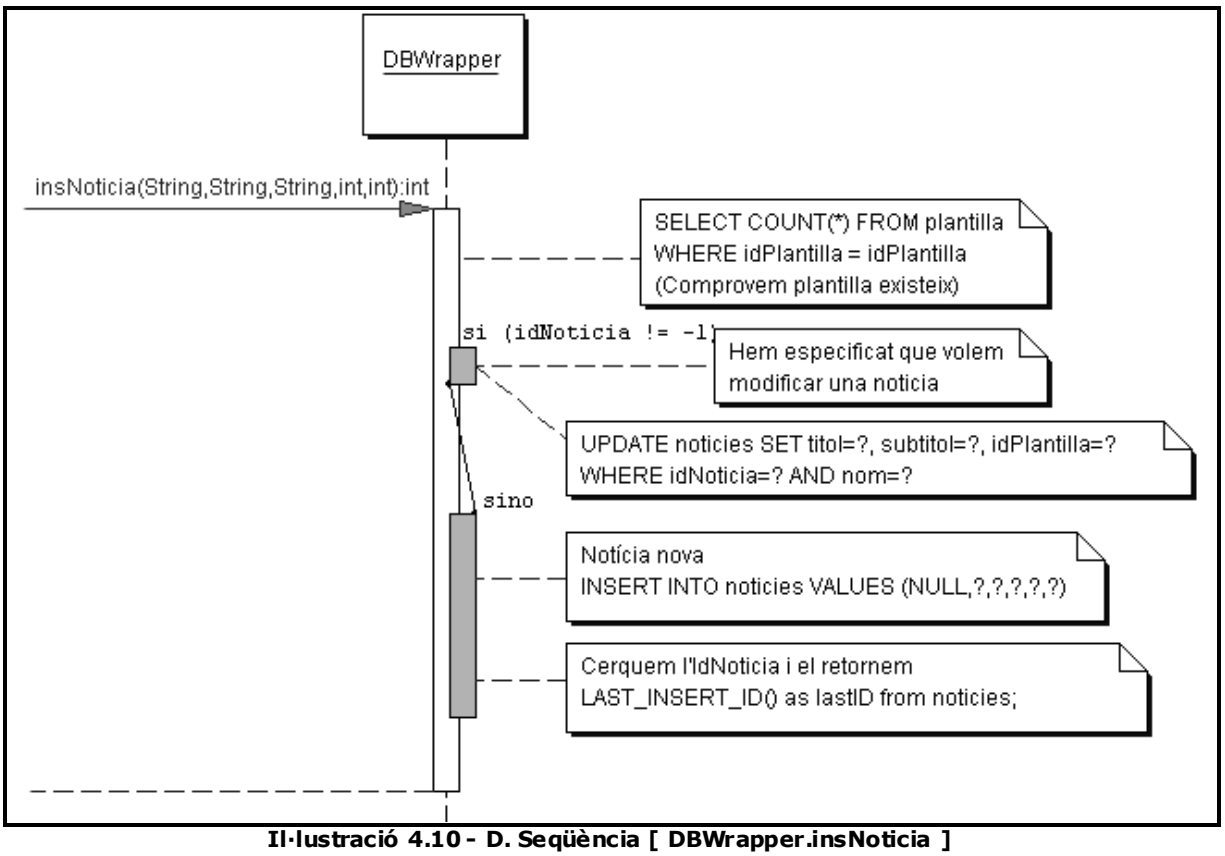


Il·lustració 4.8 - D. Seqüència [insImatge]

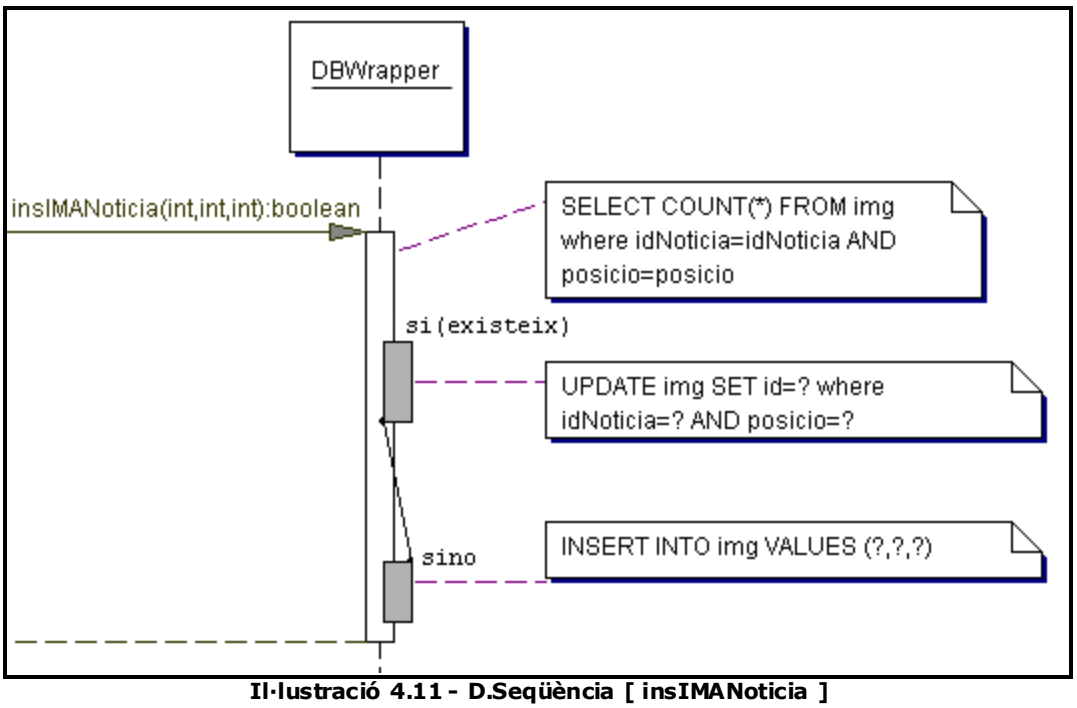
Inserir notícies:



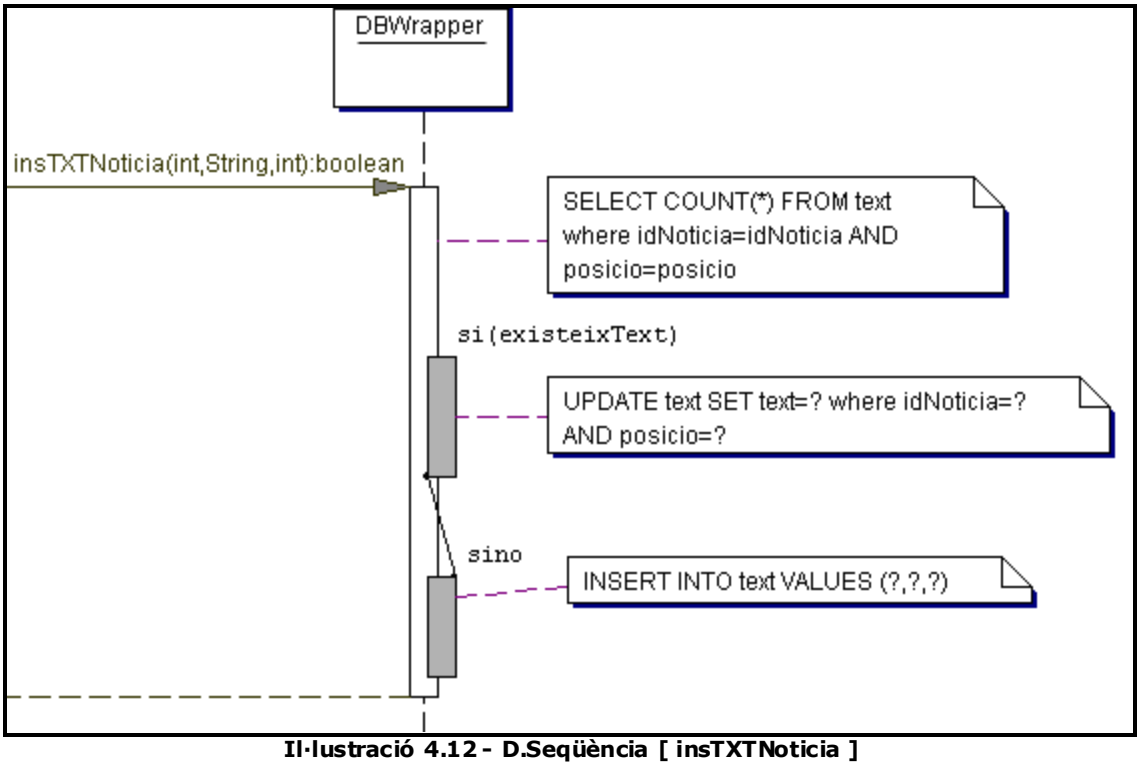
Il·lustració 4.9 - D. Seqüència [sistema.insNoticia]



Il·lustració 4.10 - D. Seqüència [DBWrapper.insNoticia]

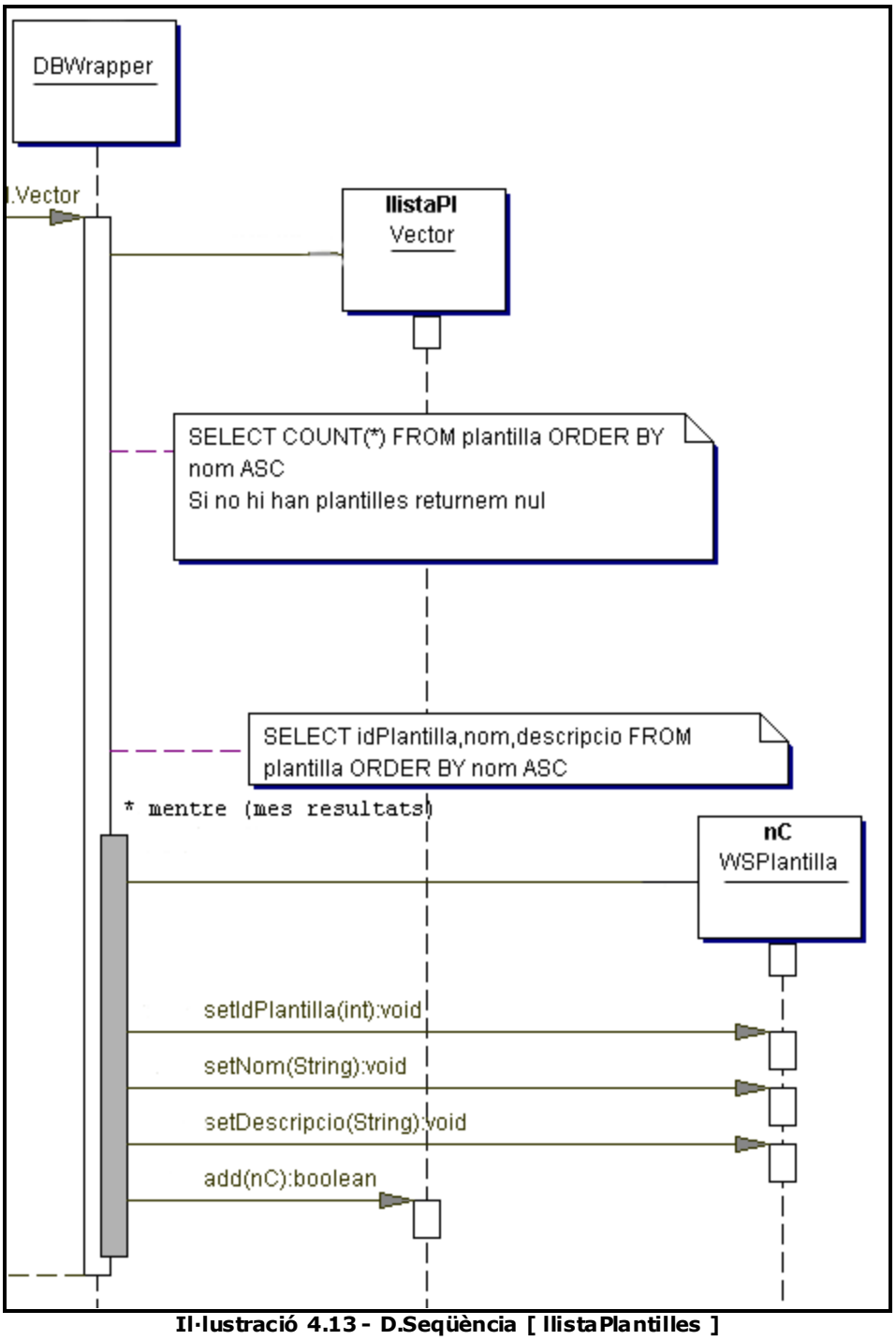


Il·lustració 4.11 - D.Seqüència [insIMANoticia]



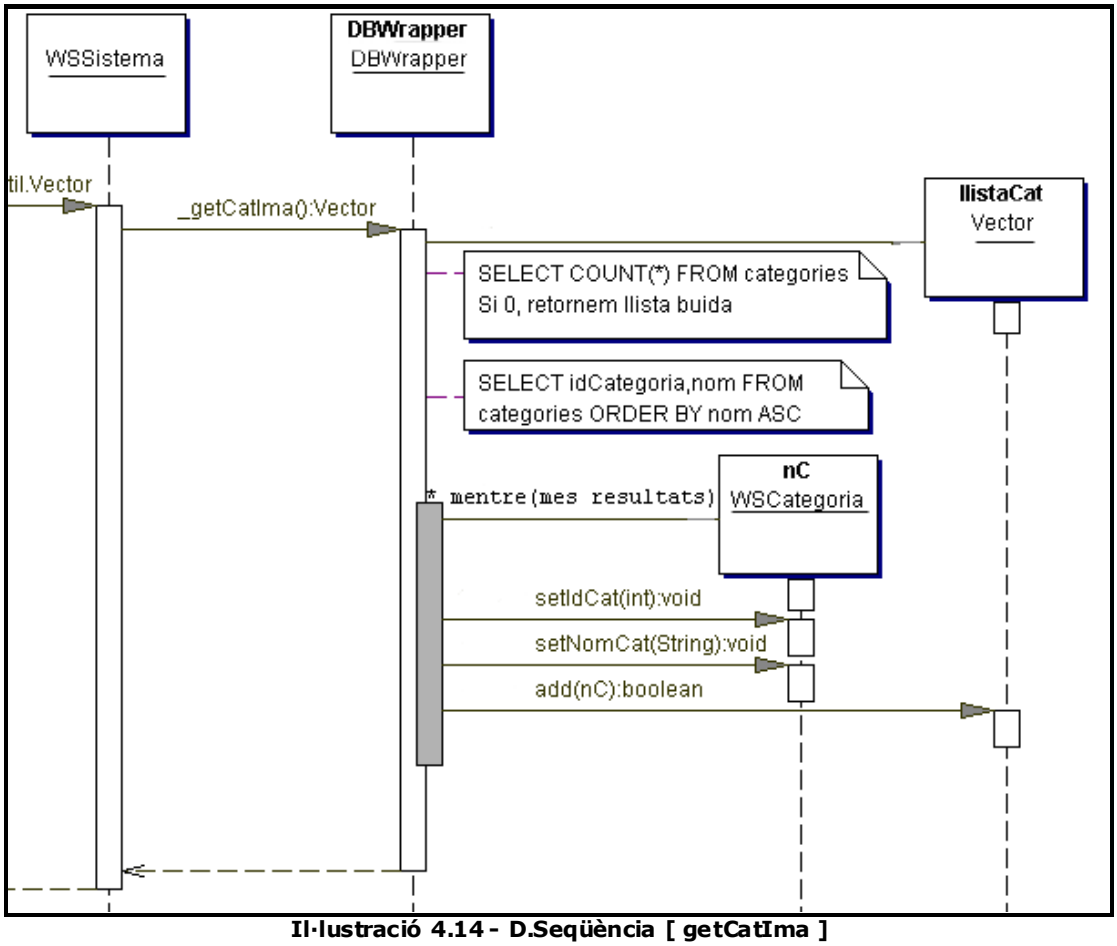
Il·lustració 4.12 - D.Seqüència [insTXTNoticia]

Recuperació de llistes/elements:
Llista Plantilles :



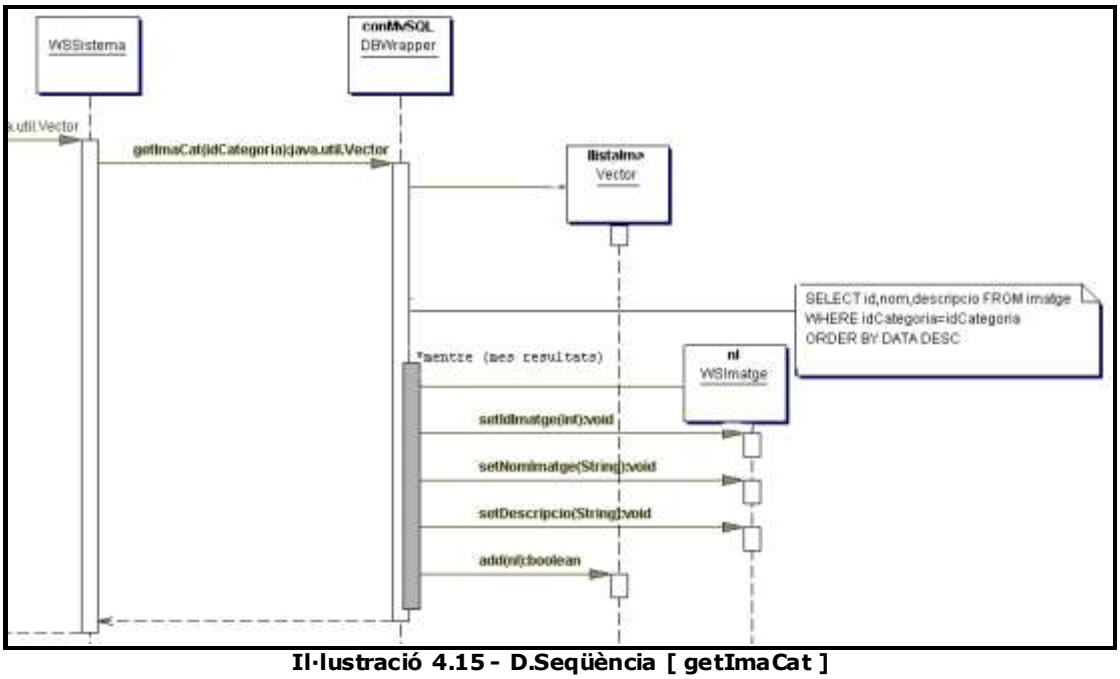
Il·lustració 4.13 - D.Seqüència [llistaPlantilles]

Llista categories:



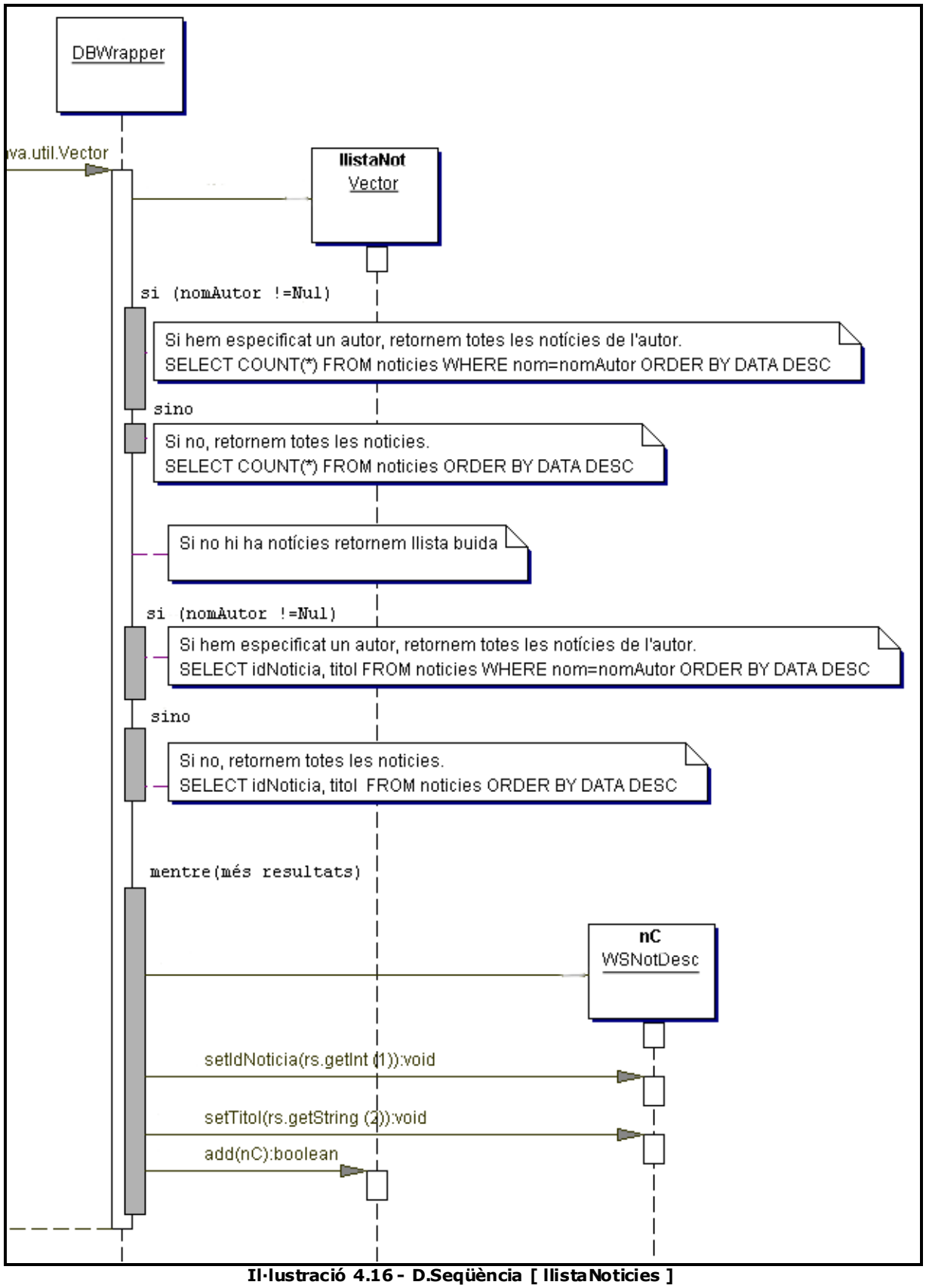
Il·lustració 4.14 - D.Seqüència [getCatIma]

Llista Imatges donada una categoria:



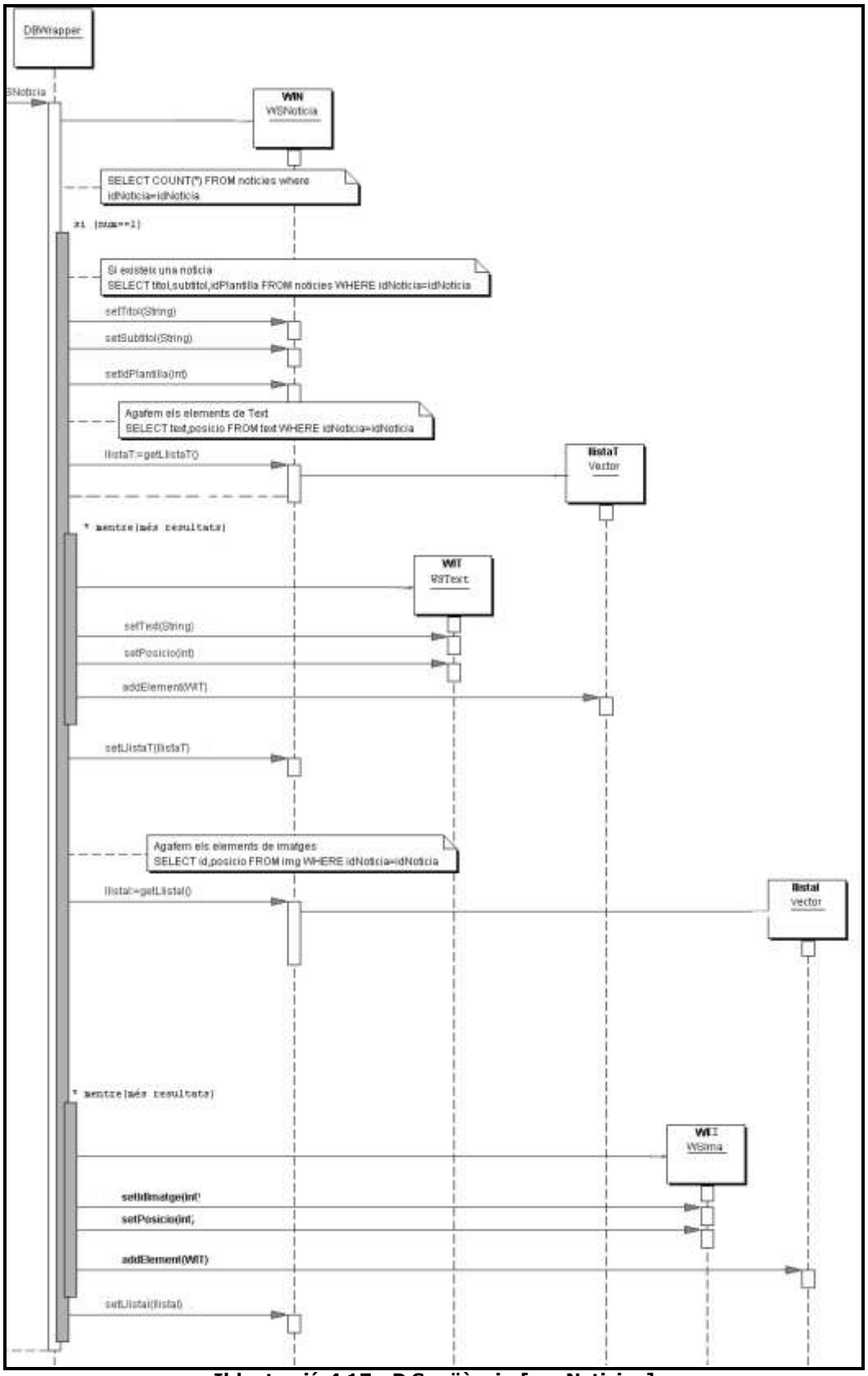
Il·lustració 4.15 - D.Seqüència [getImaCat]

Llista noticies donat un autor:

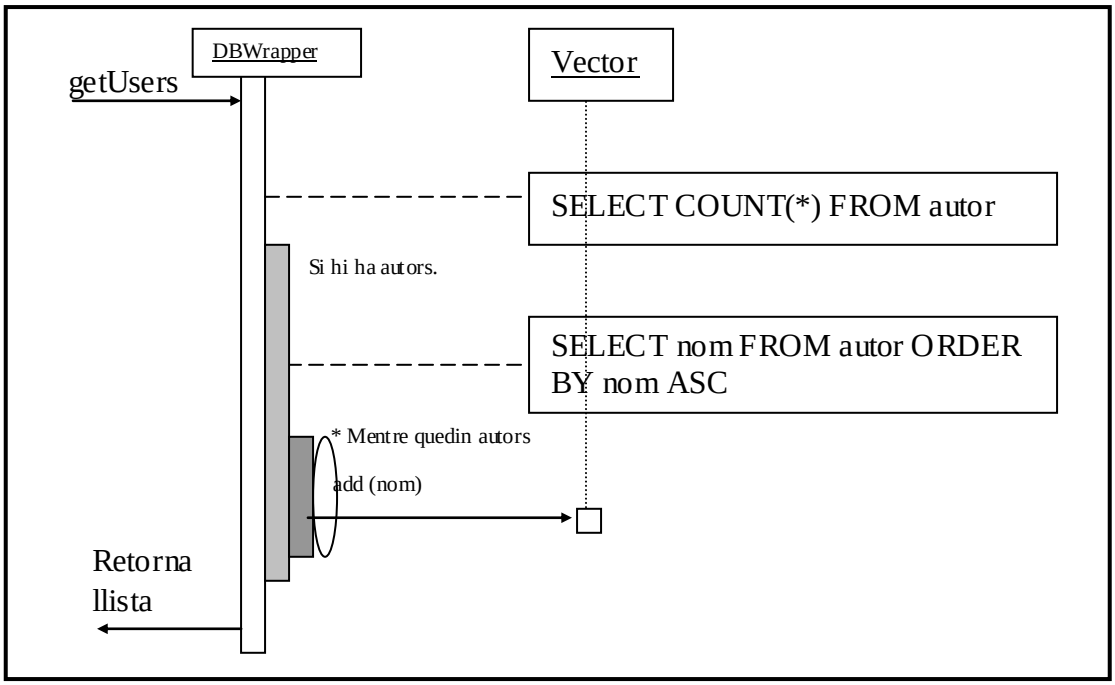


Il·lustració 4.16 - D.Seqüència [llistaNoticies]

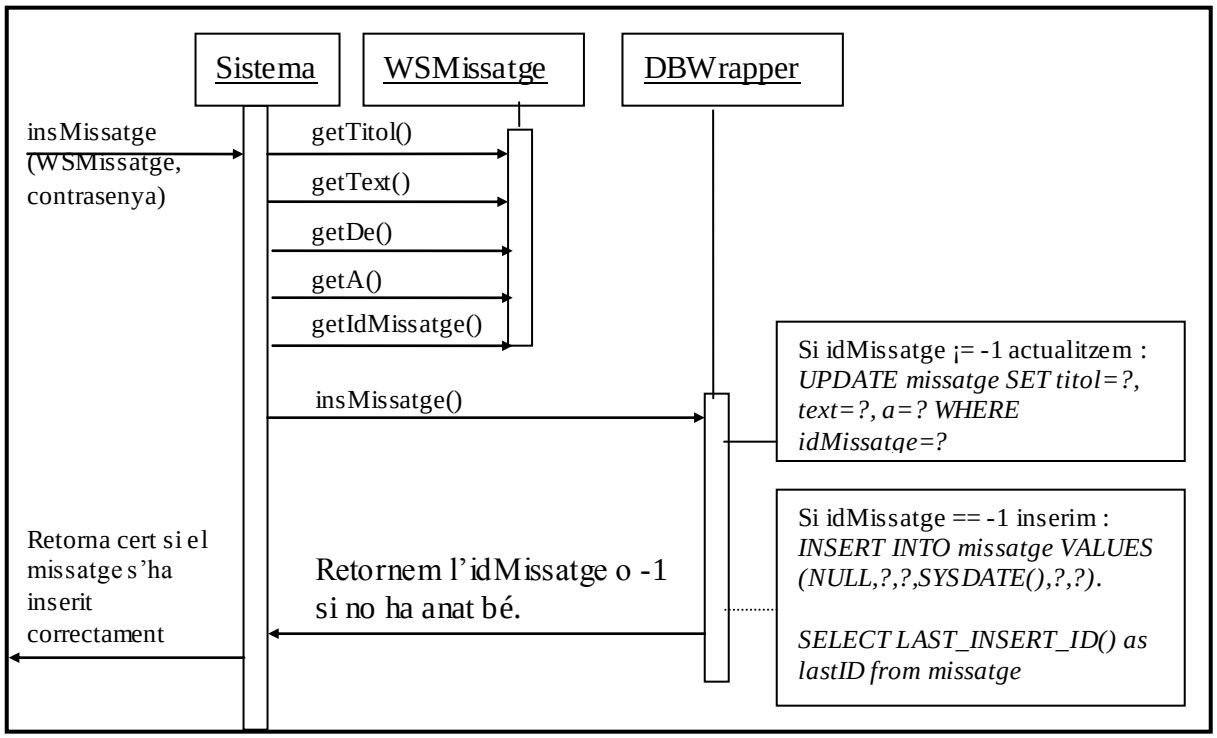
Recuperar una notícia donada la seva id:



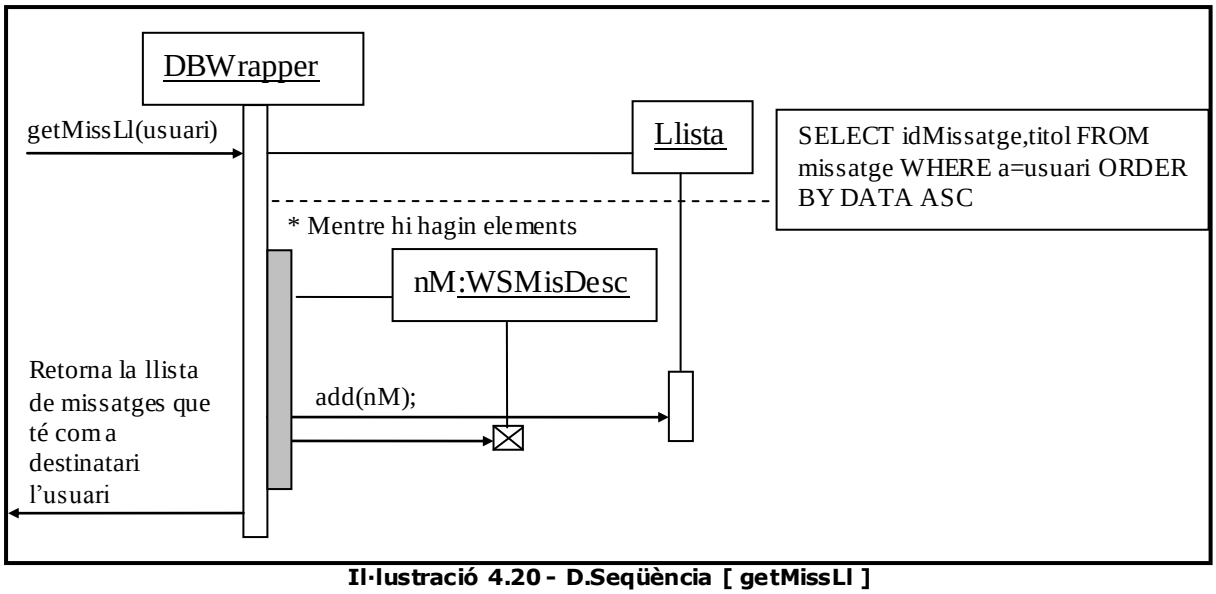
Il·lustració 4.17 - D.Seqüència [recNoticia]



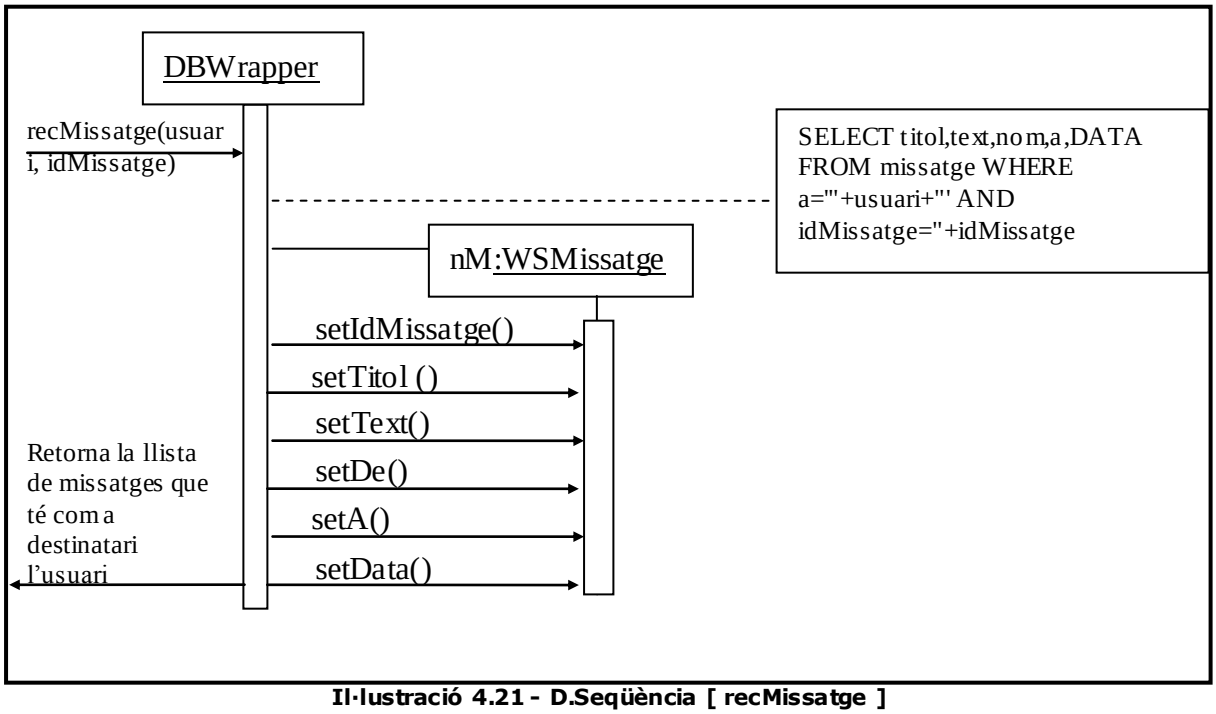
Il·lustració 4.18 - D.Seqüència - [getUsers]



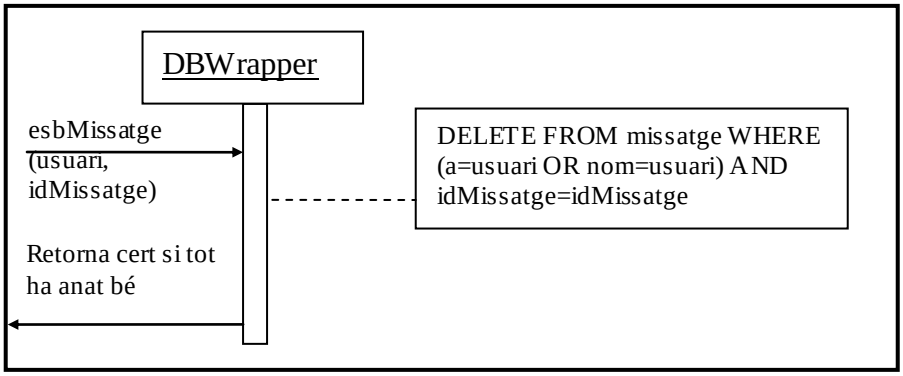
Il·lustració 4.19 - D.Seqüència - [insMissatge]



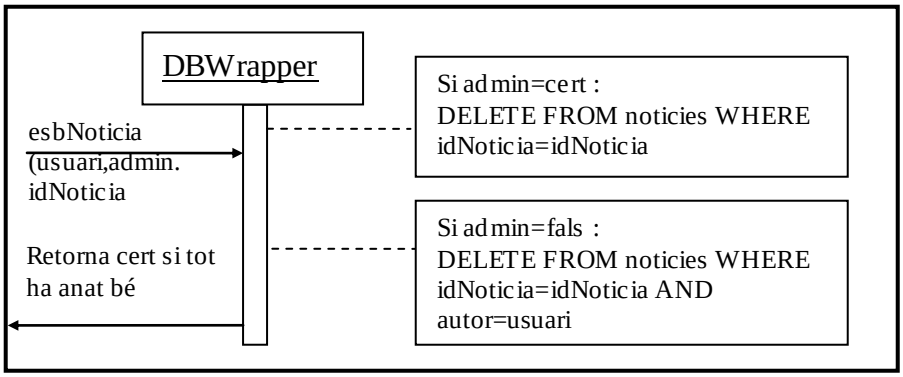
Il·lustració 4.20 - D.Seqüència [getMissLl]



Il·lustració 4.21 - D.Seqüència [recMissatge]



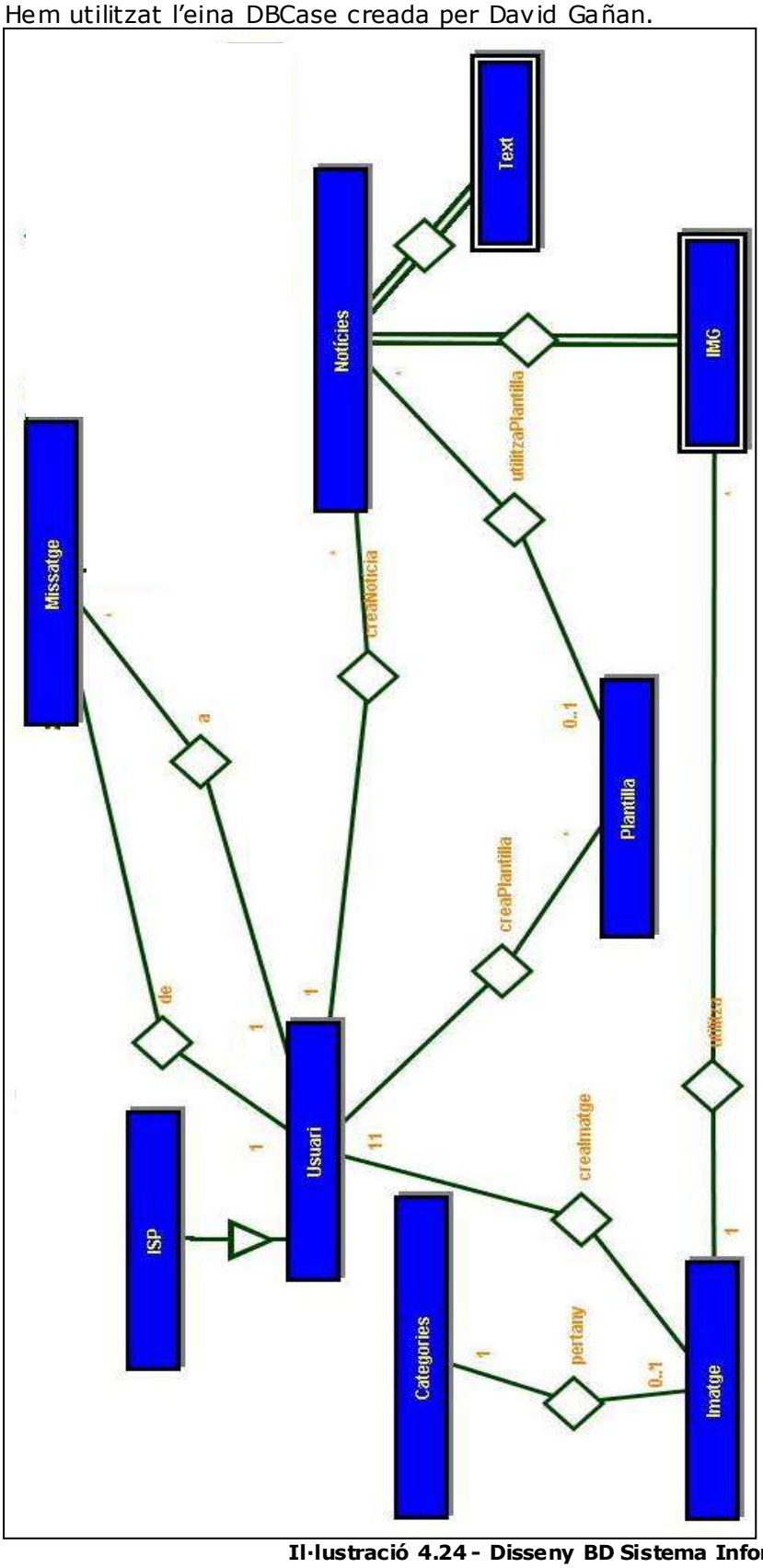
Il·lustració 4.22 - D.Seqüència [esbMissatge]



Il·lustració 4.23 - D.Seqüència [esbNoticia]

4.4.7 Disseny de la base de dades

Hem utilitzat l'eina DBCase creada per David Gañan.



