Capítol 6

Implementació i proves

6.1 Introducció

La implementació és la fase del desenvolupament d'un sistema software en que es passa dels resultats obtinguts de la fase de disseny aplicant tota una sèrie de patrons a la seva traducció al llenguatge de programació escollit.

En aquest capítol s'exposen diversos aspectes del procés d'implementació del sistema software, tals com l'estructuració del codi, descripció de les classes més rellevants, etc. ...
També es para atenció a la fase de proves a que els diferents prototipus finalitzats del projecte han estat sotmesos.

6.2 Entorn hardware i software de la implementació

Durant el procés d'implementació, s'ha fet servir un ordinador personal dotat d'un processador Intel Pentium IV, a 2GHz, amb 512 Mb de RAM, executant el sistema operatiu Windows XP professional (amb Service Pack 2).

També s'ha disposat d'una connexió a Internet de gran amplada de banda (3mbps).

D'acord amb el que s'ha esmentat amb anterioritat, el llenguatge de programació escollit ha estat J2ME (que en essència, és un subconjunt de les llibreries de la plataforma Java 2 Platform Starndard Edition 5.0).

Per a facilitar la feina del programador, la implementació s'ha fet amb l'ajut de l'eina Eclipse SDK 3.2, al que se li han afegit tot un seguit de plugins: EclipseUML, un potent mòdul que permet l'edició de diagrames de classes UML, i que genera plantilles de classes JAVA a partir d'aquests; EclipseME, que conjuntament amb el Sun Wireless Toolkit, configuren una completa suite de desenvolupament d'aplicacions en J2ME (inclou emulador, i entorn d'execució).
També s'ha disposat, per temes de seguretat, d'un repositori de versions SVN gratuït, proporcionat per la Facultat d'informàtica de Barcelona, conjuntament amb el respectiu plugin de Eclipse que en permet la sincronització.

6.3 Estructuració del codi

En aquest apartat, ens centrarem en el contingut dels diferents packages que componen la implementació del sistema software, i dins de cadascun, n'exposarem la implementació de les classes més rellevants. El codi està dividit en quatre packages que engloben cadascuna de les capes del sistema software: al package view, tenim les classes que corresponen a la capa de presentació del sistema; al package domain, les de la capa de domini; al package data, les que corresponen a la capa de dades, i al package útil, tot un seguit de classes destinades a facilitar algunes tasques concretes.

6.3.1 Package view

Essencialment, aquest package el componen les finestres de la capa de presentació, d'interfície d'observador d'aquests, i l'objecte que representa un esdeveniment de presentació.

Implementació de les finestres de la capa de presentació

Les finestres de la capa de presentació han estat construïdes mitjançant els objectes que proporciona la API de J2ME per construir objectes predeterminats, de tipus forms, és a dir, la API d'alt nivell, de manera que el programador només s'ha de preocupar de crear un formulari, i anar afegint-hi objectes (botons, caixes de text, etc. ...).

D'aquesta manera, les finestres de la capa de presentació son objectes que, en ser construïts, reben un objecte observador com a paràmetre; aquest objecte és el que està destinat a rebre els esdeveniments que es produeixin en aquesta finestra. Segons el cas, es poden diversos observadors, però en la nostra implementació, donat que fem servir un controlador de tipus Façana, amb un observador per finestra (per a totes serà el mateix observador, és a dir, el controlador Façana) en tenim prou.

D'altra banda, totes les finestres compten amb un mètode anomenat render, que

les mostra, i un altre anomenat hidra, que les oculta i retorna el control a la finestra que les ha invocat.

Com a comentari, dir que cada finestra actua com a observador d'esdeveniments externs d'ela mateixa, de manera que, quan una finestra detecta, per exemple, que l'usuari prem un botó, és ella mateixa qui recull l'esdeveniment, i llança un esdeveniment de tipus ViewEvent, amb un codi d'esdeveniment associat, que el controlador Façana s'encarregarà d'examinar per tal d'executar les transaccions corresponents.

```
public void render(Display dispDisplay)
{
        dispDisplay.setCurrent(_form);
}

public void hidra(Display dispDisplay)
{
        dispDisplay.setCurrent(_parent);
}

public void commandAction(Command arg0, Displayable arg1)
{
        try
        {
                if(arg0 == _back)
                {
                        _listen.viewEvent(new
                                        ViewEvent(ViewEvent.DONE_READ));
                }
                else if (arg0 == _reply)
                {
                        _listen.viewEvent(new ViewEvent(ViewEvent.REPLY));
                }
        }
        catch (Exception e)
        {
                _listen.errorEvent(e);
        }
}
```

· Fig. 27. Codi de les operacions comuns de les finestres de la capa de presentació

6.3.2 Package domain

Aquest package de codi conté les classes que corresponen a la implementació dels objectes conceptuals del sistema. Essencialment, es tracta de quatre classes senzilles que no aporten cap mena de dificultat en la implementació (són estructures de dades lleugeres, amb els seus corresponents mètodes modificadors i consultors).

