UNIVERSITAT OBERTA DE CATALUNYA

ESTUDIS D'ENGINYERIA EN INFORMÀTICA

Projecte Fi de Carrera

APLICACIONS WEB PER TREBALL COL·LABORATIU

"DISSENY I PROTOTIPATGE D'UN PROGRAMARI PER L'US DELS DISPOSITIUS MÒBILS EN L'APRENENTATGE ONLINE"

Consultor: Fatos Xhafa
Alumne: Ana Cartanyà Yárnoz
5 de gener de 2012

INDEX

Introducció

1.1 Propòsit del projecte

Aquest treball és el projecte final de carrera d'Enginyeria en Informàtica de la UOC. Està englobat dins l'àrea d'aplicacions web per a treball col·laboratiu. En ell farem la descripció de tot el procés, de la metodologia utilitzada i la temporalització de cada una de les etapes.

L'objectiu d'aquest projecte és crear una aplicació per dispositius mòbils que permeti interactuar amb Moodle, un entorn d'aprenentatge online. Actualment s'ha estès molt els smartphones, i cada vegada hi ha més gent connectada al mon web les 24h amb aquests aparells. I s'estan acostumant a rebre alertes cada vegada que reben un correu, un tweet, un missatge al facebook,... lo que fa que la gent estigui sempre assabentada de tot en qualsevol lloc i moment.

Aprofitant aquest concepte. Es vol generar una aplicació per mòbil que faciliti la interacció amb un campus virtual, que permeti a l'alumne està sempre assabentat mitjançant alertes, dels missatges, canvis,... que hi han a les aules. Així com crear una aplicació directe que faciliti la interacció amb el campus.

1. Descripció del projecte

2.1 Objectius

A la finalització d'aquest projecte es pretén haver dissenyat i implementat un prototip que permeti interactuar amb un campus virtual i rebre alertes a dispositius mòbils.
Els objectius que podem destacar són:

- *Consolidar les etapes d'un projecte: anàlisi i disseny, implementació*.
  Com a tot projecte, un objectiu interessant és la realització de tota la feina per etapes, passant per unes fase d'anàlisi, disseny, implementació del codi, proves del prototipus final i per últim la documentació que reculli tot el procés.

- *Entendre el concepte de web col·laborativa*.
  Al triar aquesta àrea, estem construint un prototip amb una finalitat, generar una eina que faciliti una feina a un grup de persones.

- *Aprendre l'arquitectura, interfícies i llenguatge de programació per dispositius mòbils.*
  En aquest cas qui gestionarà les alertes i la connexió al campus virtual serà el mateix dispositiu. Per aquest treball serà necessari aprendre el fonaments d'aquests dispositius per poder crear l'aplicació.

2.2 Resultats esperats

Amb la realització d'aquest treball, s'espera aconseguir realitzar tots els passos per fer un prototipus per dispositius mòbils.

- Realitzar totes les etapes d'un projecte de forma coherent. Diferenciant les etapes: que es farà, com es farà i la generació de codi.

- Entendre el funcionament de les aplicacions per dispositius mòbils. Com funcionen, com interactuen amb aplicacions webs, com es presenta la informació, com generen alertes,...

- Crear una aplicació per dispositius mòbils que interactiu amb un entorn col·laboratiu i rebi les alertes sobre les accions d'aquest campus virtual.

2.3 Motivació

Les motivacions que m'han portat a escollir aquest treballa han estat:

Realitzar el projecte final de carrera per poder obtenir la titulació d'Enginyera en Informàtica. Com a finalització dels meus estudis, és interessant realitzar un treball que faci un recull de tot allò que hem anat aprenent i que ens permeti fer tot el procés de crear un prototipus a partir d'una idea o d'una millora que faciliti la feina futura a algú. I d'aquesta forma consolidar els coneixements.

Conèixer més a fons les aplicacions col·laboratives, en aquest cas els campus virtuals. Ja que cada vegada més, encara que no sigui universitats online com la UOC, els campus estan agafant molta força. Els calendaris, les notes, les pràctiques,... avui en dia tots els estudis tenen un espai virtual en el qual l'alumne interactua amb ell. I aquest entorn genera alertes que avui en dia podem tractar per facilitar a l'alumne la planificació de la seva feina.

Aprofundir en una tecnologia emergent que cada vegada està més a l'abast de tothom. És interessant crear aplicacions que apropin el mon web a aquest tipus de dispositius. I sobretot, com en aquest cas, apropar un campus virtual a alumnes per facilitar la gestió durant els seus estudis.

2. Organització del projecte

Les tasques realitzades durant el projecte són:

3.1 Realització de les tasques

T1. Presentació i formalització del PFC

Els primers dies de classe el consultor de l'àrea va proposar uns temes. Vam fer la proposta de fer un projecte conjunt amb un altre alumne, però separat en dues parts per poder independitzar els dos treballs. El consultor va estar d'acord en la proposta.

T2. Estudi de la Documentació aportada pel consultor

La documentació aportada pel consultor va ser un llistat de punts que havia de contenir el treball. I uns exemples per poder veure per on havíem d'enfocar les diferents entregues que havíem d'anar fent.

T3. Elaboració del Pla de Treball

Aquest punt serveix per plantejar el projecte. Enfocar cap a on anirà el treball, objectius proposats, resultats esperats, motivació i planificació de les diferents tasques.

T4. Instal·lació de les tecnologies per dispositius mòbils

T4.1. Instal·lació del paquet MAMPP

Amb aquestes eines podrem posar en marxar una aplicació web, més concretament Moodle, que és una eina opensource que serveix per crear campus virtuals.

T4.2. Instal·lació de l'entorn per desenvolupar eines per dispositius mòbils

S'instal·larà un eclipse, amb les eines de java necessàries per crear l'entorn que servirà per fer el desenvolupament del prototipus.

T5. Estudi d'un campus virtual

Farem un estudi de com Moodle gestiona l'enviament d'alertes. Quin tipus d'alerta envia. I amb això generarem un llistat dels possibles requeriments que podrà contenir la futura aplicació mòbil.

T6. Estudi d'aplicacions per dispositius mòbils.

En aquest punt s'haurà d'investigar i aprendre com funcionen la gestió de les alertes a dispositius mòbils. Com es reben, com es visualitzen i quines accions es poden fer. S'haurà de començar a desenvolupar alguns exemples amb programació per a dispositius mòbils. Per entendre quin tipus d'estructura tenen. Aquesta tasca es pot desenvolupar en paral·lel mentre es fa la fase d'anàlisi de requeriments, ja que permetrà agafar el primer contacte amb la tecnologia.

T7. Fase d'anàlisi dels requeriments

Una vegada extret tota la informació de lo que haurem de construir passarem a la fase d'anàlisi on llistarem tots els requeriments que haurà de tenir la nostre aplicació.

T8. Fase de disseny

Quan sapiguem que volem fer, passarem a l'etapa de com ho farem. En aquesta fase triarem la tecnologia a emprar i dissenyarem cada un dels punts esmentats a l'apartat anterior.

T9. Fase d'implementació

En aquesta etapa s'implementarà la solució. Es generarà el codi que permetrà crear el prototipus.

T10. Fase de proves

En aquesta fase haurem de fer un llistat de les proves exhaustives que haurem de fer al prototipus per avaluar el seu bon funcionament.

T11. Manual d'instal·lació i d'usuari

S'haurà de crear un manual d'instal·lació perquè els futurs usuaris sàpiguen com instal·lar l'eina al seu dispositiu. També s'haurà de crear un manual d'usuari que expliqui les possibilitats de l'aplicació i el seu funcionament.

T12. Futures línies i conclusions

Una vegada creat el prototipus, i havent fent les proves. S'haurà de crear un llistat amb les futures línies per poder millorar el prototipus. També es farà un recull de les conclusions de tot el procés.

T13. Creació de la documentació del tot el procés

Tots els passos que hem fet per crear l'aplicació final estaran recollits en un document. On estarà detallat cada una de les etapes per així formar la part final de la memòria final del projecte.

T14. Lliurament final del projecte

Per finalitzar el PFC s'haurà de fer una entrega final amb: el prototipus, la documentació del projecte, manuals d'instal·lació i usuari i una presentació de la feina feta.

3. Estimació i planificació del projecte

3.1 Planificació temporal

La planificació del projecte estarà basat en les entregues de les PACs i lliurament final:

- **PAC1**: 26 setembre - 17 d'octubre

En la primera PAC s'escollirà el projecte, i es farà la planificació, descripció, objectius.

- **PAC2**: 18 d'octubre - 25 de novembre

En la segona PAC es farà l'anàlisi de requeriments, disseny i implementació del prototip.

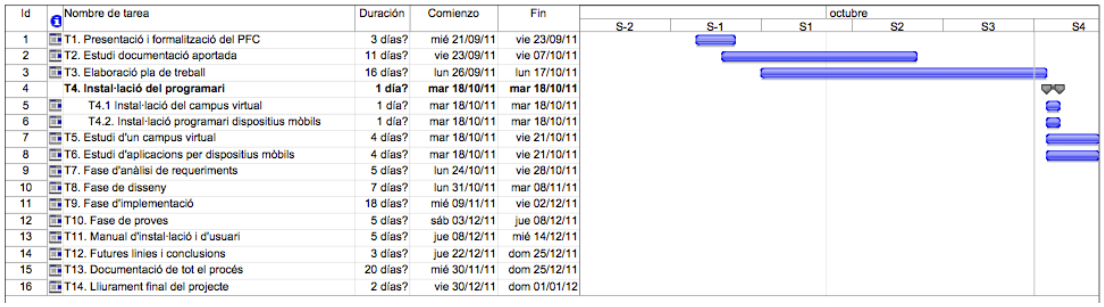
- **Entrega final**: 26 de novembre - 5 de gener

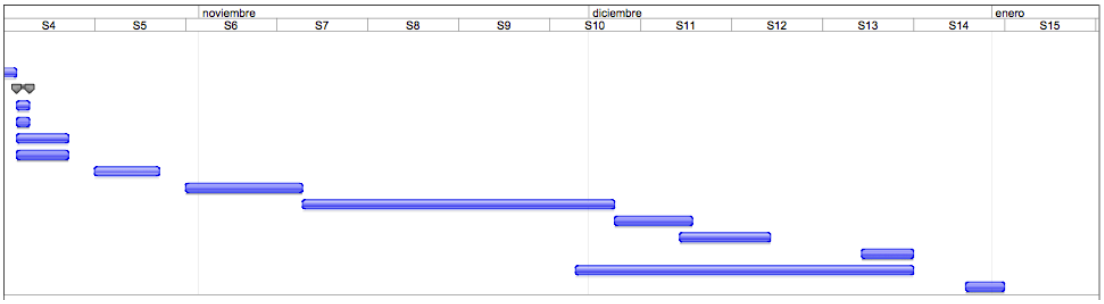
Aquesta és l'entrega final. On hi haurà el prototip, la documentació o memòria del projecte, manual d'usuari i d'instal·lació i presentació final del PFC

3.2 Diagrama de Gantt de temporització de tasques

El diagrama de Gantt és el següent:
Inici del projecte: 21/09/2011
Entrega del projecte: 01/01/2012

Id	Nombre de tarea	Duración	Comienzo	Fin
1	T1. Presentació i formalització del PFC	3 días?	mié 21/09/11	vie 23/09/11
2	T2. Estudi documentació aportada	11 días?	vie 23/09/11	vie 07/10/11
3	T3. Elaboració pla de treball	16 días?	lun 26/09/11	lun 17/10/11
4	**T4. Instal·lació del programari**	**1 día?**	**mar 18/10/11**	**mar 18/10/11**
5	T4.1 Instal·lació del campus virtual	1 día?	mar 18/10/11	mar 18/10/11
6	T4.2. Instal·lació programari dispositius mòbils	1 día?	mar 18/10/11	mar 18/10/11
7	T5. Estudi d'un campus virtual	4 días?	mar 18/10/11	vie 21/10/11
8	T6. Estudi d'aplicacions per dispositius mòbils	4 días?	mar 18/10/11	vie 21/10/11
9	T7. Fase d'anàlisi de requeriments	5 días?	lun 24/10/11	vie 28/10/11
10	T8. Fase de disseny	7 días?	lun 31/10/11	mar 08/11/11
11	T9. Fase d'implementació	18 días?	mié 09/11/11	vie 02/12/11
12	T10. Fase de proves	5 días?	sáb 03/12/11	jue 08/12/11
13	T11. Manual d'instal·lació i d'usuari	5 días?	jue 08/12/11	mié 14/12/11
14	T12. Futures línies i conclusions	3 días?	jue 22/12/11	dom 25/12/11
15	T13. Documentació de tot el procés	20 días?	mié 30/11/11	dom 25/12/11
16	T14. Lliurament final del projecte	2 días?	vie 30/12/11	dom 01/01/12





4. Altres consideracions

El cost de per desenvolupar aquest projectes serien només les hores dedicades pel recurs en fer tot el procés. S'ha triat un campus on-line opensource, que no té cap mena de cost. I per desenvolupar eines i llenguatge que també permeten la creació d'un codi lliure i sense cap mena de cost. Per lo tant dins el pressupost només es contemplarien les hores del recurs assignat.

La planificació dividida en entregues deixa poc temps per la part d'implementació. Però les setmanes compreses dins la PAC2 seran les que més feina tindran, i podria ser que alguna part de la implementació s'hagués de deixar pel lliurament final.

Aquest projecte forma part d'un altre PFC d'un alumne de la UOC, *"Disseny i prototipatge d'un programari per l'ús dels dispositius mòbils en l'aprenentatge on-line"*, que s'encarrega de la creació d'un prototipus que permeti la comunicació entre els dispositius mòbils i el campus virtual.
Gràcies a ell, es crearà un middleware que permetrà la interacció per poder rebre les alertes i notificacions i per poder connectar l'aplicació del dispositiu mòbil a l'aplicació web.
Això s'haurà de tenir present perquè el prototipus que desenvoluparem en tot el procés tindrà una força dependència amb l'altre part del projecte. Ja que sense ell no hi haurà aquesta capa de comunicació.