Il·lustració 4.24 - Disseny BD Sistema Informació

Les taules que surten del disseny són les següents:

Autor (nom, contrasenya, email, permis)
Isp (*nom*, ftp, ftpContrasenya, automatic)
Plantilla (idPlantilla, *nom*, descripcio, quadre, nomAutor)
Noticies (idNoticia, títol, subtítol, nom, *idPlantilla*)
Text (i*dNoticia*, posició, text)
Categories (idCategoria, nom)
Imatge (id, nom, descripcio, DATA, dades, *nomAutor*, *idCategoria*)
Img (i*dNoticia*, posicio, *id*)
Missatge (idMissatge, text, títol, DATA, a, *nom*)

A continuació mostrarem les sentències SQL (CREATE TABLE) necessàries per a la inserció de les taules a la base de dades mySQL.

Utilitzarem, però, algunes modificacions en les claus foranes; són les següents:

- **Notícies**
  - Clau forana plantilla: com que no hi ha un ON DELETE SET DEFAULT, en tota aquella part que s'esborri una plantilla, s'ha de fer un update per tal de tenir la notícia amb la plantilla per defecte.
  - Clau forana d'autors eliminada: si s'elimina l'autor les seves notícies es mantindran amb el seu nom, per tant s'ha de violar la clau forana i per això s'elimina la restricció.
- **Imatges**
  - El mateix cas que amb les noticies (categories, i autors). En el cas de les categories convertim la categoria a la categoria per defecte.
- **Plantilles**
  - La plantilla no s'elimina tot i eliminar-ne l'autor, per tant s'elimina la restricció.
- **img i text**, si desapareix la notícia s'elimina l'element.
- **Img,** si desapareix la imatge s'elimina l'element.
- **Missatges**, si s'elimina alguns dels elements a que estan lligats s'esborren.

```
CREATE TABLE autor (
        nom varchar(30) NOT NULL default '',
        contrasenya varchar(8) NOT NULL default '',
        email varchar(255) default NULL,
        permis varchar(10) NOT NULL default '',
        PRIMARY KEY  (nom)) TYPE=INNODB

CREATE TABLE isp (
        nom varchar(30) NOT NULL default '',
        ftp varchar(20) NOT NULL default '',
        ftpContrasenya varchar(16) NOT NULL default '',
        automatic int(11) NOT NULL default '0',
        PRIMARY KEY  (nom),
        FOREIGN KEY  (nom) REFERENCES autor (nom) ON DELETE CASCADE) TYPE=INNODB

CREATE TABLE plantilla (
        idPlantilla int(11) NOT NULL auto_increment,
        nom varchar(30) NOT NULL default '',
        descripcio varchar(255) default NULL,
```

```
        quadre varchar(36) NOT NULL default '',
        nomAutor varchar(30) NOT NULL default '',
        PRIMARY KEY  (idPlantilla)
        ) TYPE=INNODB

CREATE TABLE text (
        idNoticia int(11) NOT NULL default '0',
        Text longtext,
        posicio int(11) NOT NULL default '0',
        PRIMARY KEY  (idNoticia,posicio),
        FOREIGN KEY  (idNoticia) REFERENCES noticies (idNoticia) ON DELETE CASCADE) TYPE=INNODB

CREATE TABLE categories (
        idCategoria int(11) NOT NULL auto_increment,
        nom varchar(30) NOT NULL default '',
        PRIMARY KEY  (idCategoria),
        UNIQUE KEY nom (nom)) TYPE=INNODB

CREATE TABLE imatge (
        id int(11) NOT NULL auto_increment,
        nom varchar(30) default NULL,
        descripcio varchar(255) default NULL,
        DATA date NOT NULL default '0000-00-00',
        dades longblob NOT NULL,
        nomAutor varchar(30) NOT NULL default '',
        idCategoria int(11) NOT NULL default '1',
        PRIMARY KEY  (id),
        INDEX id_ind (idCategoria),
        FOREIGN KEY (idCategoria) REFERENCES categories (idCategoria)) TYPE=INNODB

CREATE TABLE img (
        idNoticia int(11) NOT NULL default '0',
        posicio int(11) NOT NULL default '0',
        id int(11) NOT NULL default '0',
        PRIMARY KEY  (idNoticia,posicio),
        INDEX id_ind (id),
        FOREIGN KEY (id) REFERENCES imatge (id) ON DELETE CASCADE,
        FOREIGN KEY (idNoticia) REFERENCES noticies (idNoticia) ON DELETE CASCADE) TYPE=INNODB

CREATE TABLE missatge (
        idMissatge int(11) NOT NULL auto_increment,
        text longtext,
        títol varchar(30) default NULL,
        DATA date NOT NULL default '0000-00-00',
        a varchar(30) NOT NULL default '',
        nom varchar(30) NOT NULL default '',
```

```
        PRIMARY KEY  (idMissatge),
        INDEX nom_ind (nom),
        INDEX nom2_ind (a),
        FOREIGN KEY (nom) REFERENCES autor (nom) ON DELETE CASCADE,
        FOREIGN KEY (a) REFERENCES autor (nom) ON DELETE CASCADE,
        ) TYPE=INNODB
```

4.5 Programació dels WS al PC.

En aquesta ocasió, i donada que la complexitat dels Web Services a implementar és major, utilitzarem l'eina Jakarta Ant[1] per ajudar-nos en la compil·lació. Ara generarem un .jar i realitzarem un *deploy* a Axis per tal de poder definir els paràmetres del comportament dels nostres WS.

Crearem el codi dels WS en un **package** *es.butlleti*

4.5.1 Jakarta ANT

L'script **XML** d'ANT per a compil·lar i crear el jar l'agafarem i modificarem per adaptar-lo a les nostres necessitats. Col·loquem a continuació l'script, que es col·locarà en l'arrel d'on penjarà el directori es/butlleti/. /es/butlleti/ és on hi haurà el codi a compil·lar. Mostrem i subratllem els noms que s'han de canviar per a compil·lar un altre projecte en l'script.

```
<?xml version="1.0"?>
<project basedir="." default="dist" name="WSImatge">
    <property name="base" value="."/>
    <property name="additional.path" value="."/>
    <property name="src" value="."/>
    <property name="build" value="build"/>
    <property name="dist" value="dist"/>
    <property environment="env"/>
    <path id="WSImatge.class.path">
        <pathelement path="${env.classpath}"/>
        <pathelement path="${env.additional.path}"/>
    </path>

    <property name="cp" refid="WSImatge.class.path"/>

    <target name="prepare">
        <echo>
            [ Setting time stamp and creating the directories ]
        </echo>
```

[1] http://ant.apache.org/ , "Apache Ant is a Java-based build tool. In theory, it is kind of like *make*, without *make*'s wrinkles."

```
        <tstamp/>
        <mkdir dir="${build}"/>
        <mkdir dir="${dist}"/>
    </target>

    <target depends="prepare" name="jar">
        <echo>
            [ Creating the JAR file now ]
        </echo>
        <jar jarfile="${dist}/WSImatge.${DSTAMP}-${TSTAMP}.jar">
            <fileset dir="${build}"/>
        </jar>
        <copy file="${dist}/WSImatge.${DSTAMP}-${TSTAMP}.jar" tofile="H:/Archivos de
programa/Apache Group/Tomcat 4.1/webapps/axis/WEB-INF/lib/WSImatge.jar"/>
    </target>

    <target depends="prepare" name="compile">
        <echo message="Classpath = ${cp}"/>
        <javac debug="on" destdir="${build}" srcdir="${src}" verbose="off">
            <classpath refid="WSImatge.class.path"/>
        </javac>
    </target>

    <target depends="prepare" name="compile-optimized">
        <javac debug="off" destdir="${build}" optimize="on" srcdir="${src}" verbose="on">
            <classpath refid="WSImatge.class.path"/>
        </javac>
    </target>

    <target depends="prepare" name="compile-Wall">
        <javac debug="on" deprecation="on" destdir="${build}" srcdir="${src}" verbose="on">
            <classpath refid="WSImatge.class.path"/>
        </javac>
    </target>

    <target depends="compile" name="dist">
        <echo>
            [ Copy all of the properties files into the Build directory structure ]
        </echo>
        <antcall target="jar"/>
    </target>

    <target name="clean">
        <echo>
            [ Removing the Build tree ]
        </echo>
        <delete dir="${build}"/>
```

```
    </target>
</project>
```

Si resseguim una mica l'script veurem que és una mena de **make**. Al compil·lar col·loquem el **jar** creat al directori de l'Axis corresponent, preparant-lo pel *deployment* (sol necessitar-se fer un *restart* d'Axis o de tomcat per a carregar la nova classe*)*.

4.5.2 Classes de suport (Java Beans)

El que hem de construir primer són les classes de suport. Aquestes segueixen el paradigma dels **java Beans** (la raó d'aquesta decisió la veurem al fer el *deployment*). Una de les característiques per ser **Java Bean** és que els seus atributs tinguin mètodes d'accés i modificació anomenats **Get** / **Set** + **el nom del atribut**.

Definim com a exemple les classes encarregades de transportar la notícia; la primera és l'encarregada de guardar els elements d'imatges (idImatge i posicio):

```
package es.butlleti;
public class WSInfoIma {
        int idImatge = 0;
        int posicio = 0;

        public WSInfoIma () { }
        public void setIdImatge (int iIdImatge) { idImatge = iIdImatge; }
        public int getIdImatge ( ) { return (idImatge); }
        public void setPosicio (int iPosicio) { posicio = iPosicio; }
        public int getPosicio ( ) { return (posicio); }
}
```

La següent s'encarrega de transportar els elements de text:

```
package es.butlleti;
public class WSInfoText {
        String text = null;
        int posicio = 0;

        public WSInfoText () { }
        public void setText ( String sText) { text = sText; }
        public String getText ( ) { return (text); }
        public void setPosicio (int iPosicio) { posicio = iPosicio; }
        public int getPosicio ( ) { return (posicio); }
}
```

Finalment, tenim la classe **WSInfoNoticia,** que utilitza les dues anteriors per a formar una noticia. Els elements de text i d'imatge es col·loquen en un vector per a transportar-les mitjançant WS. L'elecció d'un vector es realitza per conveniència, ja que és un contenidor molt senzill d'utilitzar en Axis i en la part de kSOAP.

```
package es.butlleti;
public class WSInfoNoticia {

        String títol = null;
        String subtítol = null ;
        int idPlantilla = 1;
        /* Vector d'WSInfoText */
        java.util.Vector llistaT = new java.util.Vector ();
        /* Vector d'WSInfoIma */
        java.util.Vector llistaI = new java.util.Vector ();



        public WSInfoNoticia () { }

        public void setTitol (String sTitol) { títol = sTitol; }
        public String getTitol ( ) { return (títol); }

        public void setSubtitol (String sSubtitol) { subtítol = sSubtitol; }
        public String getSubtitol ( ) { return (subtítol); }

        public void setIdPlantilla (int iIdPlantilla) { idPlantilla = iIdPlantilla; }
        public int getIdPlantilla ( ) {return (idPlantilla); }

        public void setLlistaT (java.util.Vector vLlistatT) { llistaT = vLlistatT; }
        public java.util.Vector getLlistaT ( ) { return (llistaT); }

        public void setLlistaI (java.util.Vector vLlistatI) { llistaI = vLlistatI; }
        public java.util.Vector getLlistaI ( ) { return (llistaI); }

}
```

Aquestes classes s'hauran de replicar en el client per tal de poder tenir accés a les seves dades.

4.5.3 Classes d'accés a la base de dades

Construïm després de les classes de suport la classe encarregada d'accedir a la base de dades **WSDBWrapper** que utilitzarà el paquet java.sql.* i el **mysql.jar**[1] per tal d'accedir a mySQL. Aquesta classe tindrà la implementació dels diagrames de seqüència que hem creat anteriorment. Es pot veure el codi d'aquesta part al CD-ROM que acompanya aquest document.

Posteriorment a la classe anterior crearem la classe WSSistema, que s'encarregarà de definir la lògica de negoci de l'aplicació (cridar als mètodes de la base de dades correctes per a construir la notícia, per exemple). Aquesta classe no fa res més que cridar als

[1] És el mysql-connector J, en la seva última versió. (www.mysql.com)

procediments de WSDBWrapper amb els paràmetres corresponents; destaquem el mètode **insNoticia**, que s'encarrega d'inserir la noticia:

```
public boolean insNoticia (WSInfoNoticia wN,String nomAutor,int idNoticia)  {
       idNoticia = conMySQL.insNoticia (wN.getTitol(),wN.getSubtitol(),nomAutor,
wN.getIdPlantilla(),idNoticia);

       if (idNoticia == -1) return false;

       for (java.util.Enumeration e = wN.getLlistaT().elements(); e.hasMoreElements () ;){
              WSInfoText wT = (WSInfoText) e.nextElement();
              conMySQL.insTXTNoticia (idNoticia , wT.getText(), wT.getPosicio() );
       }
       for (java.util.Enumeration e = wN.getLlistaI().elements(); e.hasMoreElements () ;){
              WSInfoIma wI = (WSInfoIma) e.nextElement();
              conMySQL.insIMANoticia (idNoticia , wI.getIdImatge(), wI.getPosicio() );
       }
       return (true);
}
```

4.5.4 Deployment.

Una vegada tenim el codi compilat podem realitzar el deployment corresponent. En aquest apartat comentarem per què hem de fer un deployment específic i per què hem construït les classes de suport i de transport.

Al realitzar la comanda de compil·lació ant, i si tot ha anat correctament, observarem que tenim al directori axis\WEB-INF\lib el jar corresponent (**WSImatge.jar**); l'únic que ens resta és fer que Apache Axis conegui que hi ha una nova aplicació.
Per fer-ho necessitem un script de deployment, anomenat deploy.**wsdd:**

```
<!--   Per a instal·lar el WS fer:                                  -->
<!--   java org.apache.axis.client.AdminClient deploy.wsdd    -->
<!--   mentre Axis està funcionant                                  -->
<deployment name="WSImatge"
    xmlns="http://xml.apache.org/axis/wsdd/"
    xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
  <service name="WSImatge" provider="java:RPC" >
     <parameter name="className" value="es.butlleti.WSImatge"/>
     <parameter name="allowedMethods" value="*"/>
     <parameter name="sendMultiRefs" value="false"/>
  </service>
</deployment>
```

Aquest fitxer de deployment, tot i que funciona, no ens donarà el resultat esperat; ho podem comprovar amb una URL, que ens cridarà al WS i ens retornarà el resultat

suposat (hem d'introduir algunes plantilles a la Base de Dades i un usuari amb permisos, prèviament).

http://localhost:8080/axis/services/WSImatge?wsdl
Si aconseguim veure el WSDL tindrem el deployment fet correctament (podem necessitar tornar a arrancar Axis).

http://localhost:8080/axis/services/WSImatge?method=getPlantilla

En aquest moment hem de poder veure un error d'execució del WS; expliquem el que ha passat:

El mètode getPlantilla retorna un vector que conté ítems de la classe ***WSInfoPlantilla***. Quan **Axis** intenta crear l'XML necessari per a transportar el vector a través de la xarxa genera el primer nivell, que representa a la classe vector (que coneix com serialitzar-la correctament); fet això AXIS ha de serialitzar els ítems, que tot i que contenen tipus senzills, són de la classe WSInfoPlantilla i **Axis** no sap serialitzar-les, per tant apareix un error d'execució.

Per tal de que Axis pugui serialitzar correctament hem d'ampliar l'script de serialització per a que contingui la informació dels tipus ***WSInfoxxxx***.

```
<!--   Per a instal·lar el WS fer:                                  -->
<!--   java org.apache.axis.client.AdminClient deploy.wsdd          -->
<!--   mentre Axis està funcionant                                  -->

<deployment name="WSImatge"
    xmlns="http://xml.apache.org/axis/wsdd/"
    xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">

  <service name="WSImatge" provider="java:RPC" >
            <parameter name="className" value="es.butlleti.WSImatge"/>
            <parameter name="allowedMethods" value="*"/>
            <parameter name="sendMultiRefs" value="false"/>
      <typeMapping
        xmlns:ns="http://WSImatge"
        qname="ns:WSInfoUsuari"
        type="java:es.butlleti.WSInfoUsuari"
        serializer="org.apache.axis.encoding.ser.BeanSerializerFactory"
        deserializer="org.apache.axis.encoding.ser.BeanDeserializerFactory"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      />
       <typeMapping
        xmlns:ns="http://WSImatge"
        qname="ns:WSInfoCategoria"
        type="java:es.butlleti.WSInfoCategoria"
        serializer="org.apache.axis.encoding.ser.BeanSerializerFactory"
        deserializer="org.apache.axis.encoding.ser.BeanDeserializerFactory"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      />
```

```
...
  </service>
</deployment>
```

Observem el que hem afegit a l'script: els elements **typeMapping** defineixen els tipus de l'usuari, així com els seus serialitzadors / deserialitzadors. En el nostre cas, escollir seguir l'estàndard **Java Bean** en les nostres classes de suport ens ha permès que el Serialitzador / deserialitzador pugui ser l'establert i programat ja a Apache Axis, estalviant-nos la tasca d'escriure'l.
Si executem un altre cop la mateixa sentència anterior hauríem de veure una cosa semblant a aquesta:

```
<soapenv:Envelope>
<soapenv:Body>
    <getPlantillaResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <getPlantillaReturn xsi:type="ns1:Vector">

<item xsi:type="ns2:WSInfoPlantilla">
    <descripcio xsi:type="xsd:string">Es la Plantilla per defecte</descripcio>
    <idPlantilla xsi:type="xsd:int">1</idPlantilla>
    <nom xsi:type="xsd:string">Defecte</nom>
</item>
<item xsi:type="ns3:WSInfoPlantilla">
    <descripcio xsi:type="xsd:string">Plantilla amb ordre diferent</descripcio>
    <idPlantilla xsi:type="xsd:int">2</idPlantilla>
    <nom xsi:type="xsd:string">Plantilla Diferent</nom>
</item>
<item xsi:type="ns4:WSInfoPlantilla">
   <descripcio xsi:type="xsd:string">Plantilla sense res</descripcio>
   <idPlantilla xsi:type="xsd:int">3</idPlantilla>
   <nom xsi:type="xsd:string">Plantilla rara</nom>
</item>
</getPlantillaReturn>
</getPlantillaResponse>
</soapenv:Body>
</soapenv:Envelope>
```

Podem observar ara la gran utilitat d'aquesta possibilitat de testeig del WS en Axis: si no fos d'aquesta manera hauríem de crear un client per a testejar cadascuna de les funcionalitats. Realitzada aquesta part només ens resta crear el client mòbil.

5. Creació del prototipus mòbil per l'ABM Imatges.

En la nostra primera aproximació volem crear un client en J2ME que accedeixi a les capacitats multimèdia del mòbil, per a adquirir les imatges i enviar-les a través del WS creat.

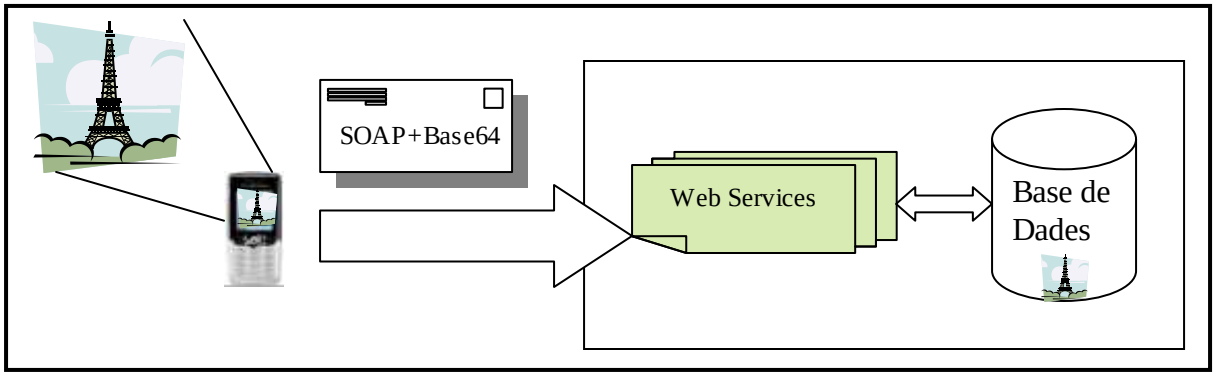
5.1 Introducció

En la primera fase d'estudi, a través dels diversos documents[1] trobats, no es va trobar cap manera d'utilitzar un midlet per a agafar una imatge ja feta en l'aplicació del mòbil. Per tant, l'única solució passava per tenir que capturar la imatge directament des de l'aplicació J2ME.

La tasca de recerca en aquest camp ens va portar cap al **MMAPI**[2] (Mobile Multimedia API), una API que ampliava el llenguatge J2ME per a poder utilitzar tots els elements multimèdia d'un mòbil (So, Vídeo, Fotografia). Tot i que a priori semblava la solució, la nostra primera prova en hardware real no va ser satisfactòria.

En els terminals de telefonia mòbil en trobem de compatibles i no compatibles amb l'API multimèdia. En el nostre cas vam veure a la Web del fabricant del **T610** (Sony Ericsson) que el mòbil amb càmera només era compatible amb l'API multimèdia en un subconjunt de funcions que no incloïa la funcionalitat de la càmera de vídeo.

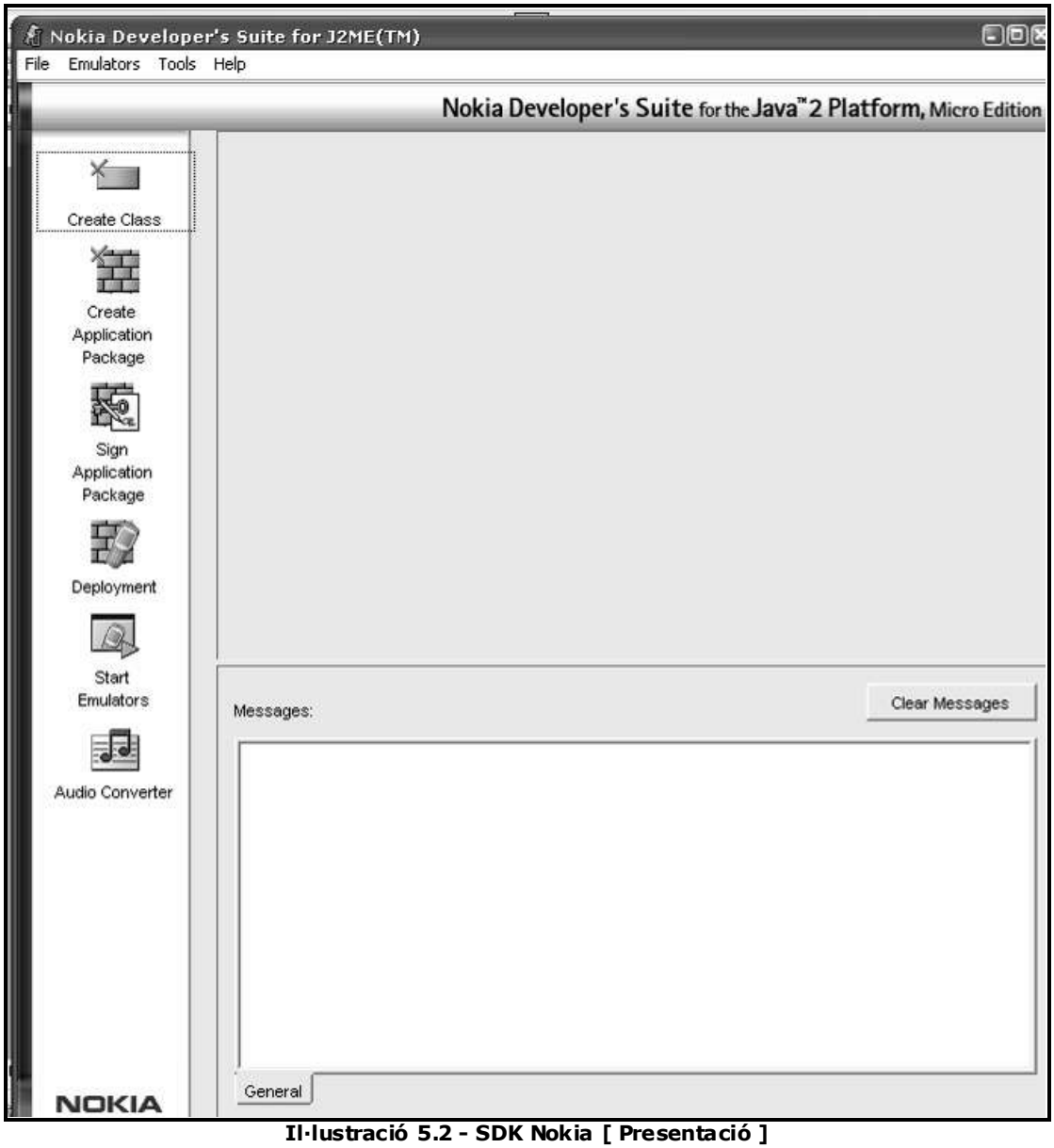
Això no passava en el terminal **Nokia 6100** (del que vam disposar per a realitzar les proves sobre un hardware real), ja que a la web oficial no s'indicava res, la seva càmera era un afegit, un perifèric. Per tant només podíem conèixer les possibilitats del terminal realitzant una petita aplicació semblant a la prova pilot, però sobre el sistema d'informació actual.



Il·lustració 5.1 - Esquema Client Imatge

[1] Podem trobar els documents al CD-ROM, i a la bibliografia.
[2] http://java.sun.com/webapps/device/device?api=41, aquesta llista però s'ha demostrat que està incompleta i és errònia.

Seguint les premisses esmentades anteriorment per tal d'estalviar memòria (utilitzar el mínim de classes possibles, etc...) crearem l'aplicació utilitzant l'SDK de Nokia, del que podem observar el seu aspecte en la següent il·lustració:



Il·lustració 5.2 - SDK Nokia [Presentació]

Per tal de poder crear una aplicació, crearem un directori (podem copiar un dels que hi ha d'exemple) i hi col·locarem el codi al subdirectori src. Per a executar-lo hem d'empaquetar l'aplicació després de compil·lar-la; ho podem fer amb un simple javac *.java.

5.2 Passos per a crear un projecte multimèdia

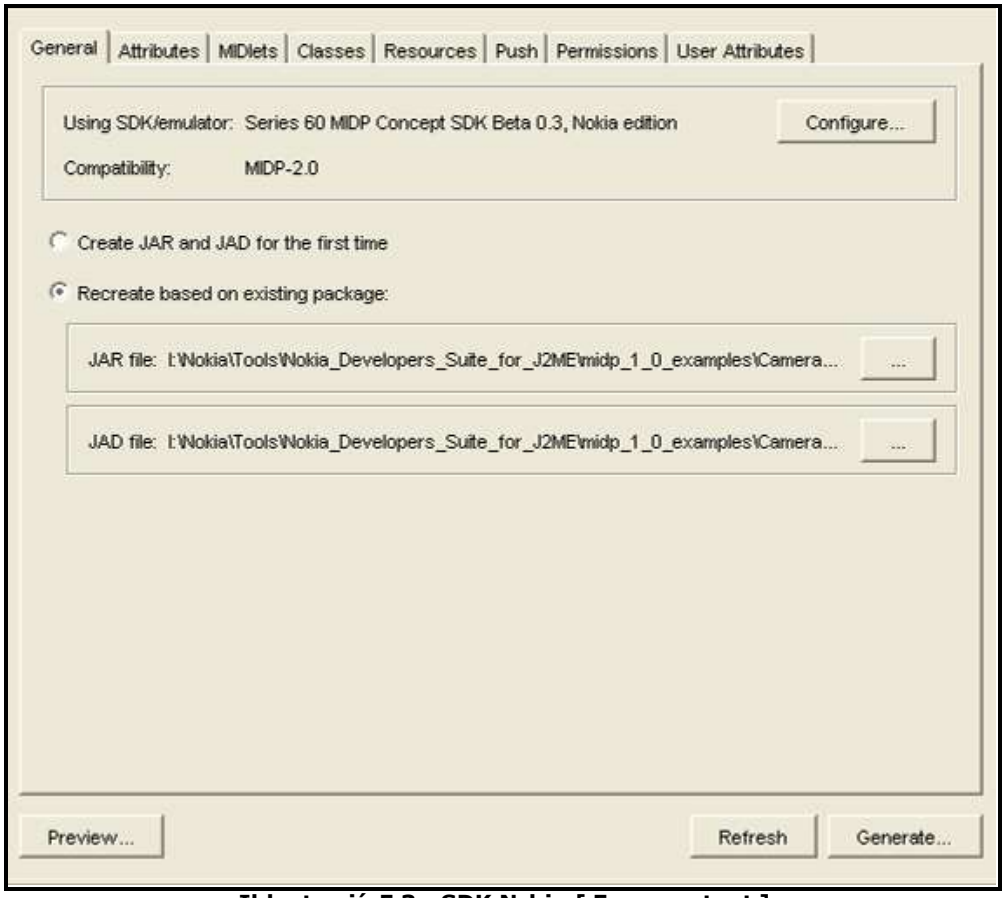
Compil·lació :

Per a compil·lar necessitem la llibreria **mmapi.jar**[1] (podem descarregar l'SDK de MMAPI des de la Web de Java Sun), també necessitarem les classes de kSOAP (posteriorment realitzarem la seva instal·lació) i finalment les classes de l'SDK de Nokia. Podem observar l'ordre de compil·lació a continuació[2]:

```
javac *.java -classpath
i:\Nokia\Tools\Nokia_Developers_Suite_for_J2ME\midp_1_0_examples\CameraMidlet\classes;I:\Nokia
\Devices\Nokia_Series_40_MIDP_Concept_SDK_Beta_0_3\lib\classes.zip;C:\WTK104\wtklib\devices\MM
Emulator\mmapi.jar
```

Les classes que compilem s'han de moure cap al subdirectori classes de l'aplicació, per a poder empaquetar-les i portar-les cap al mòbil.

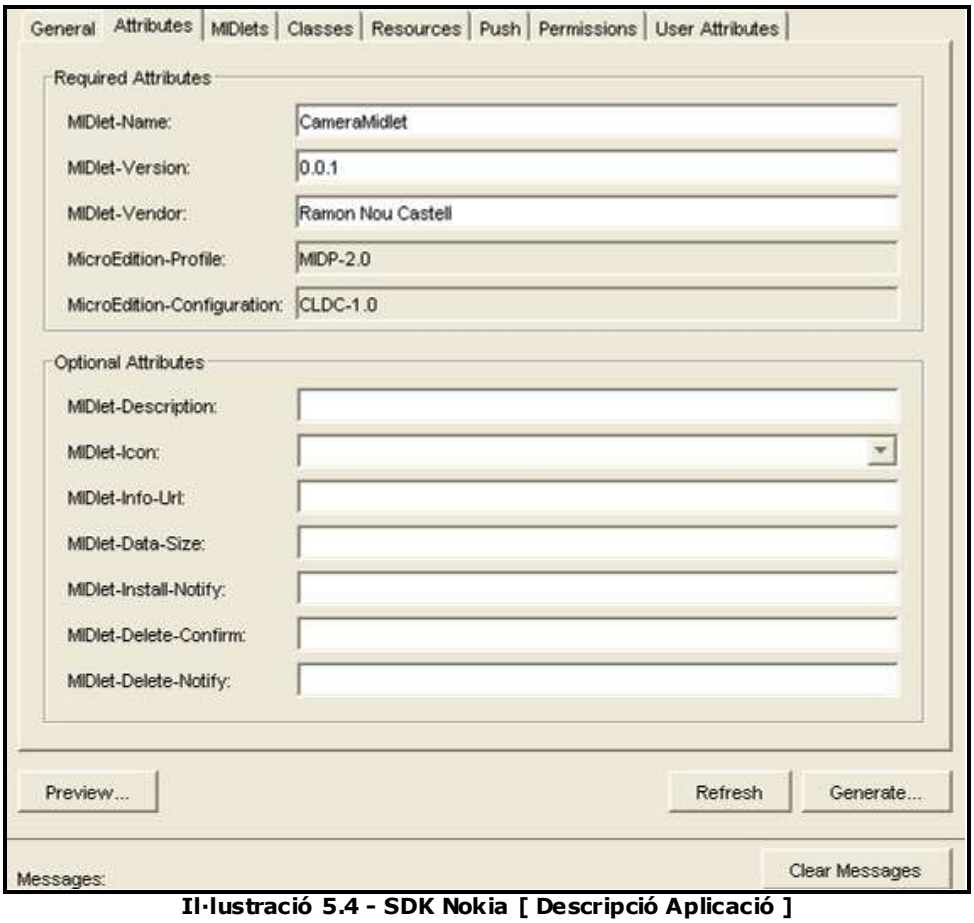
Per tal de fer-ho clicarem a *Create Application Package* (hem de configurar l'entorn per a que apunti al directori de l'aplicació que volem empaquetar / executar (File/preferences)). Seguidament hem de recrear el JAR i el JAD de l'aplicació:



Il·lustració 5.3 - SDK Nokia [Empaquetant]

[1] http://java.sun.com/products/mmapi/index.jsp
[2] Podem afegir –g:none per a que no generi informació de debug i reduir així la mida.

Premerem *Create JAR and JAD for the first time*, seleccionarem i escriurem els atributs de l'aplicació que crearem:



Il·lustració 5.4 - SDK Nokia [Descripció Aplicació]

A l'apartat de Classes seleccionarem les classes que volem incorporar al JAR; en aquest lloc només hem de col·locar les estrictament necessàries per a que l'aplicació funcioni. Finalment, a User Attributes hi podem col·locar les propietats (clau -> valor) que volem incorporar al JAD (un fitxer de text amb informació del Midlet).