6.3.3 Package data

Aquest package conté les classes que corresponen a la implementació dels resultats obtinguts del procés de disseny de la capa de dades. En aquest package, podem trobar tant les classes destinades a realitzar consultes i insercions a la BBDD local, com les que s'encarreguen d'establir canals de comunicacions remots.

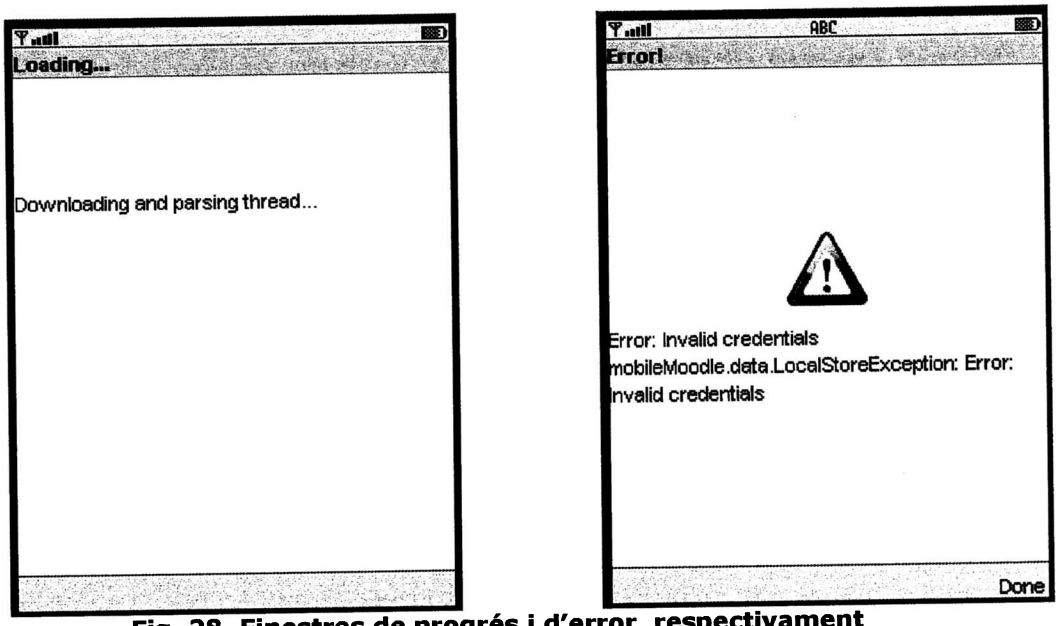
Implementació de les classes Façana de la capa de dades

Donada la simetria dels dos controladors façana de la capa de dades que es dediquen a gestionar l'accés a la BBDD local i les comunicacions remotes, en descriurem tan sols un dels dos: el que s'encarrega de buscar un perfil d'usuari a la BBDD local, i si no el troba, realitza una crida al webservicelib per emmagatzemar els resultats.

Controlador Façana UserStore

La principal característica d'aquesta classe, és el fet que implementa d'interfície observador de la classe que gestiona les comunicacions, la classe MoodlePipe. Quan el controlador rep una petició de cerca d'un usuari (mitjançant una crida estàtica, recordem que es tracta d'una classe Singleton) aquesta crida al controlador façana de la BBDD local, que s'encarrega de realitzar les crides necessàries als mètodes de la API J2MESDLIB, per cercar un usuari. Si el resultat és nul, aleshores, obre una nova MoodlePipe, invoca al mètode de crida al webservicelib, i espera (retornant el control al sistema, però no a l'usuari) rebre un esdeveniment de la classe ConnectionEvent: si l'esdeveniment és del tipus WEBSERVICE_INVOKED, aleshores descarrega els resultats, els interpreta mitjançant les llibreries de kXML, construeix un objecte conceptual Usuari, l'emmagatzema a la BBDD local mitjançant el controlador Façana pertinent, i el retorna com a resultat de la crida del mètode que ha desencadenat aquesta seqüència d'esdeveniments (cal dir que durant tot el procés, tot i que l'usuari no té el control de l'aplicació, sap quin és l'estat de les tasques que s'estan executant).