5. Instal·lació i configuració de l'entorn

5.1 Instal·lació

5.1.1 Instal·lació de l'eina Moodle

Existeix un paquet al mercat que ja permet instal·lar un Moodle i un MAMP que podem descarregar a: http://moodle.org/downloads/

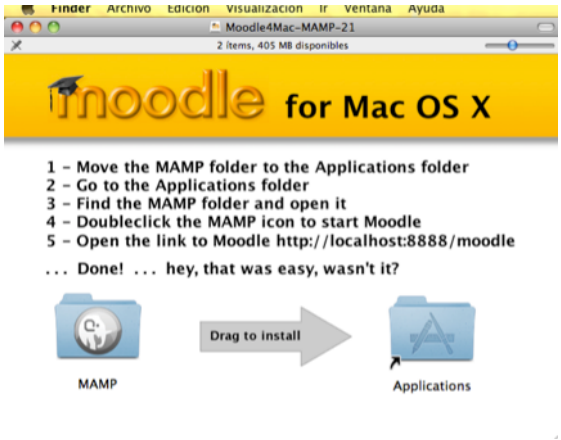
El MAMP és un instal·lable que incorpora una sèrie d'eines que permet aixecar una aplicació web feta amb PHP, com és el cas de Moodle. Les sigles corresponen a (MAC, Apache, MySQL, PHP):

- Apache: és el servidor d'aplicacions necessari per les aplicacions desenvolupades en aquest llenguatge.

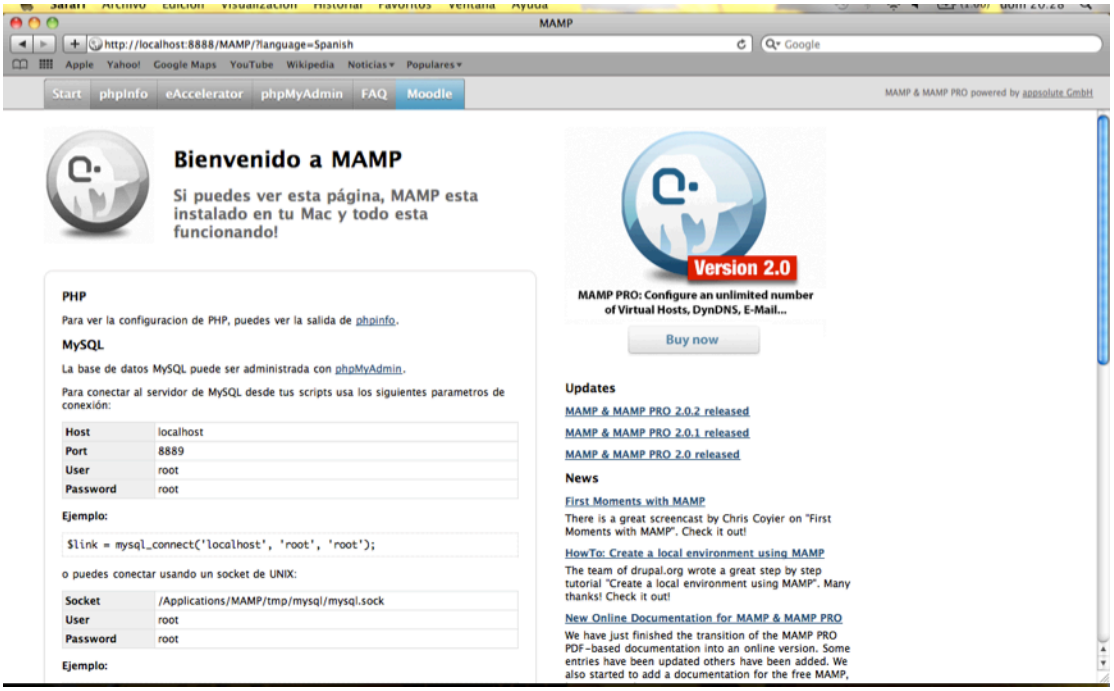
- PHP: és el llenguatge de programació amb el que està desenvolupat Moodle i es necessita un intèrpret perquè l'Apache l'entengui.

- MySQL: és la base de dades que utilitza Moodle per guardar tota la informació del portal web.

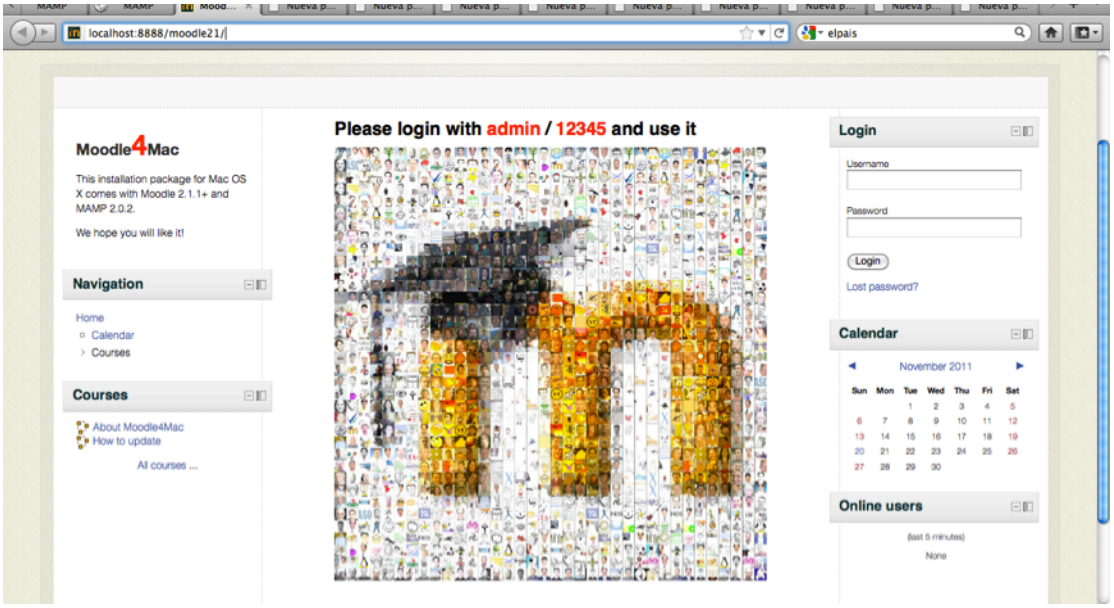
El procés d'instal·lació és molt senzill. Com el paquet ja porta tant el portal com les eines per fer-lo funciona. Una vegada instal·lat només cal desplegar l'aplicació Moodle.



Una vegada instal·lat, anem a la carpeta d'aplicacions i allà trobem l'enllaç a Moodle. Cliquem i sens obre la consola de configuració del MAMP:



I si li donem a la pestanya de *Moodle* veiem que ja ens apareix la primera plana amb el Moodle configurat amb un usuari i contrasenya creats per defecte com administrador:



Ja tenim llest l'entorn per fer la configuració bàsica per fer un petit estudi de l'eina.

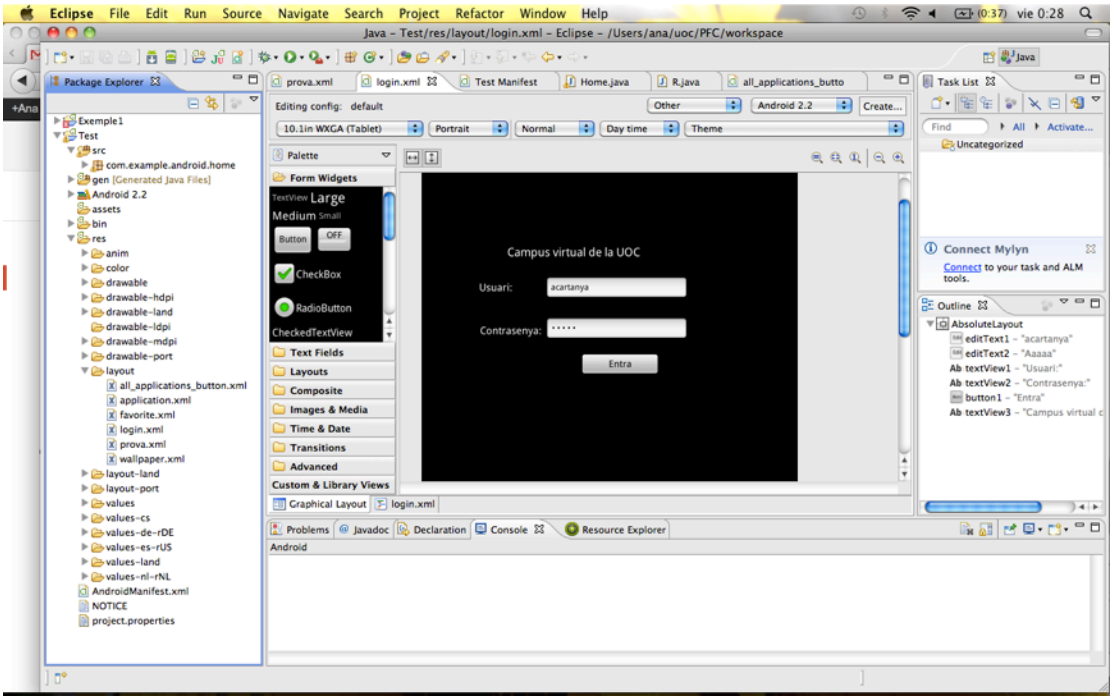
5.1.2 Instal·lació de l'entorn de desenvolupament

Per poder desenvolupar l'aplicació necessitarem instal·lar unes eines que ens facilitin i permetin treballar:

- *Eclipse*: és l'entorn que muntarem per desenvolupar l'aplicació. És compatible amb molts llenguatges i permet tenir totes les eines de programació centralitzades, de tal forma que tenim un editor de codi i fàcilment podrem desplegar l'aplicació desenvolupada per fer les proves.

- *SDK de Android*: més endavant farem l'estudi dels diferents dispositius mòbils existents al mercat. En el nostre cas com triarem la tecnologia Android, necessitarem l'API per tenir les llibreries de totes les funcions disponibles per desenvolupar.

Per provar que totes les eines funcionen i estan ben configurades, crearem un projecte d'exemple:



6. Anàlisi

6.1 Casos d'ús

6.1.1 Actors

Els actors que podem diferenciar en el nostre sistema són:

- **Administrador**: encarregat del manteniment del portal web. De la configuració de l'aparença, menús, logos,...

- **Professor**: és el responsable d'un curs o assignatura. I gestionarà el calendari d'entregues, les qualificacions, els qüestionaris,...

- **Alumne**: és el que rep el curs. Interactuarà amb el fòrum, rebrà missatges, interactuarà amb les assignatures, rebrà les qualificacions, es comunicarà amb professors i alumnes,..

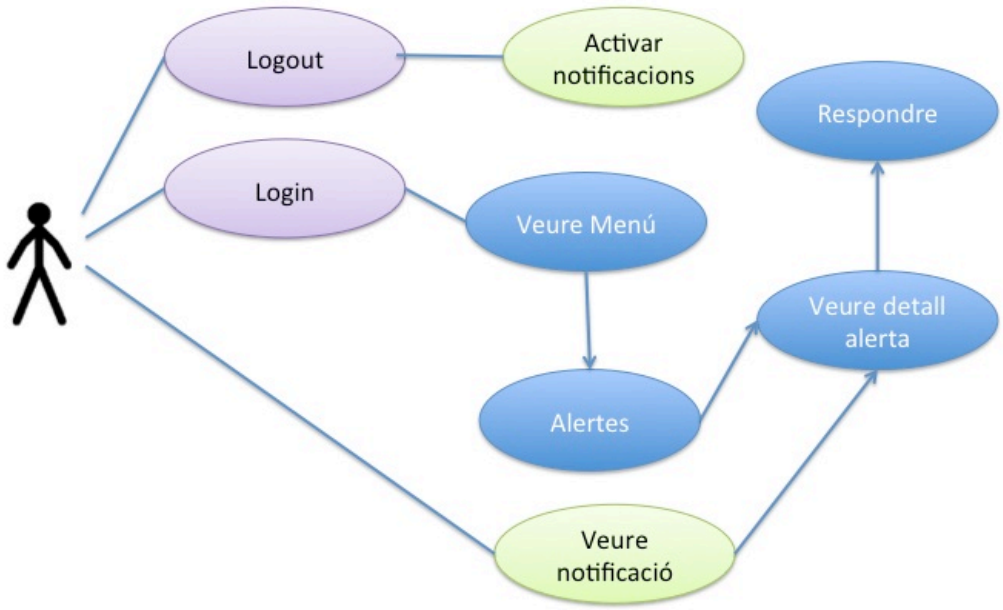
Encara que tinguem aquests tres tipus d'usuaris, a efectes d'interactuar amb les funcionalitats de l'aplicació seran les mateixes. Ja que el rol de l'usuari només afectarà al tipus d'alertes que podran rebre. Però el funcionament de l'eina serà la mateixa pels tres. Per això en els cassos d'ús parlarem d'un usuari genèric que interactuarà amb l'aplicació mòbil. Quan fem el detall del tipus de notificacions existents, llavors ja distingirem els diferents tipus d'usuaris.

6.1.2 Casos d'ús

Els cassos d'ús del nostre sistema estan relacionats amb la interacció que tindrà l'usuari amb l'aplicació per a mòbil que interactua amb Moodle.

D'una banda tindrem l'aplicació mòbil on s'emmagatzemen les alertes i per l'altra tindrem l'opció d'activar la recepció de les alertes per rebre-les com a notificacions al nostre mòbil.