Tot i que no entra dins l'abast d'aquest projecte, explicarem mínimament l'api de **MMAPI** i com podem aconseguir una aplicació que capturi la informació de la càmera.

5.3 Primera Part: llibreria MMAPI

Per a realitzar aquest tipus d'aplicació necessitem crear un nou **Canvas** (superfície de l'aplicació) que sigui capaç de tenir una imatge capturada de la càmera (temps real); per fer-ho ens basarem en un exemple que hi ha a l'**SDK** del **MMAPI**.

En primer lloc necessitem importar totes les classes necessàries:

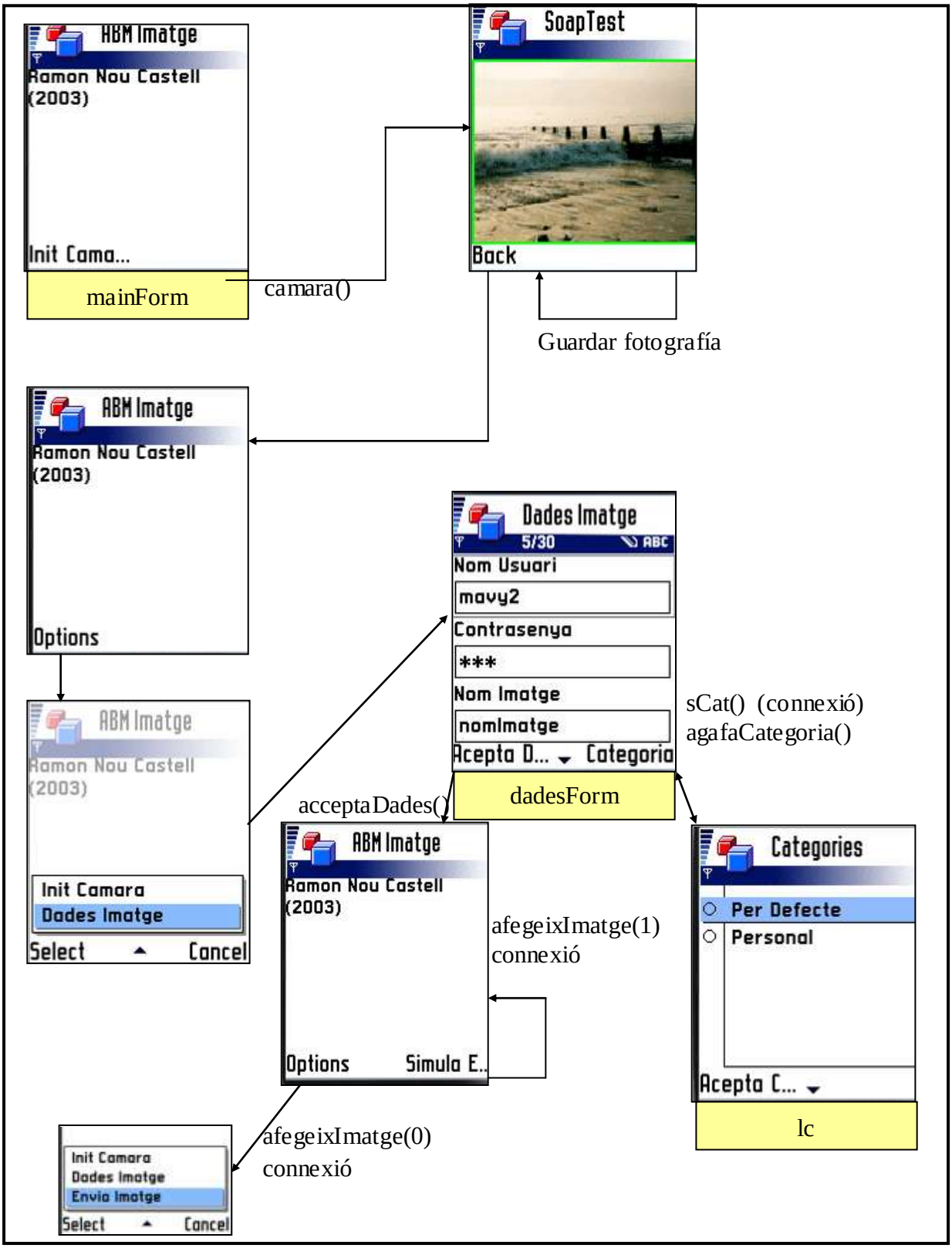
```
import javax.microedition.lcdui.*;
import javax.microedition.media.MediaException;
import javax.microedition.media.control.VideoControl;
```

Finalment, necessitem crear una classe (**CameraCanvas**) que hereti d'un **Canvas** per a redefinir els mètodes *paint* i *KeyPressed.*

```
public class CameraCanvas extends Canvas {
  private SoapTest mSnapperMIDlet;

  public CameraCanvas(SoapTest midlet, VideoControl videoControl) {
    int width = getWidth();
    int height = getHeight();

    mSnapperMIDlet = midlet;

    videoControl.initDisplayMode(VideoControl.USE_DIRECT_VIDEO, this);
       ...
    videoControl.setVisible(true);
  }

  public void paint(Graphics g) {
    int width = getWidth();
    int height = getHeight();

    // Draw a green border around the VideoControl.
    g.setColor(0x00ff00);
    g.drawRect(0, 0, width - 1, height - 1);
    g.drawRect(1, 1, width - 3, height - 3);
  }

  public void keyPressed(int keyCode) {
    int action = getGameAction(keyCode);
    if (action == FIRE)
      mSnapperMIDlet.captura();
  }
}
```

La constructora de la classe inicialitza el vídeo per a que tingui un accés directe al display del mòbil. També definim la classe *keyPressed* perquè cridi el mètode captura de l'aplicació quan es prem el botó de captura.

5.4 Flux de l'aplicació



Il·lustració 5.5 - Flux ABM Imatge Prototipus

5.5 Esquelet bàsic per a una aplicació multimèdia en J2ME

A continuació exposarem els passos necessaris per a crear una aplicació multimèdia:

```
public class SoapTest extends MIDlet implements Runnable, CommandListener{
        Form mainForm = new Form("ABM Imatge");
```

Definim un nou formulari (o pantalla) amb títol ABM Imatge

```
        Command getCommand = new Command("Init Camara", Command.SCREEN, 0);
```

Definim un nou element del menú amb nom **"Init Camara"** amb prioritat no especificada (0).

```
        public SoapTest() {
           mainForm.addCommand(getCommand);
           mainForm.setCommandListener(this);
        }
```

Al crear el midlet afegirem l'element del menú al formulari i hi assignarem un CommandListener (tal com es fa en el Java tradicional)

```
        protected void startApp(  ) throws MIDletStateChangeException
        {  Display.getDisplay(this).setCurrent(mainForm);   }
```

Quan s'engega l'aplicació es defineix mainForm com a "pantalla" actual.

```
        ...
        public void commandAction( Command p1, Displayable p2 )    {
               if (p1.getLabel().equals ("Init Camara")) camara();
```

I finalment col·loquem les crides que es fan a l'activar-se les funcions del menú.

```
               if(p1.getLabel().equals ("Back"))
                  Display.getDisplay(this).setCurrent(mainForm);
        }
}
```

Anem a introduir el codi necessari per a crear el flux de l'aplicació dissenyada, per tal que l'aplicació funcioni correctament i que el **garbage collector**[1] no s'engegui. Crearem tots els formularis a l'inici de l'aplicació i els deixarem creats fins que es tanqui.

Per tant, tindrem dos formularis: el principal i l'encarregat d'adquirir les dades de la imatge:

```
        Form mainForm = new Form("ABM Imatge");
        Form dadesForm = new Form("Dades Imatge");
```

Les propietats les aconseguirem del fitxer **.jad**, introduïdes gràcies al camp *user attributes* de l'entorn d'empaquetat:

```
        String nomUsuari = getAppProperty("usuari");
        String contraUsuari = getAppProperty("contrasenya");
        String urlWS = getAppProperty("url");
```

[1] Procés que recull totes les variables, classes, etc... que ja no s'utilitzrn per tal de lliberar memòria.

Crearem la llista de Categories que necessitem, així com tots els TextFields i cadenes:

```
java.util.Vector llistaC = null;


String nomImatge = new String ("nomImatge");
String descImatge = new String ("Descripcio");
int catImatge = 1;


TextField usuariField = new TextField("Nom Usuari", nomUsuari, 30, TextField.ANY);
TextField contrasenyaField = new TextField("Contrasenya", contraUsuari, 8,
TextField.PASSWORD);
...
TextField catImaField = new TextField("Categoria", new Integer(catImatge).toString(), 30,
TextField.NUMERIC);


StringItem copyr = new StringItem("Ramon Nou Castell (2003)", "");
javax.microedition.lcdui.List lc = new javax.microedition.lcdui.List("Categories",
javax.microedition.lcdui.List.EXCLUSIVE);
```

Els TextField vénen definits per una etiqueta i pel valor, així com per la seva longitud màxima i el seu tipus. Alguns dels tipus poden ser:

TextField.ANY → Qualsevol contingut
TextField.PASSWORD → S'emmagatzema amb * de contrasenya
TextField.NUMERIC → Només admet números.

Afegim també una cadena (**StringItem**) al formulari, que conté una etiqueta i el contingut de l'String. Finalment **lc** és una llista de selecció (Exclusiva), que ens servirà per escollir una opció entre totes les categories.

En els **midlets,** una llista de selecció és un formulari especial i ha de ser tractat com a tal (*commandListener*, afegir elements del menú...)

Acabem amb les dades globals afegint la comanda inicial i variables de control per a evitar inicialitzar dues vegades els elements dels formularis. També inclourem les variables necessàries per a guardar la imatge capturada i per capturar-la.

```
Command getCommand = new Command("Init Camara", Command.SCREEN, 0);


boolean imatge = false;
boolean acepta = false;
boolean dades = false;
boolean catNew = false;


private Player mPlayer;
private VideoControl mVideoControl;
private byte [] lastImage;
```

En el creador afegirem la comanda "**Init Camara**" al menú i l'String que hem creat. A més, hi connectarem els events:

```
public SoapTest() {
 mainForm.append(copyr);
 mainForm.addCommand(getCommand);
 mainForm.setCommandListener(this);
}
```

El flux d'esdeveniments esmentat anteriorment s'aconsegueix definint correctament el mètode *commandAction* :

```
public void commandAction( Command p1, Displayable p2 ) {
	if (p1.getLabel().equals ("Init Camara")) camara();
	if (p1.getLabel().equals ("Accepta Dades")) acceptaDades ();
	if (p1.getLabel().equals ("Accepta Categoria")) {
		agafaCategoria();
		dadesImatge();
	}
	if (p1.getLabel().equals ("Envia Imatge")) afegeixImatge(0);
	if (p1.getLabel().equals ("Simula Enviament")) afegeixImatge(1);
	if (p1.getLabel().equals ("Dades Imatge")) dadesImatge();
	if (p1.getLabel().equals ("Categoria")) sCat();
	if (p1.getLabel().equals ("Back")) Display.getDisplay(this).setCurrent(mainForm);
}
```

Comencem a comentar les funcions creades, en l'ordre real en que es criden.

5.5.1 Mètode camara

És l'encarregada de materialitzar el formulari CameraCanvas i cridar a la funcionalitat de captura de vídeo.

```
public void camara (){
Player mPlayer = null;
	try {
		mPlayer = Manager.createPlayer("capture://video");
```

Creem un dispositiu de captura de vídeo (si el volem de so, seria **capture://audio** o similar).

```
		mPlayer.realize();
		mVideoControl = (VideoControl)mPlayer.getControl("VideoControl");
		Canvas canvas = new CameraCanvas(this, mVideoControl);
```

Afegim el control de vídeo a un nou Canvas del tipus CameraCanvas.

```
		getCommand = new Command("Back", Command.SCREEN, 0);
		Canvas.addCommand(getCommand);
		Canvas.setCommandListener(this);
```

```
                Display.getDisplay(this).setCurrent(canvas);
                mPlayer.start();
```

Afegim l'element del menú per a retornar d'aquest formulari a l'anterior, i finalment engeguem la captura.

Si recordem la construcció de ***cameraCanvas***, teníem a l'acció **FIRE** una crida a un mètode anomenat *captura*. Aquest mètode s'encarrega d'agafar les dades de la imatge de la càmera i guardar-les en un vector de bytes per al seu ús posterior; també avisa a l'aplicació general que pot seguir el flux (és a dir, activar la comanda Dades Imatge).

```
void captura () {
        lastImage = mVideoControl.getSnapshot(null);
```

El mètode getSnapshot, com el seu nom indica, retorna a lastImage el flux de bytes de la imatge capturada.

5.5.2 Mètode Dades Imatge

Una vegada tenim la imatge en la variable lastImage, podem cridar a Dades Imatge per a omplir tota la informació relacionada amb la imatge (Autor, contrasenya, nom de la imatge, descripció i categoria)

```
public void dadesImatge (){
if (dades == false)    {
        getCommand = new Command("Accepta Dades", Command.SCREEN, 0);
        dadesForm.append(usuariField);
        dadesForm.append(contrasenyaField);
        dadesForm.append(nomImaField);
        dadesForm.append(descImaField);
        dadesForm.addCommand(getCommand);
        getCommand = new Command("Categoria", Command.BACK, 0);
        dadesForm.addCommand(getCommand);
```

Afegim els TextFields corresponents, i les dues noves comandes (Accepta Dades i Categoria)

```
        llistaC = recuperaCat ();
```

En aquest punt fem una crida al mètode que s'encarrega de recuperar la llista de Categories per a guardarles a llistaC.

```
        dadesForm.setCommandListener(this);
        dades = true;
        }
        Display.getDisplay(this).setCurrent(dadesForm);
}
```

5.5.3 Mètode recuperaCat

El mètode *recuperaCat*[1] en aquesta primera part és nul, ja el comentarem quan afegim la part de **kSOAP**.

[1] Podem trobar el diagrama de seqüència d'aquest tipus d'operació al punt 6.6.

Quan fem una pulsació sobre l'element menú Categoria el flux ens porta cap al mètode sCat, que té com a funció omplir la pantalla de selecció de Categoria amb la llista recuperada posteriorment.

```
public void sCat () {
        int i = 0;
       for (i = 0; i< lc.size();) lc.delete(i);
```

Esborrem el contingut de la pantalla lc.

```
       for (java.util.Enumeration e = llistaC.elements(); e.hasMoreElements () ;)
       { lc.append (((WSInfoCategoria) e.nextElement()).getNomCat(),null);       }
```

Afegim els elements de la llistaC a lc

```
       if (catNew == false)  {
       Command comando = new Command ("Accepta Categoria",Command.OK,0);
       lc.addCommand(comando);
       lc.setCommandListener(this);
       catNew = true;
       }
```

Si aquesta funcionalitat no ha estat cridada, afegim la comanda Accepta Categoria i activem l'escoltador d'events.

```
       Display.getDisplay(this).setCurrent(lc);
```

Finalment, col·loquem com a display actual la llista.

5.5.4 Mètode agafaCategoria

Quan sortim de la llista, es crida al mètode *agafaCategoria*[1] que s'encarrega de col·locar el número de la categoria en la variable corresponent.

```
public void agafaCategoria () {
       int num = 0;
       num = lc.getSelectedIndex();
       catImatge = ( (WSInfoCategoria) llistaC.elementAt ( num )).getIdCat();
}
```

5.5.5 Mètode acceptaDades

Finalment, solament ens queda acceptar les dades i habilitar les comandes d'enviament a través d'Internet. Tot això ho fem en el mètode **acceptaDades**:

```
public void acceptaDades () {
       nomUsuari = usuariField.getString();
       contraUsuari = contrasenyaField.getString();
       nomImatge  = nomImaField.getString();
       descImatge = descImaField.getString();
       catImatge = (new Integer (0).valueOf(catImaField.getString())).intValue();
       if (acepta == false)  {
```

[1] Podem trobar el diagrama de seqüència d'aquest tipus d'operació al punt 6.6.

```
                getCommand = new Command("Envia Imatge", Command.SCREEN, 0);
                mainForm.addCommand(getCommand);
                getCommand = new Command("Simula Enviament", Command.BACK, 0);
                mainForm.addCommand(getCommand);
                acepta = true;
        }
        mainForm.setCommandListener(this);
        Display.getDisplay(this).setCurrent(mainForm);
}
```

Amb aquest codi tenim l'esquelet de l'aplicació fet; només ens faltaria afegir el codi SOAP per tal d'agafar les dades del servidor Axis.

Per a provar l'aplicació podem utilitzar un emulador. Després de compil·lar l'aplicació, i empaquetar-la en un jar/jad, podem anar a l'opció de l'SDK de Nokia anomenada Start Emulators i escollir l'emulador correcte (un **Sèrie 60**, ja que és l'únic compatible amb la càmera).

5.6 Segona Part : Afegint kSOAP al projecte.

En primer lloc afegirem totes les classes estàndards de kSOAP, i posteriorment afegirem també les classes necessàries per a transportar dades binàries, és a dir, els serialitzadors / deserialitzadors en Base64 que no són al paquet estàndard, ja que ocupen espai innecessari.

kSOAP es pot descarregar d'aquí: http://ksoap.enhydra.org/. En concret necessitem la versió **ksoap-midp.zip,** que conté les classes compilades; la seva mida total és de 73 kbytes.

Per tal d'incorporar-les al projecte, les copiarem al directori del projecte, concretament al subdirectori classes, i haurem d'especificar les classes que volem afegir al projecte (de moment totes) mitjançant l'aplicació de Nokia. D'aquesta manera al crear el Jar/Jad ja s'inclouran (hem d'anar en compte de no incloure més fitxers que els .class, ja que poden sorgir errors).

5.6.1 Serialitzar/deserialitzar les classes de suport

Una vegada tenim aquest pas fet podem començar la programació de les parts necessàries. En primer lloc tenim que deixar clar que com kVM és una versió reduïda de la màquina virtual de Java no inclou la possibilitat de serialitzar / deserialitzar les classes (per exemple); per tant, hem de construir-los nosaltres. Si observem l'API de ***kSOAP*** (concretament de ***kObjects***) o mirem el directori *org\kobjects\serialization* del paquet veurem tres classes: *ElementType*, *PropertyInfo* i *KvmSerializable,* que són les encarregades d'aquesta tasca.

Per aquesta primera aplicació necessitem dos classes de transport: *WSInfoUsuari* i *WSInfoCategoria*. Aquestes classes, construïdes a la banda del servidor, han d'incorporar-se al projecte, però al mateix temps s'han de convertir en una subclasse de *KvmSerializable* i crear els mètodes corresponents. Si **kVM** tingués la possibilitat de serialitzar automàticament, com ho fa Java, normalment no caldria fer-ho.

Anem a modificar la primera classe **WSInfoUsuari.java:**

```
import org.kobjects.serialization.PropertyInfo;
import org.kobjects.serialization.ElementType;
import org.kobjects.serialization.KvmSerializable;
```

Importem les tres classes responsables de la serialització / deserialització.

```
public class WSInfoUsuari implements KvmSerializable {
```

Assenyalem que volem implementar la interfície KvmSerializable.

```
  boolean valid = false;
  String email = null;
  String permis = null ;

  private static int PROP_COUNT = 3;
```

PROP_COUNT conté el nombre de propietats de la classe.

```
  private static PropertyInfo PI_valid = new PropertyInfo("valid",ElementType.BOOLEAN_CLASS);
  private static PropertyInfo PI_email = new PropertyInfo("email",ElementType.STRING_CLASS);
  private static PropertyInfo PI_permis = new PropertyInfo("permis",ElementType.STRING_CLASS);
```

Es defineix cadascuna de les propietats amb el seu nom i la seva classe.

```
  private static PropertyInfo[] PI_PROP_ARRAY = {PI_valid,PI_email,PI_permis};
```

Finalment es construeix un array de propietats amb els tres atributs.

```
   public WSInfoUsuari () { }
   public void setValid ( boolean bValid) { valid = bValid; }
   public boolean getValid ( ) { return (valid); }

   public void setEmail ( String sEmail) { email = sEmail; }
   public String getEmail ( ) { return (email); }

   public void setPermis ( String sPermis) { permis = sPermis; }
   public String getPermis ( ) { return (permis); }

    public Object getProperty(int param) {
        if ( param == 0 ) {
            return new Boolean(getValid());
        } else if ( param == 1 ) {
            return getEmail();
        } else if ( param == 2 ) {
              return getPermis();}
        else { return null; }
    }
```

El mètode getProperty s'encarrega de retornar el valor de la propietat definida pel paràmetre. Normalment cridarà al mètode **get** de cadascuna, però pot incloure altres funcionalitats com veurem més endavant.

```
public void setProperty(int param, Object obj) {
        if ( param == 0 ) {
```

```
            boolean l = ((Boolean) obj).booleanValue();
            setValid( l );
        } else if ( param == 1 ) {
            setEmail( (String) obj );
        } else if ( param == 2 ) {
            setPermis( (String) obj );
        } else {  }
    }
```

El mètode **setProperty** realitza la tasca contrària, assigna a la propietat definida pel primer paràmetre el valor que hi ha al segon paràmetre. Normalment es crida al mètode set de cada propietat.

```
    public int getPropertyCount() {
        return PI_PROP_ARRAY.length;
    }
    public void getPropertyInfo(int param, org.kobjects.serialization.PropertyInfo
propertyInfo) {
        propertyInfo.name = PI_PROP_ARRAY[param].name;
        propertyInfo.nonpermanent = PI_PROP_ARRAY[param].nonpermanent;
        propertyInfo.copy(PI_PROP_ARRAY[param]);
    }
}
```

Finalment *getPropertyInfo* retorna informació de cada propietat.

Ara tenim la classe ***WSInfoUsuari*** llesta per rebre i enviar la seva informació a través de **kSOAP**. Tenim que realitzar la mateixa tasca per a ***WSInfoCategoria*** (el codi està al CD-ROM que acompanya aquest document).

Copiem els dos .java al directori corresponent del projecte i comencem a editar el fitxer principal per a generar l'aplicació.

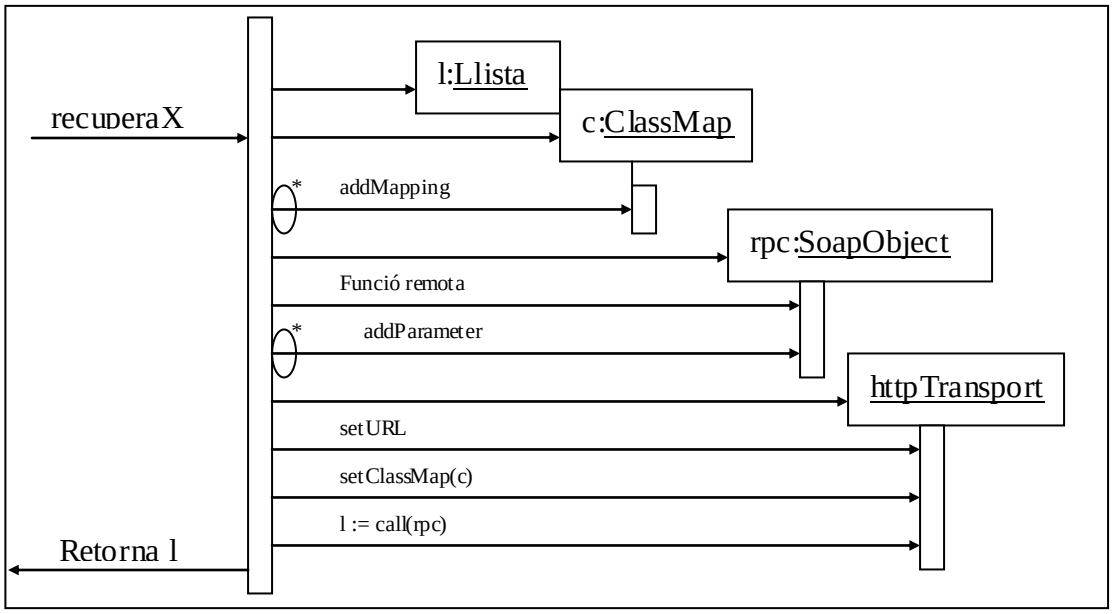
5.7 Afegint la funcionalitat kSOAP a l'aplicació principal.

Les modificacions que hem de fer a l'aplicació són les següents (deixarem per la última l'enviament de la imatge, ja que necessita incorporar el serialitzador de *Base64*):

1 – Importar les classes corresponents:

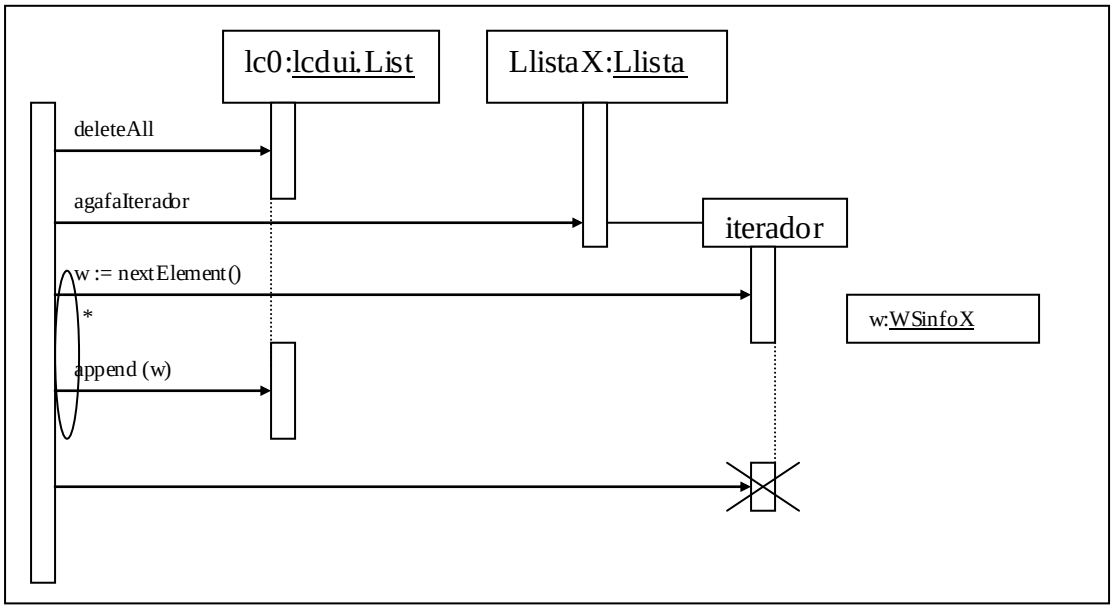
```
import org.ksoap.SoapObject;
import org.ksoap.transport.HttpTransport;
import org.ksoap.ClassMap;
```

2 – Creació/Modificació del mètode ***recuperaCat***:
Aquest mètode connectarà amb el WS i descarregarà la llista de categories. Per fer-ho demanarem el servei al mètode remot anomenat ***getCatIma***. A continuació mostrarem el diagrama de seqüència genèric per a aquest tipus de funcions.



Il·lustració 5.6 - D.Seqüència [recuperaX]

Quan tenim la llista recuperada és el moment d'utilitzar el mètode *preparaX* per a preparar la llista (formulari), per a que l'usuari seleccioni un element. Mostrem el diagrama de seqüència a continuació:



Il·lustració 5.7 - D.Seqüència [preparaX]

Finalment, ens queda la funció que recull l'índex seleccionat per l'usuari; el cerca a la llista recuperada (que normalment hi tindrà més informació) i la col·loca en la variable corresponent.

Per fer-ho simplement utilitzem els mètodes *getSelectedIndex* de la ***lcdui.List*** per a obtenir l'índex de l'element seleccionat, i a continuació utilitzem el mètode *elementAt* per a recuperar l'element, que és al mateix índex en la llista original (que té el mateix ordre).

A continuació veurem el codi dels mètodes descrits (***recuperaCat***, ja que els altres han aparegut abans):

```
public java.util.Vector recuperaCat () {
   java.util.Vector llista = null;
   ClassMap classMap = new ClassMap();
```

La classe ClassMap permet definir un mapeig entre un "descriptor" SOAP i una classe Java; en aquest cas la necessitem per a dos tasques: definir un mapeig entre la classe Vector de J2SE i J2ME, i definir el mapping de WSInfoCategoria amb la nostra classe Categoria que acabem de crear:

```
   classMap.addMapping("http://WSImatge", "WSInfoCategoria",   new
WSInfoCategoria().getClass() );
   classMap.addMapping("http://xml.apache.org/xml-soap", "Vector",   new
java.util.Vector().getClass() );
```

Aquesta és la sintaxi. En primer lloc indiquem l'espai de noms de la classe que ens arribarà (o enviarem) en el primer lloc (*WSInfoCategoria), l'*hem definida a ***Axis*** com una classe de l'espai de noms http://WSImatge. Posteriorment li indiquem el nom de la classe que ens arriba (WSInfoCategoria), i finalment la classe on es mapejarà en aquesta banda.

```
 SoapObject rpc = new SoapObject("WSImatge","getCatIma");
```

Creem l'objecte SOAP amb el nom del WS com a primer paràmetre i el mètode remot com a segon paràmetre. En aquest punt podríem afegir els possibles paràmetres de l'aplicació, però en aquest cas no en té.

```
 HttpTransport ht = new HttpTransport("http://"+urlWS+"/axis/services/WSImatge","WSImatge");
```

Creem el medi de transport per SOAP (HTTP que és el més comú i l'únic disponible normalment). Afegim l'URL del WS i el nom (aquest paràmetre, no té cap funció ara).

```
 ht.debug = true;
 ht.setClassMap( classMap );
```

Col·loquem el transport en mode depuració i assignem el mapeig de classes que hem fet anteriorment.

```
 llista = (java.util.Vector)(ht.call(rpc));
 return (llista);
```

Cridem al mètode call enviant el soapObject a través del transport http. El mètode retorna la llista ja.

Si provem l'aplicació ara tindrem tota la funcionalitat, excepte la d'enviament d'imatges. Anem a afegir tot seguit les classes necessàries per a serialitzar les dades de la imatge.

5.8 Afegint la serialització / deserialització en Base64 a l'aplicació principal.

Necessitem descarregar el fitxer *MarshalBase64.java* de la web de **kSOAP** i les classes necessàries de http://kobjects.dyndns.org/kobjects/ , la web de kObjects. Després de col·locar-les al seu subdirectori corresponent podem començar a programar el mètode **afegeixImatge**. A aquest mètode se l'indicarà mitjançant un paràmetre si volem simular l'enviament o volem enviar la imatge.

```
public void afegeixImatge(int tipus) {
  ClassMap classMap = new ClassMap();
  StringItem resultItem = new StringItem("", "");
  mainForm.append(resultItem);
  try   {
    String operacio = null;
    if (tipus == 0) operacio = new String ("insImatge");
       else operacio = new String ("tstImatge");
```

insImatge i **tstImatge** són els dos mètodes remots del WS que cridarem.

```
    SoapObject rpc = new SoapObject("WSImatge",operacio);
    rpc.addProperty("nom", new String (nomUsuari.getBytes("UTF-8")));
    rpc.addProperty("contrasenya", contraUsuari);
    rpc.addProperty("nomImatge", new String (nomImatge.getBytes("UTF-8")));
    rpc.addProperty("descripcio", new String (descImatge.getBytes("UTF-8")));
    rpc.addProperty("Categoria", new Integer(catImatge));
```

El mètode **addProperty** afegeix al missatge SOAP el nom del paràmetre i les dades. Com que la codificació del mòbil és **ISO-LATIN-1** i la d'AXIS és per defecte **UTF-8**, hem de transformar les dades de les cadenes a **UTF-8** per al seu transport. Si no ho féssim es podria produir un error d'execució a la banda del servidor; en el cas contrari tindríem que fer el mateix mètode per a la classe *WSInfoCategoria*, però ho deixem per més endavant.

```
    if (tipus == 0) {
       MarshalBase64 mb64 = new MarshalBase64();
       classMap.addMapping("http://www.w3.org/1999/XMLSchema", "base64Binary",
MarshalBase64.BYTE_ARRAY_CLASS, mb64) ;
```

ClassMap té un altre mètode addMapping en el qual se li passa la classe que es fa servir per a serialitzar/deserialitzar les dades; en aquest cas fem servir la classe MarshalBase64 que traduirà a base64Binary.

```
       rpc.addProperty("Dades", lastImage);
    }
```

L'aplicació, quan hagi de rebre o enviar una dada binaria **kSOAP**,agafarà l'element i el passarà per la classe **MarshalBase64**, que s'encarregarà de treure el flux en Base64 de les dades. En el cas de rebre la dada es transformarà en un tipus byte[].

Podem compil·lar l'aplicació i executar-la en l'emulador per tal de provar-la. Dins de l'emulador podríem activar algunes opcions de depuració i tracejat; lamentablement l'SDK de Nokia no dóna facilitats per mostrar algunes de les opcions (com són per exemple el tracejat de reserves/liberacions de memòria), que fan que l'emulador no faci aparició (es bloqueja) ja que traceja l'emulador també. Quan accedim a la pantalla Dades Imatge el Midlet es connecta al servidor i es guarden les categories a la llista interna.

Enviem la imatge i observem a la base de dades com s'ha inserit:

Il·lustració 5.8 - ABM Imatge, enviament correcte

id	nom	descripcio	DATA	dades	nomAutor	idCategoria
35	Imatge Ex1	Imatge desde el movil,	2003-12-08	[**BLOB** [1]- 15.8 KB]	mavy2	1

Taula 5.1 - ABM Imatge - resultat de l'enviament

5.9 Prova en el terminal Nokia 6100 + Càmera.

Com que a la pàgina oficial no hi havia cap informació sobre la compatibilitat de la MMAPI amb el terminal que disposàvem, ja que tenia com a característica una càmera externa, no va quedar cap més remei que provar-ho.

Donat que el terminal mòbil del que disposem no permet utilitzar **Midlets** amb extensions multimèdia (**MMAPI**) per a controlar la càmera, i no podem utilitzar els **Midlets** per accedir al sistema de fitxers per utilitzar les imatges capturades, hem optat per una altra solució per enviar les imatges a la base de dades.

Hi havia diferents alternatives per fer-ho: tenim la possibilitat de transmetre la imatge per infrarojos a un altre dispositiu mòbil (Pocket PC) i transmetre-la aleshores mitjançant un client Web Service allí. L'altra alternativa i l'adoptada per tal d'assolir l'objectiu d'independència i poder-ho fer tot des d'un mòbil, és la d'utilitzar la possibilitat d'enviar correu electrònic a través de **GPRS**.

[1] Binary Large Object

6. Procés ProcMail (Processat d'Email)

6.1 Comentaris de la solució

La solució la dissenyarem per a un sistema operatiu Windows, ja que per Linux tenim moltes més opcions per realitzar-lo. En primer lloc necessitem un servidor de correu que puguem utilitzar lliurement. Cercant per la xarxa es va trobar la següent opció:

http://www.pmail.com/overviews/ovw_mercury.htm

Per tal de fer-ho utilitzarem la capacitat d'enviar imatges per *correu electrònic* (similar als missatges MMS) que disposa el terminal. El correu electrònic ha de dirigir-se a una adreça dedicada a aquest motiu, desestimant la possibilitat d'utilitzar un servidor de correu que tingui l'usuari (a un ISP); per evitar problemes de seguretat (hem de llegir el correu, per tal de distingir els missatges que ens interessen) s'ha decidit instal·lar un nou servidor de correu a l'ordinador de l'usuari/administrador.

S'ha escollit, com en la resta del projecte, un servidor gratuït de correu (Mercury) per Windows. A part de ser gratuït, aquest programari ens permet crear daemons que tractin el correu quan arriba. Aquest servidor pot tenir SMTP i POP3; en el nostre cas només necessitem SMTP (*Simple Mail Transfer Protocol*).

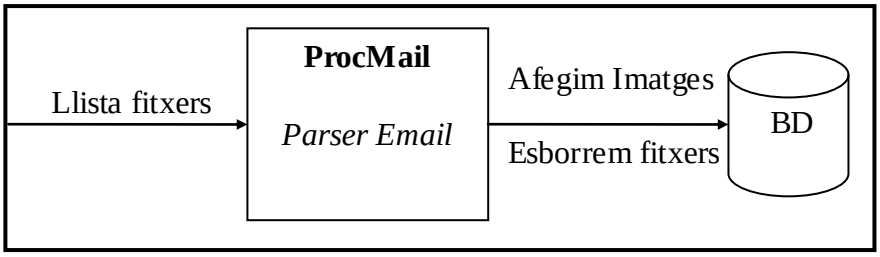
La instal·lació és molt senzilla: en primer lloc ens demana el nom del domini on rebrem els emails; si no en tenim cap podem utilitzar l'adreça ip de la següent manera: [80.25.193.222], d'aquesta manera l'usuari ens enviarà el correu a la següent adreça: postmaster@[80.25.193.222]

Podem tenir problemes si el servidor des d'on enviem el correu no accepta aquest tipus d'adreces (no tots ho accepten). Per tant, o busquem un altre servidor de sortida de correu o utilitzem algun servei de nom gratuït que ens ofereixi un nom de domini i ens doni d'alta al DNS.

Utilitzant un *template* que inclou Mercury (Mercury no inclou documentació), s'ha intentat crear un daemon, però no ha tingut molt d'èxit ja que penjava el servidor i el feia reiniciar. Donada la manca d'informació i desestimant la possibilitat de comprar els manuals, es va decidir una altra solució més general i portable a un altre servidor de correu.

La solució final es basa en utilitzar la cua de missatges que el servidor de correu emmagatzema al sistema de fitxers de la màquina en què s'executa. Per això crearem un procés que cada x segons cerqui els missatges que han arribat a la cua, analitzi cercant els missatges que continguin una imatge, i enviï aquesta imatge a la base de dades.

6.2 Disseny del procés



Hem utilitzat C++ pel procés per executar-lo estalviant recursos (és un procés que s'executarà sempre).

Anem a comentar el funcionament dels diferents procediments que trobem.