Si es produeix un error d'autenticació durant aquest procés (si per exemple, les credencials introduïdes són errònies), el sistema software notifica l'usuari mitjançant la pertinent finestra d'error.



· Fig. 28. Finestres de progrés i d'error, respectivament

Cal afegir que, en la implementació dels controladors Façana de la classe de dades s'ha contemplat la possibilitat que continguin més d'un observador, donat que, és possible que més d'un objecte es vegi afectat pels canvis provocats en un objecte durant el transcurs d'una transacció. Com per exemple, en el cas que actualitzem un perfil d'usuari, i algun dels camps existents hagi canviat: caldrà notificar a la capa de presentació, a fi que aquesta mostri correctament els canvis introduïts, i a la capa de domini també, per tal que s'encarregui de propagar els canvis a la capa de dades.

```
public static void loadUser(String strUser, String strPass)throws Exception
{
        _user = strUser;
        _pass = strPass;
        _userObject = LocalStore.getUser(strUser, strPass);
        if (_userObject != null)
        {
                fireEvent(new StoreEvent(StoreEvent.USER_LOADED,""));
        }
        else
        {
                _con = new MoodlePipe();
                _con.addConnectionListener(this);
                _con.webserviceCall(_user,_pass);
        }
}
```

· Fig. 29. Codi de la operació de cerca d'un usuari

En el fragment de codi anterior, es pot apreciar com, si es dona el cas que l'usuari ja existeix en la BBDD local (retornat per la crida al controlador Façana LocalStore), el controlador UserStore llança un esdeveniment mitjançant fireEvent().

```
public void ConnectionEvent(ConnectionEvent event)
{
        if                    (event.getEventType()                    ==
ConnectionEvent.WEBSERVICE_INVOKED)
        {
                try
                {
                        Reader reader =
                        new InputStreamReader(_con.getInputStream());

        fireEvent(newStoreEvent(StoreEvent.MESSAGE,"Downloading    and
parsing user profile..."));
                        _userObject = parse(reader);
                        fireEvent(new
StoreEvent(StoreEvent.MESSAGE,"Done."));

                        if (_userObject==null)
                        {
                                fireEvent(new                StoreEvent(new
Exception("Error: invalid credentials.")));
                        }
                        else
                        {
                                fireEvent(new
StoreEvent(StoreEvent.USER_LOADED,""));
                                LocalStore.storeUser(_userObject);
                        }
                }
                catch (Exception e)
                {
                        e.printStackTrace();
                }
                finally
                {
                        _con.closeMDConnection();
                }

        }
}
```

· Fig. 30. Implementació de l'interfície ConnectionListener

En aquest fragment de codi es pot apreciar la implementació que realitza el controlador UserStore dels mètodes de l'interfície ConnectionListener, de manera que, quan una invocació del webservicelib assoleix l'estat desitjat, la classe

MoodlePipe genera un esdeveniment de tipus ConnectionEvent, amb el codi corresponent a WEBSERVICE_INVOKED, aquest és recollit pel controlador UserStore, i passa a descarregar-ne el contingut del resultat.
Cal a dir que, com a millora, el parsejat de la informació es fa en temps de descàrrega, és a dir, es descarrega un fragment del resultat obtingut, i es parseja. No obstant, el resultat no s'emmagatzema fins que s'ha parsejat completament, a fi d'evitar possibles inconsistències a la BBDD local.

Controlador de comunicacions remotes MoodlePipe

Aquesta classe inclou la implementació dels mètodes necessaris per establir connexions remotes de crida al webservicelib, i al servidor de moodle. A diferència dels altres controladors de la capa de dades, aquest no és un controlador Singleton, de manera que, cada objecte de la classe representa un canal de comunicacions. És a dir, si volem realitzar tres crides al webservicelib, haurem d'instanciar tres objectes diferents, per invocar els mètodes pertinents a cadascun d'ells.
Aleshores, quan volem establir una connexió remota, depenent de quin sigui el nostre objectiu, el controlador de xarxa ens proporciona una sèrie de mètodes que s'encarreguen de realitzar les pertinents crides al webservicelib, de descarregar un RSS, o de preparar la connexió per contestar a un missatge del fòrum.