Anem a diferenciar cada un dels casos d'ús del nostre sistema:



Activar l'aplicació:
Aquest cas d'us contempla logar-se i deslogar-se de l'aplicació:

Codi	C000
Nom	Login
Actors	Usuari
Descripció	L'usuari haurà d'autenticar-se a l'aplicació amb un usuari i contrasenya per demostrar que està donat d'alta al portal web i per rebre aquelles alertes que tingui activades des de Moodle.
Precondició	L'usuari no està autenticat contra l'aplicació
Postcondició	L'usuari quedarà logat dins l'aplicació

Codi	CU0001
Nom	Logout
Actors	Usuari
Descripció	Existirà la possibilitat de desconnectar-se de l'aplicació i així tancar la recepció de notificacions.
Precondició	L'usuari ha d'estar autenticat contra l'aplicació
Postcondició	L'usuari no estarà autenticat

Aplicació

Aquest apartat correspon a l'aplicació per mòbil que ens mostrarà un històric de les notificacions. Tindrem un menú principal dels grups i un submenú que llistarà totes les alertes d'aquell grup. Si cliquem una d'elles veurem el detall:

Codi	CU0002
Nom	Veure grups d'alertes
Actors	Usuari
Descripció	Les notificacions estan classificades per categories. La primera pantalla de l'aplicació ens mostrarà totes les categories
Precondició	L'usuari ha d'estar autenticat contra l'aplicació
Postcondició	

Codi	CU0003
Nom	Llistar alertes
Actors	Usuari
Descripció	Dins del grup podrem veure el llistat de totes les alertes generades. Així sempre les podrem consultar. Tindrem les actuals i les històriques
Precondició	L'usuari ha d'estar autenticat.
Postcondició	

Codi	CU0004
Nom	Veure detall alerta
Actors	Usuari
Descripció	Al seleccionar una alerta concreta, podrem veure el detall del missatge que ens han enviat.
Precondició	L'usuari ha d'estar autenticat i ha d'haver triat un grup d'alertes. O entrar a través de les notificacions del mòbil.
Postcondició	

Notificacions:
Les notificacions són aquells missatges que apareixen al menú principal del mòbil i permeten l'accés ràpid a certs missatges. En el nostre cas podrem activar la recepció de la notificació i podrem llegir-la:

Codi	CU0005
Nom	Activar notificacions
Actors	Usuari
Descripció	Activant el servei de notificacions, el nostre mòbil mostrarà una alerta cada vegada que rebi una notificació de Moodle. Això permetrà que l'usuari llegeixi ràpidament el missatge.
Precondició	L'usuari ha d'estar autenticat i ha de tenir les alertes de notificacions activades a l'aplicació.
Postcondició	

Codi	CU0006

Nom	Veure notificació
Actors	Usuari
Descripció	Una vegada cliquem a sobre de la notificació, podrem veure en detall el contingut de la notificació. Aquest cas, ens reconduirà cap a l'aplicació web per mòbil, dins el grup que pertoqui l'alerta i mostrant la informació.
Precondició	L'usuari ha d'estar autenticat i ha de tenir les alertes de notificacions activades a l'aplicació.
Postcondició	

6.2 Funcionalitats més importants

Del plantejament del treball i de les interaccions amb Moodle podem detallar les funcionalitats que contemplarà aquest projecte:

6.2.1 Capa d'accés al webservice

Per comunicar el dispositiu mòbil amb Moodle, s'utilitzarà una capa o middleware que oferirà uns webservices disponibles per poder interactuar.

D'aquesta forma per l'aplicació, tota la gestió de les notificacions per part de Moodle serà transparent. Ja que únicament caldrà saber l'estructura de cada un dels webservice per poder així fer la crida, enviar els paràmetres (en cas que siguin necessaris) i rebre una resposta.
Aquesta capa facilita la feina, ja que ens permet sense interactuar amb cap objecte del portal poder rebre la informació necessària.

D'aquesta forma la comunicació amb Moodle sempre es produirà a través d'aquesta capa.

L'estructura de forma simplificada seria com la següent imatge, on el Middleware serà l'encarregat de la comunicació i es veu que l'aplicació mòbil no atacarà mai el portal:

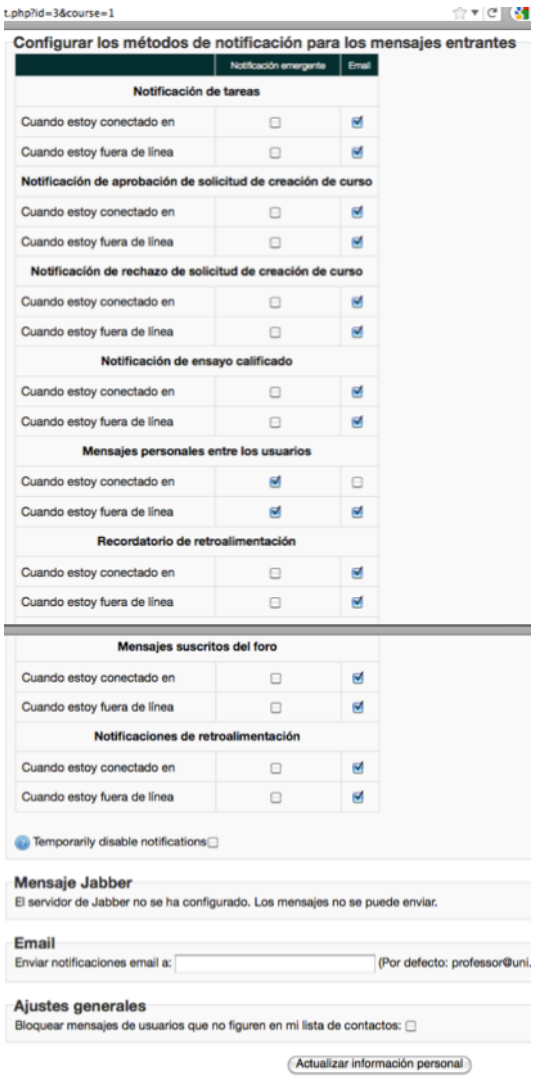
6.2.2 Aplicació mòbil

Per entendre com funcionen les notificacions de l'eina de Moodle, hem hagut de fer un petit estudi previ per veure com es generen i quin tipus de notificacions contemplen actualment el portal.

Farem una explicació de les alertes generades per Moodle que són configurables al menú de *Configuració – Missatges.*:

Les notificacions que es mostraran al dispositiu mòbil són les alertes generades pel Moodle. Dins de l'aplicació web hi ha un menú on podem configurar i activar aquestes accions:
L'usuari haurà d'entrar a Moodle per activar/desactivar les notificacions que l'interessi rebre al mòbil (Com hem comentat abans aquest treball forma part d'un altre que s'està construint en paral·lel, on s'afegirà una tercera columna que sigui mòbil per controlar l'activació dels missatges).

Aquesta imatge representa el menú de configuració dins Moodle:



A continuació farem una descripció de cada una de les notificacions actuals de Moodle:

Notificació	Descripció
Notificació de les tasques	Són les tasques pendents d l'usuari.
Notificació d'acceptació de creació d'un curs	És el missatge conforme s'ha creat correctament un curs.
Notificació de rebuig de creació d'un curs	És el missatge que apareixerà en cas de no ser. Acceptat el curs
Notificació de qualificació	És el missatge que es mostrarà sobre les qualificacions de les assignatures.
Notificació de missatges personals	Són els missatges que rebrà l'usuari, com una bústia de correu.
Recordatori de feedback	Aquest apartat contempla la possibilitat de crear un "feedback" sobre l'opinió de l'usuari sobre algo. Com per exemple fer qüestionaris sobre l'assignatura, o aportacions de millora. Aquesta acció s'encarregarà de recordar a l'usuari que té una petició pendent.
Notificació de missatges al fòrum	Notificació sobre els missatges escrits al fòrum de les diferents assignatures i del curs.
Notificació de feedback	Notificació conforme s'ha respost un qüestionari.

De les alertes mencionades anteriorment, farem una tria i definirem uns bloc s funcionals que classificaran les alertes el diferents components per definir unes subcategories. Aquestes seran:

Bloc	Descripció
Missatges	Missatges que pot rebre l'alumne per la bústia del campus.
Fòrum	Són els missatges relacionats amb el fòrum.
Campus	Són aquelles alertes relacionades amb les assignatures que cursa o que porta l'usuari, les qualificacions, les tasques, els qüestionaris, acceptació de creació d'un curs...
Xat	Són els missatges personal instantanis que pot rebre l'usuari.
Calendari	Són les notificacions relacionades amb els events del calendari.

També caldrà definir unes *retroaccions* sobre aquestes alertes, o sigui, què podrà fer l'usuari una vegada hagi rebut l'alerta. En aquest cas dependrà del bloc funcional on estigui l'alerta. Generalment l'acció que podrà realitzarà l'usuari serà la de respondre el missatge. Que una vegada omplerts els camps, s'enviarà a un webservice perquè rebi la resposta i la pugui processar dins de l'entorn de Moodle per crear una nova notificació cap a l'usuari que sigui el receptor.

En els blocs definits anteriorment, només hi hauran retroaccions en: Missatge, Fòrum i Xat. Els missatges del Campus i del Calendari només seran de consulta però no es podrà interactuar amb elles una vegada rebudes. Considerem que com son events de sistema, no estan generats per un usuari físic i no té sentit la possibilitat de respondre la notificació.
En els altres permetrà una resposta cap al remitent del missatge rebut.

6.2.3 Notificacions al dispositiu mòbil

Aquestes alertes que anirà rebent l'usuari hauran de ser configurables per rebre-les com a Notificacions al mòbil. D'aquesta forma l'usuari estarà assabentant en el mateix moment de les novetats que hi ha hagut al portal web. Des de l'aplicació mòbil hauran de ser configurables per activar-les o desactivar-les.

7. Disseny

7.1 Característiques dels dispositius mòbils

Abans de començar a especificar el disseny de l'aplicació mòbil, hem de tenir present per quin tipus de dispositius es pretén fer el desenvolupament. Per ser conscients de les característiques i limitacions que té. Un s'ha de treure del cap la idea que desenvolupar aplicacions per mòbil és lo mateix que les tradicionals però per dispositius amb pantalla més petita:

Pantalla:
La primera limitacions amb la que ens trobarem serà el tamany de la pantalla on es visualitzarà l'aplicació. Cada vegada estan sortint al mercat dissenys de dispositius mòbils amb pantalles molt més grans. Encara així l'espai és limitat. Això significa que l'aplicació haurà de ser molt senzilla, amb menús molt simples i fàcilment navegable.

Tàctil:
Molts dispositius d'aquest tipus presenten una pantalla i teclat tàctil. Això fa guanyar en espai físic al dispositiu. Però al mateix temps és necessari que els enllaços siguin lo suficientment grans perquè sigui fàcilment manipulable. Ja que es perd precisió a l'hora de clicar un enllaç.

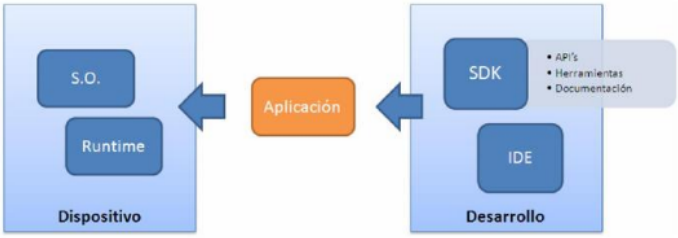
Potència:
No hem d'oblidar que parlem de dispositius petits amb poca potencia, per lo tant l'aplicació haurà de ser senzilla i no gaire pesada.

Espai:

El nivell de memòria també és limitat. I l'aplicació ha de ser lleugera i ocupar poc.

Accés a Internet:
S'ha de tenir present que l'aplicació Moodle estarà a la xarxa. El dispositiu mòbil que utilitzi aquesta aplicació haurà de tenir connexió via wifi o 3G a Internet per poder connectar-se al campus on-line. Avui en dia tots els smartphones tenen accés a Internet, però és un aspecte a tenir present.

Tipus de mòbil:
Abans de començar a desenvolupar ens haurem d'informar del sistema operatiu que té el terminal i les possibilitats que ofereix. Al mercat existeixen molt dispositius diferents (Iphone, Adroid, Blackberry, Symbian,...) amb característiques ben diferents. S'haurà de triar per quin tipus de dispositiu mòbil es desenvoluparà l'aplicació per així triar la tecnologia.



7.2 Arquitectura de la Tecnologia triada

Per fer aquest projecte hem triat les següents tecnologies segons cada una de les parts de la solució:

-Tecnologia PHP
Aquesta tecnologia ve donada per Moodle, ja que aquesta eina està implementada amb ella. Per fer l'anàlisi hem hagut d'instal·lar el paquet Moodle per poder fer un estudi previ de l'eina.
Aquesta part només l'haurem d'instal·lar per poder fer l'estudi previ. Però no caldrà fer cap modificació al portal.

-Tecnologia Mòbil
Per fer el desenvolupament haurem d'escollir algun dels sistemes que actualment permeten fer aplicacions per dispositius mòbils. Al mercat en temes d'smartphones tenim diferents opcions, anem a definir algunes:

- *Iphone – OS*: El llenguatge que utilitza és el Objective-C. És un llenguatge de programació orientat a objectes creat basat en C. És un sistema més restrictiu.

- *Blackberry*: utilitza Java ME, una plataforma estàndard del sector que defineix els conjunts comuns de l'API de Java per dispositius inal·làmbrics. Una aplicació Java ME en un dispositiu Blackberry s'executa

en una màquina virtual Blackberry Java Virtual Machine, que proporciona tots els serveis de temps d'execució a les aplicacions i realitza les funcions d'assignar memòria, comprovar seguretat i recollida de dades.

Una particularitat d'aquests dispositius es que estan orientats a negoci, pensats per ser clients d'una xarxa corporativa que inclogui: correu electrònic, serveix web,...

- *Symbian*: és un sistema operatiu per a dispositius mòbils on Nokia és el major accionista. Un dels problemes que hi ha es que encara que portin el mateix sistema operatiu, existeixen mòbils amb característiques molt diferents a la interfícies d'usuari. De fet Symbian defineix una sèrie de plataformes que permet definir famílies de mòbils segons el seu sistema operatiu i les seves característiques d'IU (a la majoria de Nokia és S60). Cada versió de cada plataforma consisteix en un conjunt d'APIs que donen accés a les funcions de les que disposa el mòbil. El conjunt de funcions que necessita l'aplicació determinarà la versió de la plataforma que necessitarem. Una vegada definida la plataforma podrem començar a treballar amb la SDK corresponent.

- *Android*:
  És un sistema operatiu basat en el nucli de Linux, dissenyat per dispositius mòbils, tablets, reproductors de mp3, netbooks, pcs. Desenvolupat per l'empresa Google inicialment i després per Open Handset Alliance (una agrupació de 48 empreses de hardware, software i telecomunicacions liderada per Google) compromesa en la promoció d'estàndards oberts per dispositius mòbils. Això significa que qualsevol desenvolupador pot crear i desenvolupar aplicacions per aquests dispositius.