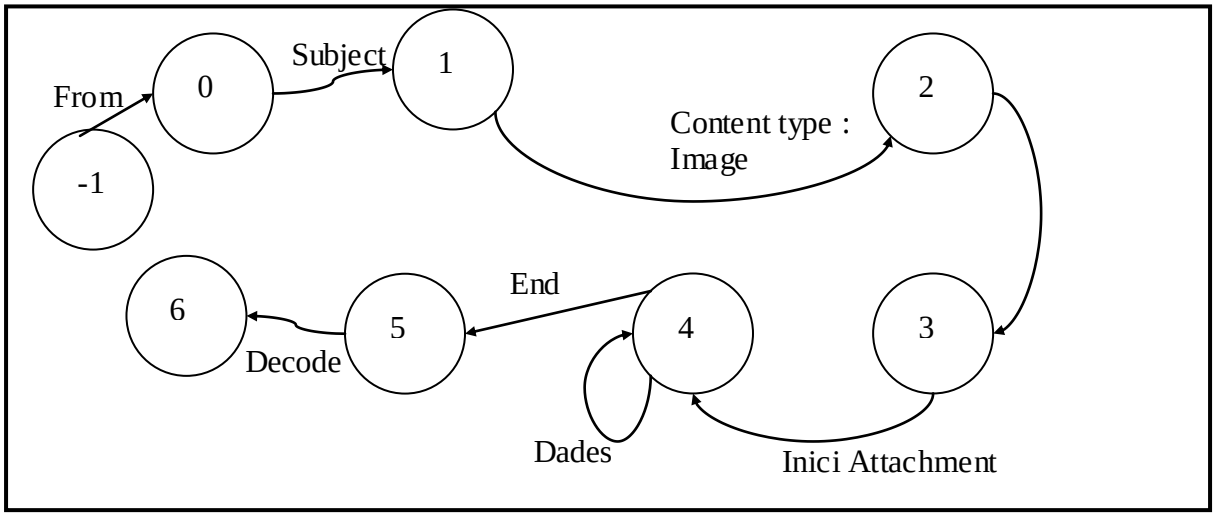
6.2.1 Programa Principal.

Consta d'un bucle infinit i un sleep de x segons; donada la baixa utilització d'aquest servei, un temps d'espera de 60 segons pot ser suficient per tal de tractar els correus que arriben. En aquest bucle, realitzem una cerca del directori corresponent a la cua i col·loquem els fitxers en una llista (list), si la llista no és buida la *processem* i un cop processada *esborrem* els fitxers que hi havien.

6.2.2 Extrau Imatges

S'ha decidit utilitzar el *subject* de l'email com a descripció de la imatge i com a títol utilitzarem el nom del fitxer enviat. Per tal d'evitar errors en la banda del client i reduir al màxim l'escriptura de dades del client mòbil, la categoria de la imatge serà la definida per defecte i l'autor de les imatges serà l'administrador.

El processat de l'email es realitza utilitzant un autòmat finit (DFA) seguint l'estructura trobada al RFC 1521[1] i 2183:



Il·lustració 6.2 - Autòmat Finit del parser del correu

[1] http://www.faqs.org/rfcs/rfc1521.html i http://www.ietf.org/rfc/rfc2183.txt

En l'estat 5, realitzem la conversió de Base-64 a binari, per tal de poder guardar la imatge. En l'estat 6 inserim les dades a la base de dades. El codi el podem trobar al CDROM que acompanya el document.

6.3 Proves.

L'aplicació es va dissenyar utilitzant un correu enviat des d'un PC, per tant les capçaleres podien canviar quant s'enviés des d'un mòbil, així com altres apartats (separacions, línies en blanc, etc...). Quan vàrem tornar a tenir disponible un terminal amb la característica d'enviar correus electrònics, es va provar el servidor, i com s'esperava una de les capçaleres no era igual (el mòbil enviava IMAGE/JPEG en compte de "image/jpeg" i canviava l'ordre d'algunes capçaleres). Altres decisions en el disseny que es van utilitzar es van haver de modificar per tal que el procés funcionés correctament.

Un altre dels problemes que vam trobar va ser que el telèfon mòbil enviava el subject (que s'utilitza com a descripció de la imatge) amb una codificació ISO-8859-1, enviant-ho amb la capçalera següent: *?=ISO-8859-1?Q?descripció=?.* Per tant, es va modificar l'autòmat per tal de que eliminés la resta de la informació.

Una vegada adaptat l'autòmat a aquests petits canvis es va tornar a provar, amb un resultat satisfactori.

Aquesta solució, tot i que no permet el grau d'integració desitjat (s'elimina el concepte d'autor de les imatges per fer que autor sigui el text agafat del "from" -per això també s'ha relaxat la clau forana de la taula imatges- i la categoria s'agafa per defecte), ens permet utilitzar una funcionalitat que donava J2ME i que no sol estar en els mòbils amb càmera externa.

6.4 Codi – Parts rellevants

Per a construir el codi, es va fer servir el paquet ***DEV-CPP***[1], que utilitza el GCC per a compil·lar. Per a inserir a la base de dades ***mySQL*** necessitàvem la llibreria (DLL) de ***mySQL*** per a C, que podem descarregar com un paquet del ***DEV-CPP*** o directament des de la pàgina de mySQL. Per a utilitzar-la tenim que indicar al compilador la localització de la llibreria *(../lib/libmySQL.a)* i realitzar un include de *mysql.h.*

Per a la transformació de MIME en base 64 a binari necessitàvem un conversor que realitzes les tasques comentades a la **RF*C 1113***; el conversor que es va utilitzar transformava fitxers de base 64 a binari, per tant la tasca que tenia que realitzar el programa era la de crear un fitxer amb el contingut en Base 64 de les dades.

Anem a analitzar i comentar algunes parts del codi de **ProcMail**, concretament la part en la que transmetem les dades binaries a la base de dades:

```
char *szSQL;
char *dades;
char *dadesEscapped;
```

[1] http://sourceforge.net/projects/dev-cpp/ o http://www.bloodshed.net/devcpp.html

```
outfile = fopen ("MIMECVT.DAT","rb");
long mida;
fseek (outfile,0L,SEEK_END);
mida = ftell(outfile);
fseek (outfile,0,SEEK_SET);
```

Obrim el fitxer amb el contingut ja en binari i hi mirem la mida:

```
dades = (char * ) malloc (mida);
dadesEscapped = (char * ) malloc (mida*2+1);
```

Com que les dades són en binari, el contingut s'ha d'escapar per a que no es trobin caràcters que facin avortar la transmissió. En el cas pitjor, en el qual tots els caràcters s'han d'escapar, el contingut del fitxer es duplica i se li afegeix un caràcter; per això s'ha de reservar mida*2+1.

```
fread (dades,mida,1,outfile);
fclose (outfile);
szSQL = (char * ) malloc (mida*2+500+1);
```

szSQL és la sentència **SQL** a transmetre, hi afegim 500 bytes més per al contingut de la sentència.

```
 mysql_escape_string (dadesEscapped,dades,mida);
```

Escapem les dades.

```
free (dades);
sprintf(szSQL,
    "INSERT INTO imatge VALUES(NULL, \'%s\', \'%s\', SYSDATE(), \'%s\', \'%s\', %d )",
        nomFitxer.c_str(), descripcio.c_str(), dadesEscapped, autor.c_str(),categoria);
mysql_query(myData, szSQL);
```

I finalment creem la sentència **SQL** i l'envíem a la base de Dades.

Aquest procés, tot i no ser 100 % segur, pretén demostrar com es pot resoldre un dels problemes trobats i aconseguir si més no uns resultats semblants a si ho féssim amb WS.

7. Creació del prototipus mòbil per l'ABM Notícies.

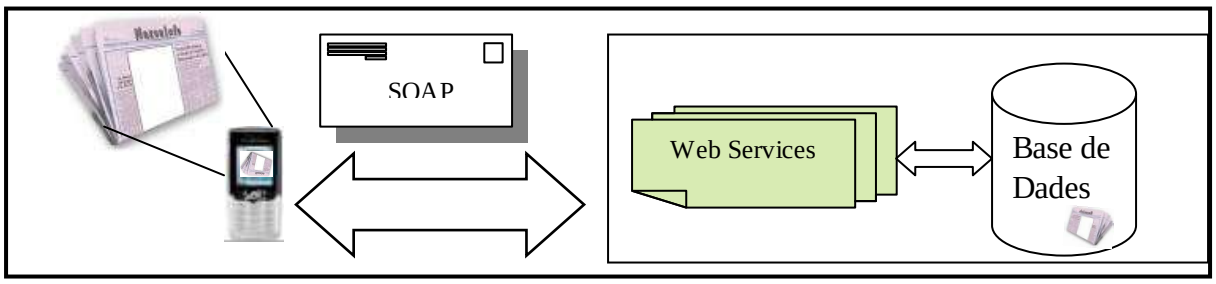
En aquest segon prototipus volem afegir la funcionalitat d'inserir / modificar notícies a través del mòbil. Tot i que pot semblar que la dificultat d'aquesta aplicació és més petita que l'anterior (ja que no necessitem accedir, en principi, a cap capacitat extra del mòbil, com pot ser la càmera), veurem que no és així.

7.1 Introducció i motivació

En aquesta aplicació l'intercanvi de dades és fonamental i per tant s'ha de tenir especial cura en les diferents codificacions de les dades que hi trobem (ISO-8859-1 (ISO-LATIN-1) en J2ME i UTF-8 en Axis). L'aplicació, en quant a flux de formularis i dades, també és més complexa. Per a fer cinc cèntims, els fluxos que hi trobem entre el WS i el dispositiu mòbil són els següents: llista de Plantilles, llista de notícies, comprovació usuari/contrasenya, intercanvi de la notícia amb els seus elements (text i imatges), llista de categories d'imatge i la llista de les imatges de la categoria escollida. Tot això realitzant-ho amb una cura especial cap a l'usuari, ja que l'aplicació ha de seguir un ordre lògic.

La usabilitat és un component important de la programació d'aplicacions per a dispositius amb capacitats visuals i d'entrada limitades, com pot ser un mòbil. Si bé no és un dels objectius realitzar una aplicació amb una usabilitat elevada, si que s'intentarà donar uns consells sobre què s'ha d'intentar fer en una aplicació d'aquestes característiques per a que sigui més o menys usable. Hem de tenir en compte en aquest punt que no tenim un control directe d'on apareixerà un determinat element o en quin lloc es col·locarà l'element X del menú, ja que cada implementació/dispositiu el col·locarà on vulgui i el presentarà a l'usuari de la manera en què la màquina virtual de Java estigui programada per fer-ho. Dit això, la primera regla s'extreu amb facilitat, donada l'experiència que es té amb un altre llenguatge (de marques, per això) amb un comportament similar, l'**HTML**. En una aplicació J2ME hem de dissenyar la interfície gràfica pel seu contingut lògic i no pel físic.

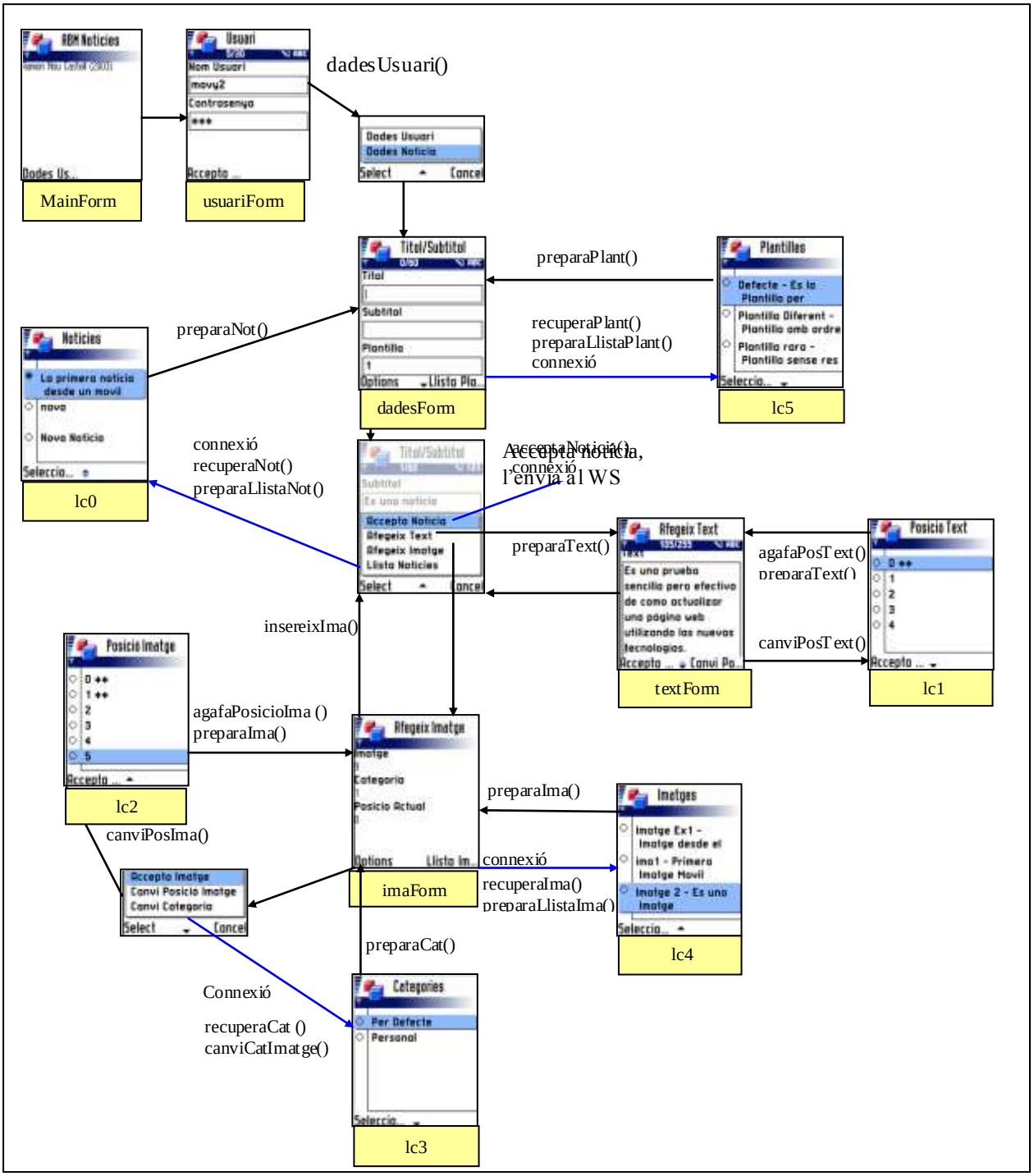
En l'*ABM Notícies* no necessitem cap element multimèdia, per tant no necessitem el MMAPI i podríem escollir qualsevol dels emuladors de l'SDK de Nokia per a testejar. Seguim els mateixos passos per a crear una aplicació base (en aquest cas s'utilitzarà l'SDK de la sèrie 40) per a **MIDP-1.0** (el més compatible). Les classes que hem de copiar al directori classes són les mateixes que l'aplicació anterior, excepte el serialitzador de **Base64**, ja que no ens serà necessària en aquest punt.



Il·lustració 7.1 - Esquema Client Notícies

Comentem ara el flux de l'aplicació que dissenyarem mitjançant les pantalles del prototipus.

7.2 Flux de l'aplicació



Il·lustració 7.2 - Flux ABM Notícies

La connexió amb el servidor mitjançant **GPRS** només es realitza el primer cop, per tal d'estalviar flux de dades. En aquest prototipus no s'inclou la possibilitat d'esborrar o de no assignar una imatge a una posició que ja ha estat assignada, o de canviar de posició un text o una imatge. Per fer-ho s'ha d'escollir primer la posició (si no s'aniran assignant per ordre correlatiu) i afegir posteriorment la imatge o el text.

7.3 Notes d'implementació

En primer lloc necessitem crear les classes de suport i transport de dades; aquesta aplicació inclou les anteriors més les que faltaven per incloure. Cal afegir que en aquest cas utilitzarem les classes per a transmetre dades també (com és el cas de la notícia i els seus elements).

Cal remarcar que en les classes en les quals s'ha de transmetre o enviar informació, s'ha intentat que la informació sigui l'estrictament necessària per a poder identificar l'element que necessitem i escollir-lo. L'èxit de l'elecció, però, dependrà en major part de la descripció o del nom de l'element.

7.3.1 Classes de suport

Com a classe per a rebre (en el cas de voler recuperar una notícia) i enviar la notícia crearem tres classes: la primera **WSInfoNoticia** s'encarregarà de dur la informació general de la notícia (títol, subtítol i la plantilla) així com la llista d'elements de text i d'imatges que conté.
Per a aquest dos elements també necessitem classes de suport per a transmetre'ls (ja que no són dades bàsiques); ho farem amb les classes **WSInfoText** i **WSInfoIma.**

L'aspecte de WSInfoNoticia és el següent: cal destacar l'ús de la classe vector, definida com a VECTOR_CLASS en ElementType

```
public class WSInfoNoticia implements KvmSerializable {
        String títol = null;
        String subtítol = null ;
        int idPlantilla = 1;
        /* Vector d'WSInfoText */
        java.util.Vector llistaT = new java.util.Vector ();
        /* Vector d'WSInfoIma */
        java.util.Vector llistaI = new java.util.Vector ();

        private static int PROP_COUNT = 5;

        private static PropertyInfo PI_títol =
                new PropertyInfo("títol",ElementType.STRING_CLASS);

        private static PropertyInfo PI_subtítol   =
                new PropertyInfo("subtítol",ElementType.STRING_CLASS);

        private static PropertyInfo PI_idPlantilla   =
                new PropertyInfo("idPlantilla",ElementType.INTEGER_CLASS);

        private static PropertyInfo PI_llistaT   =
                new PropertyInfo("llistaT",ElementType.VECTOR_CLASS);
        private static PropertyInfo PI_llistaI   =
                new PropertyInfo("llistaI",ElementType.VECTOR_CLASS);
        private static PropertyInfo[] PI_PROP_ARRAY =
                {PI_títol,PI_subtítol,PI_idPlantilla,PI_llistaT,PI_llistaI};
```

```
        public WSInfoNoticia () { }
        ...

        public Object getProperty(int param) {
         if ( param == 0 ) {
               return (getTitol());
         } else if ( param == 1 ) {
                return getSubtitol();
         } else if ( param == 2 ) {
                return ( new Integer(getIdPlantilla()));
         } else if ( param == 3 ) {
                return ( getLlistaT());
         }
           else if ( param == 4 ) {
                return ( getLlistaI());
         }
         else { return null; }
    }

    public void setProperty(int param, Object obj) {
         if ( param == 0 ) {
             setTitol((String) obj );
         } else if ( param == 1 ) {
             setSubtitol((String) obj );
         } else if ( param == 2 ) {
             setIdPlantilla( ((Integer)obj).intValue() );
         } else if ( param == 3 ) {
             setLlistaT( (java.util.Vector) obj );
         } else if ( param == 4 ) {
             setLlistaI( (java.util.Vector) obj );
         } else {
         }
    }
}
```

Cal dir que aquestes classes encara no són definitives, ja que es modificaran per afegir una solució a un error que trobarem més endavant, però s'ha escollit fer-ho així per a poder veure l'ordre natural de construcció de l'aplicació.

Les altres classes es poden trobar al CD-ROM que acompanya aquest document.

Una vegada tenim aquestes classes construïdes podem començar a dissenyar l'aplicació. Per fer-ho continuarem amb les mateixes premisses per a aconseguir un baix ús de memòria i recursos, és a dir, utilitzar el mínim de classes possibles i dissenyar els formularis que apareixen en pantalla un cop i reutilitzar-los.

A continuació mostrarem algunes parts del codi de l'aplicació.

7.3.2 Variables Globals

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;

import org.ksoap.SoapObject;
import org.ksoap.transport.HttpTransport;
import org.ksoap.ClassMap;

public class SoapTest extends MIDlet implements Runnable, CommandListener
{
          Form mainForm = new Form("ABM Noticies");
          Form usuariForm = new Form("Usuari");
          Form dadesForm = new Form("Títol/Subtítol");
          Form textForm = new Form("Afegeix Text");
          Form imaForm = new Form("Afegeix Imatge");
```

En aquesta aplicació tenim 5 formularis, com podem observar en el flux d'operació presentat anteriorment. Per tant creem cinc formularis per a dissenyar cadascuna de les pantalles.

```
        TextField usuariField = new TextField("Nom Usuari", nomUsuari, 30, TextField.ANY);
        TextField contrasenyaField = new TextField("Contrasenya", contraUsuari, 8,
TextField.PASSWORD);
        TextField titolField = new TextField("Títol", "", 60, TextField.ANY);
        TextField subtitolField = new TextField("Subtítol", "", 60, TextField.ANY);
        TextField plantillaField = new TextField("Plantilla", "1", 60, TextField.NUMERIC);
        TextField textField = new TextField("Text", "", 255, TextField.ANY);
```

Aquests TextField's són els que incorporaran l'entrada de dades per part de l'usuari, en total són sis: nom d'usuari, contrasenya, títol i subtítol de la notícia, el numero de la plantilla (que té com a tipus d'entrada TextField.**NUMERIC**) i finalment el d'entrada del text de la notícia (que hem considerat de 255 caràcters per bloc).

```
        StringItem posicioT = new StringItem("Posicio Actual", new
Integer(posicioText).toString());
        ...
```

Els StringItems contenen la informació de sortida de l'aplicació, és a dir, tots els missatges. Entre els missatges hi ha la informació de la imatge seleccionada, la posició, o la categoria, entre d'altres. La funcionalitat d'assignar una etiqueta a aquests StringItems correspon a una de les característiques de J2ME, d'aquesta manera sempre podem identificar una etiqueta sigui on sigui: J2ME la podria col·locar en qualsevol lloc (imaginem per exemple un display de dues línies de text).

```
        javax.microedition.lcdui.List lc0 = new javax.microedition.lcdui.List("Noticies",
javax.microedition.lcdui.List.EXCLUSIVE);
        javax.microedition.lcdui.List lc1 = new javax.microedition.lcdui.List("Posició Text",
javax.microedition.lcdui.List.EXCLUSIVE);
        ...
```

Per acabar amb els elements de la interfície gràfica tenim les llistes (recordem que són tractats per J2ME com a formularis); en aquesta aplicació en necessitem sis que enumerarem a continuació:

lc0 per a la llista de Notícies rebudes.

lc1 per a la llista de les posicions plenes/buides dels elements de text.

lc2 per a la llista de posicions plenes/buides dels elements gràfics.

lc3 per a la llista de les categories de les imatges rebudes.

lc4 per a la llista d'imatges de la categoria escollida.

I finalment ***lc5*** per a la llista de plantilles disponibles.

```
    WSInfoNoticia noticia = new WSInfoNoticia();
```

Finalment noticia és la notícia que estem redactant o modificant.

```
public SoapTest() {
      Command getCommand = new Command("Dades Usuari", Command.SCREEN, 0);
      StringItem copyr = new StringItem("", "Ramon Nou Castell (2003)");
      mainForm.addCommand(getCommand);
      mainForm.append(copyr);
      mainForm.setCommandListener(this);
```

A l'inicialitzar l'aplicació creem tots els formularis; en primer lloc hem creat el formulari principal (**mainForm**) afegint la comanda Dades Usuari i també la informació personal com a un StringItem. Command.**SCREEN** significa que s'assigna al botó que es fa servir normalment per a acceptar la comanda (si hi ha més d'una apareixerà un menú amb l'ordre definit pel número indicat (0 = sense preferència)).

```
      Command acceptaUsuari = new Command("Accepta Usuari", Command.SCREEN, 0);
      usuariForm.append (usuariField);
      usuariForm.append (contrasenyaField);
      usuariForm.addCommand(acceptaUsuari);
      usuariForm.setCommandListener(this);
```

Definim a continuació el formulari de l'usuari, és a dir, el que introduïm el nom de l'usuari i la contrasenya (o apareix si ha estat introduïda al .jad), i afegim també la comanda del menú Accepta Usuari.

```
      dadesForm.append (titolField);
      dadesForm.append (subtitolField);
      dadesForm.append (plantillaField);
      dadesForm.append (errorN);
      Command acceptaNoticia = new Command("Accepta Notícia", Command.SCREEN, 0);
      dadesForm.addCommand(acceptaNoticia);
      acceptaNoticia = new Command("Afegeix Text", Command.SCREEN, 0);
      dadesForm.addCommand(acceptaNoticia);
      acceptaNoticia = new Command("Afegeix Imatge", Command.SCREEN, 0);
      dadesForm.addCommand(acceptaNoticia);
      ...
```

El formulari que podem anomenar eix de l'aplicació és el de **títol/subtítol;** en ell introduirem o ens introduiran el títol de la notícia, el subtítol i l'identificador de la plantilla. En aquesta pantalla la part més important és crear una interfície correcta per a navegar per les diferents opcions (cinc opcions per a escollir).

Les col·locarem seguint la següent lògica: totes les comandes que necessiten connexió amb el WS les col·locarem al botó de cancel·lar (concretament seria el que no és per acceptar), d'aquesta manera trobem a **Command.BACK** l'accés a la Llista de Notícies i a

la Llista de Plantilles (de fet, tot i que l'hem definit així, Llista Plantilles per alguna estranya raó apareix a **Command.SCREEN**). Finalment posarem al botó d'acceptar la resta d'opcions (*Accepta la Noticia, Afegir Text i Afegir Imatges*). Cal remarcar que reutilitzem Command, ja que identifiquem les comandes amb un nom diferent cadascuna i per tant podem reutilitzar la variable. Si no ho féssim així hauríem d'utilitzar la variable per a identificar que s'ha produït l'event corresponent.

```
       Command acceptaText = new Command("Accepta Text", Command.SCREEN, 0);
       textForm.addCommand(acceptaText);
       textForm.append (textField);
       textForm.append (posicioT);
       acceptaText = new Command("Canvi Posició Text", Command.BACK, 0);
       textForm.addCommand(acceptaText);
       textForm.setCommandListener(this);
```

El formulari per a afegir text conté un *TextField* gran i un indicador de posició; com a menú col·loquem al botó de no acceptar el de canvi de posició, i al d'acceptar el de validar les dades i retornar al menú anterior.

Els altres mètodes i parts del codi són similars: commandAction conté la lògica de l'aplicació, és a dir, controla quins formularis i mètodes s'executen al realitzar una acció. Anem a comentar algunes de les accions tipus que podem trobar:

```
public void commandAction( Command p1, Displayable p2 ){
   if (p1.getLabel().equals ("Accepta Usuari")) {
       dadesUsuari ();
       Display.getDisplay(this).setCurrent(mainForm);
   }
   if (p1.getLabel().equals ("Dades Usuari"))    {
       Display.getDisplay(this).setCurrent(usuariForm);
   }
```

Quan acceptem les dades de l'usuari (***Accepta Usuari***) cridem al mètode dadesUsuari que s'encarrega de validar-les (sense connectar-se al WS) i permet seguir el flux de l'aplicació. Si volem tornar a introduir les dades de l'usuari (polsem sobre Dades Usuari) l'aplicació canvia el formulari actiu pel de l'usuari (**usuariForm**), que només té disponible l'opció Accepta Usuari.

7.3.3 Mètode dadesUsuari

```
public void dadesUsuari (){
       nomUsuari = usuariField.getString();
       contraUsuari = contrasenyaField.getString();
       if (dUsuari == false) {
          Command getCommand = new Command("Dades Notícia", Command.SCREEN, 0);
          mainForm.addCommand(getCommand);
          dUsuari = true;
       }
}
```

Observem que dUsuari es posa a true per tal d'indicar-nos que la comanda ***Dades Notícia*** ha estat activada. Les dades de l'usuari es guarden en les variables nomUsuari i contraUsuari.

```
   if (p1.getLabel().equals ("Dades Notícia"))  {
       Display.getDisplay(this).setCurrent(dadesForm);
   }
   if (p1.getLabel().equals ("Accepta Notícia"))   {
       if (acceptaNoticia()==true)   {
           /* Envia ok */
           Display.getDisplay(this).setCurrent(mainForm);
       }
       else   {
           Display.getDisplay(this).setCurrent(dadesForm); }
    }
```

El flux següent que tenim disponible (ara estem al formulari de *dadesForm*) és el d'acceptar la notícia. Aquesta opció crida al mètode **acceptaNoticia**, que s'encarrega de validar si s'ha introduït el títol (condició mínima per a acceptar una notícia) i realitza l'enviament amb la contrasenya i l'usuari corresponent (juntament amb tots els elements de text i imatges introduïts). Si es produeix qualsevol error el flux de l'aplicació retorna al formulari de dades, i si tot ha anat correctament tornem a la pantalla principal.

7.3.4 Mètode acceptaNotícia

```
public boolean acceptaNoticia () {
   if  (titolField.getString() != "")
```

Si no especifiquem un títol retornem error.

```
 {
   noticia.setTitol(titolField.getString());
   noticia.setSubtitol(subtitolField.getString());
   ...
   ClassMap classMap = new ClassMap();
   classMap.addMapping("http://WSImatge", "WSInfoNoticia",   new WSInfoNoticia().getClass() );
   classMap.addMapping("http://WSImatge", "WSInfoText",   new WSInfoText().getClass() );
   classMap.addMapping("http://WSImatge", "WSInfoIma",   new WSInfoIma().getClass() );
   classMap.addMapping("http://xml.apache.org/xml-soap", "Vector",   new
java.util.Vector().getClass() );
```

Definim tots els mapejos necessaris per a la transferència de la Notícia a través de kSOAP.

```
   SoapObject rpc = new SoapObject("WSImatge","insNoticia");
   rpc.addProperty("wN", noticia);
   rpc.addProperty("nomAutor", new String (nomUsuari.getBytes("UTF-8")));
   rpc.addProperty("contrasenya", new String (contraUsuari.getBytes("UTF-8")));
   rpc.addProperty("idNoticia", new Integer (idNoticia));
```

Per acabar, afegim al missatge SOAP els paràmetres de la funció remota insNoticia, és a dir, la noticia (enviem la classe **WSInfoNoticia**), el nom de l'autor, la contrasenya i el id (per si és una modificació).

```
   HttpTransport ht = new HttpTransport("http://"+urlWS+"/axis/services/WSImatge","WSImatge");
```

```
    ht.debug = true;
    ht.setClassMap( classMap );
```

Assignem el mapeig de classes al transport **http.**

```
    ht.call(rpc);
```

I finalment l'enviem.

Cal remarcar la possibilitat d'enviar tota la classe noticia sencera (recordem que aquesta classe té un vector de WSInfoText i de WSInfoIma).

```
if (p1.getLabel().equals ("Accepta Text")) {
     insereixText();
     Display.getDisplay(this).setCurrent(dadesForm);
}
if (p1.getLabel().equals ("Afegeix Text")) {
     preparaText();
     Display.getDisplay(this).setCurrent(textForm); }
```

El següent menú és el d'afegir text; quan accedim a *Afegeix Text* es crida al mètode **preparaText** encarregat d'incrementar l'indicador de la posició actual del text (recordem que si no l'especifiquem és correlatiu) i d'omplir el *textField* amb el contingut del text (s'itera pel vector de *WSInfoText* cercant la posició actual), i s'assigna el contingut al contenidor.

7.3.5 Mètode preparaText

```
public void preparaText() {
    java.util.Vector llistaT = noticia.getLlistaT();
    WSInfoText wIT = new WSInfoText ();
    boolean modifica = false;
    if ((posicioText+1) > 4) posicioText = 0;
    else posicioText++;
    for (java.util.Enumeration e = llistaT.elements(); e.hasMoreElements () ;)    {
       wIT = (WSInfoText) e.nextElement();
       if (wIT.getPosicio() == posicioText ) {
           modifica = true;
           break;
       }
    }
    if (modifica == true)
       textField.setString(wIT.getText());
    else
       textField.setString("");

    posicioT.setText(new Integer(posicioText).toString());
}
       ...
    if (p1.getLabel().equals ("Llista Imatges")) {
        if ( loadIma == true ) llistaI = recuperaIma(catIma);
       loadIma = false;
       preparaLlistaIma();
```

```
        Display.getDisplay(this).setCurrent(lc4);
        }
    if (p1.getLabel().equals ("Selecciona Imatge")) {
        preparaImatge();
        Display.getDisplay(this).setCurrent(imaForm);
        }
```

Recuperar la llista d'imatges es realitza en dues passades: en primer lloc es crida al procediment remot per a omplir la llista de les imatges si és necessari (primera crida, o canvi de categoria); finalment es prepara la llista (**lc4**) per a mostrar-la per pantalla.

El mètode **recuperaIma (categoria)** recupera la llista de les imatges de la categoria especificada. Per a fer-ho segueix el procediment habitual d'afegir el mapeig correcte (classe **Vector** i classe **WSInfoImatge**), cridar al mètode corresponent del WS getImaCat, enviar els paràmetres i rebre el resultat. Una vegada amb la llista plena s'assigna el contingut al formulari ***lc4*** (esborrant el contingut antic) i s'afegeix la informació corresponent.

Cal tenir en compte que a una llista (que és un formulari) també hi hem d'afegir una comanda de menú (*Selecciona Imatge*), i per tant necessitem un booleà per tal d'evitar que s'introdueixi més d'un cop.

Finalment, quan s'accepta l'ítem de la llista es crida al mètode **preparaImatge** per a agafar l'índex seleccionat, cercar a la llista corresponent (que tindrà el mateix ordre) i finalment afegir l'id de la imatge a la cadena corresponent.

7.3.6 Mètode preparaImatge

```
public void preparaImatge (){
     int num;
     num = lc4.getSelectedIndex();
     idImatge = ( (WSInfoImatge) llistaI.elementAt ( num )).getIdImatge();
     imatge.setText (new Integer (idImatge ).toString());
}
```

7.3.7 Mètode insereixText

El mètode **insereixText** () es crida quan s'accepta el text.

```
public void insereixText () {
        java.util.Vector llistaT = noticia.getLlistaT();
        boolean modifica = false;
        int pos = 0;
        WSInfoText wIT = new WSInfoText ();
        for (java.util.Enumeration e = llistaT.elements(); e.hasMoreElements () ;) {
             wIT = (WSInfoText) e.nextElement();
             if (wIT.getPosicio() == posicioText ) { modifica = true;  break; }
             pos++;
        }
```

En primer lloc cerquem al vector si hi ha algun text a la mateixa posició ja inserit (per tant és una modificació) i actualitzem la variable modifica.

```
        if (modifica == true) {
```

```
        try {
                wIT.setText(textField.getString());
        } catch (Exception e) {;}
        llistaT.setElementAt (wIT,pos);
```

En el cas que s'hagi de modificar assignem el text a l'element de la llista corresponent.

```
        }
        else   {
            wIT = new WSInfoText ();
            try {
                wIT.setText(textField.getString());
                } catch (Exception e) {;}
             wIT.setPosicio (posicioText);
             llistaT.addElement (wIT);
```

Sinó, simplement afegim l'element a la llista.

Quan volem canviar la posició de l'element de text o de l'element d'imatge es realitza una cerca en el vector d'imatge/text, per tal de trobar les posicions ocupades o lliures. Finalment, les mostra en una llista assenyalant les ocupades amb dos asteriscos.

7.3.8 Mètode demanaNoticia

Per acabar mostrarem el codi del mètode encarregat d'obtenir una noticia (text i imatge) del WS. Observem que el procediment retorna un **WSInfoNoticia**.