Cal dir que, en la implementació del controlador de xarxa, també s'ha previst la subscripció de diversos observadors a un únic objecte, de manera que, quan un objecte MoodlePipe realitza una sèrie de transaccions, els seus observadors (com per exemple, el controlador Façana UserStore de la capa de dades, i el controlador Façana de la capa de Domini) n'estan al corrent en tot moment.

```
public void webserviceCall(String strUser, String strPassword)throws
MDConnException, IOException
{
        final String user = strUser, password = strPassword;
        new Thread()
        {
                public void run()
                {
```

```
                    try
                    {
                            _conn =
                            (HttpConnection)Connector.open(Constants.WE
                            BSERVICE_URL+"user="+user+"&pwd="+pass
                            word);
                            _conn.setRequestProperty("user-
            agent","Profile/MIDP-2.0 Configuration/CLDC-1.1");
                            _conn.setRequestProperty("Content-
Type","application/x-www-form-urlencoded");
                            _in = _conn.openInputStream();
                            fireEvent(new
ConnectionEvent(ConnectionEvent.WEBSERVICE_INVOKED));
                    }
                    catch (IOException e)
                    {
                            fireEvent(new ConnectionEvent(e));
                    }
                    catch (Exception E)
                    {
                            fireEvent(new ConnectionEvent(E));
                    }

            }
        }.start();

}
```

Fig. 31. Fragment de codi corresponent a la crida al webservicelib

Cal parar especial atenció al fet que les pipes de comunicacions s'executen, tal i com s'ha esmentat amb anterioritat, sobre un fil d'execució diferent al principal.

També s'ha de destacar la especial importància que té el fet que, les comunicacions amb el servidor de moodle es realitzen de manera que el servidor, en tot moment, creu que qui obre la connexió és un navegador web estàndard, i que s'està realitzant un procés d'ingrés en el sistema mitjançant el seu formulari HTML. Així doncs, el procés per a ingressar al sistema, fent servir els mètodes que implementa la interfície de xarxa, és el següent:

- En primer lloc, cal preparar les cookies, segons l'esmentat a 5.4.3. Això s'aconsegueix mitjançant el mètode públic de MoodlePipe getCookies(). Aleshores, quan les cookies estan disponibles, rebrem un esdeveniment ConnectionEvent de tipus COOKIES_STORED.

- Rebem l'esdeveniment COOKIES_STORED, i aleshores invoquem el mètode públic MDLogin(). Aquest mètode, quan hagi finalitzat les transaccions corresponents, generarà un esdeveniment ConnectionEvent de tipus LOGGED.

- Rebem l'esdeveniment LOGGED, i invoquem al webservicelib a fi de que ens retorni l'identificador del primer missatge del tema de discussió al que volem contestar (del qual ja en tenim casi totes les dades necessàries, després de descarregar-les amb el perfil d'usuari).
  Quan hagi finalitzat el procés, aquest mètode generarà un esdeveniment ConnectionEvent de tipus FIRST_POST_ID_READY.

- Recollim l'esdeveniment FIRST_POST_ID_READY, i construïm un objecte conceptual ForumMessage, amb els paràmetres següents:

  - Identificador del curs a que pertany el fòrum (obtingut de les dades del RSS, al perfil d'usuari)

  - Identificador de la instància de fòrum al servidor de moodle (obtingut de les dades del RSS)

  - Identificador del tema de discussió al que volem contestar (obtingut de l'enllaç que conté el RSS a l'apartat dedicat a aquest tema de discussió)

  - Identificador del primer missatge del tema de discussió (obtingut de la crida al webservicelib

  - Identificador de l'usuari (obtingut del perfil d'usuari)

- Afegim el text i el títol del missatge, obtingut mitjançant la corresponent finestra de la capa de presentació. Enviem el missatge, mitjançant el mètode postToForum() de la classe MoodlePipe.

Seguint el procés descrit, el servidor de moodle ja sap de la nostra presència, i ens permetrà accedir a les zones privades. Aleshores, mitjançant un mètode HTTP de tipus POST generat pel mètode postToForum() de la classe MoodlePipe, rebrà una petició amb aquest contingut al seu cos:

- *"course=cursID&forum=forumID&parent=parentID&discussion=discussID &userid=userID"&subject=titol&message=missatge&format=1"*

6.3.4 Package util

En aquest package podem trobar la implementació de les classes auxiliars que s'han fet servir per a facilitar i agrupar algunes tasques que es realitzen a més d'un mòdul (en comptes de duplicar codi de forma innecessària).
Dins del package util, s'hi troba la classe Constants, que conté algunes dades (tals com l'adreça URL de mòdul dels fòrums de moodle, o l'adreça URL del webservicelib) constants. També podem trobar una classe que s'encarrega de codificar i descodificar cadenes de text en format UTF-8, format en que moodle envia les dades RSS per defecte.