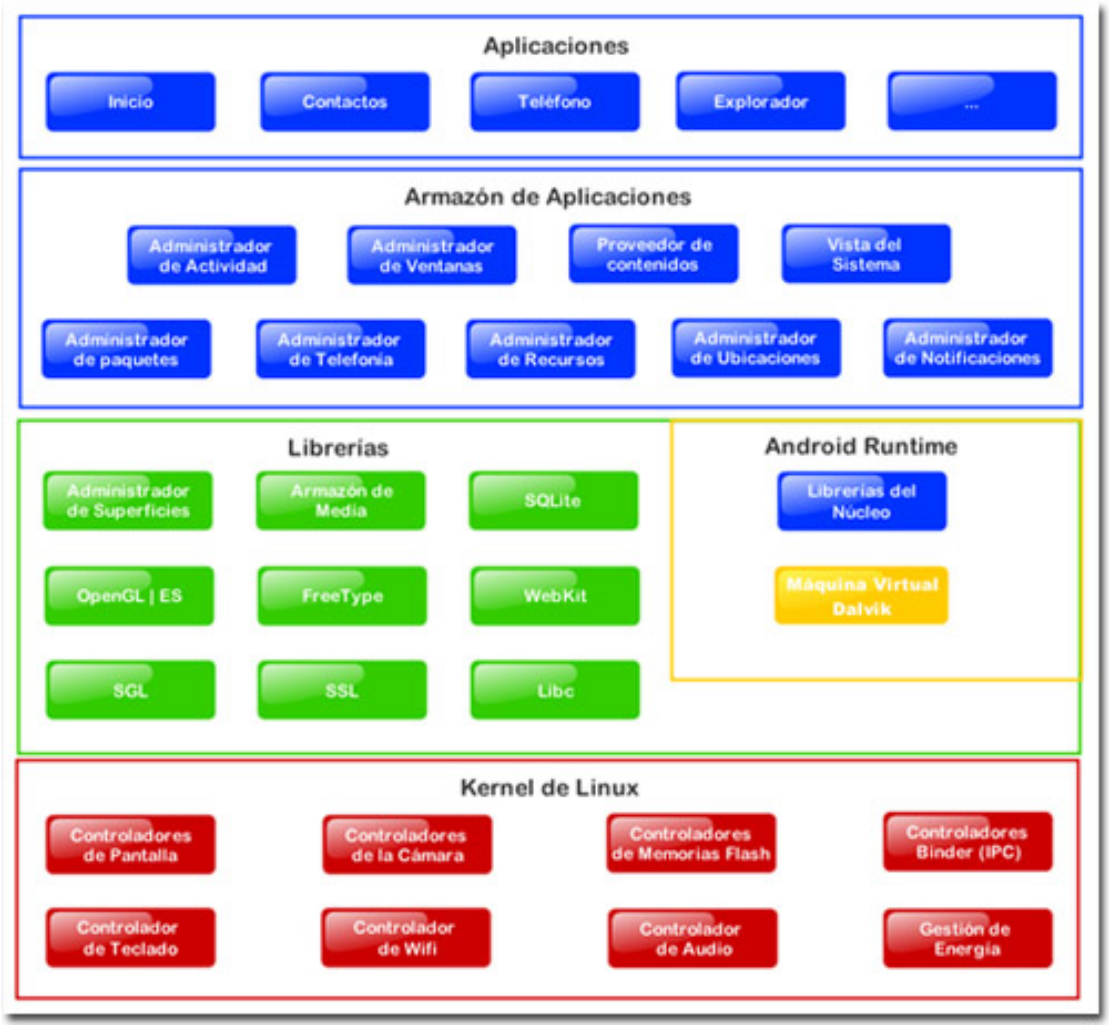
  Característiques del sistema:
  - Framework d'aplicacions que permet reemplaçar i reutilitzar els components.
  - Navegador integrat basat en el motor open source Webkit
  - SQLite base de dades per l'emmagatzemen estructurat que s'integra directament amb les aplicacions.

Arquitectura Android

Hem triat Android, ja que és un llenguatge opensource que permet desenvolupar sense costos. Actualment la tecnologia Android s'ha estes molt i existeixen molts dispositius mòbils (HTC, Samsung, Sony) que l'utilitzen així com diferents tipus de tablets.

A part aquesta aplicació permetrà que estigui disponible tant per mòbils com per tablets.

Un esquema de l'arquitectura que utilitza és:



Funcionament:
Anem a explicar l'estructura de l'arquitectura d'aquest llenguatge per començar a esquematitzar la feina que haurem de fer:
Quan creem un projecte a Eclipse del tipus Android, per defecte sens creen unes carpetes, entre elles està:

/res:
Que inclou els subdirectoris:

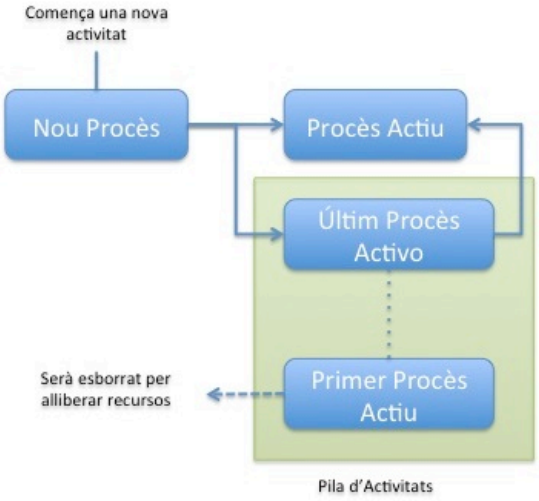
- drawable: amb els següents subdirectoris Drawable-hdpi, drawable-ldpi, drawable-ldpi, directoris on s'emmagatzemen les imatges que utilitzarà l'aplicació.
- values: directori que contés les cadenes de text que utilitzarà l'aplicació. No és obligatori definir totes, però és recomanable.
- layout: conté el XML que representen les interfícies gràfiques.

/gen:
On trobem el fitxer anomenat R.java. Aquest fitxer no s'ha de modificar, perquè l'eclipse s'encarregarà de posar el codi que genera ell mateix. Serveix per enllaçar les coses que anem fent en XML (les diferents pantalles o interfícies) amb la programació Java.

/src:
En aquesta carpeta tindrem els diferents úblic amb les classes **Activity** corresponent. Les classes que crearem hauran d'extendre de la classe Activity.

Les aplicacions d'Android funcionen sota un esquema d'Activitats. Una activitat presenta una interfície gràfica (XML) que permet a l'usuari interactuar amb l'aplicació. Cada aplicació té varies activitats que es van mostrant a l'usuari segons les vagi necessitant.

Una activitat representa una pantalla, la qual interactua amb l'usuari. S'ha de tenir clar el concepte de pila, cada aplicació Android està corren amb el seu propi procés, el qual s'executa i s'inclou en una pila de processos. Cada vegada que l'aplicació ho necessiti, s'insereix una nova activitat a la pila i quan no es necessiti més es fa un push i es col·loca en primer lloc de la pila.



A continuació mostrem un petit exemple d'una activitat:

```
úblic com.prova.exemple;
import android.app.Activity;
import android.os.Bundle;
úblic class HolaMon extends Activity{
        úblic void onCreate(Bundle savedInstanceState){
                super.onCreate(saveInstanceState);
                setContentView(R.layout.exemple);
        }
        ...
}
```

El mètode *onCreate* és el que s'executa en el moment d'iniciar l'aplicació.
El mètode *setContentView*, definirà qual de les interfícies gràfiques creades al directori layout serà la utilitzada. Com a paràmetre s'utilitza el R.layout.exemple, que com hem comentat abans, és una classe generada automàticament per Eclipse cada vegada que creem un nou component per l'aplicació. Dins d'ella es creen classes estàtiques amb variables que representen cada un dels components.

Anem a definir els mètodes que té la classe Activity:

onCreate: Mètode per crear l'activitat per primer cop.

onStart (): Per que l'activitat se visualitzi.

onResume (): Per que l'activitat comenci a interactuar amb l'usuari.

onPause (): Quan un altre activitat s'invoca i l'actual ha de passar al fons. Així es guarda l'estat de l'activitat.

onStop (): Quan una activitat és invocada i l'actual ha d'està en segon pla durant un temps determinat per després utilitzar-se.

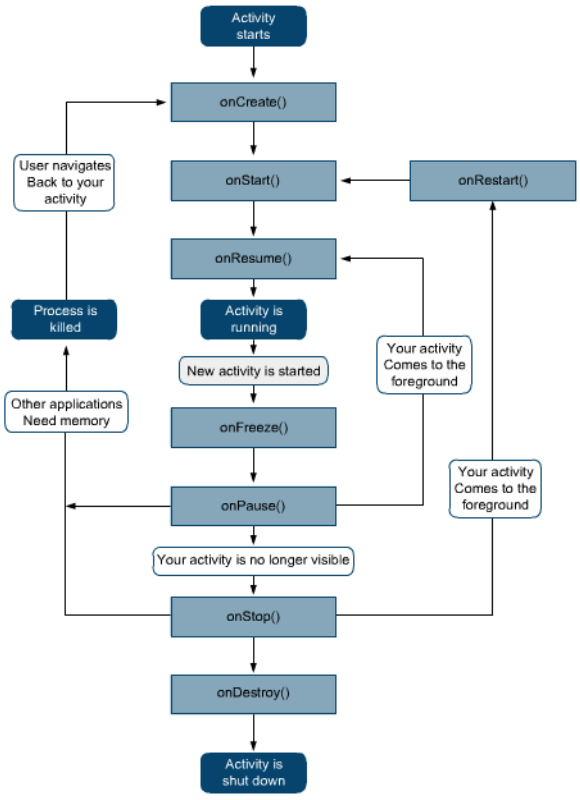
onRestart (): Quan una activitat estava en segon pla i es vol tornar a utilitzar.

onDestroy (): S'utilitza per destruir una activitat, degut a que ja no és necessària.

onSaveInstanceState (Bundle): S'utilitza per guardar l'estat.

onRestoreInstanceState (Bundle): S'utilitza per recuperar l'estat guardat pel mètode anterior definit.

I el cicle de vida d'una activitat és el següent:



AndroidManifest.xml
Totes les aplicacions hauran de tenir aquest fitxer i no es pot canviar de nom. En ell s'especifiquen les opcions generals del programa, com el paquet principal, l'activitat que s'haurà d'executar a l'iniciar l'aplicació, la definició de totes les activitats, els permisos,... Aquí tenim un petit exemple de la seva estructura:

```
<xml>
<manifest xmlns:android=”http://schemas.android.com/apk/res/android” ..>
<application android:icon=”@drawable/icon” android:label=”@string/app_name”>
 <activity android:name=”.HolaMon”
 android:label=”@string/app_name”>
 <intent-filter>
 <action android:name=”android.intent.action.EXEMPLE” />
 <category android:name=”android.intent.category.LAUNCHER” />
 </intent-filter>
 </activity>
</application>
 <uses-sdk android:minSdkVersion=”4” />
</manifest>
```

7.2 Disseny Lògic

A part de l'explicació que de l'apartat anterior, que ens servirà per plantejar l'aplicació, definirem les classes que necessitarà l'aplicació mòbil:

Alerta:
Serà la classe principal de l'aplicació. Que farà referencia a les notificacions que s'aniran rebent de Moodle.

Activities
Al desenvolupar una aplicació per Android, els fonaments del disseny lògic seran les activitats. En tindrem una per cada pantalla física de l'aplicació.
Podem destacar:

Activitat	Descripció
Login	Activitat que enllaçarà amb la vista on l'usuari es llogarà
Menú	Activat que s'enllarà amb el la vista que mostrarà els diferents grups dels quals es reben les notificacions. Caldrà seleccionar un per veure el detall.
Missatge	Activitat que mostrarà el llistat de missatges rebuts i emmagatzemats a la base de dades.
Fòrum	Activitat que mostrarà el llistat de missatges rebuts del fòrum i emmagatzemats a la base de dades.
Campus	Activitat que mostrarà el llistat de missatges rebuts del campus i emmagatzemats a la base de dades.
Xat	Activitat que mostrarà el llistat de missatges instantanis

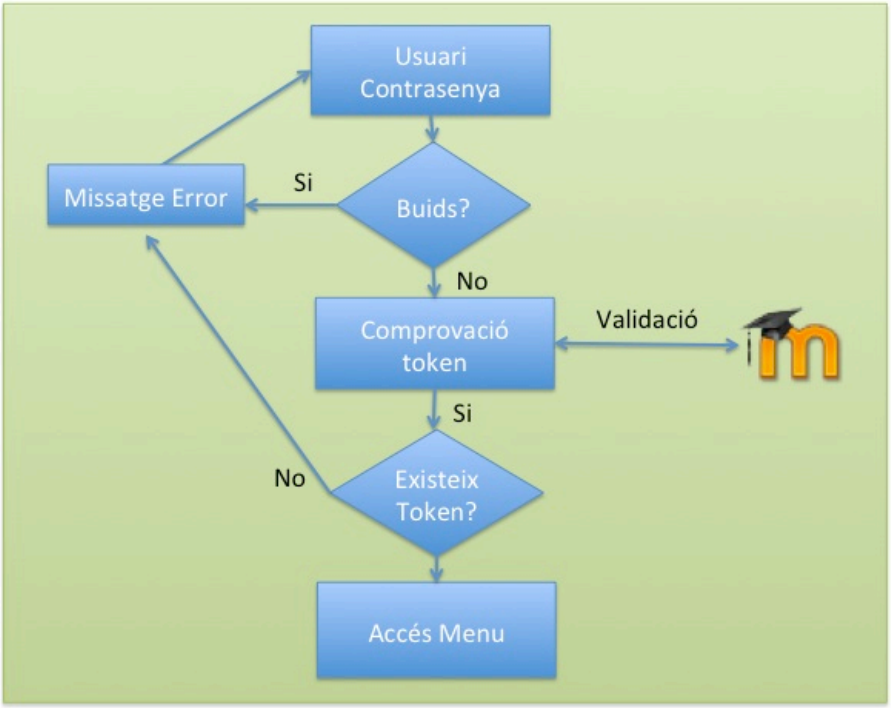
	rebuts.
Calendari	Activitat que mostrarà el llistat de missatges rebuts dels events del calendari.
Veure detall	Activitat que enllaça amb la vista que mostra el detall del missatge.
Crear alerta	Des de certes notificacions, l'usuari podrà respondre els missatges que ha rebut.