```
public WSInfoNoticia demanaNoticia (int id) {
WSInfoNoticia WSIN = new WSInfoNoticia();
   ClassMap classMap = new ClassMap();
   classMap.addMapping("http://WSImatge", "WSInfoNoticia",   new WSInfoNoticia().getClass() );
   classMap.addMapping("http://WSImatge", "WSInfoText",   new WSInfoText().getClass() );
   classMap.addMapping("http://WSImatge", "WSInfoIma",   new WSInfoIma().getClass() );
   classMap.addMapping("http://xml.apache.org/xml-soap",      "Vector",                  new
java.util.Vector().getClass() );
   SoapObject rpc = new SoapObject("WSImatge","recNoticia");
   rpc.addProperty("idNoticia", new Integer (id));
   HttpTransport ht = new HttpTransport("http://"+urlWS+"/axis/services/WSImatge","WSImatge");
   ht.debug = true;
   ht.setClassMap( classMap );
   try {
       WSIN = (WSInfoNoticia)(ht.call(rpc));
       return (WSIN);
    }
```

La resta de codi és similar i la seva consulta es pot fer a través del CD-ROM que acompanya aquest document.

7.4 Proves i primers errors.

Al realitzar els primers tests l'aplicació responia correctament. Però hi havia un error en principi difícil de detectar. Quan s'escrivia text amb accents o caràcters fora de l'estàndard, o bé Axis o bé l'aplicació mòbil treien errors.

La intuïció deia que era pel joc de caràcters, així que es va realitzar una cerca en aquest camp; en primer lloc es va trobar que Axis enviava per defecte **UTF-8**. Tot i que ho podíem canviar no ens interessava, ja que per operar amb altres clients i per comprovar l'adaptació del terminal mòbil a aquesta situació ens interessa que Axis fos el més estàndard possible. Per tant es rebutja la possibilitat de canviar la codificació d'Axis.

Posteriorment es van trobar alguns mètodes (al SDK de J2ME) per a conèixer la codificació que utilitza el terminal mòbil i la codificació que s'utilitzava va ser ISO-5589-1 és a dir **ISO – LATIN – 1**.

No costa gaire veure que no tenim, en principi, cap problema per a convertir de ISO – LATIN -1 a UTF-8, ja que és un petit subconjunt, però si que tenim problemes per a fer la tasca contraria. Hi ha caràcters a **UTF-8** que són irrepresentables a ISO-LATIN-1.

7.4.1 De ISO-LATIN-1 a UTF-8

La primera part del problema (passar de ISO-LATIN-1 a UTF-8) la podem resoldre a l'enviar dades a través de ***kSOAP*** o a les classes de suport mitjançant una petita conversió que s'encarrega de fer la classe String de J2ME.

Anem a veure-ho en el mètode getProperty de la classe ***WSInfoText*** :

```
public Object getProperty(int param) {
  if ( param == 0 ) {
    try {
       return ( new String (getText().getBytes("UTF-8")));
    } catch (Exception e) {;}
    return (getText());
  } else if ( param == 1 ) {
     return (new Integer (getPosicio()));
  }
  else {
    return null;
  }
}
```

Anem a explicar què fem: agafem l'String guardat en ISO-LATIN-1 mitjançant el mètode getText() de la classe WSInfoText, aconseguim el seu flux de bytes transformat amb la codificació UTF-8, i posem el resultat en un nou String que és el que retornarem quan el serialitzador ho demani.

Els resultats en aquesta part són satisfactoris, tots els caràcters que escrivim es transmeten i visualitzen correctament a través del PC. Bé, no tots, el símbol de l'Euro no es transforma correctament; analitzant la seqüència de bytes transmesos mitjançant un

sniffer trobem que la codificació que s'envia no correspon al símbol de l'euro. Pot ser un problema d'estandardització del mòbil o de la màquina virtual de J2ME.

7.4.2 De UTF-8 a ISO-LATIN-1

El procés contrari s'ha intentat realitzar amb una sentència similar a l'anterior, però **J2ME** no realitza la conversió, tot i que el joc de caràcters utilitzat pertany al subconjunt ISO-LATIN-1. Per tant, s'ha decidit crear una classe de suport que realitzi la conversió.

Per a crear-la s'ha obtingut la taula de caràcters que pertanyen a ISO-LATIN-1 i la seva codificació en UTF-8, i s'ha realitzat un mètode que realitza el mapeig invers. Quan es troba un caràcter extra a **UTF-8** es codifica com a dos caràcters. Els que pertanyen a **ISO-LATIN-1** tenen com a primer caràcter els codis { *0xC2, 0xC3, 0xCB* }. Una vegada hem trobat aquest caràcter agafem el següent i descodifiquem el caràcter corresponent utilitzant la taula. Mostrem el codi construït per a realitzar aquesta tasca a continuació:

```
public class UTF2ISO  {
       private static char mapping (int tipo,int valor)    {
           char a = '0';
              if (tipo == 0xC3) {
                     switch (valor) {
                            case 0x80 : a = 'À';break;
                            case 0x82 : a = 'Â';break;
                            case 0x8A : a = 'Ê';break;
                            case 0x8B : a = 'Ë';break;
                            case 0x88 : a = 'È';break;
                            case 0x8D : a = 'Í';break;
                            case 0x8E : a = 'Î';break;
                            case 0x8F : a = 'Ï';break;
                            ...
                            case 0xB9 : a = 'ù';break;
                            case 0xBB : a = 'û';break;
                            case 0xBC : a = 'ü';break;
                            case 0x86 : a = 'Æ';break;
                            case 0x98 : a = 'Ø';break;
                            case 0xA6 : a = 'æ';break;
                            case 0xB8 : a = 'ø';break;
                     }
                     return (a);
              }
              if (tipo == 0xC2){
                     switch (valor) {
                            case 0x80 : a = '€';break;
                            case 0xB0 : a = '°';break;
                            case 0xA2 : a = '¢';break;
                            case 0xA3 : a = '£';break;
                            case 0xA7 : a = '§';break;
```

```
                        case 0xB6 : a = '¶';break;
                        case 0xAE : a = '®';break;
                        case 0xA9 : a = '©';break;
                        ...
                    }
                }
                if (tipo == 0xCB){
                        switch (valor) {
                        case 0x86 : a = '^';break;
                        case 0x9c : a = '˜';break;
                    }
                }
                return (a);
        }
        public static
        String conversion (String a)  {
           String b = new String ();
           int aL = a.length();

           for (int i=0;i<aL;i++){
               if (a.charAt(i) != 0xC3
                   && a.charAt(i) != 0xC2
                   &&  a.charAt(i) != 0xCB)
                  b += a.charAt(i);
               else    {
                  int actual = a.charAt(i);
                  int siguiente = a.charAt(i+1);
                  i++;
                  b += mapping(actual,siguiente);
               }
           }
        return (b);
        }
}
```

7.4.3 Canvis en la interfície gràfica

S'ha detectat també una sèrie d'errors quant a la presentació de la interfície gràfica en l'emulador **Nokia**. En aquest emulador les comandes del menú que havíem especificat que es posessin en el botó dret (el de cancel·lar o tornar a enrere) es posaven igualment al botó esquerre (el d'acceptar). La prova de l'aplicació en un terminal **Sony** ha donat en canvi els resultats esperats. Tot i això, el llibre **J2ME in a Nutshell** ja indica que és perfectament normal que això passi.

8. Ampliació i finalització del sistema.

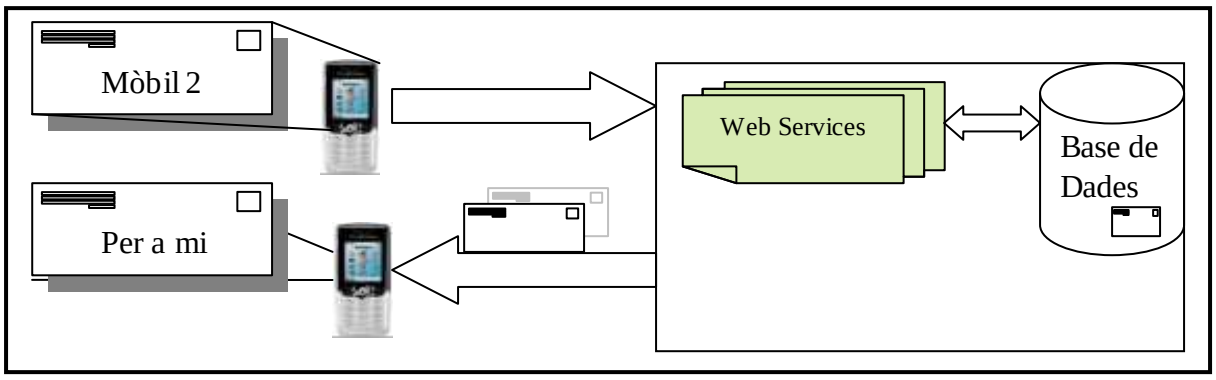
Hem finalitzat el sistema afegint la funcionalitat de missatgeria, així com algunes funcions en el prototipus de notícies, com són esborrar notícies. Hem aprofitat per introduir la funcionalitat de poder especificar l'URL del servidor *Tomcat/Axis* des de dins del Midlet i una millora en la recuperació davant dels errors de xarxa (que es van detectat en la fase de proves). D'aquesta manera, si la xarxa cau (pèrdua de connexió, cobertura, etc.) en la majoria dels casos obtindrem un formulari buit i podrem continuar.

Durant la fase de proves s'ha detectat que si tenim l'enviament del missatge **SOAP** amb el mode debug activat (***httpTransport***), hi ha alguns missatges que no arriben a sortir del mòbil i s'activa una excepció. Per tant, en l'etapa de producció hem col·locat *ht.debug = false;* per a estalviar memòria.

En aquesta part també s'ha provat l'aplicació exhaustivament des d'un viatge a l'estranger, i s'han incorporat algunes millores per tal de fer el seu ús més senzill.

8.1 Creació de l'AB missatges

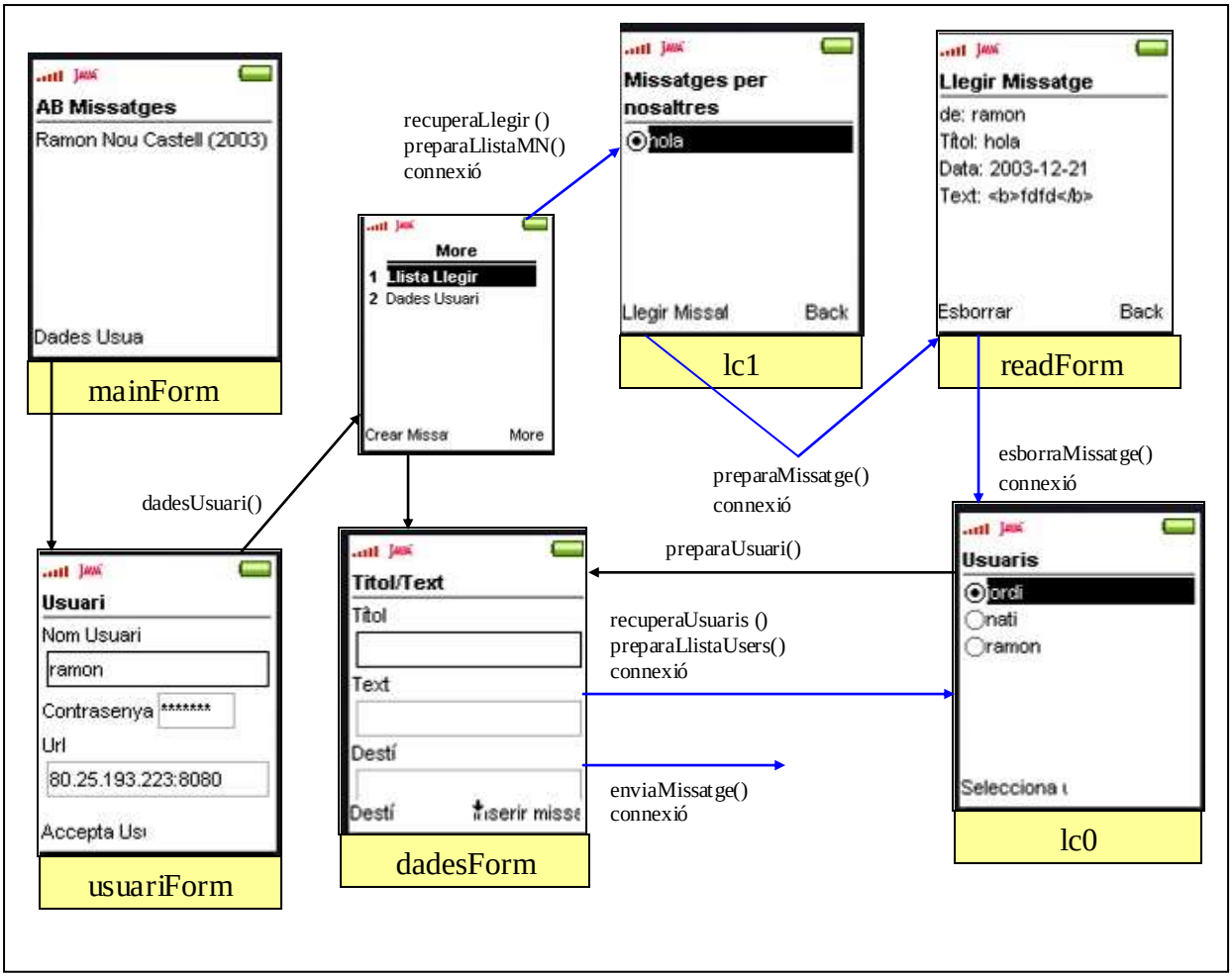
Finalment, hem creat l'aplicació per a poder llegir, crear i esborrar missatges del servidor. Per fer-ho hem seguit la mateixa estructura que la resta d'aplicacions que hem creat. En aquesta aplicació hem introduït algunes millores (que també apareixeran en ABM Notícies), com són la possibilitat de canviar l'URL des de l'aplicació i un control més ampli dels errors produïts per fallades de xarxa.



Il·lustració 8.1 - Esquema Client Missatges

8.2 Flux de l'aplicació.

Mostrarem a continuació el flux de l'aplicació; per fer-ho utilitzarem l'emulador de J2ME de Sony.



Il·lustració 8.2- Flux AB Missatges

8.3 Parts rellevants.

Aquest Midlet no conté cap part rellevant sobre les altres aplicacions, per tant no mostrarem cap part del codi.

8.4 Proves de l'aplicació.

En les proves hem testejat el límit dels textField i la transmissió de gran quantitat de text, hem trobat que tot i que podem especificar un nombre molt gran en el **TextField**, depenent del perfil **MIDP** i de la màquina virtual de **kVM**, tindrem més o menys espai disponible per al TextField. En l'emulador d'un *Nokia 7100* tenim disponibles 900 caràcters per TextField.

Les proves van ser satisfactòries en tots els casos d'ús. En el pròxim capítol mostrem les proves de rendiment i de transmissió de dades que hem realitzat.

8.5 Montatge del client final

Aprofitant que en un .jar podem col·locar més d'una aplicació, hem fusionat els dos codis (***ABM Notícies*** i ***AB Missatges***) en un directori, l'hem compilat i finalment hem creat el paquet. A més, hem afegit dos icones en format *.png* per a que apareguin en el menú de selecció d'aplicació. Tot això ho hem pogut fer de forma senzilla des de l'SDK de **Nokia**, en l'apartat de *Create Application Package,* en la pestanya Resources i Midlets.

Amb això aconseguim que en poc espai tinguem les dues aplicacions juntes (63 Kbytes), ja que les classes que pertanyen a ***kSOAP*** estan compartides.

Il·lustració 8.3 - Menú de selecció

9. Rendiment de l'aplicació.

Per a analitzar el rendiment de l'aplicació hem pogut obtenir un mòbil **Sony Ericsson T610**. A part de poder comprovar que l'aplicació és totalment compatible sense recompil·lar, l'SDK de Sony-Ericsson inclou unes funcionalitats extres que el fan molt útil per a mesurar el consum de memòria i el consum de pila en l'aplicació corrent al dispositiu real.

També inclou la funcionalitat de permetre utilitzar la xarxa del PC per a fer les transferències que es realitzen per **GPRS** normalment.

9.1 Mesures

Aquestes mesures no pretenen ser exhaustives, ja que el programari no permet fer gaires proves ni dóna moltes facilitats per fer-ho. En aquestes proves s'ha utilitzat un JAR/JAD de l'aplicació compilada (*sense ofuscar*[1]) que ocupa uns 61 kb. Per a la realització de la taula s'ha utilitzat la mitja de 10 execucions amb els mateixos paràmetres.

9.1.1 Memòria

	Sony (T. Real)	**Nokia (Emulador)**
	Lliure (Utilitzada)	*Lliure (Utilitzada)*
Mòbil net.	262144 (0)	215040 (0)
Engeguem	218088 (44056)	163724 (51316)
Introduïm nom usuari	216928 (45216)	160112 (54928)
Pantalla Dades Notícia	216444 (45700)	156388 (58652)
Crida a llista Plantilles (3)	126016 (136128)	61256 (153784)
Crida a llista Noticies (7)	109064 (153080)	48000 (167040)
Recuperació d'una noticia	7340 (254804)	70600 (144440)
Introducció de text	168332 (93812)(l'aplicació recupera memòria)	23288 (191752)

Taula 9.1- Memòria utilitzada ABM Notícies

La taula anterior consisteix en petites mesures de la memòria disponible pel procés (pila) durant diferents fases de la seva execució. Observem, per exemple, que hi ha un moment crític quan es recupera una notícia, ja que la memòria baixa molt. De totes maneres, l'observació en la mateixa execució de la recuperació de memòria per part de kVM a l'introduir més text a la notícia, dóna a entendre que tot i que la memòria és escassa, quan el **Garbage Collector** fa la seva feina es recupera en bon grau. Cal recordar que aquestes mesures es realitzen amb mode debug (o alguna cosa que s'hi assembla) i per tant els consums i el funcionament del Garbage Collector no es poden assegurar que siguin reals.
Si es coneguès alguna cosa més del funcionament d'aquest mode, segurament podríem aconseguir estimadors per fer aquestes mesures més acurades.

[1] En podem trobar un a http://sourceforge.net/projects/jarg/

9.1.2 Dades transmeses/rebudes

En aquest apartat utilitzarem l'emulador del Nokia 7210. Aquest emulador permet veure la quantitat de dades que es transmeten per la xarxa; d'aquesta manera podem realitzar càlculs d'overhead, per exemple. Donat que l'emulador de la sèrie 60 no ens permet realitzar aquest tipus de mesures, l'aplicació Camera no es pot analitzar.

AB Missatges

	Destinataris (2)	**Enviar**	**Total**
Creació missatge 900 caràcters.	Mòbil -> PC : 622 bytes Mòbil <- PC : 768 bytes	Mòbil -> PC : 1919 bytes Mòbil <- PC : 649 bytes	M -> PC : 2541 bytes M <- PC : 1417 bytes
	Llista Missatges (1)	**Rebre**	**Total**
Rebre missatge 900 caràcters.	Mòbil -> PC : 788 bytes Mòbil <- PC : 879 bytes	Mòbil -> PC : 809 bytes Mòbil <- PC : 1915 bytes	M -> PC : 1597 bytes M <- PC : 2794 bytes

Taula 9.2 - Tràfic en AB Missatges

El cost d'enviament del missatge és elevat, donat que no es poden transmetre les dades amb una codificació ASCII estàndard, i per tant hi ha un overhead important en aquest punt.

ABM Notícies

	Llista Plantilles (1)	**Categoria(1)**	**Imatge (3)**
Creació Notícia. (300 caràcters)	M -> PC : 653 bytes M <- PC : 961 bytes	M -> PC : 650 bytes M <- PC : 862 bytes	M -> PC : 725 bytes M <- PC : 1421 bytes

Taula 9.3 - Tràfic preliminar ABM Notícia

	Enviament	**Total**
Creació Notícia. (300 caràcters)	M -> PC : 1787 bytes M <- PC : 645 bytes	M -> PC : 3815 bytes M <- PC : 3889 bytes

Taula 9.4 - Tràfic per a enviar una notícia

	Llista Notícies (3)	**Rebre**	**Total**
Modificar Notícia. (300 caràcters)	M -> PC : 816 bytes M <- PC : 1235 bytes	M -> PC : 723 bytes M <- PC : 1600 bytes	M -> PC : 1539 bytes M <- PC : 2835 bytes

Taula 9.5 -Tràfic per a rebre una notícia

Per tant, estem en una mitjana d'uns 800 bytes per petició (sense informació). En una tarifa normal GPRS (nacional) la transferència costaria uns 0,04 € (1,89 € per a 80 Kbytes). El cost d'ús per a un usuari és elevat, per això necessitem comprar "abonaments GPRS" de 1 Mb de transferència al mes per exemple per 6 €, quantitat

suficient per a un elevat ús del sistema. Cal remarcar que en les proves realitzades del sistema a l'estranger el cost de GPRS va ser molt elevat per qüestions de roaming.

9.2 Ofuscació del codi

Tot i que la utilitat d'un ofuscador de codi Java és en un primer moment la seguretat (ja que el fa més difícil desensamblar-lo), en un entorn J2ME en el que la memòria és crítica la seva utilitat és ben diferent.

Ofuscant un jar, podem reduir el nom de les classes i els mètodes, i estalviar d'aquesta manera memòria que podem utilitzar per a altres tasques (o pel propi procés).

En una de les proves que s'ha realitzat d'ofuscació, un .jar que ocupava 61 Kb ha passat a ocupar 50 Kb utilitzant l'ofuscador **jarg**[1]. Observem a continuació el nivell de compressió que s'aconsegueix en cada classe:

```
CameraCanvas.class (75%)
SoapTest.class (69%)
UTF2ISO.class (57%)
WSInfoCategoria.class (82%)
WSInfoIma.class (81%)
WSInfoImatge.class (81%)
WSInfoNotDesc.class (82%)
WSInfoNoticia.class (79%)
WSInfoPlantilla.class (81%)
WSInfoText.class (82%)
WSInfoUsuari.class (80%)
org/kobjects/serialization/ElementType.class (85%)
...

[Statistics]
                 OLD :      NEW :  REDUCED :    RATIO :
  class sz :   115596 :    91781 :    23815 :       79%
  class  # :       42 :       42 :        0 :      100%
  cp size  :    60318 :    56420 :     3898 :       93%
  cp entr# :     5324 :     5120 :      204 :       96%
  field  # :      258 :      258 :        0 :      100%
  method # :      364 :      364 :        0 :      100%
  bcode sz :    23124 :    22693 :      431 :       98%
  bcode  # :    10881 :    10733 :      148 :       98%
```

[1] El podem trobar a http://sourceforge.net/projects/jarg/

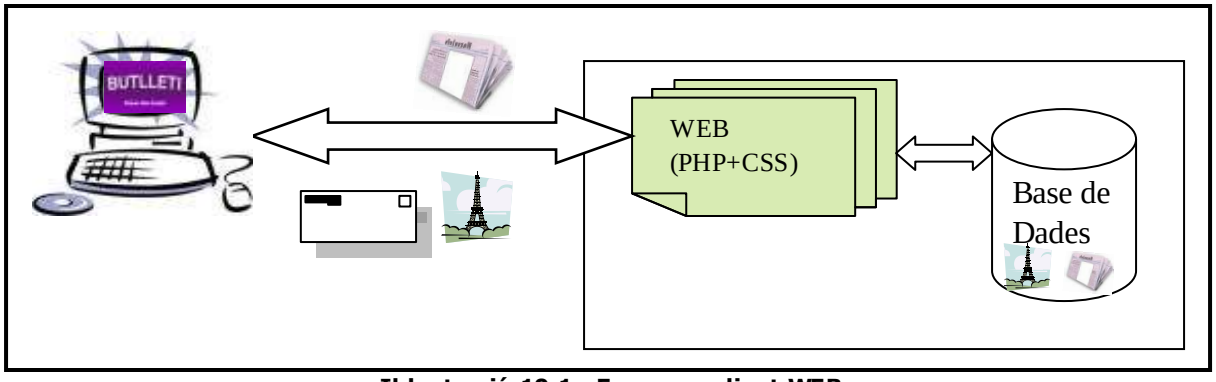
10. Client WEB

10.1 Introducció

El client WEB, tot i que a priori semblava una de les parts més senzilles a realitzar, algunes de les funcionalitats (com són per exemple la creació de plantilles o de les mateixes notícies) requerien una gran quantitat de formularis per a transportar la informació d'una pàgina a l'altra, finalitzant l'operació inserint les dades a la base de dades.

Aquesta WEB s'ha creat al mateix temps que els diferents clients, així doncs vam començar creant la part de la prova pilot, després la part de control d'usuaris i després la part del ABM Imatges (aprofitant gran part del prototipus) amb capacitat d'inserir/modificar i eliminar imatges de la base de dades. Posteriorment es va crear la funcionalitat de les plantilles (ABM) i s'hi va incorporar tota la gestió de les notícies, tant per a visualitzar-les com a per crear-les i modificar-les.

Per a acabar es va introduir la funcionalitat de la missatgeria (la més senzilla de totes) i l'actualització remota.



Il·lustració 10.1 - Esquema client WEB

Per a la seva creació es va utilitzar el llenguatge PHP, i es van reaprofitar gran part de les seqüències SQL creades per als Web Services. En una primera part el PHP utilitzava variables Globals (register_globals = on) per a facilitar la tasca de desenvolupament. Finalment, quan estava tot funcionant, es van realitzar els canvis oportuns per a fer servir variables de sessió i millorar així la seguretat del sistema WEB.

Finalment, per a intentar millorar l'aspecte gràfic de la WEB es van incorporar codi HTML per a poder utilitzar CSS. D'aquesta manera podem canviar l'aspecte de la web (colors, fonts, estils, etc...) canviant pràcticament el .CSS.

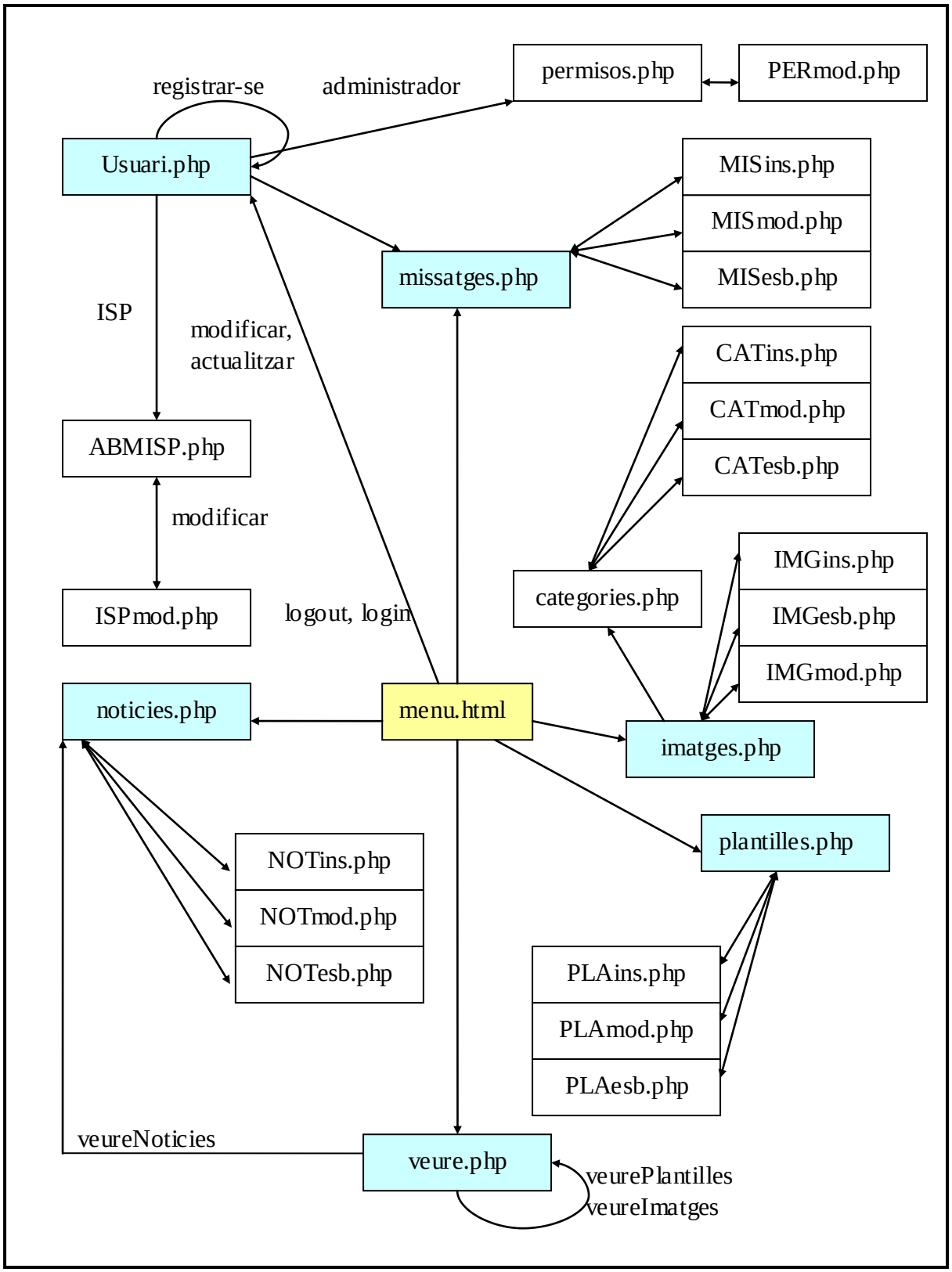
La web, a l'igual que els altres components, té un fitxer anomenat **config.php**, en el qual hi ha els paràmetres de la base de dades del sistema.

10.2 Components de la part WEB

El client web s'ha dividit en les següents funcionalitats:

- **Usuaris**
  - Usuari.php (control login/logout, alta)
  - Permisos.php (controlador modificació permisos)
    - PERmod.php (modificació permisos)
  - ABMISP (controlador modificació dades ISP)
    - ISPmod.php (modificació dades ISP)
- **Imatges**
  - Imatges.php (controlador alta/baixa/modificació)
    - IMGins.php (inserir imatges)
    - IMGesb.php (esborrar imatges)
    - IMGmod.php (modificar imatges)
  - Categories.php (controaldor alta/baixa/modificació)
    - CATins.php (inserir categoria)
    - CATesb.php (esborrar categoria)
    - CATmod.php (modificar categoria)
- **Missatges**
  - Missatges.php (controlador A/B/M)
    - MISins.php (inserir missatge)
    - MISesb.php (esborrar missatge)
    - MISmod.php (modificar missatge)
- **Noticies**
  - Noticies.php (controlador A/B/M)
    - NOTins.php (inserir missatge)
    - NOTesb.php (esborrar missatge)
    - NOTmod.php (modificar missatge)
- **Plantilles**
  - Plantilles.php (controlador A/B/M)
    - PLAins.php (inserir plantilla)
    - PLAesb.php (esborrar plantilla)
    - PLAmod.php (modificar plantilla)
- **Visualitzar**
  - Veure.php (Controlador de la part de visualització)

Hem creat una col·lecció de funcions (funcions.php) per a utilitzar en els diferents PHP que hem creat. Entre elles podem trobar sortides de llistes de selecció pel navegador, creació de thumbnails per a les imatges, actualització i creació dels fitxers necessaris per a l'actualització remota. Visualització de notícies, missatges, etc...



Il·lustració 10.2 - Flux Pàgina WEB

10.3 Funcions.php

A continuació mostrarem la capçalera de la funció PHP i la seva funcionalitat. En el cas que fos necessari mostraríem també el diagrama de seqüència del codi creat i/o la sentència SQL que executa (ometem els diferents controls sobre les variables d'entrada a la sentència).

Descripció	Crida	Sentències SQL i Exemples.
Mostra un error	mysql_die($error)	
Retorna el permís de l'usuari	retPermis ($dbase,$user)	SELECT permis FROM autor WHERE nom=.$user.
Missatge de denegació de permís.	missPermis ($dbase,$user,$posicio,$ABM, $de,$anterior)	p.ex : missPermix ($dbase,$usuari,$CNOTI,'inserir', 'noticies', 'noticies.php'); comprobaria l'usuari pel permís CNOTI. Si no fos correcte tornaria el missatge : 'Ho sentim, no té permís per inserir noticies' i es redirigiria a la pàgina noticies.php.
Retorna la cadena $file concatenada amb l'extensió especificada (jpg,png o gif)	addExtension ($tipus,$file)	
Retorna el nom de la imatge amb l'extensió correcta. Si tipus és 0 retornem file directament.	show_href($file,$ruta,$tipus)	
Crear uns thumbnails per a les imatges, retorna tag html de <img src....> amb l'informació correcta.	show_thumbnail($file,$ruta,$tipus)	El màxim de amplanda o llargada està específicat com a 120 pixels. Per a poder trobar l'amplada $ruta.$file ha de apuntar a una URL o un fitxer existent.
Retorna en una llista desplegable el resultat de la sentència $sql i assigna la selecció a la variable $nom. $cvalor, conté el nom de la columna que s'assigna a valor i $csortida, conté el nom de la columna que es veu.	llista ($dbase,$sql,$nom,$cvalor,$csortida,$tamany)	P.ex Llista ($dbase,"SELECT idNoticia, títol FROM....",$noticia, 'idNoticia','títol', 5); Al acceptar el formulari la variable $noticia tindria el valor de l'idNoticia seleccionada. L'usuari només veuría per a seleccionar la columna títol de la taula.
Similar a l'anterior però amb la possibilitat de seleccionar un valor de la taula tal que $cvalor == $select	llistaSeleccio ($dbase,$sql,$nom,$cvalor,$csortida,$select,$tamany)	
Retorna una llista desplegable amb els autors.	autors ($dbase)	SELECT * FROM autor ORDER BY nom ASC
Retorna una llista desplegable amb les notícies de l'autor	noticies ($dbase,$autor)	SELECT idNoticia,títol FROM noticies WHERE nom=.$autor. ORDER BY DATA DESC, idNoticia DESC
Retorna una llista desplegable amb les notícies.	noticiesTotes ($dbase)	SELECT idNoticia,títol FROM noticies WHERE 1 ORDER BY DATA DESC, idNoticia DESC
Similar a llista, pero especialitzada per a les imatges. Torna una descripció amb més camps per a facilitar la selecció de les imatges.	llista_imatges ($dbase,$sql)	nom -- descripcio -- DATA
Retorna una llista despregable	imatges ($dbase, $usuari,	SELECT id,nom,descripcio,DATA

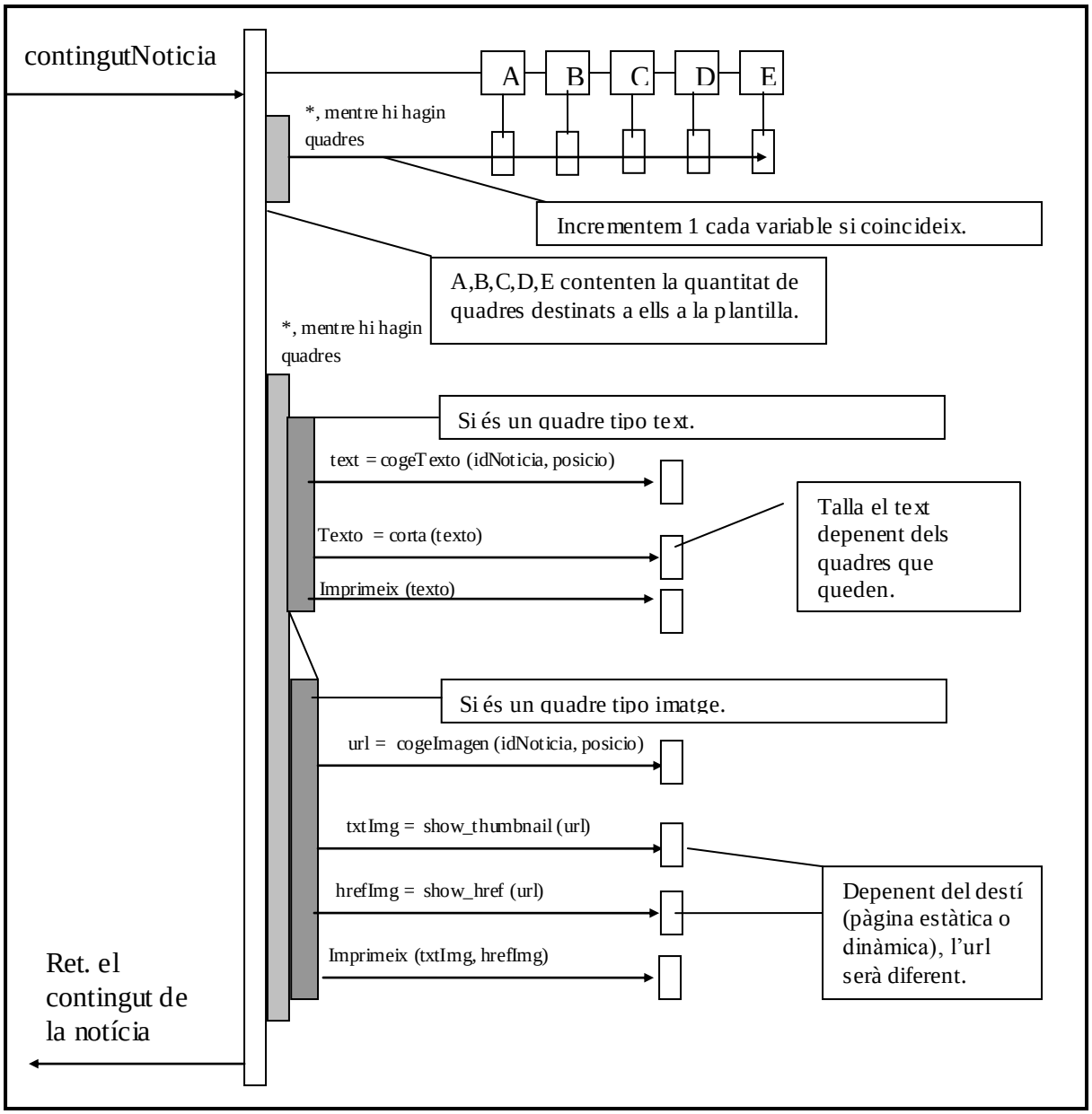
amb les imatges de l'autor (si es un administrador totes)	$permis)	FROM imatge WHERE nomAutor=$usuari ORDER BY DATA DESC o SELECT id,nom,descripcio,DATA FROM imatge WHERE 1 ORDER BY DATA DESC
Retorna una llista despregable amb les imatges de la categoria.	imatgesPCat ($dbase, $categoria)	SELECT id,nom,descripcio,DATA FROM imatge WHERE idCategoria=$categoria ORDER BY DATA DESC
Retorna una llista amb les url de les imatges.	imatgesPCatURL ($dbase, $categoria)	SELECT id,nom,descripcio,DATA,nomAutor FROM imatge WHERE idCategoria=$categoria ORDER BY DATA DESC. Retorna una llista de links amb les dades de la imatge per a poder-la visualitzar.
Similar a l'anterior però retorna una taula amb els thumbnails de les imatges. 3 columnés per fila.	imatgesPCatTABLE ($dbase, $categoria)	
Retorna una llista despregable amb les categories disponibles. Pot seleccionar la categoria amb idCategoria == $defecte	categories ($dbase,$defecte)	SELECT idCategoria,nom FROM categories ORDER BY nom ASC
Retorna llista despregable amb les plantilles disponibles, pot preseleccionar la idPlantilla == $defecte.	plantilles ($dbase,$defecte)	SELECT idPlantilla,nom FROM plantilla ORDER BY nom ASC
Retorna una llista despregable amb les plantilles disponibles del usuari.	plantillesUsuari ($dbase,$defecte,$autor)	SELECT idPlantilla,nom FROM plantilla WHERE nomAutor=$autor ORDER BY nom ASC
Retorna una llista despregable amb Text A..E i Imatge 1..6 i Buit, per a poder crear les plantilles. La variable associada a la selecció es 'ele' concatenat amb $num.	elePlant ($num,$select)	
Crea una taula per a visualitzar la plantilla que se li passa per paràmetre.	veurePlantilla ($quadre)	Utilitza elePlant i el seu paràmetre $select per a preseleccionar l'element correcte.
Retorna una llista despregable amb les posicions disponibles / ocupades de text.	omplePos2 ($dbase,$ident)	SELECT posicio FROM text WHERE idNoticia=$ident ORDER BY posicio ASC
Retorna una llista despregable amb les posicions disponibles / ocupades d'imatges.	omplePosIm2 ($dbase,$ident)	SELECT posicio FROM img WHERE idNoticia=$ident ORDER BY posicio ASC
Retorna els text de la noticia,posicio	retornaText ($dbase,$ident,$posicio)	SELECT Text FROM text WHERE idNoticia=$ident AND posicio =$posicio
Retorna l'id de l'imatge de la posicio/noticia	buscaImatge($dbase,$ident,$posicio)	SELECT i.id FROM img AS i,imatge AS j WHERE i.id=j.id AND i.idNoticia=$ident AND i.posicio =$posicio;
Agafa el text de una noticia/posició	cogeTexto ($dbase,$idNoticia,$ultimo)	És una crida a retornaText, però $ultimo esta especificat com a 'A'..'E'
Retorna l'URL de la imatge. Si tipus = 0 no cal extensió a la sortida.	cogeImagen ($dbase,$idNoticia,$ultimo,$tipus)	Utilitza el mètode buscaImatge per a trovar l'id de la imatge i crear l'URL "mostraImatge.php?id=$id"

Talla el text donada una llargada determinada (es talla sempre en espais en blanc)	corta ($texto,$uD,$larg)	
Donada una plantilla, un id de noticia, i un tipus crea la notícia corresponent. tipus = 0 indica la creació de la notícia estàtica.	contingutNoticia ($dbase,$quadre,$idNoticia,$tipus)	Mostrarem a continuació el diagrama de seqüència d'aquesta funció.
Retorna la notícia debidament formatejada. * tipus 3, notícia per a la web estàtica. * tipus 1, notícia resumida. * tipus 0, notícia per a la web dinàmica.	mostraNoticia ($dbase,$idNoticia,$tipus)	En el tipus 3 és retorna una cadena amb la notícia. En els tipus 1 i 0 no és retorna res, sino que s'imprimeix la notícia directament. També mostrarem el diagrama de seqüència d'aquesta operació
Esborra el directori dst_dir del ftp, així com tots els fitxers i subdirectoris.	ftp_rmAll($conn_id,$dst_dir)	
Actualitza el FTP, amb el número de notícies indicadés per a numNot	actFTP ($link, $ftplogin,$ftpurl, $ftpContrasenya, $ftpPATH, $numNot)	SELECT idNoticia FROM noticies WHERE 1 ORDER BY DATA DESC, idNoticia DESC LIMIT 0, $numNot SELECT i.id FROM img AS i,imatge AS j WHERE i.id=j.id AND i.idNoticia=idNoticia Es mostrarà el diagrama de seqüència d'aquesta operació.
Retorna una llista de selecció dels missatges que provenen de la sentència SQL.	llistaMissatge ($dbase,$sql)	El format de sortida és el següent: Títol de nom a nom – (DATA)
Retorna la llista de missatges de l'usuari especificat (dirigits a ell o fets per ell)	missatges ($dbase,$usuari)	SELECT títol,idMissatge,nom,a,DATA FROM missatge WHERE a=$usuari OR nom=$usuari ORDER BY DATA DESC
Retorna la llista de missatges de l'usuari especificat (creats per a ell)	missatgesMOD ($dbase,$usuari)	SELECT títol,idMissatge,nom,a,DATA FROM missatge WHERE nom=$usuari ORDER BY DATA DESC
Retorna el número de missatges dirigits per a l'usuari que tenim a la bústia.	countMissatges ($dbase,$usuari)	SELECT COUNT(*) FROM missatge WHERE a=$usuari
Retorna el text necessari per a mostrar el missatge	printMissatge ($link,$idMissatge)	SELECT * FROM missatge where idMissatge=$idMissatge

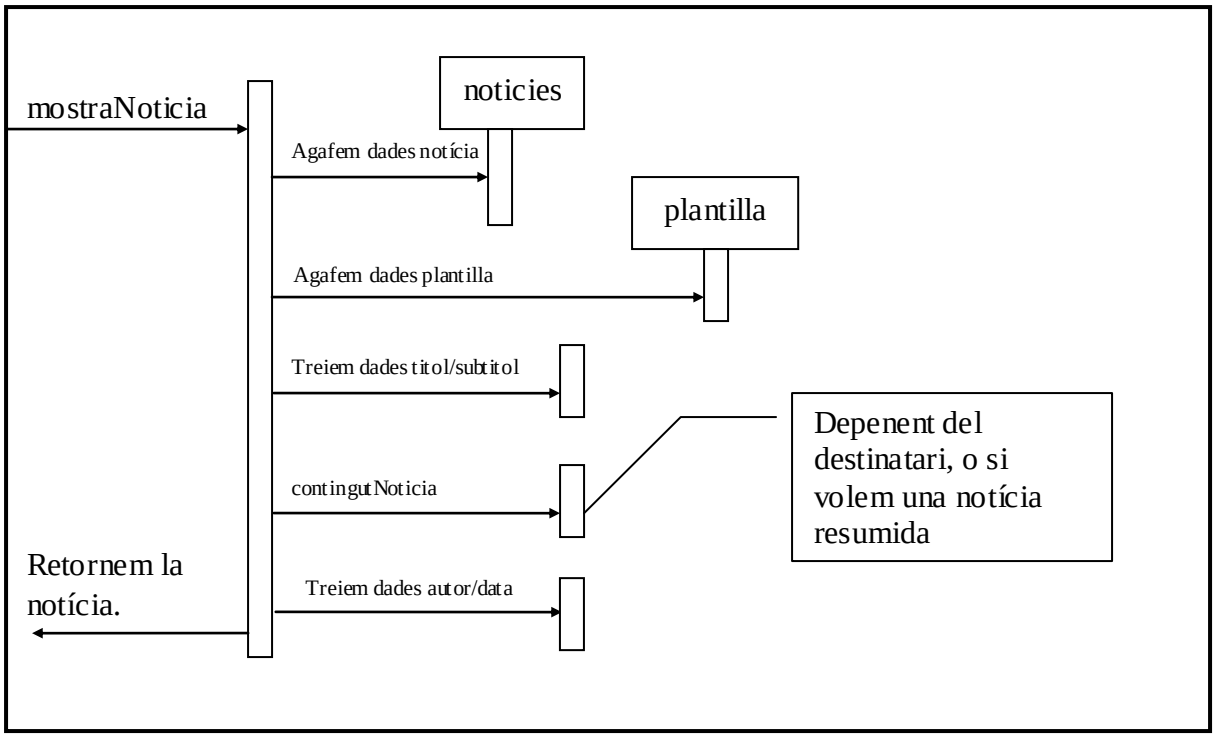
Taula 10.1 - Funcions de funcions.php

10.3.1 Diagrames de seqüència

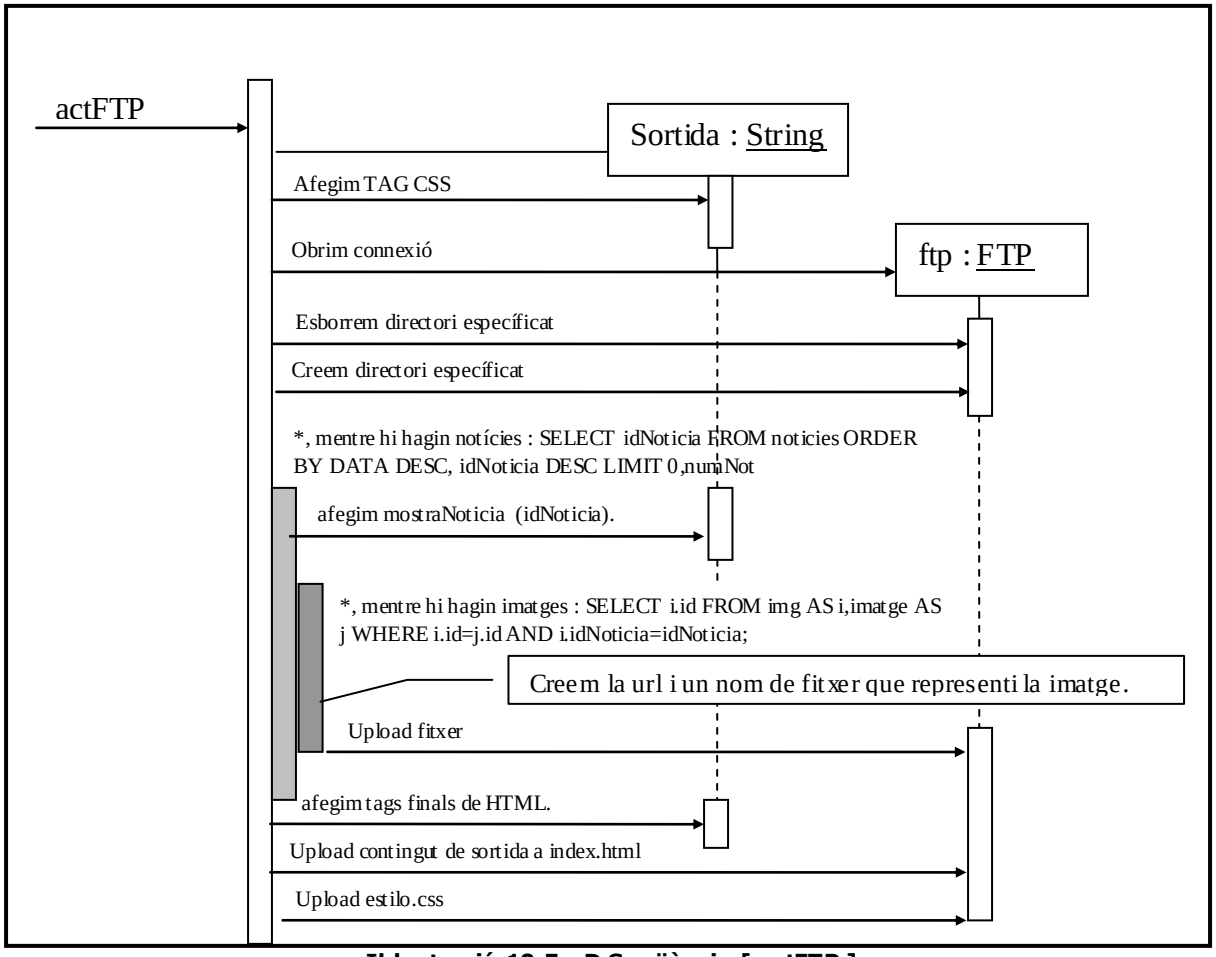
A continuació mostrarem els diagrames de seqüència de les funcions més complicades.



Il·lustració 10.3 - D.Seqüència [contingutNoticia]



Il·lustració 10.4 - D.Seqüència [mostraNoticia]



Il·lustració 10.5 - D.Seqüència [actFTP]

10.4 Controladors

Els controladors són molt senzills: comencen amb la connexió a la base de dades (si és necessari) utilitzant les dades de config.php; posteriorment, defineixen la barra de navegació (menú) i comencen amb la lògica del programa. Normalment el controlador es crida a si mateix amb la variable **apartat** prenent com a valor: inserir, modificar, esborrar, i l'script es dirigeix a la part corresponent a la funció:

```
if (isset($_SESSION['login']) && isset ($_GET['apartat']) && $_GET['apartat'] == "Modificar"
&& $_SESSION['login']==1 )
```

A aquesta part només s'entraria si s'està logat i l'apartat és igual a Modificar.

Posteriorment, dins d'aquesta part es presenta un formulari amb la informació necessària per a inserir/modificar o esborrar; aquest formulari al ser acceptat anirà a la pàgina corresponent **xxxmod.php**, **xxxins.php** o **xxxesb.php**, que amb les dades rebudes realitzarà la funció requerida.

10.5 Seguretat

Tot i que no s'han fet grans esforços en la seguretat, les variables que mantenen l'estat de la sessió (logat, usuari, contrasenya, etc.) són variables de sessió i per tant no poden modificar-se des del navegador. Per altra banda les operacions comproven sempre si el permís de l'usuari que la realitza és correcte.

Al no necessitar variables globals podríem especificar al servidor PHP que no acceptes variables globals, però per culpa del que sembla un bug de PHP (segons fòrums públics de PHP) tenim que activar-les per a poder transferir imatges del PC al servidor. Si aquesta part s'arregla podríem desactivar les variables globals.

10.6 Personalització i CSS

En el directori de la WEB podem trobar el fitxer títol.gif encarregat de mostrar la imatge de capçalera del menú, i podem canviar-la per una altra més adient. A continuació mostrarem les classes que hem creat per a la visualització / modificació de l'aspecte de la pàgina WEB i per a poder-la adaptar amb facilitat a les nostres necessitats.

`a:link`	Links
`a:visited`	Links visitats
`a:hover`	Links amb el ratolí a sobre
`a:hover IMG`	Links amb Thumbnail
`BODY`	Cos del html (color de fons, pex.)

`img`	Imatges (recuadre extern)
`.noticia`	Classe global notícia, específica el color de fons de la caixa, la mida, el recuadre...
`.títol`	Especifica el text del títol de la notícia, així com el color de fons, etc...
`.subtítol`	Específica el text del subtítol de la notícia, així com el color de fons
`.cos`	Específica l'aspecte del cos de la notícia
`.peu`	Específica l'aspecte del peu de la notícia (recuadre, text...)
`#form table`	Defineix els colors i l'aspecte dels formularis amb id = form
`#form td`	Defineix els colors i l'aspecte dels formularis amb id = form
`#navleft`	Creació de la barra de menú esquerra, colors, recuadre...
`#navtop`	Creació de la barra de menú superior.
`#dibuix`	Creació dels recuadres per als thumbnails de la galeria.
`#dibuix2`	Creació dels recuadres per als thumbnails de la notícia.

Taula 10.2 - Estils CSS

Mostrem un exemple de com modificar l'aspecte de la WEB en no massa temps. En la següent figura podem veure l'aspecte original de la WEB amb tonalitats blaves. A continuació modificarem el CSS per a adaptar la WEB a una associació ecologista amb tonalitats verdes.

Il·lustració 10.6 - Aspecte original

Anem a assenyalar els passos necessaris per a realitzar la conversió:

a) Canvi del color de fons (TAG : BODY) i li afegim una imatge per al fons (Imatges/fondo.jpg).

```
BODY { background-color: #80B080;   background-image: url(Imatges/fondo.jpg); }
```

b) TAG #navleft, adaptem el color del menú de navegació.

```
#navleft        { … background-color: #80B080; …}
```

c) Eliminem el color de fons dels links (TAG A)

```
a:link          {... background-color: none;}
a:visited       {... background-color: none;}
```

d) Fem que es subratlli amb un color verd florescent.

```
a:hover         {...  background-color: lime;}
```

e) a #navleft i #navtop, canvíem l'estil del requadre.

```
#navtop         {border: black 3px dotted;…}
```

f) Modifiquem finalment els elements de la notícia: en primer lloc el títol al qual li afegirem una imatge de fons (TAG : títol) i el color de la notícia en general (TAG notícia) , per a utilitzar una tonalitat més verda (TAG cos, subtítol, títol per a adaptar el color de fons i els requadres).

```
.títol          {background-color:#A0A0FF ; background-image:url(Imatges/titoltaula.jpg); … }
.noticia        { background-color:#F0FFF0; … }
...
```

g) Canviarem l'aspecte del peu de la notícia (TAG: Peu) .

```
.peu    {
        background-color:#e0FFe0;letter-spacing:2px;text-align:right;font-size: small;
        border: 3px solid #F0FFF0;…
}
```

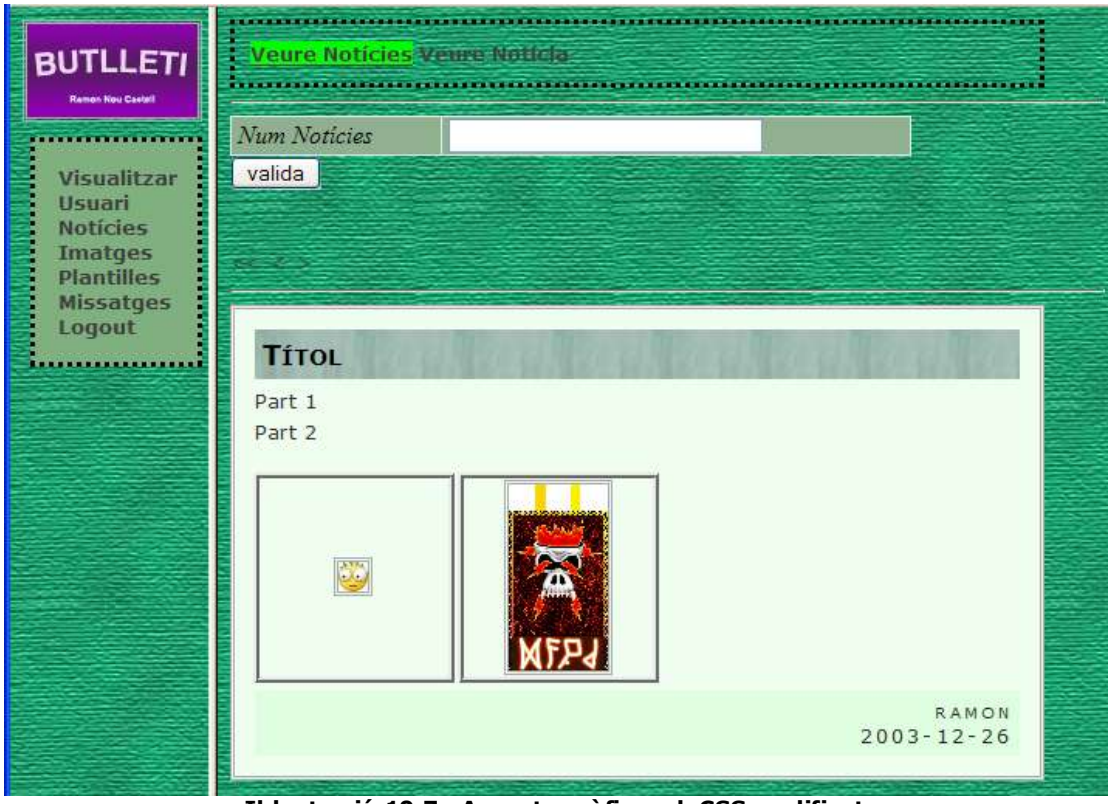
h) Acabarem modificant els colors dels formularis (TAG form) i dels thumbnails de la galeria (TAG dibuix), i el tamany de la notícia i el menú superior a 500 pixels.

```
.noticia        { …  width: 500px; }
```

i) Finalment i per a demostrar la potència dels CSS, eliminarem la linea del subtítul. (TAG subtítol).

```
.subtítol       {... display:none;}
```

El resultat és el següent:



Il·lustració 10.7 - Aspecte gràfic amb CSS modificat.

10.7 WEB d'instal·lació

Per tal de crear les taules de la base de dades i instal·lar l'usuari inicial de manera senzilla es va crear un script PHP. L'script esborra la base de dades (drop) i crea una nova base de dades amb les dades del config.php.

Finalment, crea correctament les taules (amb el mode **INNODB** de mySQL) per a poder fer servir claus foranes.

A l'acabar la creació de les taules, afegim l'usuari principal del sistema, el qual en un primer moment s'encarregarà de donar els permisos als usuaris.

11. Planificació final i desviació

La planificació inicial va ser pessimista en algunes tasques i massa optimista en altres. Per altra banda, a l'inici de la segona part del projecte es va decidir realitzar un prototipus de l'aplicació i per tant la planificació d'aquesta part es va tenir que avançar. Mostrem a continuació el resultat:

Tasca	Prevista	Real
Cerca d'informació sobre WS (PC)	16	8
Cerca d'informació sobre WS (Pocket PC)	8	6
Instal·lació Tomcat/Apache/mySQL/PHP	3	1
Instal·lació Apache Axis	1	1
Primer contacte amb Axis (proves, documentació)	8	8
Especificació de la prova pilot	4	2
Disseny de la prova pilot	8	8
Disseny (BD) prova pilot	1	1
Implementació BD	1	0,5
Implementació WS a AXIS	16	8
Cerca d'informació sobre PHP	8	8
Implementació façana WEB	16	20
Instal·lació entorn prog Pocket PC	8	3
Estudi de la implementació mitjançant .NET	24	12
Estudi de la implementació mitjançant EVT	24	24
Especificació/Disseny part Pocket PC	4	4
Implementació amb .NET	16	8
Implementació amb EVT	16	24
Proves i conclusions	8	8
Documentació.	16	16
Cerca d'informació sobre mòbils	16	16
Especificació del sistema d'informació	40	30

Taula 11.1 - Planificació esperada - real

11.1 Comentaris de les desviacions

Cerca d'informació sobre WS (PC) - (16 -> 8)

La cerca es va reduir a informació sobre AXIS, ja que les altres alternatives eren o massa cares o la seva implementació era massa complicada.

Cerca d'informació sobre WS (Pocket PC) - (8 -> 6)

Aquesta cerca ha estat ràpida també, ja que hi havia poques alternatives, entre elles havia la de realitzar un client amb tecnologia Flash (Macromedia).

Instal·lació Tomcat/Apache/mySQL/PHP – (3 -> 1)

La instal·lació d'Apache/mySQL i PHP es simplifica utilitzant un paquet que instal·la i configura els 3 elements. La instal·lació de Tomcat és també molt senzilla ja que Apache té un paquet d'instal·lació per a Windows.

Especificació de la prova pilot (4 -> 2)

El cas pilot era molt senzill i per tant la seva especificació no mereixia més de 2 hores.

Implementació WS a AXIS (16 -> 8)

La primera implementació del WS a Axis va ser sorprendentment ràpida i sense gaires problemes (no necessitavem classes de suport tampoc).

Implementació façana WEB (16 -> 20)

La implementació de la primera façana WEB va ser costosa, trigant 4 hores més de les previstes per a resoldre problemes amb la sortida de les imatges pel navegador.

Instal·lació entorn prog Pocket PC (8 -> 3)

L'entorn Pocket PC només té una possibilitat, Microsoft; per tant, la instal·lació i cerca del programari va ser ràpida. La part de PocketSoap ja l'havíem trobat en la tasca "Cerca d'informació sobre WS (Pocket PC)".

Estudi de la implementació mitjançant .NET (24 -> 12)

L'estudi de la implementació amb .NET va ser extremadament senzill, els manuals indicaven que pràcticament amb 4 clicks podríem tenir un client fet i per a totes les plataformes on arriba el .NET Framework. Per tant només es va requerir un estudi de 12 hores.

Estudi de la implementació mitjançant EVT (24 -> 24)

En aquesta part si que hi vam trobar dificultats, tot i que les 24 hores estimades eren una estimació molt pessimista: gairebé ens quedem curts. PocketSOAP és una llibreria complicada: en un primer moment, la seva dificultat disminueix si programem amb Visual Basic ja que s'adapta molt bé. Però al mateix temps aquesta solució incrementa el temps d'implementació, al ser Visual Basic per EVT un llenguatge amb poca funcionalitat.

Implementació amb .NET (16 -> 8)

El temps d'implementació es va reduir a la meitat per la sorprenent facilitat de creació d'un client WS en .NET. Per fer-nos una idea, vàrem trigar més en realitzar la interfície gràfica que en crear la part client.

Implementació amb EVT (16 -> 24)

Tal com hem avançat, la implementació amb visual Basic ens va fer perdre temps; la utilització de coses com obrir fitxers binaris (les imatges) i ficar-les en una variable poden arribar a ser molt complicades amb Visual Basic per EVT.

Especificació del sistema d'informació (40 -> 30)

El sistema va resultar més senzill d'especificar del que s'esperava ja que no incloïa cap element complicat.

Prototipus (Activitat no programada)

Tasca	Previst (no programat)	Real

Cerca d'informació sobre mòbils (programació)	4	3
Instal·lació SDK i selecció	8	6
Cerca Informació sobre mòbils (programació)	4	3
Disseny	30	15
Disseny BD	10	4
Implementació BD	4	4
Creació WEB Control	24	16
Implementació WS	8	6
Especificació/Disseny móbil Imatges	4	2
Creació Client mòbil Imatges	24	16
Proves emulador	8	2
Especificació/Disseny móbil Notícies	8	4
Creació Client mòbil Notícies	24	12
Testeig Client Mòbil amb terminal real	8	6
Creació de les solucions als problemes trobats	0	16
Proves client correu	0	4

Taula 11.2 – Planificació prototipus.

La part del prototipus no estava planificada i es van haver d'avançar alguns elements (com el disseny de la base de dades i el disseny del sistema, que només es va realitzar per la part a implementar). Posteriorment, es va acabar d'implementar el que faltava.
En la següent taula hem descomptat el temps que vam exhaurir en la creació del prototipus; també hem afegit algunes tasques que no vam planificar a l'inici.

Cerca d'informació sobre mòbils (PROG) (4 -> 3)

Navegant a través de les pàgines corporatives dedicades als desenvolupadors dels diferents fabricants de mòbils, vam escollir una sèrie de fabricants que disposaven de SDK per a J2ME.

Instal·lació dels diferents SDK i Selecció d'un SDK (8 -> 6)

Com que teníem només l'opció de Nokia al prototipus, només vam instal·lar l'SDK de Nokia. Posteriorment vam poder instal·lar l'SDK de Sony.

Disseny (0 -> 15, Activitat Avançada)

Per tal de dissenyar i implementar el prototipus (amb un timeline de novembre) vam realitzar la selecció de funcionalitats que volíem implementar. Es van escollir dues per la seva possible dificultat i transcendència:

La primera va ser la funcionalitat de les imatges, ja que implicava la investigació en l'ús de la càmera.
La segona era la implementació del sistema de notícies, ja que implicava la transferència de diferents tipus de dades i una mostra molt representativa de l'aplicació final.

Disseny BD (0 -> 4, AA)

El disseny de la base de dades necessitava part del disseny del sistema, i com que es va fer la base de dades definitiva va requerir molta cura la seva realització. Cal destacar que l'eina DBCASE va fallar un parell de cops, destruint l'arxiu XML que contenia l'especificació de la base de dades i fent-nos perdre temps.

Implementació BD (0 -> 4, AA)

La implementació va durar unes 4 hores, ja que incloïa una llarga llista de taules i es van haver de mirar algunes característiques de mySQL que es desconeixien.

Creació Web de Control (0 -> 16, AA)

La web va ser difícil de realitzar, ja que incloïa conceptes com user / contrasenya, dades que es transmetien d'una pàgina a altre, modificació al vol d'imatges, així com un

estudi de la implementació de les plantilles web mitjançant taules. Aquestes 16 hores, per sort, es podran restar de la creació de la façana WEB final.

També es va construir la funcionalitat d'actualització d'una pàgina web estàtica.

Implementació WS (0 -> 6, AA)

El WS, tot i no ser difícils eren molts, i la seva programació va consumir 6 hores. Igualment, com el cas anterior, aquestes 6 hores es podran descomptar de la programació final.

Creació Client Mòbil Imatges (0 -> 16, AA)

En un principi vam estimar la duració del prototipus, que tenia poca funcionalitat comparant-lo amb el client final, en 10 hores; però el desconeixement de la **MMAPI** va fer que l'aplicació es realitzes amb 16 hores. Per sort l'experiència adquirida en aquesta tasca pot reduir el temps total, que era de 24 hores, en 20.

Creació Client Mòbil Notícies (0 -> 12, AA)

Amb l'experiència del client Imatges, vam trigar només 12 hores en construir el client encarregat de la part de les notícies. Aquest client inclou tota la funcionalitat necessària, a excepció de l'eliminació de notícies que podem incorporar amb poc temps, sense arribar a les 24 hores estimades.

Testeig Client Móvil (amb terminal real) (0 -> 6, AA)

Les proves amb un terminal real van ser ràpides, ja que s'havia provat prou bé amb els emuladors. Tot i això, aquest testeig ens va crear 2 tasques més per fer: la creació de la solució de processat d'Email i la solució del mapeig entre codificacions.

Creació de les solucions als problemes (0 -> 16, ANP)

Aquesta tasca extra inclou la cerca d'informació sobre les codificacions de caràcters UTF-8 i ISO-LATIN-1, així com la cerca d'un servidor de correu, la seva instal·lació i la creació del procés ProcMail.

Finalització Disseny sistema	15	8
Finalització Client ABM notícies	12	4
Especificació/Disseny part missatges	2	2
Programació ABM Missatges amb J2ME	4	4
Proves/Solució problemes	2	2
Creació de la façana WEB	8	18
Proves Aplicació general	0	16
Creació WEB Instal·lació	0	2
Creació CSS i personalització	0	8
Preparació de la part d'instal·lació	8	16
Proves instal·lació	4	8
Creació de la memòria	100	100

Taula 11.3 - Planificació real de l'etapa final.

Disseny del sistema d'informació (15 -> 8)

Vam utilitzar part del temps que teníem dedicat al disseny en crear els diagrames de seqüència de les operacions que mancaven.

Programació ABM Notícies (12 -> 4)

Una vegada acabat el prototipus i realitzades les proves, es van aprofitar les hores que mancaven per a augmentar la funcionalitat del client i incorporar la funcionalitat d'esborrar notícies, així com diverses millores en seguretat.

Proves Aplicació general (0 -> 16, ANP)

Aprofitant un viatge es van realitzar proves exhaustives del sistema, sobretot d'enviament d'imatges i problemes de connexió a la xarxa mitjançant el mòbil.

Façana WEB (8 -> 18)

La programació de la façana WEB es va acabar en aquest punt; teníem tota la funcionalitat i es van millorar aspectes com la seguretat i la presentació.

WEB Instal·lació (0 -> 2, ANP)

Tot i no estar programada, es va realitzar una pàgina WEB d'instal·lació per a facilitar la introducció de les dades inicials a la base de dades. En aquesta WEB no es va realitzar cap disseny especial.

CSS i personalització (0 -> 8, ANP)

Per tal de facilitar la conversió de la pàgina WEB (en quant a disseny) a una altra organització o necessitats, es va crear un fulla d'estils (CSS) i es va modificar la pàgina WEB per a utilitzar-los. També es van crear un parell d'exemples.

Part instal·lació (8 -> 16)

En un primer moment la instal·lació era molt senzilla, però no tenia en compte coses com l'adreça IP (que pot tenir diferents un servidor) o que ja hi hagi un servei al port 80, per exemple. Per tant, es va modificar i ampliar l'instal·lador per a poder-se utilitzar en una gran varietat de casos.

Proves instal·lació (4 -> 8)

Per a poder-lo instal·lar es van provar diferents màquines amb diferents configuracions.

Finalment, la suma aproximada d'hores és de 530, que s'aproxima suficientment al temps estimat a l'inici, gràcies a que es va realitzar una planificació pessimista i s'han afegit tasques no planificades inicialment.

12. Conclusions

El projecte ha arribat al seu objectiu: si bé no hem pogut crear un sistema d'informació de qualitat, cal veure que el S.I. era una eina per a arribar a un objectiu, que en aquest cas era comprendre les dificultats que hi ha a l'implementar clients de Web Services basats amb tecnologies emergents com són Pocket PC i J2ME, i acostar-se a totes les tecnologies que envolten a la seva creació.

12.1 Anàlisi de costos

Analitzarem per separat les tres plataformes implicades: Pocket PC, Mòbil i PC.

12.1.1 Pocket PC

La implementació en un Pocket PC és segurament la menys complicada. El fet de que les aplicacions es construeixin en un entorn visual facilita en gran manera la tasca d'implementació i permet reduir el temps necessari per a crear els diferents formularis de l'aplicació. El cost monetari en aquesta part és el més important, ja que un dels programaris requereix una llicencia, que no és especialment assequible.

Cost monetari

Analitzem primer que ens costa una llicencia de ***Visual Studio .NET 2003***[1] en la seva versió mínima (professional), que tot i treure'ns algunes funcionalitats, no les necessitem per a realitzar aplicacions basades en WS i en Windows CE.

El preu (10/12/2003) és de $1,079, que al canvi són uns 900 €, una quantitat força elevada. També podem utilitzar una versió de prova[2] durant 60 dies per tal de provar el software.

L'altra alternativa, la d'utilitzar ***l'EVT*** (Embedded Visual Tools) i ***PocketSOAP***, ens permet crear clients WS de baix cost d'implementació. Si bé no té la facilitat i potència de .NET quan a disseny d'aplicacions, els clients no han de ser gaire complicats i per tant podem sortir-nos amb èxit amb aquesta alternativa.

Cost d'implementació

Aquí és on obtenim algunes de les diferencies més notables.

Visual Studio .NET ens permet realitzar una aplicació en molt poc temps. Aquest estalvi de temps d'implementació es pot utilitzar per a fer una aplicació millor o més depurada, o simplement estalviar-se-la en recursos. Per fer-nos una idea de la facilitat d'implementació, la construcció de la primera capa de l'aplicació (la que accedeix al procediment remot) només és qüestió d'introduir l'adreça (URL) del WSDL del Web Service al que volem accedir. En qüestió de segons tindrem disponible una classe que ens encapsularà tots els accessos al WS.

[1] http://msdn.microsoft.com/vstudio/howtobuy/choosing.aspx
[2] http://msdn.microsoft.com/vstudio/productinfo/trial/default.asp

Per altra banda, amb ***PocketSOAP*** i ***EVT*** aquesta capa l'hem d'acabar fent nosaltres. Aquesta tasca pot ser més o menys complicada, depenent del grau de familiaritat que tinguem amb ***PocketSOAP***. Cal recordar que la implementació més natural de ***PocketSOAP*** és amb el llenguatge Visual Basic. Visual Basic en la versió que conté EVT no inclou encara els conceptes de classe (o els inclou d'una manera molt relaxada). Això provoca que la implementació no adquireixi mai els nivells d'una aplicació amb .NET.

Cal destacar també que amb .NET les aplicacions gaudeixen de més eficiència: l'API per exemple és molt més potent i amb poques línies podem realitzar tasques que amb l'antiga API de WinCE no podem aconseguir.

En els dos casos les capacitats de depuració i emulació dels Pocket PC adquireixen un grau d'exactitud molt alts i per tant no trobem diferències entre els dos mètodes (permeten execució pas a pas, etc...)

Conclusió Pocket PC

L'elecció d'un mètode o un altre ha d'estar condicionat per la feina que hem de fer i el seu possible rendiment. Donat que és una tecnologia molt capdavantera, el risc tecnològic és molt important i està present. Ningú farà servir una aplicació que a l'hora de fer-la funcionar li costa massa de manegar o és incomoda, per tant aquest risc és el que ens ha de dir si utilitzar .NET o EVT+PocketSOAP. Cal també conèixer els recursos humans dels que disposem i el temps/diners que els tenim disponibles. Si aquest consum de recursos humans en l'alternativa EVT+PocketSOAP arriben al preu de la llicència, possiblement sigui més recomanable utilitzar .NET, ja que l'aplicació serà (segurament) més robusta i la realitzarem en un temps menor.

Pel que fa a la dificultat de la implementació del client Web Service, tampoc podem considerar de gran dificultat la implementació en PocketSOAP, ja que els Pocket PC han deixat de ser un dispositiu amb característiques de memòria o de CPU limitades.

12.1.2 J2ME, tecnologia mòbil

En els terminals mòbils ens trobem poques opcions en quant a llenguatge i llibreria, només disposem (i a dia d'avui (desembre-2003) la implementació de Web services per a J2ME de SUN encara segueix en public draft) de la implementació mitjançant kSOAP (i kSOAP 2, tot i que no és recomanable el seu ús per ser poc estable).

Com a llenguatge, l'elecció és única: Java en la seva versió J2ME (MicroEdition.), Per tant el seu cost en quant a llenguatge es 0. Hem de tenir en compte si volem algun entorn integrat (IDE) per a facilitar-nos la tasca d'implementació, però donada la poca dificultat (de moment) dels clients i aplicacions en J2ME, realment no cal. Si volem escollir podem agafar la plataforma SunOne o jBuilder per exemple.

Cost monetari

El cost monetari és nul quan a programari. No requerim cap llicència per a utilitzar J2ME, ni per a kSOAP. En quant a maquinari tampoc, tot i que pot semblar estrany podríem arribar a crear una aplicació complerta mitjançant un emulador de J2ME i assegurar-nos al 100% de que l'aplicació funcionarà en un terminal mòbil real.

L'emulació mitjançant la utilització dels SDK dels fabricants (Nokia, Sony-Ericsson, Siemens...), que són gratuïts (en molts casos no tenim ni que registrar-nos), permeten una emulació perfecta del terminal en qüestió (podent modificar diferents paràmetres, com la mida de la pila o de la memòria disponible).

Si volem adquirir un terminal real, a part del cost del terminal (franja 200-400 €) hem de tenir en compte el cost de les comunicacions en GPRS a l'hora de fer les proves. Aquest punt pot arribar a ser crític, però per sort trobem en alguns SDK com en el de Sony-Ericsson la possibilitat d'habilitar un túnel entre la connexió a la xarxa de l'ordinador on s'executa i el mòbil real. Això ens permet estalviar el cost de connectivitat per GPRS i provar l'aplicació en un mòbil real.

El cost d'ús per a un usuari és elevat, per això necessitem comprar "bonos GPRS" de 1 Mb de transferència al mes per exemple per 6 €, quantitat suficient per a un elevat ús del sistema.

Cost d'implementació

El cost d'implementació ve marcat pel tipus d'aplicació i la dificultat del disseny dels diferents formularis i fluxos de l'aplicació. El fet que les aplicacions hagin de construir-se sense tenir en compte conceptes com modularitat fan que la seva construcció (al menys en la última fase) sigui feina d'una persona i facin difícil el seu manteniment.

Podem observar que tot i intentar aplicar aquesta tècnica a l'aplicació **ABM Notícies**, la mida de l'aplicació era crítica (61 Kb) quan MIDP-1.0 ens assegura solament que com a mínim hi hauran 64 Kb per a l'aplicació (per sort això es troba en comptades ocasions ja). Això es pot solucionar en gran manera ofuscant el codi una vegada compilat i reduir així la mida de l'aplicació fent que les etiquetes, mètodes, variables i noms de classes ocupin menys espai (recordem que Java és un llenguatge on a les classes compilades es mantenen els noms).

Per tant ens trobem en un cost de temps d'implementació elevat en comparació amb el poc codi que hi tenim que generar.

kSOAP, com PocketSOAP, és una llibreria fàcil d'utilitzar, i tot i que té pocs exemples disponibles, costa ben poc acostumar-se a ella. De fet, amb el codi que hem creat en aquest projecte fi de carrera, no creiem que el futur desenvolupador o usuari necessiti gaire més possibilitats d'ús. (Recordem, enviament en Base64 de dades binaries, i enviament de classes complexes).

Conclusió J2ME

La utilització de WS amb J2ME és també molt perillosa, no per l'aplicació en sí, sinó per tenir un risc tecnològic molt més elevat que en Pocket PC. No cal recordar el que dèiem al començar aquesta memòria: la quota d'utilització de WAP en un telèfon mòbil és molt petita. Per a fer-ho atractiu per a un usuari, l'aplicació o servei que es creï ha de ser útil i usable.

J2ME no dóna moltes facilitats per a que sigui usable, ja que la col·locació final dels elements es delega finalment al **kVM**; una cosa curiosa que hem trobat en la realització de les proves sobre terminals reals és que **Nokia** no fa gaire cas a la col·locació dels menús (que podem especificar a quin botó s'assigna) i per exemple en un terminal **Sony** els menús es col·loquen en el lloc correcte.

Així doncs, més que a la implementació s'ha d'assignar més recursos a l'estudi de la usabilitat per a construir interfícies còmodes per a l'usuari. Podem acabar dient que el cost de la implementació en J2ME és el cost dels recursos humans que és facin servir.

12.1.3 PC, servidor i programació

En el PC hi fem servir una gran quantitat de programari. servidor web, servidor d'aplicacions, servidor de correu, servidor de Web Services, SGBD. L'elecció de tot el programari que hem utilitzat s'ha basat amb el cost quan a llicències i en la seva portabilitat al món Linux i al món Windows. L'elecció ha estat suficientment senzilla ja que el programari és molt utilitzat arreu del món.

Així tenim el servidor web Apache amb el mòdul d'extensió per a interpretar contingut en PHP, PHP com a llenguatge de programació en la xarxa en comptes de l'alternativa de Microsoft (.ASP), i finalment el Sistema Gestor de Base de Dades mySQL, que tot i no ser tant potent com els comercials, pel nostre sistema (i per molts altres projectes a la web) és més que suficient i té una integració perfecta amb PHP.

Per la banda dels Web Services requeríem com a mínim de Apache Tomcat, servidor d'aplicacions, i per realitzar els Web Services ho podíem fer a mà, amb alguna plataforma de pagament (WebSphere, per exemple) o amb l'opció que hem triat, Apache Axis. Apache Axis ens permetia reduir l'estudi dels WS de la banda PC al mínim i centrar-nos en l'aspecte central del PFC, que és la seva implementació en terminals mòbils.

Cost monetari

En aquests moments el cost monetari del programari en la banda PC és nul també. Tot el programari és d'ús gratuït o requereix una llicència mínima per a ús comercial. Hem de tenir en compte altres aspectes per a calcular el cost d'aquesta banda, un d'ells és el cost de manteniment d'un ordinador (el servidor) connectat les 24 hores a casa. Tot i que normalment ara els ordinadors (pel seu ús quotidià) solen estar-hi connectats, el cost mensual pot arribar a uns 12 €; cal també tenir en compte la connexió a la banda ampla que pot arribar als 39 € en la connexió mínima que ofereix Telefònica.

Cost d'implementació

El cost d'implementació el podem dividir en els diferents programaris que hem fet servir.

En primer lloc tenim l'SGBD, mySQL, on hem hagut d'introduir les taules de la base de Dades; podem assegurar que el cost d'implementació d'aquesta part és el mateix que en qualsevol SGBD ja que gairebé podem introduir taules amb SQL Standard i llestos.

Web Services: el cost aquí bé donat pel coneixement que tinguem amb els WS i Axis. Donat que el coneixement va ser adquirit durant la realització d'aquest PFC el cost és elevat en els primers WS que realitzem, però el grau d'expertesa que s'hi aconsegueix en poc temps és molt elevat (WS amb classes de suport, etc) i els resultats s'obtenen amb un temps gairebé record.

Façana WEB: aquesta part és la que porta més feina donat que es necessita fer la feina d'un dissenyador, cosa que és prou difícil, i d'un programador WEB. Adquirir el coneixement per a utilitzar PHP costa el mateix que aprendre un llenguatge de programació, i com a tot llenguatge de programació els primers resultats no són molt bons fins que no tenim un cert grau d'experiència. Els resultats però han de servir de base per a altres PFC's o persones que amb més coneixements de disseny i de PHP puguin fer una façana WEB a l'altura de l'aplicació.

Conclusió PC

Tota l'elecció del programari ha seguit la mateixa pauta: facilitat d'utilització, facilitat d'aprenentatge i cost nul o quasi nul. Programari amb aquestes característiques hi ha

poc, però per sort hi ha. A més, s'ha intentat triar el programari, que l'autor d'aquest document coneixia (com és el cas de PHP, Tomcat i mySQL), i per tant poder reduir el temps dedicat a estudiar-lo. En el cas de la façana WEB és una feina més de disseny gràfic que de programador, i per tant aconseguir que un dissenyador WEB ens faci la façana pot fer guanyar molts punts al programari desenvolupat.

12.2 Rendiment i Futur

Quan s'estava fent aquest PFC, varen posar-se de moda els WebLogs: pàgines webs personals o d'entitats que es converteixen en una mena de diaris personals amb gran afluència de públic. Amb aquest boom dels weblogs també hi ha empreses al darrere que han creat un seguit de software per tal que actualitzar una pàgina web amb un weblog sigui tant senzill com obrir una aplicació, escriure el contingut que hi volem afegir i enviar-lo per la xarxa.

Llegint les notícies sobre aquesta nova moda, va aparèixer una nova paraula que definia perfectament el software que s'estava fent (tot i que la base era més rudimentària); s'anomenava MobLog (Mòbil Log), i pretén ser el mateix que un WebLog però utilitzant fotografies obtingudes per un mòbil. Cal destacar que un dels primers MobLog és català.

Els MobLogs solen utilitzar la funcionalitat d'enviament de correu per a crear la "notícia" i enviar la imatge. Per tant no utilitzen la tecnologia WS.

Aquesta és la notícia que va aparèixer al Diari de Barcelona. (*podeu veure més noticies a la web del moblog de Cardedeu* http://cardedeu.textamerica.com/):

> Un veí de Cardedeu (Vallès Oriental) s'ha embarcat en una croada contra l'incivisme dels ciutadans i la despreocupació dels responsables municipals. "A Cardedeu no anem bé" és la seva arma: un 'moblog' ('weblog' o quadern de bitàcola a Internet amb fotografies preses amb telèfons mòbils) en què Albert Cuesta exposa proves gràfiques del comportament desaprensiu que es troba pels carrers de la localitat. El multimèdia arriba així a un àmbit, el dels diaris personals a la Xarxa, que han assolit un alt grau de popularitat entre els internautes.
>
> "A Cardedeu no anem bé" es planteja com una eina de participació ciutadana, ja que accepta les aportacions d'altres ciutadans, que poden inserir directament imatges per correu electrònic o bé des del telèfon mòbil multimèdia, a més d'afegir els seus comentaris a les fotos. El 'moblog' es proposa així ampliar les funcions de la llista de correu ciutadana que el mateix Albert Cuesta administra i que funciona des de fa set anys. "Com més siguem, més irregularitats aconseguirem recollir", afirma l'autor, que explica que "el propòsit és aconseguir que viure en aquest poble resulti més còmode per als ciutadans, sobretot per als qui no anem en cotxe ni tenim interessos immobiliaris. Es tracta de posar en evidència tant les actituds incíviques d'alguns com la ineficiència general de les administracions públiques".

Aquests **MobLogs** tenen com a plataforma empreses que "ofereixen" els seus serveis, però la seva infraestructura no es basa en WS sinó amb aplicacions dedicades, formularis Web o correus que s'analitzen i s'incorporen a la pàgina. Una infraestructura amb WS donaria la possibilitat d'utilitzar més dispositius i crear clients amb gran facilitat.

Per tant, aquest PFC arriba en un moment ideal per a l'explotació de les idees que poden sorgir de la tecnologia que hem emprat: oferir serveis de MobLogs en temps real, crear comunitats/fòrums amb dispositius mòbils, creació d'un servei d'ajut al treballador (agenda cooperativa, localització de companys, etc), manteniment i consulta de serveis, funcionalitats de domòtica, transmissió de dades des de l'ordinador personal....

També s'obre una important part d'estudi per a intentar millorar el rendiment de les aplicacions i de les llibreries pocketSOAP i kSOAP, així com la futura (esperem) llibreria de Sun, que s'integrarà amb J2ME.

Com a continuació d'aquest projecte, s'està ofertant en aquests moments (desembre-2003) un nou projecte final de carrera, que parteix dels resultats obtinguts d'aquest. Entre les seves tasques hi trobem l'anàlisi del rendiment, seguretat, creació de nous serveis, millora del client Web utilitzant Apache Cocoon, estudi de les noves llibreries de J2ME, ...

12.3 Problemes i sol·lucions

A continuació oferirem una llista de tots els problemes que hem trobat i les seves solucions:

- Mòbil sense suport a MMAPI o sense suport complet.
  - Creació d'un sistema d'enviament d'imatges per correu electrònic a través del mòbil.
- Problemes entre els jocs de caràcters de J2ME i Axis, incapacitat de realitzar la conversió UTF-8 a ISO-LATIN-8.
  - Creació d'una classe que realitza la conversió del text d'UTF-8 a ISO-LATIN-8 (subconjunt del mòbil).
  - Enviament del flux de bytes en UTF-8 automàticament.
- Creació del daemon al servidor de correu (mercury) reiniciava el servidor.
  - Creació d'un procés extern en C++ per a realitzar la tasca ja que no hi havia documentació gratuïta.
- Enviament de correus sense nom de domini.
  - Mercury ofereix la possibilitat, però hi ha servidors d'ISP (el de l'UPC per exemple) que no permeten enviar els correus. La solució és cercar servidors que ofereixin la possibilitat (hotmail), o demanar un nom de domini o subdomini.
- Errors de xarxa i respostes ignorades als clients mòbils.
  - Al treure tota la informació de debug i crear codi més compacte per a utilitzar menys recursos el problema desapareix.
- Seguretat en PHP.
  - Per a que la part WEB fos segura s'ha retirat l'opció del PHP register_globals per a impedir que es passin variables globals a través de l'url.
- Register_globals a off provoca la reestructuració de les variables del codi.
  - Després del desenvolupament es va modificar el codi del PHP per a utilitzar variables de sessió.
- Register_globals a off provoca que no es puguin pujar imatges.
  - Un bug no resolt de PHP no permet que es pugin imatges, per tant s'ha d'activar register_globals; això ja no provoca un problema de seguretat donada la reestructuració de les variables.
- Alguns JPG no retornen la informació de la mida correctament.
  - PHP té un bug que fa que la funció getImageSize() no funcioni correctament; una actualització de la versió de PHP va resoldre el problema.
- L'aspecte gràfic de la façana WEB requereix modificació.
  - Donat que el disseny ha de realitzar-lo un dissenyador WEB, el codi html que es genera té tags d'informació sobre les classes a les que pertany cada part.
  - Es creen CSS (Cascade Style Sheets) per a permetre una modificació global i ràpida de la façana WEB.
- Reducció del tràfic entre el mòbil i el servidor.

    - Per a reduir el tràfic s'ha decidit que molta de la informació que rebem s'emmagatzemi al client en una mateixa execució (llista d'imatges, usuaris, etc...).
- Les plantilles requereixen una sortida HTML configurable.
    - Utilització d'una taula i de la comanda COLSPAN per a repartir el text entre cel·les contigües.
- Proves d'instal·lació requereixen un PC net.
    - Utilització d'emuladors de PC per a simular un entorn recent instal·lat i poder provar els scripts d'instal·lació.
- El servidor de correu pot caure i perdre els missatges.
    - Enviament del correu a un servidor de correu d'un ISP com a mesura de seguretat, donat que no utilitzem més tràfic GPRS al fer-ho i garantim que la imatge no es perdrà.
- Instal·lació al servidor del PFC d'en Rubén Barrio, problemàtica a l'existir el servidor de correu i un servei al port 80.
    - Modificació del PHP i del servidor apache per a permetre la instal·lació a un altre port (9999).
    - Modificació del procés de captura de correu per a funcionar amb cues d'usuari en comptes de la cua global del servidor de correu.
- Axis no serialitza/deserialitza correctament l'enviament de classes compostes.
    - Necessitat de crear un java Bean per a la classe i definir un serialitzador/deserialitzador al deployment.
- L'script d'instal·lació necessita conèixer l'IP de l'ordinador i el sistema ha de ser fàcil d'utilitzar.
    - Creació d'un script amb la capacitat de mostrar totes les adreces IP de l'ordinador i deixar que l'usuari esculli una. Un script en PERL modifica la configuració de l'apache i de Mercury.
- L'usuari ha de realitzar les mínimes accions possibles.
    - Instal·lació dels serveis a windows per a que Apache, Tomcat i MySQL s'engeguin automàticament.
- Problemes d'interoperativitat amb .NET del WS final.
    - Reduir un nivell d'abstració en els Vectors.
- Mòbil Nokia 6100 no pot enviar correus (estàndards) amb attachments.
    - Modificació de l'autòmat de ProcMail per a que pugui capturar missatges sense subject. "Missatge MMS" s'utilitza com a descripció.

12.4 Valoració Personal

Com a coses positives del projecte m'agradaria destacar els coneixements adquirits en molts camps que no solem trobar dins del pla d'estudis. Només cal observar la taula (Taula 2.1 - Elements utilitzats.) per a fer-se una idea de tots els elements que hi intervenen; destacaria entre ells el coneixement de programació amb PHP, eines visuals com EVT o Visual Studio .NET, i els Web Services. Com a punt fort s'ha obtingut un coneixement, que considero molt bo, en la matèria de programació en J2ME, fins i tot de com aprofitar les capacitats multimèdia d'un telèfon mòbil i de tota la problemàtica associada al tenir poca memòria disponible.

Les coses que canviaria si tornés a fer el projecte de nou anirien en dos vessants: la primera seria realitzar un PFC que tingués com a objectiu fer el sistema d'informació complet, ja que el considero molt interessant; i l'altra és al revés, posar més pes a la implementació i estudi dels dispositius en J2ME per tal d'extreure més dades de depuració i de rendiment, per exemple. Però donat el resultat final del projecte que s'ha obtingut, el de tenir una visió generalista de la implementació de Web Services i tot el que l'envolta, el resultat ha estat satisfactori. A més, cal destacar que el sistema és totalment funcional.

13. Bibliografia

Dedicada a J2ME

J2ME in a Nutshell
Kim Topley
O'Reilly
ISBN: 0-596-00253-X

Els documents que hi ha a continuació són de les pàgines dedicades als desenvolupadors que tenen Nokia (www.nokia.com) i Sony-Ericsson (www.sonyericsson.com)

A Brief Introduction to MIDP Clients for Web Services
Forum Nokia
(Fitxer A_Brief_Introduction_to_MIDP_Clients_for_Web_Services_v1_0.pdf)

Java support in Sony Ericsson mobile phones P800/P802/T610
Sony Ericsson
(Fitxer developersguide_p800_t610.pdf)

Camera MIDlet: A Mobile Media API Example
Forum Nokia (Gener 2003)
(Fitxer Camera_MIDlet_A_Mobile_Media_API_Example_v1_0.pdf)

The Developer Platform 1.0 for Series 40 API Overview
Nokia (Juny 2003)
(Fitxer S40_API_over.pdf)

Optimizing the Client/Server Communication for Mobile Applications
Forum Nokia (Febrer 2003)
(Fitxer Optimizing_client_server_part1.pdf i Optimizing_client_server_part2.pdf)

Java MIDP Application Developer's Guide for Nokia Devices
Forum Nokia (novembre 2002)
(Fitxer Java_MIDP_App_Dev_Guide_v1_0.pdf)

Dedicada a Axis

http://www.sosnoski.com/presents/index.html
Conjunt de presentacions sobre Axis, interoperativitat, i Java vs .NET
(Fitxer axis-java-soap.pdf del CD-ROM)

Leveraging open source for web services development
TheServerSide.com (Juny 2003)
Chris Peltz i Claire Rogers. (devresource.hp.com)

(Fitxer WSOpenSource.pdf del CD-ROM)
Cal afegir la propia documentació d'AXIS (http://ws.apache.org/axis/)

Dedicada als Web Services

Web Service Programming Using Axis

Christian Gross
Presentació a O'Reilly Open Source Convention (Juliol 2003)
(Fitxer gross_christian_axis.pdf del CD-ROM)

A comparison of building XML-based web services (J2EE vs .NET)

Chad Vawter i Ed roman (Juny 2001)
(Fitxer J2EE-vs-DotNET.pdf)

The Java Web Services Tutorial

Eric Armstrong, Stephanie Bodoff, Debbie Carson *et al.*
Un tutorial que surt de l'àmbit del projecte, però que abarca la creació de WS sense Axis, utilitzant J2EE
JavaWSTutorial.pdf

Dedicada a PHP

Guia Esencial PHP

Christopher Cosentino
ISBN: 84-205-3326-2
Prentice Hall

http://www.php.net/

Projectes externs relacionats :

Analysis of Java Applications on Mobile Devices Based on a Prototype for Sony Image Station

Adrian Jochinger
Juliol 2003
(Fitxer Jochinger.2003.pdf)

http://www.cs.ust.hk/~scc/csit510/proposal/
Universitat de Hong Kong de Ciència i Tecnologia (HKUST)
Conté diferents treballs (pràctiques) d'un curs de desenvolupament de Software, relacionats en aplicacions d'E-Business. (COMP610E)
(http://www.cs.ust.hk/~scc/comp610e/)

Entre elles trobem el disseny d'un sistema mòbil de compra de tickets.
http://www.cs.ust.hk/~scc/csit510/proposal/kelvin.progress.pdf
http://www.cs.ust.hk/~scc/csit510/proposal/ip.pdf

Podem trobar que el software utilitzat és molt semblant al que s'utilitzarà, en excepció d' Apache Soap, que s'ha substituït per Apache Axis.

14. ANNEX-A – dispositius J2ME

14.1 Característiques dels dispositius J2ME

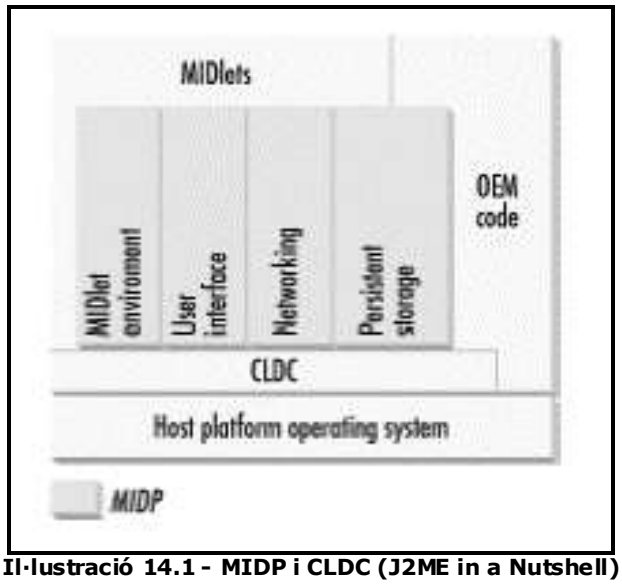
(Ref – J2ME in a Nutshell)

CLDC – Connected Limited Device Configuration
CLDC especifica el mínim conjunt de classes i funcionalitat de Java que pot implementar-se en un dispositiu d'aquestes característiques.

Trobem, per exemple, que només podem accedir a les següents col·leccions de dades: Hashtable, stack, Vector, Enumeration.

MIDP – Mobile Information Device Profile.

Els perfils (profiles) afegeixen classes adicionals al dispositiu; en aquest cas MIDP afegeix classes per a utilitzar la xarxa, interfície gràfica, i emmagatzament.



Il·lustració 14.1 - MIDP i CLDC (J2ME in a Nutshell)

Els telèfons de que disposem tenen el perfil MIDP 1.0, aquest perfil té els següents requeriments de maquinari :

Memòria :
- 128 Kb de memòria RAM per a guardar-se ella mateixa.
- 32 Kb de memòria per a la pila de Java
- 8 Kb de memòria no volàtil per a poder emmagatzemar informació

Display :
- 96 (ample) x 54 (alt) píxels.
- 2 colors.

Dispositius d'entrada :

- Possibilitat d'entrar números del 0 al 9, tecla de selecció i 4 botons per a moure el cursor (o equivalents)

Xarxa :

- Suport per a HTTP 1.1ç

14.2 Característiques de la màquina virtual kVM

Kilobyte Virtual Machine és una maquina virtual de funcionalitat reduïda que utilitza poca memòria i té un garbage collector optimitzat per a dispositius amb poca memòria.

Les classes incorporades de J2SE a J2ME han de ser similars, és a dir, han de tenir el mateix comportament, però poden ometre algunes funcionalitats encara que mai afegir.

Entre les característiques retallades hi trobem, per exemple, el suport per a punt flotant, la classe java.lang.reflect, java.lang.ref, JNI...

14.3 Compatibilitat amb dispositius amb el mateix MIDP

Cal destacar que podem trobar problemes de compatibilitat, ja que per exemple **Nokia** ha afegit algunes classes java al seu perfil per tal d'aprofitar al màxim aquests dispositius (accedir directament a la memòria de video, per exemple).
Per tant, ens hem d'allunyar el màxim possible d'aquestes classes.

15. ANNEX B – Desenvolupament amb Axis

15.1 Web Services amb Axis, cas pràctic

Apache Axis és un servlet, executat a través de Apache Tomcat, que permet servir Web Services. Concretament, és una implementació del protocol SOAP:

```
SOAP is a lightweight protocol for exchange of information in a decentralized, distributed
environment. It is an XML based protocol that consists of three parts: an envelope that
defines a framework for describing what is in a message and how to process it, a set of
encoding rules for expressing instances of application-defined datatypes, and a convention for
representing remote procedure calls and responses.
```

Podem descarregar la seva versió 1.1 final de http://ws.apache.org/axis/download.cgi

15.2 Instal·lació

Prerequisits :

- Apache Tomcat 4.1.x (Full)
- Java 1.3 o superior

```
<DIR>          docs
<DIR>          lib
         2.624 LICENSE
           443 README
         2.634 release-notes.html
<DIR>          samples
<DIR>          webapps
<DIR>          xmls
```

Copiem el contingut del directori **webapps** al directori **webapps** del apache tomcat.

Ara necessitarem un parser XML, el podem descarregar d'aquí http://xml.apache.org/dist/xerces-j/ i instal·lar-lo a axis\WEB-INF\lib
Finalment crearem les variables d'entorn necessàries:

```
AXIS_HOME = H:\Archivos de programa\Apache Group\Tomcat 4.1\webapps\axis

AXIS_LIB = %AXIS_HOME%\WEB-INF\lib

AXISCLASSPATH = %AXIS LIB%\axis.jar;%AXIS LIB%\commons-discovery.jar;%AXIS LIB%\commons-
logging.jar; %AXIS_LIB%\jaxrpc.jar; %AXIS_LIB%\saaj.jar; %AXIS_LIB%\log4j-
1.2.8.jar;%AXIS_LIB%\xml-apis.jar;
%AXIS_LIB%\xercesImpl.jar;%AXIS_LIB%\wsdl4j.jar;%AXIS_LIB%\mysql.jar
```

15.3 Prova

Podem carregar la pàgina inicial d'axis fent :
http://localhost:8080/axis/index.html

i pulsant *Validate the local installation's configuration* ens assegurarem de que tot funciona bé.

http://localhost:8080/axis/services/Version?method=getVersion

```
<soapenv:Envelope>
        <soapenv:Body>
        <getVersionResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <getVersionReturn xsi:type="xsd:string">
Apache Axis version: 1.1
Built on Jun 13, 2003 (09:19:43 EDT)
</getVersionReturn>
</getVersionResponse>
</soapenv:Body>
</soapenv:Envelope>
```

Una de les característiques d'Axis més útils en el procés de desenvolupament és la possibilitat de cridar als mètodes d'un WS a través de l'URL. En el cas anterior, hem cridat al mètode *getVersion* del WS *Version.*

15.4 Deployment:

Una vegada tenim un jar amb el WS que volem publicar, el copiem al directori /axis/web-inf/lib.

Reiniciem el servidor d'aplicacions, i realitzem la tasca de deployment amb descriptor:

```
java -cp "%AXISCLASSPATH%" org.apache.axis.client.AdminClient deploy.wsdd


Processing file deploy.wsdd
<Admin>Done processing</Admin>
```

Tornem a reiniciar el servidor d'aplicacions per que la nova configuració es trobi. També podem copiar les classes desempaquetades al directori /axis/web-inf/classes.

15.5 Implementació de Web Services.

Els Web Services d'Axis són simples classes Java, que tenen com a única restricció (si no volem especificar les funcions de serialització / deserialització) que s'han d'utilitzar els tipus definits:

xsd:base64Binary	byte[]
xsd: boolean	boolean
xsd:byte	byte
xsd:dateTime	java.util.**Calendar**
xsd:decimal	java.math.**BigDecimal**
xsd:double	double
xsd:float	float
xsd:hexBinary	byte[]
xsd:int	int
xsd:integer	java.math.**BigInteger**
xsd:long	long
xsd:QName	javax.xml.namespace.**QName**
xsd:short	short
xsd:string	java.lang.**String**

Taula 15.1 - Tipus de Dades d'Axis

Com a col·leccions d'objectes és recomanable fer servir el tipus Vector (java.util.vector)

15.5.1 Exemple

Anem a crear un WS que formarà part d'un package a es.exemple i que tindrà com a funció retornar una llista {url, descripció} de tots els links d'una pàgina web, especificada com a paràmetre.

Creem una estructura de directori similar a aquesta:
/LinkGet/es/exemple/
A exemple hi posarem el codi de les classes java i Descriptor que tindrà el format d'un Java Bean amb les propietats, URL i Desc.
I LinkGet, que tindrà el codi del WS.

```
package es.exemple;

public class Descriptor {
	String sURL = null;
	String desc = null;

	public Descriptor () { }

	public void setURL ( String sUrl) { sURL = sUrl; }
	public String getURL ( ) { return (sURL); }

	public void setDesc ( String sDesc) { desc = sDesc; }
	public String getDesc ( ) { return (desc); }

}
```

Aquesta classe la tindrem que replicar al client (depenent de la llibreria SOAP utilitzada), per tal de que es pugui deserialitzar/serialitzar correctament.

La resta del codi es pot trobar dins del CD-ROM que s'inclou amb aquest document.

Observem que la funció retorna un tipus Vector, però no s'especifica quin tipus conté. Per tant, hem d'especificar a Axis el tipus que pot contenir, ja que sinó podem tenir un error.

Per compil·lar i crear el .jar automàticament utilitzem Jakarta-Ant, amb un build.xml similar al que trobem al CD-ROM.

Això ens col·locarà el jar al directori \axis\WEB-INF\lib\. Podríem també copiar les classes compilades al directori \classes\, amb l'estructura de directoris corresponent.

Solament ens manca realitzar el deployment, i per fer-ho utilitzarem dos fitxers:
Deploy.wsdd i undeploy.wsdd

En el fitxer de deploy hi remarquem les característiques del WS que volem publicar; entre elles trobem, per exemple, les funcions que volem fer visibles, els mappings de tipus i les classes que s'utilitzen. També podem configurar com serà la sortida del xml.

```
<deployment name="LinkGet"
    xmlns="http://xml.apache.org/axis/wsdd/"
    xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">

  <service name="LinkGet" provider="java:RPC" >
            <parameter name="className" value="es.exemple.LinkGet"/>
            <parameter name="allowedMethods" value="*"/>
            <parameter name="sendMultiRefs" value="false"/>

        <typeMapping
         xmlns:ns="http://LinkGet"
         qname="ns:Descriptor"
         type="java:es.exemple.Descriptor"
         serializer="org.apache.axis.encoding.ser.BeanSerializerFactory"
         deserializer="org.apache.axis.encoding.ser.BeanDeserializerFactory"
         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
       />
  </service>
</deployment>
```

```
<typeMapping
        xmlns:ns=http://LinkGet     → Hauría de ser la URL del nostre servidor
        qname="ns:Descriptor"        → Nom que apareixerà al xml
        type="java:es.exemple.Descriptor"  → Classe que la implementa
       // Es un java Bean, per tant utilitzem el De/serialitzador d'Axis.
```

```
            // Aquí podríem especificar els nostres propis.
  serializer="org.apache.axis.encoding.ser.BeanSerializerFactory"
        deserializer="org.apache.axis.encoding.ser.BeanDeserializerFactory"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      />
```

L'undeploy es simplement això :

```
<undeployment name="LinkGet" xmlns="http://xml.apache.org/axis/wsdd/">
  <service name="LinkGet"/>
  </undeployment>
```

Una vegada ho tenim tot fem :

java -cp "%AXISCLASSPATH%" org.apache.axis.client.AdminClient deploy.wsdd
o
java -cp "%AXISCLASSPATH%" org.apache.axis.client.AdminClient undeploy.wsdd

Ara podem reiniciar el servidor d'Axis (o tot el Tomcat) i podrem veure, si tot a anat bé, el nostre WS aquí:

http://localhost:8080/axis/servlet/AxisServlet

Si cliquem sobre el WSDL del WS LinkGet, veurem com el deployment ha fet que al WSDL es defineixi el tipus Descriptor, tot i no sortir al codi.

http://localhost:8080/axis/services/LinkGet?wsdl

Podem aprofitar la característica d'Axis per testejar el WS que em creat.

http://localhost:8080/axis/services/LinkGet?method=getVersion

```
<soapenv:Envelope>
<soapenv:Body>
<getVersionResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<getVersionReturn xsi:type="xsd:string">LinkGet V.1.0</getVersionReturn>
</getVersionResponse>
</soapenv:Body>
</soapenv:Envelope>
```

http://localhost:8080/axis/services/LinkGet?method=getLinks&sURL=http://www.fib.upc.es

```
<soapenv:Envelope>
       <soapenv:Body>
       <getLinksResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
       <getLinksReturn xsi:type="ns1:Vector">
<item xsi:type="ns2:Descriptor">
<URL xsi:type="xsd:string">http://www.upc.es</URL>
<desc xsi:type="xsd:string"/>
```

```
</item>
<item xsi:type="ns3:Descriptor">
<URL xsi:type="xsd:string">http://www.fib.upc.es/ca</URL>
<desc xsi:type="xsd:string"/>
</item>
<item xsi:type="ns4:Descriptor">
<URL xsi:type="xsd:string">http://raco.fib.upc.es</URL>
<desc xsi:type="xsd:string"/>
</item>
...
</getLinksReturn>
</getLinksResponse>
</soapenv:Body>
</soapenv:Envelope>
```

15.6 Altres formes de fer deployment

Si la nostra classe és senzilla la podem col·locar (sense compil·lar) al directori axis\, però amb l'extensió .jws.

Si apuntem amb el navegador a http://localhost:8080/axis/EchoHeaders.jws (p.ex) aquesta classe es compil·larà i podrem veure el seu WSDL http://localhost:8080/axis/EchoHeaders.jws?wsdl i realitzar totes les funcions habituals.

15.7 Altres configuracions

Al fitxer \axis\WEB-INF\server-config.wsdd hi podem configurar diversos aspectes, que poden fer els nostres Web Services més eficients.

Un d'ells és <parameter name="sendMultiRefs" value="true"/>; si el deixem com està per defecte, les dades xml ens arribaran de la següent manera:

```
<soapenv:Envelope>
<soapenv:Body>
        <getCatImaResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<getCatImaReturn href="#id0"/>
</getCatImaResponse>

<multiRef id="id0" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="ns1:Vector">
<item href="#id1"/>    <item href="#id2"/>

</multiRef>

<multiRef id="id1" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="ns2:WSInfoCategoria">
<idCat xsi:type="xsd:int">1</idCat>
<nomCat xsi:type="xsd:string">Per Defecte</nomCat>
</multiRef>

<multiRef id="id2" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="ns3:WSInfoCategoria">
<idCat xsi:type="xsd:int">2</idCat>
<nomCat xsi:type="xsd:string">Personal</nomCat>
</multiRef>

</soapenv:Body>
</soapenv:Envelope>
```

Aquesta manera és útil si les dades es repeteixen (cosa que no passa en el nostre sistema), però afegeix un overhead si no ho fan.

```
<soapenv:Envelope>
        <soapenv:Body>
        <getCatImaResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <getCatImaReturn xsi:type="ns1:Vector">
        <item xsi:type="ns2:WSInfoCategoria">
<idCat xsi:type="xsd:int">1</idCat>
<nomCat xsi:type="xsd:string">Per Defecte</nomCat>
```

```
</item>
        <item xsi:type="ns3:WSInfoCategoria">
<idCat xsi:type="xsd:int">2</idCat>
<nomCat xsi:type="xsd:string">Personal</nomCat>
</item>
</getCatImaReturn>
</getCatImaResponse>
</soapenv:Body>
</soapenv:Envelope>
```

16. ANNEX C – Creació paquet instal·lació

16.1 Creació del paquet autoinstal·lable

Hem utilitzat el generador **Nullsoft Scriptable Install System**[1] per a crear els scripts i paquets necessaris per a la instal·lació del programari en un ordinador personal.

Els passos per a la instal·lació són els següents:

- Instal·lar Apache (Servidor Web)
- Instal·lar PHP (i modificar http.conf)
- Instal·lar mySQL
- Instal·lar/connectar els serveis a l'ordinador.
- Instal·lar el client web
- Punt de control → Testeig del client WEB

- Instal·lar Apache Tomcat
- Instal·lar Apache Axis
- Instal·lar els Web Service
- Instal·lar/connectar els serveis a l'ordinador.
- Punt de control → Testeig de la banda WS.

- Tasques addicionals

    - Instal·lació servidor de correu
    - Instal·lació de procMail
    - Instal·lació arxiu de propietats per als WS (ha d'estar a C:\ )

La principal dificultat d'aquest procés va ser estudiar un nou llenguatge i trobar les eines necessàries per a realitzar la tasca de trobar la IP de l'ordinador i instal·lar els serveis a Windows.

Al CD-ROM que acompanya el document hi podem trobar l'script creat per a generar el fitxer, així com les versions del programari utilitzat. Cal remarcar que la instal·lació, tot i que s'ha provat en diferents sistemes, pot fallar, i per tant recomano la seva instal·lació manual, ja que és més enriquidora.

16.2 PHP d'autoinstal·lació de la base de dades

Per tal de poder realitzar les tasques d'instal·lació de la base de dades des de zero, i poder crear l'usuari administrador del sistema, es va realitzar una part Web amb PHP que realitzava el procés mitjançant un navegador.

[1] http://nsis.sourceforge.net/site/index.php

PHP disposa de comandes dins del seu SDK que permeten realitzar qualsevol tasca sobre mySQL. Aquestes tasques van des de les consultes (query's) típiques fins els create database del SQL estàndard.

Mostrem un petit fragment del PHP:

```
$link = mysql_connect($db_WHER,$db_USER,$db_PASS) or mysql_die ("Unable to connect");

mysql_query("drop database ".$db_NAME);

mysql_query("create database ".$db_NAME) or mysql_die("Create database error:
".mysql_error());mysql_select_db($db_NAME,$link) or mysql_die ("Unable to connect to
database");