També s'hi troben les classes necessàries per tractar els temes de seguretat (encriptat i desencriptat de passwords en MD5), a més d'algunes utilitats de cara al tractat de les cadenes de text obtingudes al parsejar un document XML, com ara bé l'eliminació de tags HTML que el cos dels missatges d'un RSS pugui contenir, i que si es passessin per alt, farien el text gairebé inintel·ligible. Cal dir que no tots els tags HTML s'eliminen, donat que això requereix un tractat de cadenes de text ocasionalment molt grans, fet que comportaria que el sistema software hi dediqués molt de temps d'execució (recordem que tractem amb un processador de prestacions reduïdes).

```
public static String AUTHSERVER =
"http://morfeo.upc.edu/crom/";

public static String WEBSERVICE_URL =
AUTHSERVER+"mod/wiki/mobilewebservice.php?sel=exportUserPrefs&";

public static String FORUMS_URL =
AUTHSERVER+"mod/forum/post.php?reply=";

public static String WEBSERVICE_FIRST_POST =
AUTHSERVER+"mod/wiki/mobilewebservice.php?sel=getFirstPost&discussID=";
```

· Fig. 32. Contigut de la classe Constants

6.4 Proves

Durant la implementació de l'aplicació s'han anat realitzant les de proves necessàries per assegurar el compliment de les especificacions inicials. Cada cas d'ús implementat s'ha sotmès a les proves necessàries per verificar el seu funcionament.

El banc de proves en que s'ha realitzat el procés de proves dels diferents prototips que s'han anat generant durant el procés d'implementació es pot dividir en dos escenaris: un de virtual, configurat per l'emulador de dispositius mòbils que aporta el Sun Wireless Toolkit (imprescindible per a desenvolupar sistemes software J2ME), i un altre de real, basat en l'ús de dispositius mòbils compatibles amb J2ME. Concretament, s'ha fet servir una PDA Tungsten C, amb el sistema operatiu PalmOS versió 5.2.1, i un telèfon mòbil Sony Ericsson t630 (tot i que es va desestimar el seu ús, donat que no és compatible amb les especificacions CLDC 1.1).

Donat que la velocitat de funcionament del sistema software depèn, en sa gran majoria, de l'estat de la connexió de xarxa (ample de banda, disponibilitat, etc. ...) i del volum de dades descarregades, és difícil establir un banc de proves 100% fiable. També s'ha de tenir en compte que el rendiment de l'aplicació, depèn de les prestacions tècniques del hardware que l'executa. Tot i això, els resultats obtinguts han estat més que satisfactoris. Amb un ample de banda de 11mbps, proporcionat per la connexió inalàmbrica de la UPC XSF-UPC, la velocitat de descàrrega i processat de 10 kbytes d'informació (tamany que pot arribar a ocupar un RSS de tamany considerable) el temps mig ha estat d'uns 5.23 segons, temps més que acceptable, donat que, aquest procés només es realitza quan l'usuari descarrega el RSS sencer, és a dir, que en aquest temps, hi ha inclòs el temps de parsejat del document XML, el parsejat UTF-8, i l'eliminació dels marcadors HTML. Una vegada s'ha emmagatzemat el document, ja no s'ha de descarregar fins que s'actualitzi volutàriament.

Capítol 7

Planificació i estudi econòmic

7.1 Planificació

La planificació d'un projecte compren la previsió de les dates per la realització de totes les tasques relacionades. Com a previsió que es sempre s'han de preveure possibles desviacions en quant al temps requerit per tal de dur a terme l'execució de les tasques.