7.3 Disseny Funcional

A nivell funcional anem a veure com treballaran les diferents part de l'aplicació, podem distingir:

Login

En aquest diagrama de flux veurem quin és el recorregut que farà la validació de l'usuari contra el middleware de Moodle:

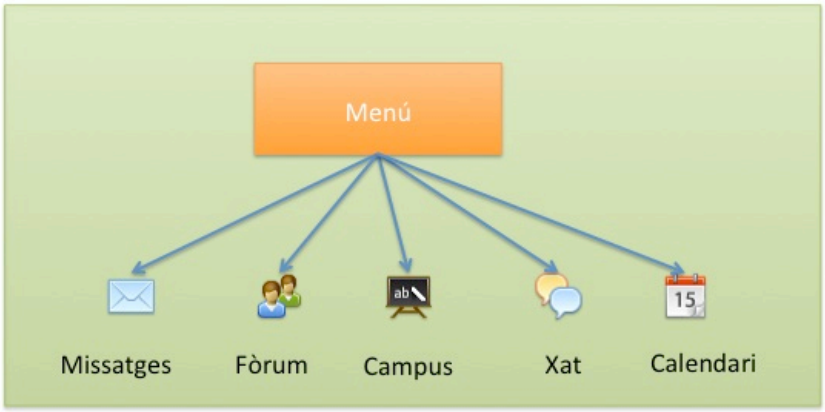


Existiran diferents validacions, que els camps tinguin valor, que l'usuari existeixi a Moodle i tingui un *token* associat. Si totes les validacions han anat correctament, a continuació es redirigirà cap a la següent *Activity* o pantalla, el menú.

Menú

Aquesta funcionalitat permetrà a l'usuari filtrar quin tipus de notificacions vol veure. Mitjançant un menú podrà triar l'opció:

El següent diagrama mostra les categories d'alertes que té actualment implementades l'aplicació:



Estaran identificades mitjançant una imatge i un nom perquè visualment siguin més accessibles.

Alertes

L'estructura de l'alerta presentarà un llistat:

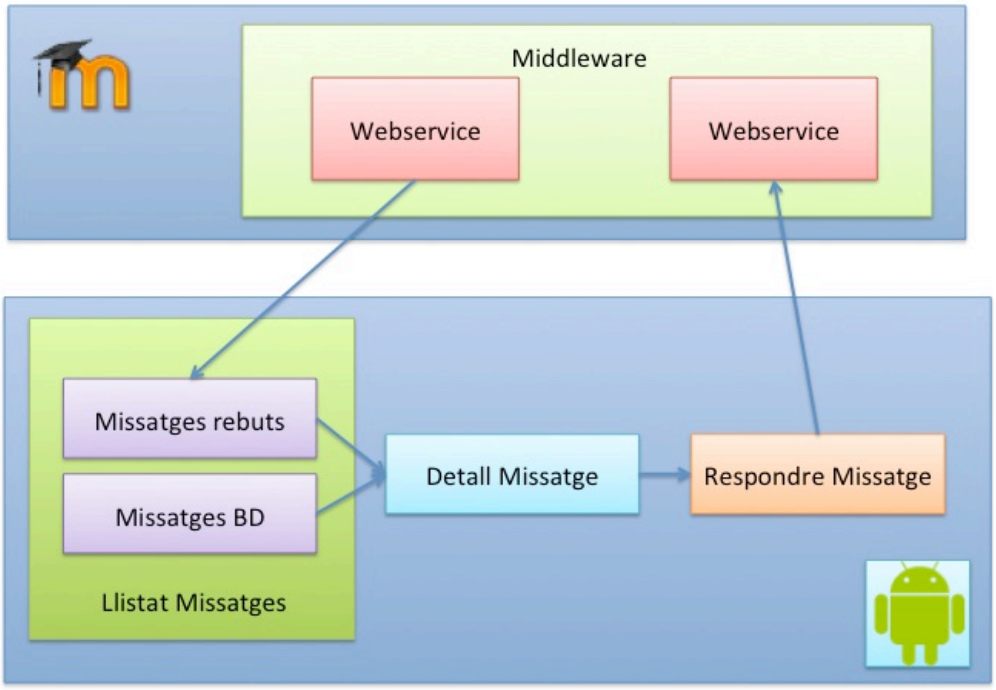
- Missatges rebuts: Alertes actuals rebudes de Moodle. A través del middleware es connectarà al webservice adequat i rebrà les alertes.

- Missatges BD: Alertes antigues emmagatzemades a la base de dades que es recuperaran a continuació de les rebudes. Aquí caldrà una connexió a la base de dades per rebre les alertes segons el tipus funcional (missatge, xat, fòrum, campus o calendari).

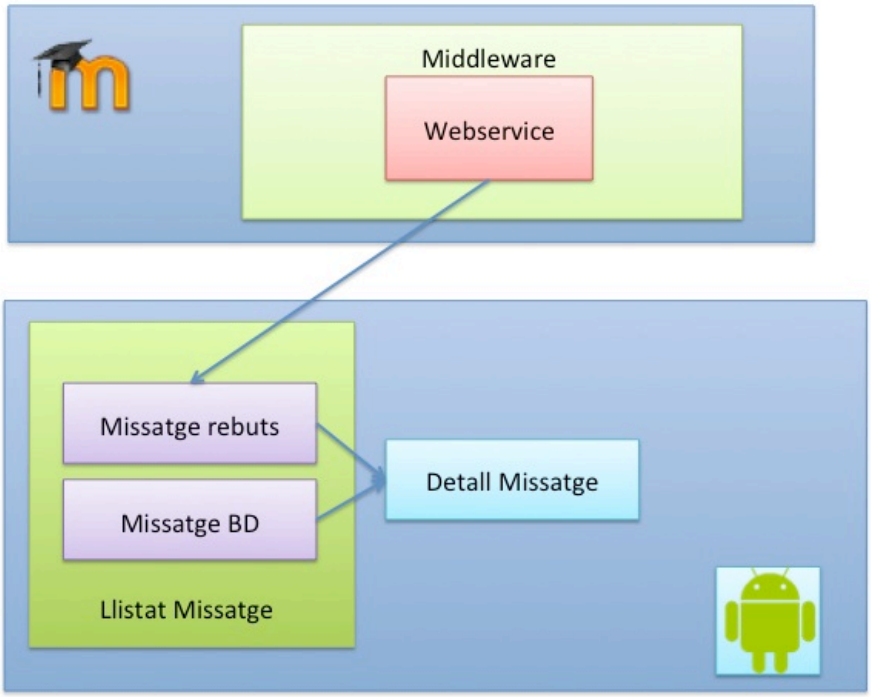
- Detall Missatge: el llistat d'alertes serà clicable i permetrà seleccionar-ne una per veure-la en detall.

- Respondre Missatge: algunes alertes, missatge, xat i fòrum, permetran retroaccions a les alertes. Existirà la possibilitat de respondre la notificació. Per fer això caldrà una nova connexió al middleware per enviar l'alerta al Moodle perquè la pugui processar com una notificació. Com es pot veure a la imatge, el webservice serà diferent al que s'ha consultat per rebre les alertes. Existirà un per cada acció diferent que es pugui realitzar cap a Moodle. D'aquesta forma per l'aplicació serà transparent i només caldrà saber a quin webservice cal fer la crida per interactuar i quins paràmetres d'entrada/sortida necessita.

La següent imatge mostra un esquema amb els dos blocs, la part de Moodle amb el middleware i la part de l'aplicació mòbil.



Aquest esquema representa les alertes que no tenen retroaccions, com és el cas del Calendari i el Campus. Una vegada vist el detall de la notificació no es podrà realitzar cap acció sobre elles.



7.4 Disseny de la Capa de persistència

Un altre punt a definir és el com emmagatzemar aquests missatges que l'usuari aniran rebent a la seva aplicació.

Android permet les opcions de:

- *Emmagatzemen intern*;

Utilitza fitxers guardats internament i són d'us privat per l'aplicació. Quan es desinstal·la, aquest fitxers són eliminats. El seu ús és similar al d'un fitxer

  - openFileOutput(): mètode per obrir el fitxer.
  - write(): mètode per escriure al fitxer.
  - read(): mètode per llegir del fitxer.
  - close(): mètode per tancar el fitxer.

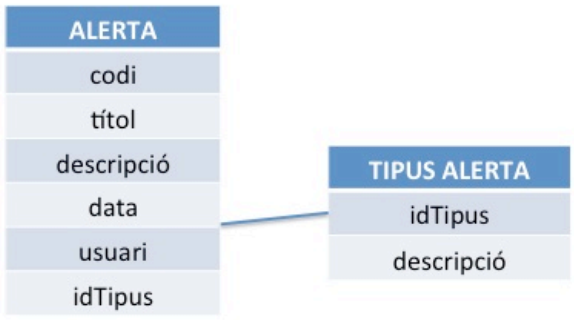
- *Emmagatzemen extern*:

Els dispositius Android, poden utilitzar recursos externs per guardar la informació. Com per exemple targetes SD.

- *Base de dades SQLite:*

És un motor de base de dades molt popular per oferir característiques interessants com el seu petit tamany, no necessita un servidor, té poca configuració, es transaccional i per suposat és de codi lliure.
Android incorpora de sèrie totes les eines necessàries per la creació i gestió de la base de dades SQLite i entre elles una completa API per dur a terme de forma senzilla totes les tasques.

Base de dades SQLite
De tots els tipus de disseny de la capa de persistència al final hem triat utilitzar la base de dades de SQLite. Així aprofundim en l'ús de Bases de dades per dispositius mòbils.
Anem a definir doncs el model entitat relació de l'aplicació, constarà de 2 taules:



Anem a especificar una mica més el diagrama:

Taules

- *Alerta*:

Guardarà totes les notificacions enviades des de Moodle. Els camps seran:

  - codi: Enter, clau primària de la taula que serà un identificador autoincremental. És la clau primària de la taula.
  - títol: Cadena, text que mostrarà una breu descripció per mostrar al llistat d'alertes.
  - descripció: Cadena, text més extens que mostrarà la notificació sencera.
  - data: Data, que mostrarà la data en que s'ha generat l'alerta.
  - usuari: Cadena, indicarà de qui és l'alerta.

- idTipus: Enter, indicarà quin tipus d'alerta és.

- *Tipus Alerta*

Guardarà els tipus d'alerta que hi haurà a l'aplicació mòbil. Inicialment els que hem definit són: Missatge, Fòrum, Campus, Xat i Calendari. Però aquests en un futur podríem ampliar-se. I d'aquesta forma els tindrem definits de forma dinàmica. Els camps seran:

- idTipus: Enter autoincremental, serà la clau primària i l'identificador del tipus. És la clau primària de la taula.
- descripció: Cadena, serà la descripció del tipus d'alerta que apareixerà als menús superiors de l'aplicació.

Relacions:

- *Missatge – Tipus de alerta*

La relació és de n Alertes pertanyen a 1 Tipus d'alerta.
Servirà per generar els llistats d'alertes segons el tipus d'alerta.

Preferències de l'usuari

Caldrà buscar també un altre tipus d'emmagatzament per poder guardar les preferències de l'usuari. Ja que l'aplicació contempla moltes vistes diferents i necessitarem guarda les credencials de l'usuari.

Android ofereix la possibilitat d'utilitzar les SharedPreferences que fan ús d'un espai intern per guardar preferències de l'usuari.

7.5 Tecnologia de comunicació client – servidor

En aquesta aplicació necessitarem la constant recepció de les alertes que es vagin produint dins del Portal. Per això haurem de buscar un mecanisme que ens faciliti la recepció instantània.

Existeixen dos mecanismes d'interacció entre client-servidor:

-Tecnologia PUSH (empènyer/enviar)
-Tecnologia PULL (estirar/atreure)

Els dos similars i lo que caracteritza a cada un d'ells es la petició de la transacció. En el cas del PULL s'origina al client i al PUSH l'encarregat de generar la petició és el servidor.

Els serveis *pull* interroguen periòdicament al servidor per saber si existeixen noves dades disponibles. Esta solució es la més fàcil d'implementar, però és la menys ineficient.

Els serveis *push* estan més orientats a models del tipus publicador/subscriptor. El client s'ha de subscriure a un o varis canals per accedir a la informació. D'aquesta forma quan hi ha un nou contingut el servidor envia la informació al

client. No cal que el client vagi preguntant cada vegada al servidor, sinó que el servidor notifica al dispositiu que disposa de les noves dades. Amb això el client haurà de deixar una connexió oberta perquè el servidor li notifiqui.

Exemples d'aplicacions que utilitzen la tecnologia push: sistemes missatgeria instantània o de xats online, el protocol SMTP dels correus electrònics, jocs, resultats d'esports

7.6 Interfícies de l'aplicació

Abans de començar a definir l'aspecte de l'aplicació mòbil, crec que és important tenir present que és una aplicació per un aparell amb una pantalla petita. Això significa que l'aplicació haurà de ser molt intuïtiva i accessible. Ja que l'espai per presentar la informació no serà molt extensa. A més molts dispositius tenen pantalles tàctils, per lo tant quant més senzilla, més fàcil serà per l'usuari navegar dins els menús i submenús.

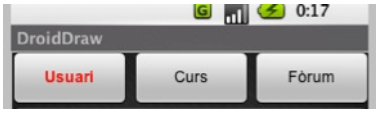
Anem a definir el conjunt de les pantalles que haurem de dissenyar per dur a terme tot el procés. Les imatges que es presenten són un prototipus de les futures pantalles, per dissenyar com hauran de ser i quina informació hauran de presentar. Més endavant farem el disseny final, una mica més elaborat que aquestes pantalles que són informatives:

- *Aplicació per a mòbil*

La idea és fer una aplicació senzilla que mostri un històric del recull de tots els missatges activats pel usuari.