mysql_query("
  CREATE TABLE autor (nom varchar(30) NOT NULL default '',
  contrasenya varchar(8) NOT NULL default '',  email varchar(255) default NULL,
  permis varchar(10) NOT NULL default '',  PRIMARY KEY  (nom))
      ")or die("Create table 1 Error: ".mysql_error());
```

1 - Ens connectem al SGBD.
2 – Destruïm la base de dades si existia.
3 – Creem la base de dades (comprovació de errors per a cada pas a partir d'ara).
4 – Ens connectem a la base de dades creada.
5 – Introduïm una taula utilitzant SQL estàndard (en negreta).

Com observem és una tasca senzilla i que dona un resultat molt professional. A l'acabar de crear les taules saltem a un altre formulari per a crear el primer usuari administrador, les categories i les plantilles per defecte.

17. ANNEX D – Manual d'ús

17.1 Manual del Procés d'instal·lació

El sistema pot funcionar sense cap dificultat en PC's obsolets (< Pentium) funcionant amb Windows 98, per exemple, o Linux.

17.1.1 Windows

Prerequisits: Tenir instal·lat el JDK de Java.

Si utilitzem el paquet autoinstal·lable creat, a l'executar-lo ens demanarà seleccionar en primer lloc el directori destí; recomanem utilitzar un directori curt i sense espais al nom (*c:\Butlleti* o fins i tot *c:*).

En segon lloc es mostraran les diferents adreces IP de l'ordinador on s'instal·la el servidor. Si no en detecta cap, agafarà *localhost* per defecte.

Si tenim problemes en la detecció de l'adreça IP, recomanem que s'editi el fitxer */apache/conf/httd.conf* i es realitzi una cerca per a "*serverName localhost*", per tal de substituir localhost per l'adreça IP o el nom de domini.

Aquesta adreça IP també s'utilitzarà en el servidor de correu; si el necessitem utilitzar i no funciona correctament o tenim la sort de tenir un nom de domini, podem modificar el fitxer *MERCURY.INI* situat al directori d'instal·lació dins la carpeta **MERCURY** i canviar l'adreça IP seleccionada pel nom del host, o la IP de sortida (cal recordar que l'adreça de correu en cas de que el domini sigui una adreça IP és la següent *postmaster@[xxx.xxx.xxx.xxx]*). El servidor de correu que s'instal·la està preparat per a que s'agafin els correus de la cua directament; si volem crear diferents usuaris és recomanable instal·lar des de l'aplicació creada pel desenvolupador.

En un altre punt de la instal·lació es copia el fitxer *WSImatge.properties* al directori *c:\;* si no volem que sigui així no tenim més remei que recompil·lar els WS. En aquest *WSImatge.properties* hi podem escriure les dades de la base de dades per al WS.

La instal·lació *ens pot demanar* el nom d'usuari i la contrasenya per la base de dades, en aquest punt hi tenim que col·locar com a user: *mavy* i com a contrasenya : *pfc2003* (cas per defecte).

Per resumir, si canviem alguna de les dades cal modificar una sèrie de fitxers per tal d'adequar la configuració:

- apache/htdocs/install/config.php
- apache/htdocs/PFCButlletí/config.php
- mercury/mercury.ini
- c:\WSImatge.properties o \WSImatge.properties en la versió Linux
- Els .jad del client mòbil
- I finalment el link a ProcMail que utilitza els paràmetres per defecte (recordar que l'últim paràmetre indica si es cerca al directori QUEUE o al directori de l'usuari).

Si tot ha anat correctament amb la instal·lació hem d'observar una pantalla del navegador d'Internet amb una pàgina de presentació.

En aquest moment s'ha creat la base de dades i les taules, i obtenim un formulari amb dos camps per a introduir les dades (user/contrasenya) de l'usuari administrador al sistema.

La instal·lació mentrestant continuarà amb el servidor Tomcat, Axis i els WS. Se'ns obrirà una altra finestra amb el WSDL d'un WS si tot ha anat correctament, i sinó apareix (el nostre sistema és lent) hem d'esperar i prémer el botó de recarrega.

Si realitzem la instal·lació manual hem de seguir els següents punts:

- Instal·lar **AppServ** (Paquet que instal·la **Apache**, **MySQL** i **PHP**) o cada component per separat.
- Copiar el client **WEB** (modificant les dades al *config.php*).
- Executar http:\\localhost\Install\install.php per a instal·lar la base de dades i afegir l'usuari Administrador.
- Recordar que cal modificar el php.ini de \windows per a activar el register_globals=On per a que funcioni la transferència d'imatges des de PHP.

- Instal·lar **J2DK**.
- Instal·lar **Tomcat**.
- Copiar directori **Axis**.
- Copiar el **WS** (realitzant el deploy, i fent un restart de Tomcat) si no hem copiat el directori AXIS del CDROM.
- Testejar WS amb el WSDL. (modificant el *WSImatge.properties* copiat a la C:\).

- Instal·lar **Mercury**.
- Configurar el servei de correu (Solament server SMTP).
- Instal·lar **ProcMail** i executar-lo amb els paràmetres correctes.

ProcMail C:\MERCURY\QUEUE\ mavy pfc2003 butlleti localhost 50000 0

On:

- *C:\MERCURY\QUEUE* és la cua de missatges del servidor.
- *Mavy* és l'user de la base de dades.
- *Pfc2003* és la contrasenya de la base de dades.
- *Butlleti* és el nom de la base de dades.
- *Localhost* és l'adreça IP de la base de dades.
- *50000* és el nombre de milisegons entre cada comprovació de la cua.
- I l'últim paràmetre al ser 0 ens indica que cerquem els missatges directament de la cua, si fos 1 l'indicariem el directori del correu de l'usuari i cercaria els missatges allí.

17.1.2 Linux

Material:

MySQL
Descarregar des de www.MySQL.com la versió 4.0 Standard per a linux.

Apache
Descarregar des de www.apache.org

- Versió 1.3.29 de l'Apache http Server
- 4.1 d' Apache Tomcat : http://jakarta.apache.org/builds/jakarta-tomcat-4.0/nightly/

PHP
Descarregar de PHP.net la versió 4.3.4

J2SDK
http://java.sun.com/j2se

Passos:

Instal·lació de mySQL :
Seguir els passos indicats a *INSTALL*, i afegir si es vol un nou usuari "*mavy*" amb contrasenya "*pfc2003*" o qualsevol altre modificant els fitxers de configuració de la WEB i el properties dels Web Services.

Instal·lació de Apache
Seguir els passos indicats a *INSTALL*, però afegint al configure *–enable-module=so*
Modificar el port al httpd.conf o al config.php de la WEB per a que siguin iguals.

Instal·lació de PHP
A l'igual que els altres components seguirem les instruccions de *l'INSTALL*. Cal remarcar la necessitat de modificar el httpd.conf per a afegir un nou tipus (*AddType*); per a les pàgines PHP (tal com indica *l'INSTALL*) també necessitarem afegir *index.php* com a nou document inicial (realitzarem una cerca per *index.html* i ho trobarem).

També necessitem copiar el *php-ini.dist* al directori */usr/local/lib/* com a *php.ini*; cal afegir la modificació de **register_globals=On** per a que funcioni correctament l'upload d'imatges.

Finalment, copiarem el contingut de la **WEB** al directori *htdocs* i apuntarem amb un navegador a http://localhost/Install/install.php per a realitzar la instal·lació inicial. Si no funciona correctament hem de modificar el *config.php* del directori *Install* i del *PFCButlleti*.

Si volem utilitzar *phpMyAdmin* per a administrar la base de dades hem de renombrar *phpadmin* a *phpMyAdmin* per a que funcioni correctament i modificar el fitxer *config.ini.php* per a adequar-lo a la configuració.

Banda WS

Instal·larem el **J2SDK 1.4.1**

Comprovarem que està tot correctament instal·lat utilitzant la comanda *set | grep JAVA* i que el JAVA_HOME apunta al directori d'instal·lació.

Instal·lem Tomcat
Copiem el directori **AXIS** (contingut al */Usuari/Deploy Axis*) al *webapps* de **Tomcat**, copiem també el *WSImatge.properties* a l'arrel de l'arbre de directoris i el configurem correctament.

Íniciem **Tomcat** i ja podem utilitzar els WS.

Per a iniciar ens col·loquem al directori del tomcat/bin i executem l'script *startup.sh*. Si es queixa del **JAVA_HOME** hem d'assignar-lo amb:
export JAVA_HOME=/usr/java/j2sdk1.4.2_03

Una vegada tenim tot el sistema instal·lat i els serveis aixecats podem començar a utilitzar-lo.

17.1.3 Clients Pocket PC

Per a instal·lar copiarem el .CAB corresponent al dispositiu i l'executarem des del Pocket PC. Cal instal·lar abans el .NET Compact Framework.

17.1.4 Clients J2ME

Mitjançant infrarrojos, cable sèrie o Bluetooth copíarem el .JAR i el .JAD al mòbil. Si no podem copiar el .JAD utilitzarem l'SDK del fabricant per a realitzar la tasca (sinó no tindríem usuari/contrasenya i URL del WS per defecte). Cal remarcar la necessitat de modificar el .JAD amb un editor de text abans de pujar-lo, per tal d'adequar-lo a la configuració.

17.2 Client WEB

17.2.1 Usuari no registrat

Si vostè és un usuari no registrat al sistema, o vol actuar com a tal, pot realitzar les següents accions:

- Visualitzar Notícies
- Visualitzar Imatges
- Visualitzar Plantilles

Per a accedir a la pàgina web, ha d'utilitzar un navegador d'internet amb accés a la xarxa i accedir a l'adreça:

http://localhost:9999/PFCButlleti/index.html (canviant localhost per l'adreça del servidor) o http://localhost/PFCButlleti/index.html si la instal·lació és al port estàndard.

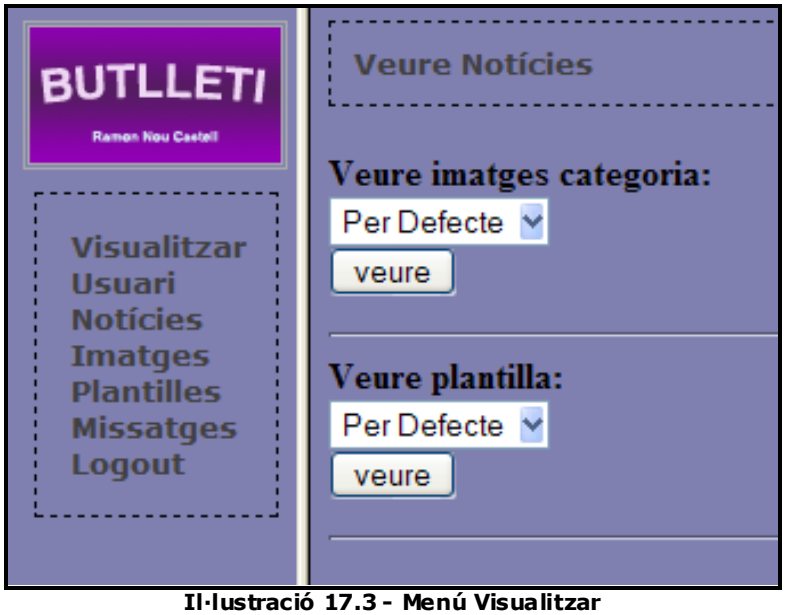
Il·lustració 17.1 - Pàgina principal

A la banda esquerra tenim el menú de navegació principal de la web i a la banda superior tenim el menú de navegació del subapartat seleccionat. En el cas d'un usuari no registrat només podem utilitzar el menú visualitzar.

Il·lustració 17.2 - Visualitzar notícies

A la *il·lustració 17.2* podem veure els controls de la visualització de les notícies. A **Num Notícies** podem escriure el número de notícies a veure per pàgina (premi valida per a acceptar). Els controls inferiors (**'<<','<','>'**) permeten navegar per les notícies. Finalment, amb l'apartat del menú **Veure Notícia** podrem visualitzar una de les notícies.

Menú Visualitzar



Il·lustració 17.3 - Menú Visualitzar

En aquest menú podem escollir una categoria i prémer veure per a visualitzar les imatges de la categoria. Podem realitzar el mateix amb les plantilles.

Passem ara a l'ús per part dels usuaris registrats.

Logar-se
Per a logar-se només tenim que prémer en l'apartat Usuari del menú de l'esquerra.

Il·lustració 17.4 - Menú usuari

Si ens volguéssim registrar només cal que introduïm un nom d'usuari i premem login, automàticament ens sortirà el formulari d'inscripció al sistema.

17.2.2 Usuari Registrat

Un usuari registrat pot enviar i rebre missatges d'altres usuaris, una vegada logat.

Al menú usuari podem veure el número de missatges que tenim per a llegir; també podem accedir directament utilitzant el menú Missatges.

Il·lustració 17.5 - Menú Missatges

Tenim l'opció d'Afegir Missatge, Esborrar i Llegir Missatges i modificar els missatges que hem creat.

Il·lustració 17.6 - Creació d'un missatge.

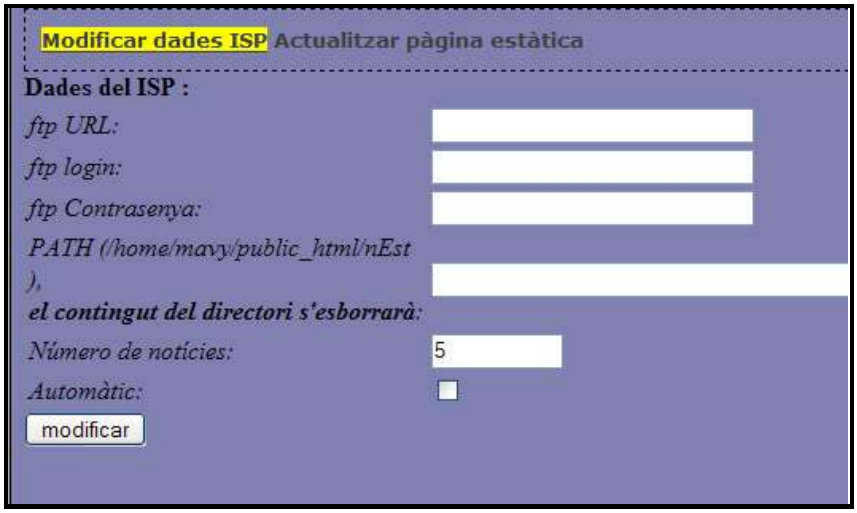
Per a enviar un missatge a un usuari hem de seleccionar l'usuari de la llista desplegable, escriure el missatge i finalment clicar sobre inserir.

17.2.3 Usuari Registrat (ISP)

Si tenim permisos d'ISP, al menú usuari hi trobarem l'opció d'actualitzar la pàgina estàtica.

Il·lustració 17.7 - Actualització de la pàgina estàtica

Quan entrem a aquesta opció (i si és la primera vegada que entrem) hem d'introduir les dades de l'ISP a actualitzar (modificar dades ISP).



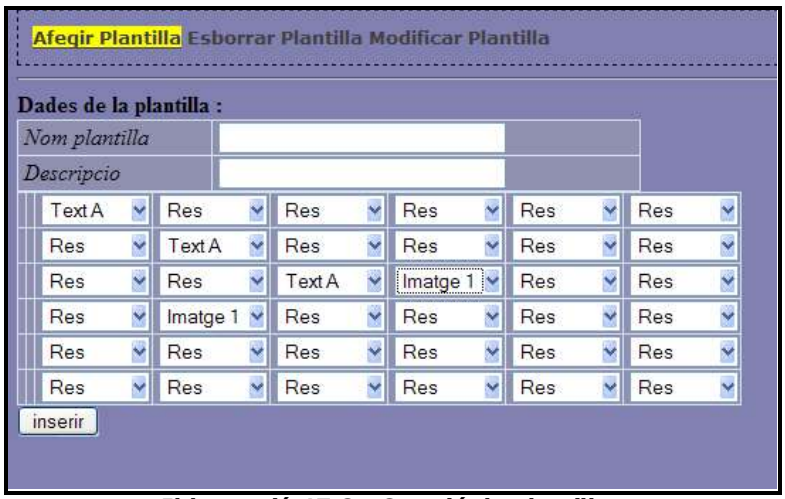
Les dades que hem d'introduir són la url del ftp (ftp.terra.es), el nom d'usuari (ftp login), la contrasenya i el path complert (no cal que existeixi el directori final).

Hem d'anar en compte, ja que s'esborrarà el directori cada cop que actualitzem. Finalment, podem especificar el número de notícies que volem que apareguin i si volem que s'actualitzi quan l'administrador realitzi una actualització de tots els ISP (automàtic).

Quan hem introduït aquestes dades correctament podem utilitzar l'opció Actualitzar pàgina estàtica per a actualitzar el directori que hem assenyalat (es crearà un index.html, la fulla d'estils i una sèrie d'imatges). Aquesta pàgina pot trigar bastant en carregar completament; hem de deixar-la oberta fins que finalitzi l'operació d'actualització.

17.2.4 Usuari Registrat (Plantilles)

Si volem crear, modificar o esborrar plantilles hem de tenir permisos per a fer-ho. Quan l'administrador ens els doni podrem accedir a la part Plantilles.



Il·lustració 17.8 - Creació de plantilles

La il·lustració anterior mostra l'aspecte de l'apartat de creació d'una plantilla. En ella hem d'introduir el nom de la plantilla i una descripció; posteriorment, amb les llistes despregables que representen un taulell, seleccionar quins elements volem que apareguin. Cal recordar que la distribució del text es fa d'esquerra a dreta primer (si volem fer un text amb columnes, cada columna ha de ser d'un text diferent).

En aquest taulell podem repetir elements d'imatges, com representem en la il·lustració.

Cal recordar que si no col·loquem tots els elements de text o d'imatges, si la notícia té més elements aquests no apareixeran.

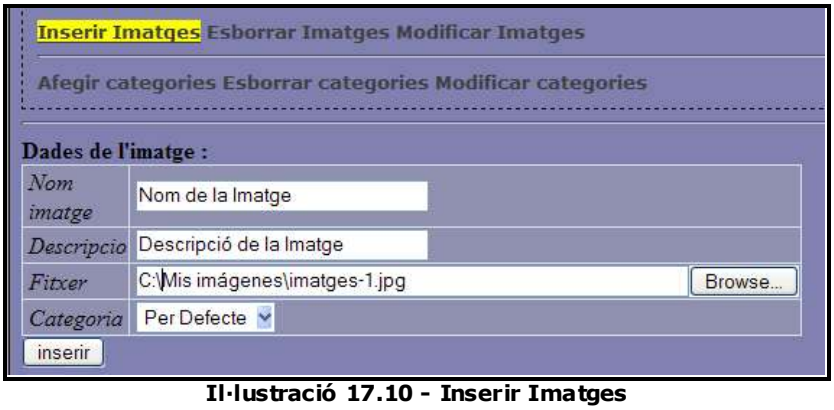
17.2.5 Usuari Registrat (Imatges)

En l'apartat Imatges podem crear / modificar i esborrar imatges, així com les categories on es classifiquen.

Il·lustració 17.9 - Menú Imatges

La part dedicada a les categories és molt senzilla. Cal remarcar que si esborrem una categoria les imatges passaran a forma part de la categoria per defecte (que no es pot esborrar).

Per a inserir imatges polsarem la primera opció del menú i omplirem les dades necessàries que ens demanen. Observem a continuació un exemple:



Il·lustració 17.10 - Inserir Imatges

L'apartat de modificar és similar al d'inserir.

17.2.6 Usuari Registrat (Notícies)

La part més complicada de la web és la construcció de notícies; en la següent imatge mostrem el menú Notícies que surt en prémer sobre Inserir Notícies:

Il·lustració 17.11 - Inserir Notícies

En primer lloc escriurem el títol i el subtítol de la notícia i seleccionarem la Plantilla; quan hem acabat polsarem sobre inserir.

Il·lustració 17.12 - Títol i subtítol acceptats.

Una vegada tenim el títol i el subtítol acceptats hem d'introduir el contingut de la notícia. Si volem introduir Text utilitzarem el botó **veure Text**, i si volem passar a introduir directament imatges utilitzarem el botó **Fi Text.**

Il·lustració 17.13 - Introduint elements de text.

Escriurem el text que vulguem inserir a la notícia, seleccionarem l'element (A, B, C, D o E) i finalment utilitzarem el botó Afegir Text per a inserir-lo (apareixerà A(Ocupada)) o Veure Text si volem passar a una posició ocupada.

Finalment utilitzarem el botó Fi text per a passar a la introducció d'imatges.

L'aspecte del formulari per a la introducció de les imatges és el següent:

Il·lustració 17.14 - Introducció d'imatges a la notícia.

En la primera llista desplegable podem seleccionar la categoria de les imatges; quan la canviem hem de prémer el botó Canvi Categoria per a recarregar el formulari i que mostri les imatges de la nova categoria. Si seleccionem una imatge i utilitzem el botó Veure imatge previsualitzarem la imatge. Finalment, per a inserir la imatge a la posició (1 a 6) hem de seleccionar tant la imatge com la posició i prémer Afegir Imatge.

Quan hem finalitzat el procés ja podem utilitzar el botó Fi Imatge per a finalitzar la introducció de la notícia.

17.2.7 Usuari Registrat (Administrador)

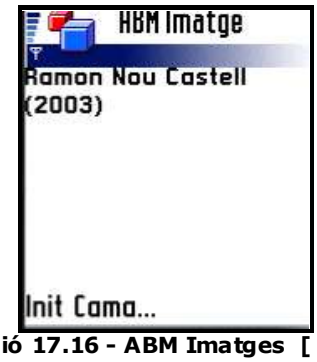
L'administrador té alguns elements més al menú usuaris, entre ells hi ha el de donar o treure permisos als usuaris o d'esborrar usuaris.

Il·lustració 17.15 - Modificació d'usuaris.

Finalment també té disponible a la part d'actualització automàtica una nova opció: actualitzar totes les pàgines estàtiques realitza una actualització massiva a tots els ISP's que tenen l'opció d'actualització automàtica activada.

17.3 Client Mòbil Imatges

Una vegada realitzada la instal·lació al mòbil de l'aplicació, prèviament amb la modificació del .jad (usuari, contrasenya i url) per a utilitzar les dades correctes en la connexió, podem executar-lo.



Il·lustració 17.16 - ABM Imatges [Principal]

Pulsarem el botó Init Camara per a iniciar la captura de l'imatge. En aquest punt podem enquadrar el mòbil i prémer el botó de captura (definit pel mòbil), fins que estiguem satisfets amb la imatge (el mòbil guardarà la última solament).

Il·lustració 17.17 - ABM Imatges [Captura]

Si la imatge és satisfactòria podem tornar enrera i escollir la nova opció **Dades Imatge** (el mòbil es connecta a la xarxa per a descarregar la llista de categories).

Il·lustració 17.18 - ABM Imatges [Dades Imatge]

En aquesta nova pantalla hem d'introduir les dades corresponents al nom d'usuari / contrasenya (si no estaven bé al .jad), així com les dades relacionades amb la imatge (nom i descripció). Si premem el botó Categoria podem escollir una categoria de les disponibles al sistema per a classificar les imatges.

Quan haguem introduït totes les dades les acceptarem quan premem Acepta Dades i tornarem al menú principal amb dues opcions més. Simula Enviament realitza tot el procés d'enviament, però sense enviar el flux de dades per a comprovar si hi pot haver algun problema. Envia Imatge, finalment, realitza l'enviament de la imatge mitjançant la xarxa cap al servidor.

17.4 Client Mòbil Missatges

Aquesta aplicació s'instal·la juntament amb el client de Notícies, per tant el .jad amb la configuració és compartit.

Iniciem l'aplicació i utilitzem l'opció Dades Usuari on introduirem, si és necessari, les dades de l'usuari i l'adreça IP:port del servidor (80.25.193.223:8080 p.ex) .

Il·lustració 17.19 - Missatges - [Dades Usuari]

Una vegada acceptem les dades podrem escollir dues opcions noves: llegir missatges (Llista Llegir) o bé crear un missatge nou (Crear Missatge).

Llista Llegir
En prémer aquesta opció el mòbil connectarà amb el servidor i descarregarà la llista de missatges que tenim per a llegir.

Il·lustració 17.20 - Missatges - [Llista Llegir]

En podem escollir un i llegir-lo; fent això tindrem disponible una altra opció, que és la d'esborrar el missatge.

Crear Missatge

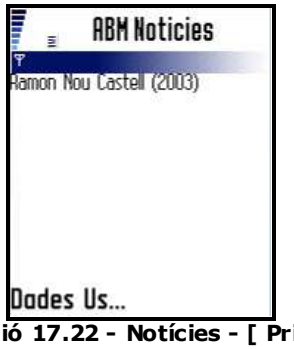
Al crear un missatge nou ens sortirà un formulari similar al de la següent il·lustració:

Il·lustració 17.21 - Missatges - [Crear Missatge]

En ella introduirem el títol i el text del missatge (uns 900 caràcters normalment) i el destí del missatge, que serà un nom d'usuari registrat al sistema (podem o bé escriure'l o descarregar la llista d'usuaris del sistema i seleccionar-lo). Finalment utilitzarem l'opció Inserir missatge per a introduir el missatge al sistema. Si tot ha anat correctament l'aplicació retornarà al formulari inicial.

17.5 Client Mòbil Notícies

El client utilitza les dades del .jad compartint-les amb el client AB Missatges. A l'executar l'aplicació observarem una pantalla com la següent, on escollirem l'única opció disponible.



Il·lustració 17.22 - Notícies - [Principal]

Introduirem les dades de l'usuari i l'adreça del servidor si és necessari (per ser incorrecta o no estar especificada al .jad). Finalment acceptarem a l'usuari.

Il·lustració 17.23 - Notícies - [Dades Usuari]

A l'acceptar l'usuari tindrem disponible l'opció Dades Noticia; en ella podrem introduir el títol/ subtítol i la plantilla a utilitzar o anar a les altres opcions disponibles.

Il·lustració 17.24 - Notícies - [Dades Notícia]

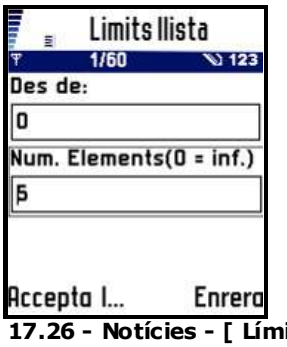
Les opcions que tenim disponibles són les següents :

- Accepta Notícia
- Afegeix Text
- Afegeix Imatge
- Llista Notícies
- Límits Llista
- Llista Plantilles

En primer lloc, si no volem la plantilla per defecte o volem seleccionar una altra des d'una llista (descarregada des del servidor), podem seleccionar l'opció Llista Plantilles i seleccionar-la.

Il·lustració 17.25 - Notícies - [Llista Plantilles]

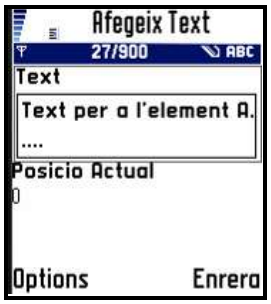
Les opcions Límits Llista i Llista Notícies estan relacionades: límits llista permet especificar quantes notícies (i des de quina) volem descarregar del servidor. La seva utilitat és evitar exhaurir la memòria i reduir la transferència de dades del servidor. Si col·loquem a Num. Elements 0 transferirem tota la llista de Notícies.



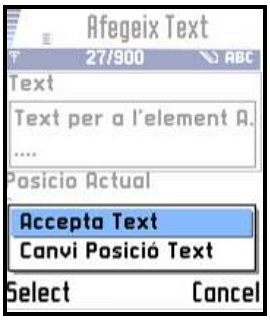
Il·lustració 17.26 - Notícies - [Límits Llista]

Una vegada escollits els límits podem demanar la llista de les notícies i rebre una notícia per a modificar-la (si és nostra, si no la replicarem) o esborrar-la (també si és nostra, és clar).
Finalment comentarem la introducció dels elements que componen la notícia.

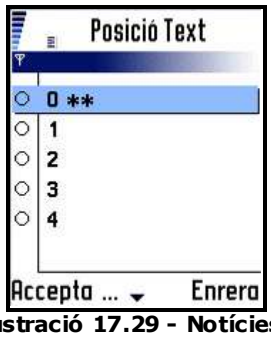
En primer lloc afegir text ens permet afegir els elements de text de la notícia (A..E, indicat com a 0 a 4).



Il·lustració 17.27 - Notícies - [Afegeix Text]

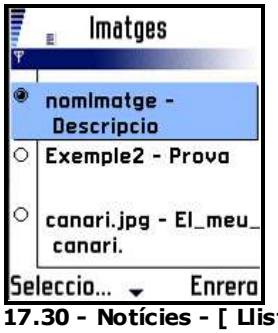


Il·lustració 17.28 - Notícies - [Accepta Text]

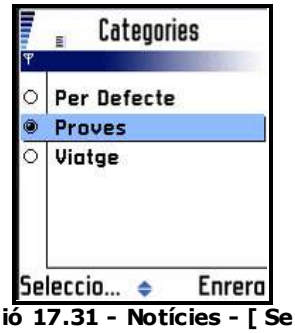


Il·lustració 17.29 - Notícies - [Posició Text]

Seleccionarem l'opció Accepta Text una vegada estem satisfets amb el text d'una posició concreta. La funció Afegeix Imatges és similar; tenim per això dues opcions més: Llista Imatges i Canvi Categoria. Amb llista d'imatges obtindrem (mitjançant connexió al servidor) la llista de les imatges de la categoria seleccionada, i la categoria l'obtindrem mitjançant també una connexió al servidor amb l'opció Canvi Categoria.



Il·lustració 17.30 - Notícies - [Llista Imatges]



Il·lustració 17.31 - Notícies - [Selecciona Categoria]

Finalment, acceptarem la imatge amb l'opció Accepta Imatge.

Il·lustració 17.32 - Notícies - [Accepta Imatge]

Quan tinguem la notícia redactada escollirem l'opció accepta notícia i la notícia s'enviarà al servidor per ser inserida o modificada.

17.6 Servidor de Correu

Al directori \Mercury\ hi trobarem el servidor de correu mercury i el procmail. Si hem utilitzat l'instal·lador podrem executar-los directament des del menú Inici de Windows. En cas contrari, podem executar els dos programes des d'una finestra de DOS.

Mercury.exe serveix per a executar el servidor de correu (ja estarà configurat); si necessitem configurar-lo recomanem llegir els manuals d'instruccions o l'ajuda en línia del programari. Amb aquesta configuració podríem crear usuaris del servei de correu, per exemple.

Procmail.exe és el procés encarregat de capturar les imatges, la seva línia de comandes és la següent:

<PATH CUA (acabat amb \)> <USERBD> <PASSBD> <BD> <IPBD> <POLLTIME m
s> <tipus (0) QUEUE / (1) USER>

- PATH CUA: Apuntarà al directori on s'emmagatzemen els correus, aquest directori serà o bé la cua (C:\MERCURY\QUEUE\ (p.ex)) o bé un directori d'usuari (C:\MERCURY\MAIL\USUARI\ )
- USERBD : Serà el nom d'usuari de la base de dades (mavy per defecte)
- PASSBD : La seva contrasenya (pfc2003)
- BD : El nom de la base de dades (pfcsample)
- IPBD : És la IP de la base de dades (localhost)
- POLLTIME : temps que es triga en capturar un correu en ms. (120000 és un bon valor, cada 2 minuts se cercarà el directori)
- TIPUS: posarem un 0 si és un directori de cua o un 1 si hem fet la instal·lació amb un nom d'usuari.

Exemple de línia de comandes:
Procmail C:\MERCURY\QUEUE\ mavy pfc2003 pfcsample localhost 120000 0

Finalment cal recordar que la finestra del procés cal tenir-la oberta.

17.6.1 Enviament d'imatges :

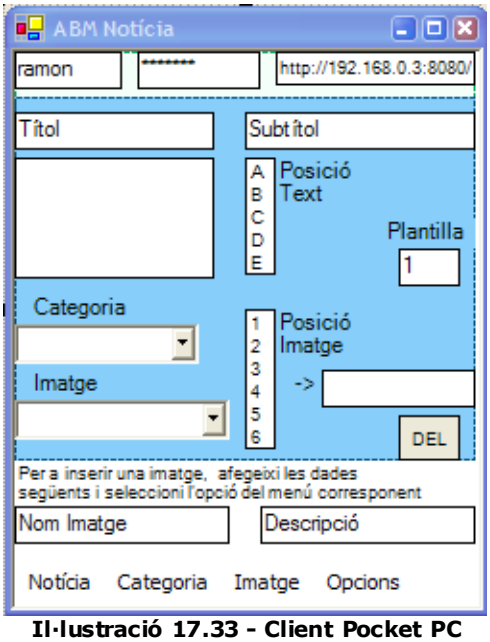
Per a enviar un correu al sistema hem d'utilitzar l'adreça postmaster@[80.25.193.223] en el cas que no tinguem nom de domini (cas per defecte) i que no haguem definit cap usuari; utilitzarem butlleti@[80.25.193.223] en el cas que haguéssim definit un usuari amb el nom butlletí.

El format del correu és el següent: al subject hi col·locarem la descripció de les imatges, el nom del fitxer serà el nom de la imatge i la @ de correu serà el nom de l'autor. Finalment, podem incloure més d'una imatge al correu com a attachments; el sistema inserirà totes les imatges alhora.

Si volem assegurar-nos que no perdrem els missatges és recomanable utilitzar l'habilitat de poder afegir un CC al missatge i enviar-lo així a un altre servidor (d'un ISP per exemple). Les imatges, si tot ha anat bé, trigaran POLLTIME / 1000 segons en inserir-se.

17.7 Client Pocket PC

Per a la seva execució necessitem copiar el .CAB corresponent al nostre processador al Pocket PC i executar-lo. Necessitem també instal·lar abans el Microsoft .NET compact framework (si no disposem d'un **Pocket PC 2003**), que podem obtenir des de les pàgines de Microsoft.



Il·lustració 17.33 - Client Pocket PC

En primer lloc observem les 3 primeres caselles: nom d'usuari, contrasenya i URL del WS; si canviem qualsevol d'aquestes podem guardar la configuració utilitzant l'opció del menú [Opcions->Save].

17.7.1 Inserir Notícies

La zona blava és la corresponent a aquest cas d'ús. En primer lloc introduïm el títol, el subtítol i el número de la plantilla (1 per defecte); utilitzant la llista (A,B,C,D,E) podem seleccionar la posició del text on volem escriure (mitjançant la caixa de text de l'esquerra).

La següent part és inserir imatges a la notícia; en primer lloc carregarem la llista de Categories del servidor [**Categoria->Carrega**]. Al seleccionar una categoria al menú despregable, carregarem del servidor la llista de les imatges al menú despregable inferior. Finalment, a l'igual que amb el text, seleccionant un element de la llista (1,2,3,4,5,6) canviarem la posició a la que volem assignar la imatge que seleccionarem amb la llista d'imatges.

Si volem esborrar l'assignació només tenim que prémer el botó **DEL**. Finalment utilitzem l'opció del menú [**Notícia->Inserir**] per a inserir la notícia.

17.7.2 Inserir Imatges

Per a inserir imatges necessitem la llista de categories primer, seleccionar-ne una i introduir el nom i la descripció que desitgem a les dues caselles inferiors. Finalment seleccionarem una imatge del PocketPC utilitzant l'opció [**Imatge->Inserir**]. Cal anar en compte amb la mida del fitxer, ja que no es recomana enviar fitxers de més de 300 Kbytes.

17.8 Manteniment

Per el manteniment del sistema, hem de fixar-nos en el directori /apache/logs i eliminar els logs cada x temps (depenent de l'útilització del sistema); també cal donar un cop d'ull

al directori /apache/lib/php/temp, on es guarden les sessions de PHP, i eventualment es podría omplir.

Finalment, per mantenir la base de dades és convenient utilitzar l'eina phpMyAdmin mitjançant un navegador (renombrant a htdocs el directori corresponent a phpMyAdmin) per tal de fer copies de seguretat de les dades, reparar la base de dades, esborrar registres, etc.

És recomanable renombrar els directoris Install i phpMyAdmin quan no s'utilitzen, així com crear index.html i index.jsp nuls a les parts de htdocs i webapps on no volem que s'accedeixi.

18. ANNEX E – Interoperativitat amb .NET

Després de la finalització del sistema i dels clients mòbils es van realitzar algunes proves d'interoperativitat amb Visual Studio .NET, realitzant l'operació d'inserir notícies i imatges.

18.1 Problemes i solucions

VS generà la classe que accedeix al Web Service automàticament donant-li l'adreça del WSDL. Com la nostra classe era una mica complicada (classes dintre de classes, col·leccions...) la classe contenia uns quants errors i hem hagut de modificar a mà el fitxer reference.vb per a que funciones bé.

La depuració l'hem fet utilitzant un capturador de paquets[1] TCP/IP per capturar el missatge SOAP de sortida del Pocket PC i comparar-lo amb el missatge obtingut amb el client mòbil.

L'error que hem trobat era que s'enviaven malament els vectors que contenen els elements de text i d'imatges de la notícia. En Java el Vector contenia els elements de tipus WSInfoText però en VS .NET s'enviava una classe Vector que tenia una propietat ítem que era una col·lecció de WSInfoText :

Java	:	Vector [[WSInfoText][WSInfoText][WSInfoText]]
VS	:	Vector [item [[WSInfoText][WSInfoText][WSInfoText]]]

Això passava ja que el que conté un Vector són "ítem" i VS agafava "ítem" com a nova classe. La solució és la següent:

```
Public llistaI(6) As WSInfoIma
Public llistaT(5) As WSInfoText
```

En compte de:

```
Public llistaI As Vector
Public llistaT As Vector
```

Finalment per a utilitzar-lo:

```
Dim texto(6) As ABMNoticies.WSImatge.WSInfoText
...
noticia.llistaT = texto
```

I enviem la notícia.

[1] http://www.pocketsoap.com/tcptrace/ , indiquem el port a escoltar i el port on redirigir la petició i obtindrem el paquet.

El següent error que hem trobat ha estat al rebre els vectors: havíem dissenyat els WS de manera que s'enviaven java.util.Vector, per tal que funcionés amb .NET hem hagut de treure aquest nivell i enviar directament els arrays dels elements, és a dir :

```
Original       : Vector [ [WSinfoXXX] [WSinfoXXX] [WSinfoXXX] ]
Modificat      : WSinfoXXX [[item1][item2][item3]]
```

Mirem per exemple el procediment de getUsers; l'original era així :

```
public java.util.Vector getUsers ()
```

i l'hem canviat per aquest :

```
public String[] getUsers ()
```

Hem obtingut algunes avantatges amb això: la primera, hem reduït una mica l'overhead dels missatges XML (una capa menys), i tot i que hem modificat el Web Service no hem hagut de modificar el client kSOAP, ja que continua funcionant.

Fet això, hem construït el client de Pocket PC per a poder inserir Notícies i inserir Imatges. L'aplicació, una vegada superats els problemes, ha estat de construcció senzilla.

18.2 Testeig

El testeig de la banda de les notícies ha estat satisfactori, en canvi l'enviament d'imatges grans (> 300 Kb) donava error a la banda del servidor. Cercant les llistes de notícies de mySQL hem obtingut la solució.

S'han augmentat alguns paràmetres de mySQL per a permetre missatges més grans (entre Java i mySQL), s'ha actualitzat el connector mySQL – Java per una versió en desenvolupament que tenia el "bug" resolt i finalment hem actualitzat la versió de mySQL a la següent versió estable.

En aquest moment hem pogut transmetre imatges de 1 Mb sense cap problema.

19. ANNEX F – Contingut del CDROM

A continuació comentarem el contingut del CD-ROM que acompanya aquest document.

Trobem quatre directoris :

```
<DIR>          Codi
<DIR>          Desenvolupador
<DIR>          Documentacio
<DIR>          Usuari
```

Codi conté tot el codi del sistema, Desenvolupador conté les eines per a desenvolupar utilitzades (EVT, Nokia SDK,...), Documentació té accés als PDF's esmentats a la bibliografia i finalment Usuari conté tot el necessari per a instal·lar el sistema (tant per Windows com per Linux).

19.1 Codi

```
<DIR>          ExempleAxis     Conté l'exemple LinkGet utilitzat en l'annex
<DIR>          Final           Codi de la versió final del sistema, tant WS com clients
<DIR>          PHP             Codi del client WEB i php.ini utilitzat en el desenvolupament
<DIR>          Pilot           Codi de la prova pilot (Client .NET i Client PocketSOAP)
<DIR>          Prototipus      Codi del prototipus (similar al Final)
<DIR>          Scripts         Codi dels scripts d'instal·lació
               ProcMail.rar    Codi del procés de captura de email
```

Directori PHP

```
<DIR>          Install         WEB Instal·lació
<DIR>          PFCButlleti     WEB Final
<DIR>          PFCImatge       WEB Pilot
```

19.2 Desenvolupador

```
<DIR>          Instalador      Conté el programa creador d'instal·lacions
<DIR>          J2ME            SDK's de Nokia i Sony-Ericsson
<DIR>          Pocket PC       EVT i SDK Pocket PC2002
<DIR>          Utilitats       TCPTrace, tracejador de peticions TCP/IP
```

Directori J2ME

```
<DIR>          KSoap           Fitxers corresponents a KSOAP 1 i KSOAP 2
<DIR>          Nokia           SDK de Nokia
<DIR>          Oficial         J2ME oficial i MMAPI
<DIR>          Sony            SDK de Sony-Ericsson
<DIR>          xerces-2_4_0
```

19.3 Documentació

<DIR>	General	
<DIR>	J2ME	
<DIR>	WS	
	Jochinger.2003.pdf	**Projecte relacionat**

Directori General

jdbc-3_0-fr-spec.pdf	**Documentació JDBC**
XML_Schema.pdf	**Informació sobre XML**

Directori J2ME

A_Brief_Introduction_to_MIDP_Clients_for_Web_Services_v1_0.pdf
Camera_MIDlet_A_Mobile_Media_API_Example_v1_0.pdf
developersguide_p800_t610.pdf
Java_MIDP_App_Dev_Guide_v1_0.pdf
ksoap.html
Nokia_UI_API_v1_0.zip
Optimizing_client_server_part1.pdf
Optimizing_client_server_part2.pdf
S40_API_over.pdf

Directori WS

axis-java-soap.pdf
Axis.pdf
gross_christian_axis.pdf
J2EE-vs-DotNET.pdf
JavaWSTutorial.pdf
WSOpenSource.pdf

19.4 Usuari

<DIR>	Client J2ME	**Conté els diferents .jar/.jad dels clients**
<DIR>	Client Pocket PC	**Conté el client per a Windows CE**
<DIR>	Components Separats	**Podem trobar els diferents elements del sistema**
<DIR>	Instalacio	**Els dos .exe d'instal·lació**
<DIR>	J2DK	**El J2SDK 1.4.1 per a instal·lar tomcat**
<DIR>	Linux	**Tots els fitxers necessaris per a Linux**

Directori Instalacio

Butlleti(APACHE9999).exe	**Instal·lació d'Apache, PHP i MySQL al port 9999**
Butlleti.exe	**Instal·lació de tot el sistema (80 Apache i 8080 Tomcat), també instal·la el servidor de correu.**

Directori Components Separats

<DIR>	Apache	**Servidor Web Apache**
<DIR>	AppServ	**Instal·lador de Apache, PHP i MySQL (alternatiu)**
<DIR>	Axis	**Zip amb la release oficial d'Axis**
<DIR>	Capturador Correu	**ProcMail i fitxers Bat per a cridar-lo**

```
<DIR>          DEPLOY AXIS          Fitxers per a realitzar el deploy dels WS, i el directori
Axis per si volem copiar-lo directament a tomcat\webapps

<DIR>          MERCURY              Servidor de correu Mercury

<DIR>          MySQL                Binaris relacionats amb MySQL

<DIR>          PHP                  Última versió estable de MySQL

<DIR>          Tomcat               Tomcat 4.1

<DIR>          WEB                  Fitxers PHP del client WEB, inclou phpMyAdmin (cal
renombrar-lo)
```