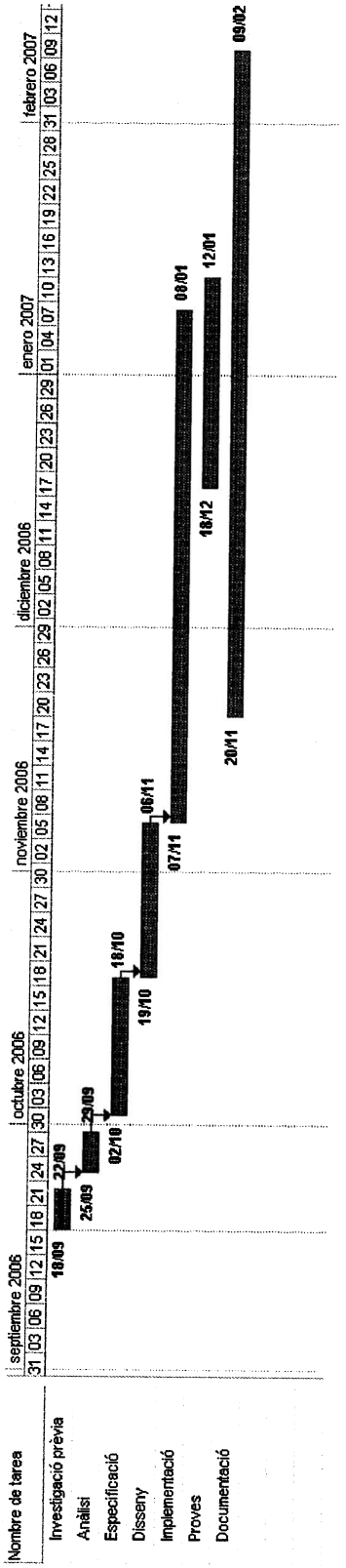
Segons la guía docent de la FIB, la càrrega de treball per cada crèdit matriculat és de 20 hores. Així doncs, un projecte de la ETIG que consta de 22.5 crèdits haurà de prendre unes 450 hores, en total. Amb una dedicació de 35 hores setmanals, la durada total del projecte hauria d'haver estat d'unes 13 setmanes. El projecte es va iniciar al setembre de 2006, i la seva data de finalització ha estat al maig del 2007. Aquesta atípica desviació de la planificació inicial, ha estat ocasionada pel fet que la dedicació en hores setmanals no ha estat la prevista, donat que l'autor compagina el desenvolupament del projecte amb el desenvolupament d'un altre projecte, al Laboratori de Càlcul de la FIB, a on hi està empleat des d'octubre del 2005. Així doncs, la dedicació en hores setmanals, s'ha vist reduïda a 25 hores setmanals, fet que ha condicionat severament la durada del projecte. Així doncs, fent el còmput de setmanes previstes a la planificació del projecte, segons aquesta dedicació, en resulten 18 setmanes, dada que reflexa de forma fidedigna la durada real del projecte.

La durada del projecte també s'ha vist compromesa per viratges de planificació, com el fet que en un principi es considerés la possibilitat d'implementar un servidor d'autenticació extern (unes 25 hores), o investigacions prèvies i familiatització amb les tecnologies emprades (cal esmentar que, en un principi, J2ME resultava una tecnologia totalment aliena a l'autor).

En qualsevol cas, la planificació inicial del projecte preveu aquests possibles canvis, i el resultat final ha estat força congruent amb l'esperat. No obstant, i com és freqüent en aquesta mena de projectes, la fase d'implementació s'allarga degut a incidències tècniques, errors, i problemes generals relacionats amb les tecnologies emprades.

Tasca	Durada prevista	Durada real
Investigació prèvia	20 hores	20 hores
Anàlisi	20 hores	20 hores
Especificació	50 hores	50 hores
Disseny	50 hores	50 hores
Implementació	180 hores	**240 hores**
Proves	80 hores	80 hores
Documentació	50 hores	50 hores
Total	450 hores	**510 hores**

· Fig. 33. Planificació en hores del desenvolupament del projecte



· Fig. 34. Diagrama de Gantt corresponent a la planificació del projecte

7.2 Estudi econòmic

L'estudi econòmic del projecte engloba el cost dels recursos emprats en el procés de desenvolupament: els recursos tècnics emprats (màquines, programari, possibles costs logístics, etc....) i el costs derivats del personal que s'hi ha vist involucrat. Amb aquestes dades, es pot obtenir un pressupost acurat del cost econòmic projecte.

Recursos Hardware

El maquinari emprat en el procés de desenvolupament ha estat un ordinador personal, propietat de l'autor, i una PDA Tungsten C, facilitada pel departament de Llenguatges i Sistemes Informàtics de la UPC, a través de Marc Alier i Forment. Per tant, les despeses econòmiques en material de tipus hardware com a nul·les.