Login:
La primera pantalla serà la que servirà a l'usuari per logar-se. Introduint un usuari i una contrasenya l'aplicació es connectarà al webservice per validar les credencials i començar a rebre les notificacions que tingui configurades l'usuari:

Veure Grups
Mitjançant blocs per classificar aquests missatges obtindrem un menú on tindrem les diferents funcionalitats:



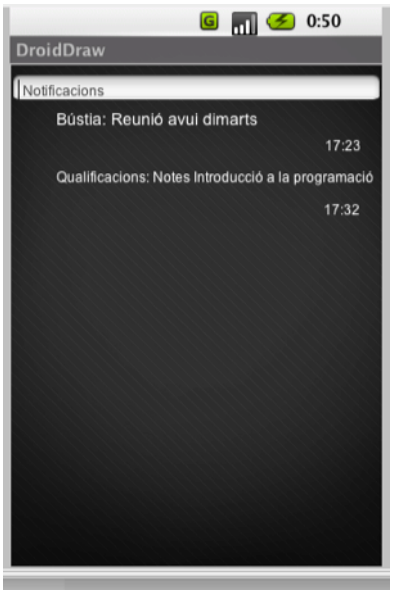
Llistat d'alertes:
Un llistat amb una breu descripció que una vegada clicat es podrà veure tot el contingut del missatge.:

Un llistat de tots els missatges referents a cada un dels blocs. Com podem veure serà una breu descripció. Si l'usuari vol llegir tot el missatge, caldrà clicar-los i apareixerà el detall del missatge

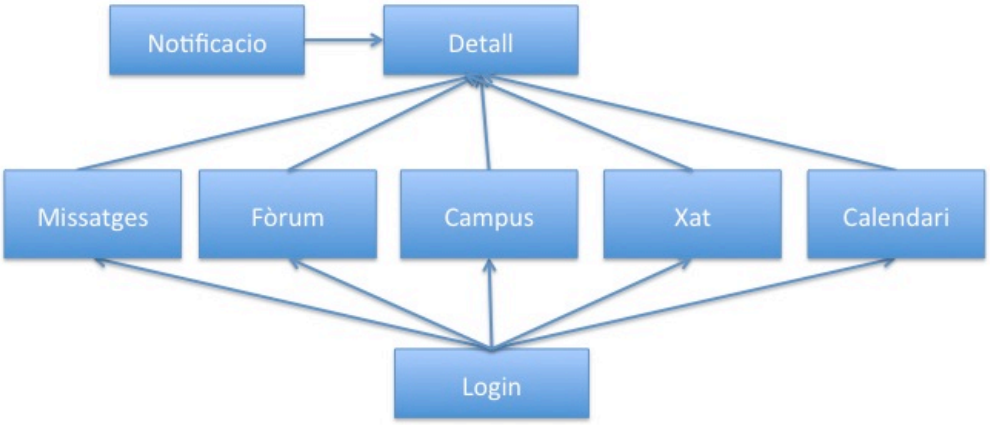
Veure detall missatge:
La informació mostrada és: el títol, la data i la descripció de la notificació.
Com es mostra a la següent imatge:

- *Notificacions al dispositiu mòbil.*

Els missatges que l'usuari rebrà s'hauran de poder activar com a notificacions al mòbil perquè cada vegada que es rebi el missatge aparegui a Notificacions per tenir un enllaç més directe i així l'usuari estigui assabentat a l'instant. Aquest apartat hauria de poder ser configurable perquè l'usuari trií quines són les alertes més importants per ell. La imatge que es mostra presenta com quedaria:



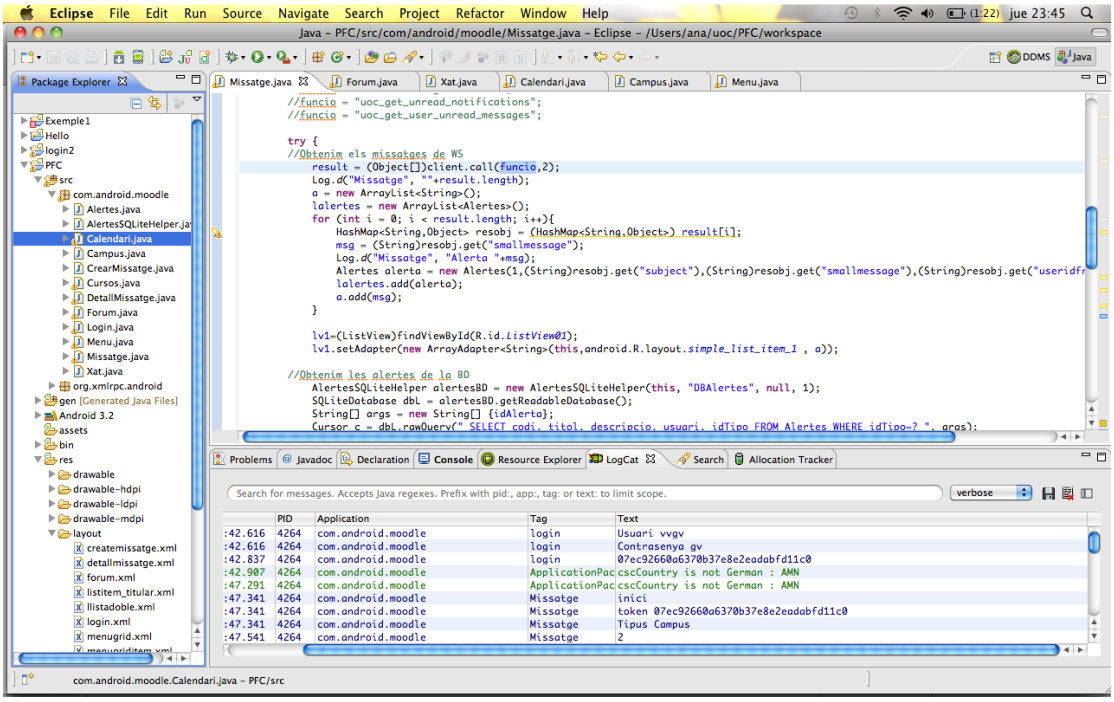
En resum la Navegació entre les diferents interfícies serà tal com mostra el següent diagrama:



8. Implementació

8.1 Estructura del projecte

Per desenvolupar aquest projecte hem utilitzat l'entorn de desenvolupament o IDE de l'eclipse. Que permet, una vegada instal·lada l'SDK d'Android, generar projectes d'aquest tipus. Al crear un nou projecte, li indiquem que serà d'Android i ell automàticament ens crear la jerarquia de carpetes pròpia del projecte. Això facilita molt la feina perquè ja tenim tota l'estructura generada automàticament.



Utilitzar un IDE per desenvolupar permet fer-ho molt més fàcilment. Ja que presenta una imatge més visual del projecte, una connexió entre tots els objectes, un control més ràpid dels errors, missatges que t'ajuden a incloure llibreries, definir correctament parts de codi.

La classe *R.java*, la qual no hem de modificar. Amb l'Eclipse, automàticament, cada vegada que creem un layout donarà d'alta dins d'aquest fitxer tots els identificadors dels diferents controls. Lo que suposa una gran ajuda per desenvolupar.

Per desenvolupar les vistes, pantalles o layout també ofereix una eina *Graphical layout*, que ajuda a visualitzar com quedarà la pantalla. O fins i tot arrossegar diferents controladors per anar configurant la pàgina.

L'SDK d'Eclipse incorpora una eina per poder debugar, el LogCat. Més endavant explicarem com funciona. Així hem pogut anar controlant els errors i les traces durant el desenvolupament de l'aplicació.

L'*emulador* que hem utilitzat per executar l'aplicació ha estat de dos tipus:

- Emulador dins de l'eclipse: un emulador que s'intal·la quan utilitzes l'SDK d'Android.
- Connexió d'un mòbil Android (HTC desire): mitjançant un cable USB hem connectat el mòbil a l'ordinador. I l'Eclipse permet definir com a emulador un mòbil extern. En aquest punt hem guanyat en velocitat, ja que el temps de càrrega de l'aplicació és molt més petit que l'emulador, i hem pogut veure de forma real com l'aplicació anava agafant forma en el dispositiu que finalment l'utilitzarà.

- **AndroidManifest**

És el fitxer de configuració xml del projecte, on es definiran diferents parts com:

- L'activitat principal que s'executa a l'iniciar l'aplicació
- La connexió a Internet
- Es dona d'alta cada una de les activitats de l'aplicació
- Es defineix la icona que tindrà l'aplicació

8.2 Capa de negoci

- **Estructura d'objectes**

Els objectes que hem hagut d'implementar en aquesta aplicació són:

- **Activities**:

Las activitats representen el component principal de la interfície d'una aplicació en Android. Es pot pensar en que hi hauran tantes activitats com pantalles a l'aplicació.
Aquestes classes ens permetran interactuar amb les vistes, recollint paràmetres, enviant dades, realitzant accions o construint llistes.

- **Objecte Alertes**

A l'aplicació es gestionaran diferents blocs d'alertes o notificacions. Però totes tindran una estructura similar. Hem creat una classe Alertes per poder gestionar aquest tipus d'objecte.
Els atributs que tindrà seran:

- Codi: identificador de l'alerta
- Títol: breu descripció
- Descripció: descripció sencera
- Usuari: qui ha enviat l'alerta
- idUsuari: identificador de qui ha enviat l'alerta
- data: data de quan s'ha enviat la notificació
- idTipo: identificador pel tipus d'alerta.

L'estructura de l'objecte serà la següent

```
public class Alertes {
    private int codi;
    private String titol;
    private String descripcio;
    private String usuari;
    private Date data;
    private int idTipo;
    private int idUsuari;

    public Alertes(int codi, String titol, String descripcio, String
usuari, Date data, int idTipo){
        this.codi = codi;
        this.titol = titol;
        this.descripcio = descripcio;
        this.usuari = usuari;
        this.data = data;
        this.idTipo = idTipo;
    }

    public Alertes(int codi, String titol, String descripcio, String
usuari,int idTipo){
```

```
        this.codi = codi;
        this.titol = titol;
        this.descripcio = descripcio;
        this.usuari = usuari;
        this.idTipo = idTipo;
    }
...
```

- **AlertesSQLiteHelper**

Aquesta classe serà l'encarregada de gestionar les accions contra la base de dades. Estendrà de la classe *SQLiteHelper*. Més endavant, a l'apartat de la capa de persistència explicarem la seva estructura i com funciona.

- **Accés a webservice**

La gran majoria de les grans webs usen aplicacions que utilitzen serveis webs (web service). Un webservice és un conjunt de protocols i estàndards que serveixen per intercanviar dades entre aplicacions. D'aquesta forma distintes aplicacions desenvolupades amb llenguatges de programacio diferents i executades sobre qualsevol plataforma, poden accedir als serveis per intercanviar dades entre xarxes d'ordinadors com Internet.
Existeixen al mercat diferents estardards, com SOAP (Simple Object Acces Protocol), REST o XML-RPC (XML Remote Procedure Call).

SOAP: defineix com dos objectes en diferents processos poden comunicar-se per mitja d'intercanvi de dades XML (missatges)

XML-RPC

Per desenvolupar l'accés al webservice, en el nostre cas hem utilitzat el mètode RPC. Hem descarregat la llibreria adequada per poder utilitzar les classes i així aconseguir la connexió cap als webservices de Moodle

La part interessant d'aquest projecte, és la connexió de l'aplicació mòbil als diferents webservices de Moodle, per així poder obtenir la informació necessària per mostrar a les diferents pantalles. Per cada una de les opcions, necessitarem connectar-nos a un servei diferent, ja que els valors de les dades que rebrem seran diferents.

L'exemple de codi de l'aplicació:

```
String serverurl = host + "/webservice/xmlrpc/server.php" + "?wstoken=" +
token;
uri = URI.create(serverurl);
XMLRPCClient client = new XMLRPCClient(uri);
//Funcio del WS
funcio = "uoc_get_user_unread_messages";
try {
       //Obtenim els missatges de WS
       result = (Object[])client.call(funcio,2);
       a = new ArrayList<String>();
       lalertes = new ArrayList<Alertes>();
       for (int i = 0; i < result.length; i++){
```

```
            HashMap<String,Object> resobj = (HashMap<String,Object>)
result[i];
       msg = (String)resobj.get("smallmessage");
       Alertes alerta = new
Alertes(1,(String)resobj.get("subject"),(String)resobj.get("smallmessage"),(
String)resobj.get("useridfrom").toString(),1);
       lalertes.add(alerta);
       a.add(msg);
}
lv1=(ListView)findViewById(R.id.ListView01);
lv1.setAdapter(new
ArrayAdapter<String>(this,android.R.layout.simple_list_item_1 , a));
```

On podem veure:
Definim una *url* on invocarem al servei, passant com a paràmetre el *token* amb la informació de l'usuari.

```
String serverurl = host + "/webservice/xmlrpc/server.php" + "?wstoken=" +
token;
```

Instanciem un objecte XMLRCPClient:

```
uri = URI.create(serverurl);
XMLRPCClient client = new XMLRPCClient(uri);
```

Fem la crida al servei del webservice que ens retornarà la informació necessària:

```
funcio = "uoc_get_user_unread_messages";
result = (Object[])client.call(funcio,0);
```

El mètode *call*, rebrà com a paràmetres la funció o servei que cridem i els paràmetres d'entrada en cas que els tingués.

A la variable *result*, tindrem l'estructura que retornarà el webservice. En aquest cas serà una estructura doble, però dependrà de cada un dels serveis.

- **Accés al token mitjançant JSON**

JSON, JavaScript Object Notation, és un format lleuger per l'intercanvi de dades. Per la seva lleugeresa s'ha convertit en un estàndard alternatiu al XML.

Per accedir als webservices de Moodle, no ho farem amb l'usuari i la contrasenya. Necessitarem un objecte anomenat token que servirà per validar-no en la crida als serveis dels diferents webservice.
Per això el login de l'aplicació consistirà en aconseguir el token donats un usuari i una contrasenya. Per fer això haurem de fer una crida a un http que ens retornarà el token.
Per fer aquesta part hem triat la tecnologia JSON, que ens permet connectar-nos via http i rebre un paràmetre:

L'exemple de com hem utilitzat la llibreria és:

```
DefaultHttpClient httpclient = new DefaultHttpClient();
servei = "uoc_ws";//uoc_ws - moodle_ws
HttpPost httpost = new
HttpPost(host+"/login/token.php?username="+us.getText().toString()+"&passwor
d="+pw.getText().toString()+"&service="+servei)
```

```
try {
       HttpResponse response = httpclient.execute(httpost);
       BufferedReader reader = new BufferedReader(new
InputStreamReader(response.getEntity().getContent(), "UTF-8"));
       String json = reader.readLine();
       JSONTokener tokener = new JSONTokener(json);
       JSONObject finalResult = new JSONObject(tokener);
   String token = finalResult.get("token").toString());
...
```

- **Control de Logs**

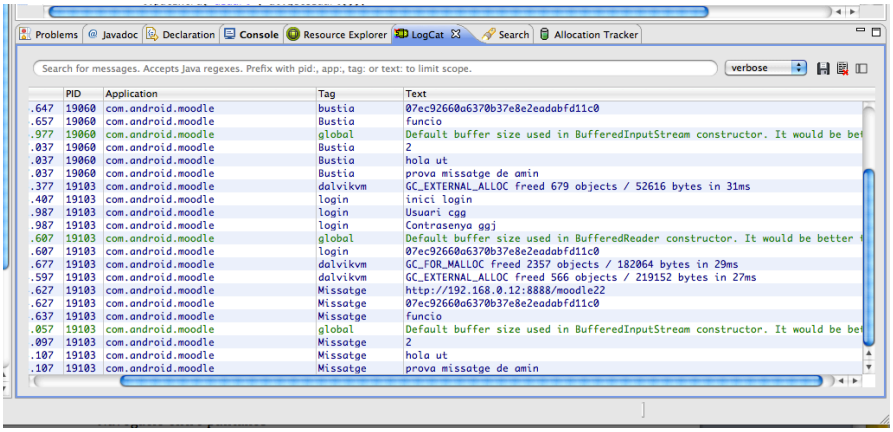
Existeix una llibreria pròpia d'Android, andorid.util.Log que ens permet poder afegir xivatos per poder debugar millor el codi.
Els missatges porten associada la següent informació:

- Data/Hora del missatge.
- Criticitat, nivell de gravegtat del missatge. On podem distingir els diferents tipus:
  - Error – e()
  - Warning – w()
  - Info – i()
  - Debug – d()
  - Verbose – v()
- PID, codi intern del procés que ha introduït el missatge.
- Tag, etiqueta identificativa del missatge
- Missatge, el text del missatge

Cada un d'aquests mètodes reb com a paràmetres l'etiqueta (tag) i el text del missatge. L'etiqueta sol ser el nom de l'aplicació o activitat que genera el missatge.
A l'aplicació hem anat utilitzant aquest component per debugar el codi, veure que les variables retornaves valors correctes o descobrir en quin punt el programa donava un error. El codi que hem utilitzat ha estat de la següent forma:

```
Log.d("login","inici del login");
```

On a la finestra de l'Eclipse del LogCat podem veure la informació recollida durant l'execució de l'aplicació:



- **Missatges del tipus Toast**

Per notificar missatges a l'usuari, Android té diferents formes de fer-ho. Una d'elles és mitjançant les notificacions anomenades Toast.
És un missatge que es mostra a la pantalla durant uns segons a l'usuari per després tornar a desaparèixer automàticament sense requerir cap tipus d'actuació per part de l'usuari i sense rebre el focus. Encara que es poden personalitzar, apareixen a la part inferior de la pantalla sobre un requadre negre. Aquests missatges són útils per mostrar informació ràpida sense requerir intervenció de l'usuari.
En el nostre cas els hem utilitzat a la validació del login. En el cas de que els camps estiguin buids, apareixerà un missatge avisant. I en el cas de que l'usuari no existeixi, també es llençarà.
La seva utilització és molt senzilla. Disposa d'un mètode estàtic *makeText()*, al que li haurem de passar com a paràmetre el context de l'activitat, el text a mostrar i la durada del missatge (LENGTH_LONG o LENGTH_SHORT). A continuació caldrà cridar al mètode *show()* per mostrar-ho per pantalla.

Un exemple de la utilització:

```
Toast toast = Toast.makeText(getApplicationContext(), "Els camps estan
buids", Toast.LENGTH_SHORT);
toast.show();
```

8.3 Capa de persistència

En aquest apartat explicarem els mètodes que hem utilitzat per emmagatzemar informació a l'aplicació:

- **SharedPreferences**

Android ens permet utilitzar una eina, les preferències compartides, per emmagatzemar dades en una aplicació sobre l'usuari. Cada preferència es guarda en forma de clau-valor, un identificador i un valor associat a l'identificador. Les dades s'emmagatzemen en un fitxer XML.

La utilització d'aquestes preferències és molt senzilla. Utilitzarem l'objecte SharedPreference disponible a l'API d'Android. I mitjançant un identificador únic diferenciarem les de cada aplicació.
Per obtenir una referència a una col·lecció utiltizarem el mètode getSharedPreferences() al que li passarem l'identificador de la col·lecció i un

mètode d'accés que indicarà quines aplicacions tenen accés i quines operacions estaran permeses.
Existeixen 3 tipus:

MODE_PRIVATE: Només la nostra aplicació té accés.
MODE_WORLD_READABLE: Totes les aplicacions poden llegir les preferències, però només la nostra aplicació les poden modificar.
MODE_WORLD_WRITABLE: Totes les aplicacions poden llegir i modificar aquestes preferències.

En la nostra aplicació hem utilitzat aquesta opció per guardar l'usuari, la contrasenya i el token. I així poder consultar-ho des de altres pantalles.

Per invocar-ho farem:

```
private SharedPreferences prefs;
prefs = getSharedPreferences("PreferenciesMoodle",Context.MODE_PRIVATE);
```

Per modificar les dades emmagatzemades:

```
private SharedPreferences.Editor editor;
editor = prefs.edit();
editor.putString("host", host);
editor.putString("logged", "logged");
```

Per consultar les dades:

```
prefs = getSharedPreferences("PreferenciesMoodle",Context.MODE_PRIVATE);
host = prefs.getString("host", "");
token = prefs.getString("token", "");
```

- **Base de dades SQLite**

SQLite és un motorde base de dades molt popular en l'actualitat per oferir característiques com: petit tamany, no es necessari un servidor, poca configuració, transaccional i de codi lliure.
Android incorpora una API per gestionar-la d'una forma senzilla. Anem a veure les diferents accions que hem realitzat a l'aplicació:
Crearem una classe java que estendrà de la classe *SQLiteOpenHelper*, la qual tindrà un constructor i dos mètodes onCreate() i onUpgrade(), que serviran per crear la base de dades i actualitzar l'estructura.

En el nostre cas hem creat la classe AlertesSQLiteOpenHelper:

```
public class AlertesSQLiteHelper extends SQLiteOpenHelper{

//Sentencia SQL per crear la taula Alertes
String sqlCreate = "CREATE TABLE Alertes (codi INTEGER, titol TEXT,
descripcio TEXT, usuari TEXT, idTipo INTEGER)";

public AlertesSQLiteHelper(Context contexto, String nombre,
                           CursorFactory factory, int version) {
    super(contexto, nombre, factory, version);
}
```

```
@Override
public void onCreate(SQLiteDatabase db) {
     //Executa la sentencia de creació de la taula
     db.execSQL(sqlCreate);
}

@Override
public void onUpgrade(SQLiteDatabase db, int versionAnterior, int
versionNueva) {
     //S'elimina la versió anterior de la taula
     db.execSQL("DROP TABLE IF EXISTS Usuarios");
     //Es crea la nova versió de la tabla
     db.execSQL(sqlCreate);
  }
}
```

On crearem una taula a la base de dades Alertes, amb l'estructura per emmagatzemar les diferents alertes que vagi reben l'aplicació.

Per interactuar amb la taula que hem creat podrem fer:

- **Inserir** nous registres

Una vegada definida la classe helper anterior, la primera acció serà obrir la base de dades. Crearem un objecte de la classe AlertesSQLiteHelper al que passarem:

- El context de l'aplicació
- El nom de la base de dades
- Un objecte CursorFactory que no sempre serà necessari (en aquest cas li passarem un null)
- La versió de la base de dades que necessitarem.

Al crear aquest objectes poden passar diferents efectes:

- Si la BD ja existeix i la versió coincideix , s'obrirà una connexió cap a ella.
- Si la BD de dades existeix, però la seva versió és anterior a la sol·licitada, cridarà al mètode *onUpgrade()* per convertir la BD a la nova versió i es connectarà a la BD actualitzada.
- Si la BD no existeix, cridarà al mètode *onCreate()* per crear-la i es connectarà amb la BD creada.

Quan tinguem referència al objecte, podrem utilitzar dos mètodes:

- getReadableDatabase() en el cas de fer consultes.
- getWritableDatabase() en el cas de fer modificacions.

Amb el mètode *execSQL()* executarem la query per inserir dades a la BD.
El mètode *close()* tancarà la connexió a la BD una vegada acabades les operacions.

El resultat del nostre codi serà així, on podem veure tots els passos explicats anteriorment:

```
//Si no està creada la BD es crea i sinó s'aconsegueix la connexió
AlertesSQLiteHelper alertesBD = new AlertesSQLiteHelper(this, "DBAlertes",
null, 1);
```

```
//Inserir a la BD
SQLiteDatabase dbE = alertesBD.getWritableDatabase();

if(dbE != null)
{
  for(int i=1; i<=5; i++)
  {
      //Generem unes dades
      int codi = i;
      String titol = "Titol" + i;
      String descripcio = "Descripcio" + i;
      String usuari = "Usuari" + i;
      int idTipo = 1;

      //Inserim les dades la taula alertes
      dbE.execSQL("INSERT INTO Alertes (codi, titol, descripcio, usuari,
idTipo) " +
                           "VALUES (" + codi + ", '" + titol +"', '" +
descripcio +"', '" + usuari +"', " + idTipo +")");
  }
  //Tanquem la BD
  dbE.close();
}
```

- **Recuperar** els registres emmagatzemats

En aquest apartat explicarem com hem recuperat les dades de la base de dades. Com hem fet anteriorment, necessitarem establir una connexió amb la base de dades instanciant l'objecte *AlertesSQLiteHelper*.

Per recuperar les dades utilitzarem el mètode *rawQuery()* de la classe *SLQiteDataBase*. Que rebrà com a paràmetres la sentència SQL completa on indicarem els camps a recuperar i els criteris de selecció.
El resultat l'obtindrem en forma de *cursor*, que posteriorment podrem recorre per obtenir les dades del resultat de la consulta.

L'exemple del codi que hem implementat és el següent:

```
...

//Si no està creada la BD es crea i sinó s'aconsegueix la connexió
AlertesSQLiteHelper alertesBD = new AlertesSQLiteHelper(this, "DBAlertes",
null, 1);

SQLiteDatabase dbL = alertesBD.getReadableDatabase();
String[] args = new String[] {"1"};
Cursor c = dbL.rawQuery(" SELECT codi, titol, descripcio, usuari, idTipo
FROM Alertes WHERE idTipo=? ", args);

ArrayList al = new ArrayList();
//Nos aseguramos de que existe al menos un registro
if (c.moveToFirst()) {
//Recorremos el cursor hasta que no haya más registros
do {
   String codi = c.getString(0);
   String titol = c.getString(1);
```

```
   Log.d("forum", titol);
   al.add(titol);
   } while(c.moveToNext());
}
lv1.setAdapter(new
ArrayAdapter<String>(this,android.R.layout.simple_list_item_1 , al));

...
```

Com hem comentat, el resultat es retornarà en un objecte *Cursor*, que haurem de recorre per obtenir els resultats, existeixen dos mètodes:

- *moveToFirst()* mou el punter del cursor al primer registre retornat
- *moveToNext()* mou el punter del cursos al següent registre retornat

Amb la interacció d'aquests dos mètodes farem un bucle que ens permeti recorre el resultat.
I per obtenir el valor concret de cada registre, utilitzarem el mètode *getString*(i), on i és la posició del resultat. Si la nostre query retorna codi, títol, descripció, usuari, idTipo. Llavors

- Codi = *getString*(0)
- Títol = *getString*(1)
- Descripció = *getString*(2)
- ...

8.3 Capa de presentació

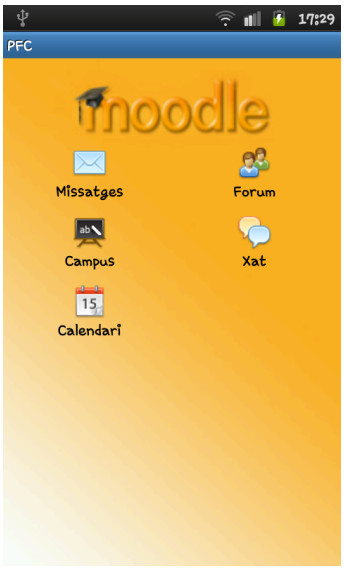
En aquest apartat explicarem tota la part relacionada amb la part visual de l'aplicació:

- Icona de l'aplicació

Hem definit una icona, perquè aparegui com a imatge dins del menú

- Estètica de l'aplicació

Hem definit una estètica diferent a la que plantejen les aplicacions en Android per defecte. Hem fons de color del mateix estil que el logo de Moodle per crear el el fons de totes les pantalles de l'aplicació. Així li dona una estètica diferent a total l'aplicació.



- Objectes de la vista

En aquest apartat definirem aquells objectes que hem utilitzat que són propis de la presentació de les aplicacions Android. Per crear les diferents vistes haurem d'utilitzar els següents objectes:

Els **layouts** són elements no visuals destinats a controlar la distribució, posició i dimensions dels controls que s'insereixen en el seu interior. Aquests estenen de la classe *ViewGroup*. Existeixen diferents tipus:

- FrameLayout: col·loca tots els controls fills alineats a la cantonada esquerra superior. Així cada control queda amagat pel següent control.
- LinearLayout: apila tots els elements de forma horitzontal o vertical segons la seva prioritat.
- TableLayout: permet distribuir als seus fills de forma tabular, definint les fileres i les columnes necessàries i la posició dins de la taula.
- RelativeLayout: permet especificar la posició de cada element de forma relativa al seu element pare o a qualsevol element dins del layout.

Un exemple a l'aplicació de la utilització del RelativeLayout:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@drawable/fondomain"
android:layout_height="fill_parent" android:layout_width="fill_parent">
...
```

El control **TextView** permet crear etiquetes que seran estàtiques. Encara que poden ser modificables des de les Activities . Aquí tenim un exemple:

```
     <TextView android:layout_width="wrap_content" android:textSize="10pt"
android:textColor="#FFFFFF" android:text="Contrasenya:"
```

```
android:id="@+id/tv_pw"
        android:layout_height="wrap_content"
android:layout_centerVertical="true"
android:layout_alignLeft="@+id/tv_un"></TextView>
```

I un exemple de la modificació d'un *TextView* desde la capa de negoci, invocant al mètode *setText()*:

```
TextView tv = (TextView)MyView.findViewById(R.id.grid_item_text);
tv.setText("Missatges");
```

El control **Button** serveix per afegir botons que permetran realitzar accions quan es faci un click sobre ells:

```
    <Button android:layout_width="100dip" android:id="@+id/btn_login"
android:layout_height="wrap_content" android:text="Login"
android:layout_below="@+id/tv_pw"
        android:layout_centerHorizontal="true"
android:layout_marginTop="26dp"></Button>
```

Amb el mètode *setOnClickListener()* activarem l'acció que proca el click del botó com per exemple la navegació entre les diferents vistes. Això es definirà a la capa de negoci amb un exemple del següent tipus:

```
Button ok = (Button)findViewById(R.id.btn_login);
ok.setOnClickListener(new View.OnClickListener() {
      public void onClick(View v) {
            //Acció a realitzar quan es faci el click
      ...
      }
}
```

El control **ImageView** permet crear espais amb imatges:

```
<ImageView android:id="@+id/grid_item_image"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center_horizontal">
</ImageView>
```

Que poden ser modificats dinàmicament amb el mètode *setImageResource()* des de la capa de negoci:

```
ImageView iv = (ImageView)MyView.findViewById(R.id.grid_item_image);
iv.setImageResource(R.drawable.email);
```

El control **EditText** permet crear espais on l'usuari introduirà text:

```
    <EditText android:id="@+id/et_pw" android:layout_height="wrap_content"
android:inputType="textPassword" android:layout_marginTop="10dp"
        android:layout_alignLeft="@+id/et_un"
android:layout_below="@+id/et_un" android:layout_alignRight="@+id/et_un"
```

```
android:layout_width="fill_parent"
          android:lines="1"></EditText>
```

A la capa de negoci el recollirem amb la següent funció:

```
EditText us = (EditText)findViewById(R.id.et_un);
String usuari = us.getText().toString());
```

El control **ListView** mostra a l'usuari una llista d'opcions seleccionables sobre el propi control. En cas de que hagin més dades de les que caben a la pantalla, permet fer scroll entre el llistat d'items.