Recursos Software

El programari emprat en el procés de desenvolupament es mostren a la següent taula, de forma que queda explicitat el seu cost. Cal observar que, segons la filosofia de desenvolupament de software de l'autor, tot el software emprat es distribueix sota llicències generals d'ús gratuït. Tan sols s'han emprat dues eines comercials, Microsoft Project, facilitada pel Laboratori de Càlcul de la FIB, i Microsoft Visio, també facilitat per l'LCFIB.

Software	Cost
Eclipse SDK 3.2	Gratuït
Java 2 MicroEdition	Gratuït
Sun Wireless Toolkit	Gratuït
Omondo EclipseUML plugin Free Edition	Gratuït
API Fast MD5	Gratuït
API kXML	Gratuït
EclipseME plugin	Gratuït
Sun OpenOffice 2.2	Gratuït
Subclipse plugin	Gratuït
Microsoft Project	Software LCFIB
Microsoft Visio	Software LCFIB
Cost total	**0 €**

· Fig. 35. Taula resum amb els costs dels recursos de Software

D'aquesta taula en deduïm que el cost de software no suposa cap cost afegit al pressupost del desenvolupament del projecte.

Recursos humans

Per últim cal considerar el cost del personal necessari per dur a terme el projecte. Primer de tot cal identificar els diferents perfils que intervenen en un projecte. La meva tasca durant el desenvolupament del projecte s'ha dividit en els perfils d'analista (que bàsicament duu a terme l'especificació i el disseny del sistema) i Programador (que s'encarrega bàsicament de la implementació). A continuació es mostra l'assignació de les tasques del projecte segons cada perfil.

Tasca	Durada	Analista	Programador
Investigació prèvia	20 hores	20 hores	
Anàlisi	20 hores	20 hores	
Especificació	50 hores	50 hores	
Disseny	50 hores	50 hores	
Implementació	240 hores		240 hores
Proves	80 hores		80 hores
Documentació	50 hores	40 hores	10 hores
Total	510 hores	180 hores	330 hores

· Fig. 36. Distribució de les hores de durada del projecte reals segons els rols de personal que hi estan involucrats

Per calcular el cost econòmic dels recursos humans del projecte, es fa servir una estimació del sou base d'un analista avui dia (uns 40 € / hora), i per als programadors, faré servir el sou base d'un becari (6 € / hora). Així doncs, el resultat és el següent:

Rol	Durada	Preu/hora	Cost
Analista	180 hores	40 € / hora	7200 €
Programador	330 hores	6 € / hora	1980 €
Total	510 hores		9180 €

· Fig. 37. Cost del projecte segons la distribució per rols i preu per hores

Així doncs, sumant el total de 0 € de costs de hardware, i els 0 € de costs de software, el preu final del projecte només queda condicionat pel preu que comporten els recursos humans associats, és a dir, **el cost del projecte és de 9.180 €**.

Capítol 8

Conclusions i treball futur

8.1 Conclusions

Durant el transcurs del procés de desenvolupament del projecte, a més de coneixements tècnics sobre tecnologies que en principi m'eren totalment alienes, crec que he obtingut els coneixements i part de l'experiència necessàries per afrontar un projecte de gran envergadura. Quan hom desenvolupa un projecte d'aquestes caraterístiques, s'enfronta a reunions de seguiment, exposicions, redacció de documents, enginyeria de software, noves tecnologíes, etc.... que al cap i a la fí, i en essència, el que suposen no és més que un escenari real d'aplicació dels coneixements obtinguts durant tota la carrera.

Aquest projecte m'ha ajudat, sobretot, a obtenir una visió global del que suposa el desenvolupament no d'un sistema software, sinó d'un projecte, en la seva totalitat, és a dir, amb totes i cadascuna de les fases que el componen; el resum de tot això, és el present document, testimoni de tot el treball realitzat.

Durant el desenvolupament del projecte, a més, també se'm va oferir la possibilitat d'impartir una classe de tres hores en el postgrau "Desenvolupament de software per PDA's i Smartphones" de la fundació UPC, fet que considero d'una gran importància; em va obligar a preparar una exposició de llargada considerable, i a "aprendre a ensenyar" tots els coneixements tècnics que he anat adquirint durant el transcurs del projecte.

Evidentment, gran part del volum de coneixements que he adquirit durant el desenvolupament del projecte, són purament de caràcter tècnic: se m'ha ofert la possibilitat de treballar amb J2ME, tecnologia totalment desconeguda per a mi fins a la data, i d'aplicar coneixements ja adquirits mitjançant l'experiència personal (i els anys d'estudi a la FIB). Eines com Eclipse SDK, Apache Tomcat, EasyPHP, i tots els coneixements adquirits a les assignatures d'Enginyeria de Software han esdevingut de gran utilitat.

M'agradaria afegir que, com a autor del projecte, estic força satisfet amb els resultats, sobretot pel fet que, actualment, no existeix cap sistema software basat en J2ME que implementi les característiques de MobileMoodle. Tanmateix, considero remarcable el fet que s'han implementat totes les funcionalitats previstes des d'un principi, adquirint uns nivells de reusabilitat i eficiència més que acceptables.

8.2 Treball futur

M'agradaria destacar el fet que, el projecte continua obert, de forma que s'hi continuaran afegint funcionalitats, per tal de donar cobertura total a la plataforma moodle des de els terminals mòbils. D'entre totes les possibles funcionalitats, la més important sens dubte, és la que permetria visualitzar les wikis d'un curs de moodle des d'un dispositiu mòbil. També és podrien introduir una sèrie de millores en la capa de presentació, a fi de fer més user-friendly el sistema.

8.3 Agraïments

Arribats a aquest punt, m'agradaria esmentar i donar les gràcies a totes les persones que, per la seva col·laboració (o falta de col·laboració) han fet que aquest projecte s'hagi tancat satisfactòriament.

En primer lloc, m'agradaria agraïr als meus companys de feina del Laboratori de Càlcul de la Facultat d'Informàtica de Barcelona, pel seu inestimable ajut i recolzament durant les fases inicials del projecte en que J2ME era poc més que unes sigles per a mi: Jose Francisco Crespo, Alain Muñoz, Núria Lara, Cristina Montañola, Marc Torruella, Joan Esteve, Jordi Creus, David Marcè, Diego Jarne, Jordi Gassó, i als que probablement em deixo. També agraîr a tots els membres del DfwikiTeam, dirigits per Marc Alier, la seva col·laboració, especialment a Oriol Nieto i a Alex Gorjón, autors del webservicelib.

Mencionar també als meus amics més íntims per recolzar-me i estar al meu costat en tot moment. Gràcies Toni, Cristian, Jose, Javi i Mikel, per fer-me passar unes estones tan divertides.

Gràcies també als meus companys de pis, concretament al meu company d'habitació, Mohammed, per suportar-me fins a tan altes hores de la matinada escrivint interminables linees de codi.

Agrair als meus pares i germà la seva comprensió, i demanar-los-hi disculpes pels injsutificables canvis d'humor derivats del procés de desenvolupament del projecte, i a la meva parella, el seu recolzament incondicional i la seva capacitat per a evitar que em desanimés en alguns moments.

Gràcies a tots els esmentats, i a tots els que sens dubte per oblit no he inclòs.

Bibliografía

- Dolors Costal – Xavier Franch – Mª Rivera Sancho – Ernest Teniente. Enginyeria del Software: Especificació. Edicions UPC, 2005
- Craig Larman : UML y Patrones. (Una introducción al análisis y diseño orientado a objetos y al proceso unificado). Prentice Hall, 2002.
- Grady Booch, James Rumbaugh, Ivar Jacobson : El Lenguaje Unificado de Modelado. Addison Wesley, 2001
- Java : http://java.sun.co/
- XML 1.1 : http://www.w3.org/TR/2006/REC-xml11-20060816/
- http://www.mobilefish.com/emulators/j2me_wt/j2me_wt_quickguide_22_emulator.html
- http://www.it.uc3m.es/java/#J2ME
- http://labs.silverorange.com/archives/2003/july/privaterss
- http://www.it.uc3m.es/celeste/docencia/j2me/tutoriales/midp2_0/
- http://developers.sun.com/techtopics/mobility/midp/articles/parsingxml/
- http://kxml.objectweb.org/software/documentation/apidocs/
- http://www.tiddlywiki.com/
- http://java.sun.com/javame/reference/apis/jsr118/
- http://www.mailxmail.com/curso/informatica/j2me/capitulo25.htm
- http://www.it.uc3m.es/celeste/docencia/j2me/tutoriales/midp2_0/PracticaIO/
- http://www.geocities.com/CollegePark/Quad/8901/cap10.htm#inputstr
- http://www.eclipse.org/
- http://eclipseme.org/
- http://www.eclipsedownload.com/
- http://subclipse.tigris.org/
- http://es.wikipedia.org/
- http://kxml.objectweb.org/
- http://www.twmacinta.com/myjava/fast_md5.php