```
<ListView android:id="@+id/ListView01" android:layout_width="wrap_content"
android:layout_above="@+id/bt1" android:layout_height="wrap_content" />
```

Aquestes llistes es construiran de forma dinàmica. Des de la capa de negoci, construirem una estructura que li passarem a aquest control.

Com podem veure el codi que tenim a continuació:

```
ArrayList a = new ArrayList<String>();
for (int i = 0; i < result.length; i++){
       HashMap<String,Object> resobj = (HashMap<String,Object>) result[i];
       msg = (String)resobj.get("smallmessage");
       a.add(msg);
}
ListView lv1=(ListView)findViewById(R.id.ListView01);
lv1.setAdapter(new
ArrayAdapter<String>(this,android.R.layout.simple_list_item_1 , a));
```

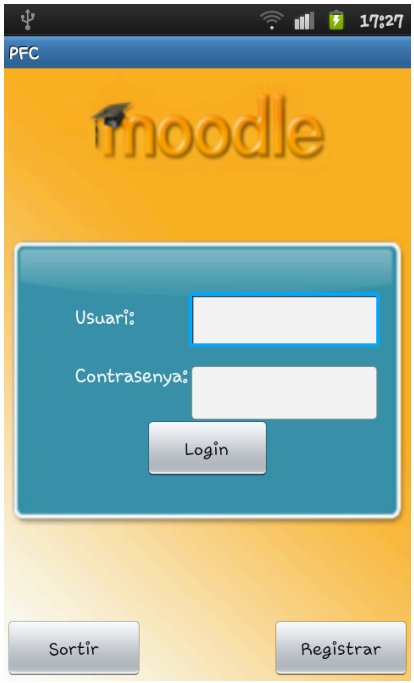
A la part de l'*Activity* construirem una estructura de dades (*ArrayList*) a, invocarem el control amb el mètode *findViewById()* i amb el mètode *setAdapter()* li passarem l'estructura a la *ListView* perquè mostri els valors. D'aquesta forma podem comprovar que aquest control serà completament dinàmic ja que dependrà de la quantitat de valors que li passem segons l'estructura.

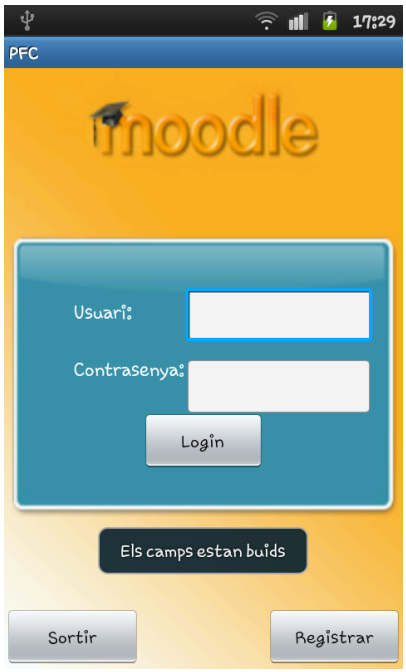
8.4 Funcionalitats detallades

En aquest apartat explicarem detalladament cada una de les funcionalitats de l'aplicació.

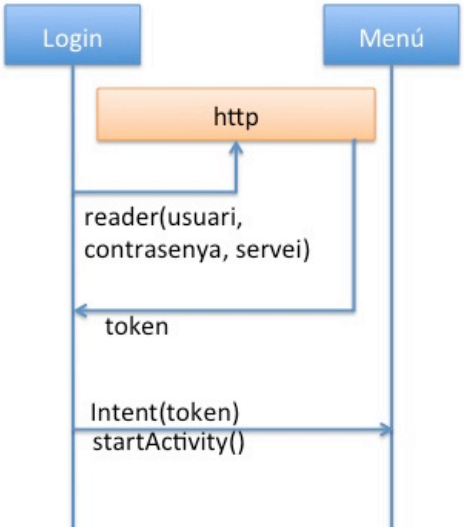
8.4.1 Login

És la pantalla d'inici de l'aplicació. On l'usuari introduirà el seu usuari i contrasenya i s'aconseguirà el token per poder utilitzar els serveis del webservice.





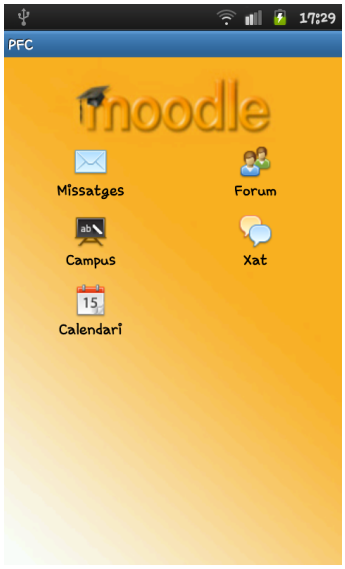
El diagrama de seqüència serà el següent:



On ens connectarem a Moodle via http utilitzant la llibreria JSON per poder obtenir el token que ens servirà per interactuar en un futur amb els diferents webservices.

8.4.2 Menú

Una vegada logats a l'aplicació, accedirem al menú on tenim totes les opcions del tipus d'alerta. Caldrà clicar sobre una de les imatges i accedirem al llistat d'alertes d'aquell tipus.



Com la visualització de les alertes serà dinàmica. Ja que totes funcionen amb la mateixa estructura, caldrà enviar uns paràmetres:

- Text: Text que identificarà el tipus d'alerta (Missatge, Xat, Fòrum)
- Funció: Indicarà quin és el webservice que ens retornarà les notificacions que necessitem per aquell tipus d'alerta.

La següent taula relaciona cada una de les alertes amb el webservice:

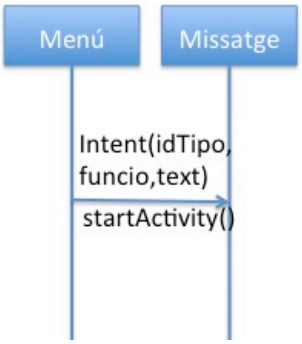
Alerta	Webservice entrada	Webservice retroacció
Missatge	uoc_get_user_unread_messages	uoc_send_messages
Fòrum	uoc_get_unread_foro	uoc_send_unread_foro
Campus	uoc_get_unread_notifications	
Xat	get_instant_messages	Send_instant_messages
Calendari	uoc_get_user_calendar_events	

- IdTipo: Paràmetre que ens filtrarà la consulta a la base de dades, perquè només ens retorni aquelles que són del tipus que volem. Aquest paràmetre ens el podríem estalviar perquè és equivalent al Text. Però per estalviar accessos a la base de dades el passarem com a paràmetre.

La relació serà del tipus:

Alerta	**idTipo**
Missatge	1
Fòrum	2
Campus	3
Xat	4
Calendari	5

El diagrama de seqüencia que tenim acontinanuació detella com serà la interacció entre les dues Activitys. En aquest cas serà molt senzill degut que la funcionalitat de Menú només ha de permetre a l'usuari triar categoria d'alerta vol visualitzar:



8.4.3 Missatge

Una particularitat que hem detectat dels diferents blocs funcionals de l'aplicació (missatges, xat, calendari, campus, fòrum) es que totes internament tenen una estructura similar.
Inicialment havíem plantejat la solució com la creació d'una Activity i un Layout per cada un dels blocs funcionals. Però això es podria simplificar creant una Activity dinàmica que mostri el llistat d'alertes segons cada bloc funcional.

Com el llistat de les alertes estarà format per dos blocs:

- Les que rebrem a través del webservice que seran les més recents i aniran situades a la part superior del menú.
- Les que estan emmagatzemades a la base de dades i que corresponen a alertes rebudes anteriorment.

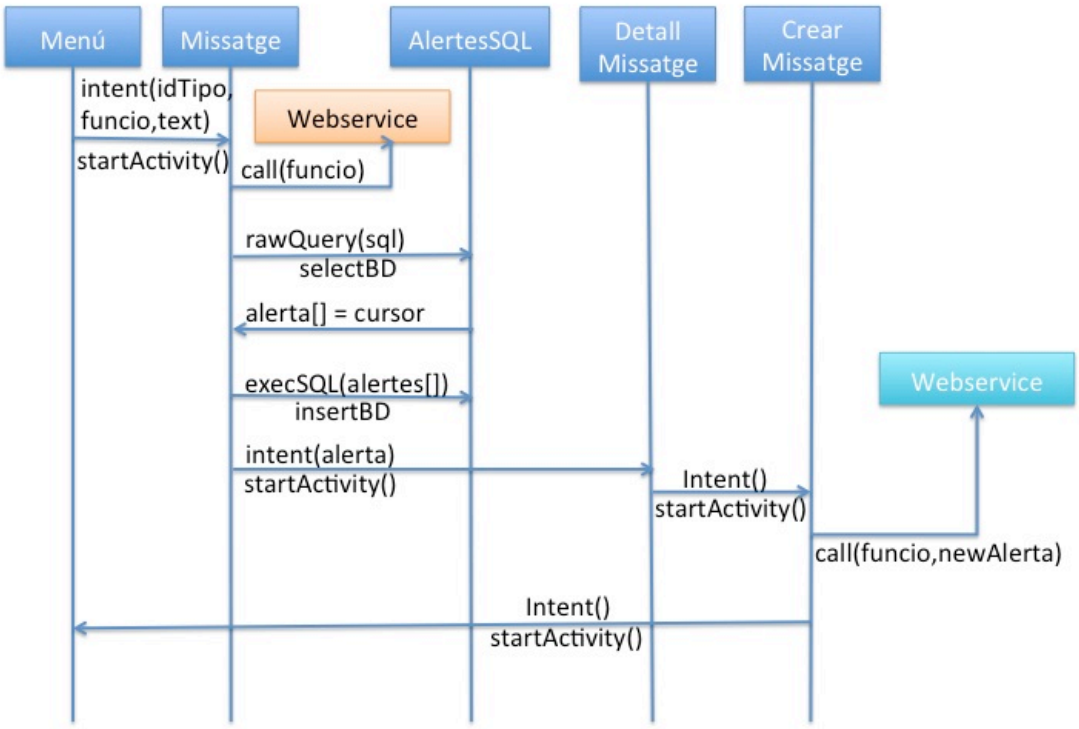
Caldrà distingir a quin webservice ens connectarem per obtenir les notificacions i quin tipus de missatge hem de recuperar de la base de dades.

L'*Activity Menú* serà l'encarregada de passar aquests paràmetres depenen de la tria que hagi fet a l'usuari dins el seu menú.
A l'Activity Missatge ens connectarem tant al webservice com a la base de dades per recuperar el llistat d'alertes.
I per visualitzar el detall d'una alerta, caldrà clicar-la i així es farà la crida a la següent Activity *DetallMissatge* on es mostrarà la informació completa de l'alerta (Titol, Usuari i Descripció).

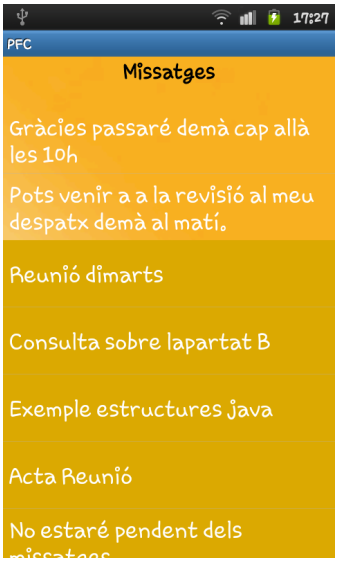
Alguns tipus d'alertes permetran retroaccions, que consistirà en la possibilitat de respondre la notificació. En aquest cas apareixerà un botó *Respondre* que activarà la següent Activity *CrearMissatge.*

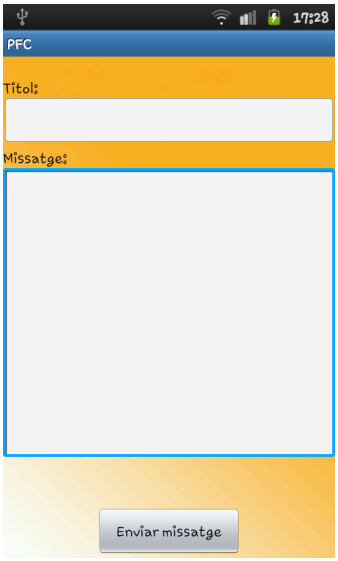
En aquesta Activity apareixerà un menú que l'usuari podrà omplir i enviar. Per fer aquest enviament l'aplicació es tornarà a connectar a un nou webservice que rebrà la notificació de resposta i la processarà segons Moodle.

El següent diagrama de seqüencia detalla lo que hem explicat anteriorment amb els mètodes que utilitza per fer tot el procés:



Exemple de les tres pantalles que compose una alerta:





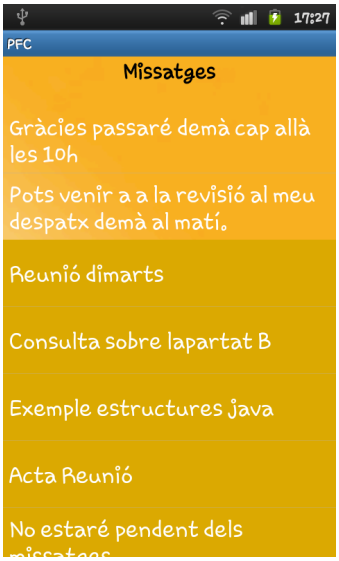
9. Escenari d'ús: Exemple Moodle i l'aplicació

En aquest apartat anem a descriure un petit exemple de l'us de l'aplicació:

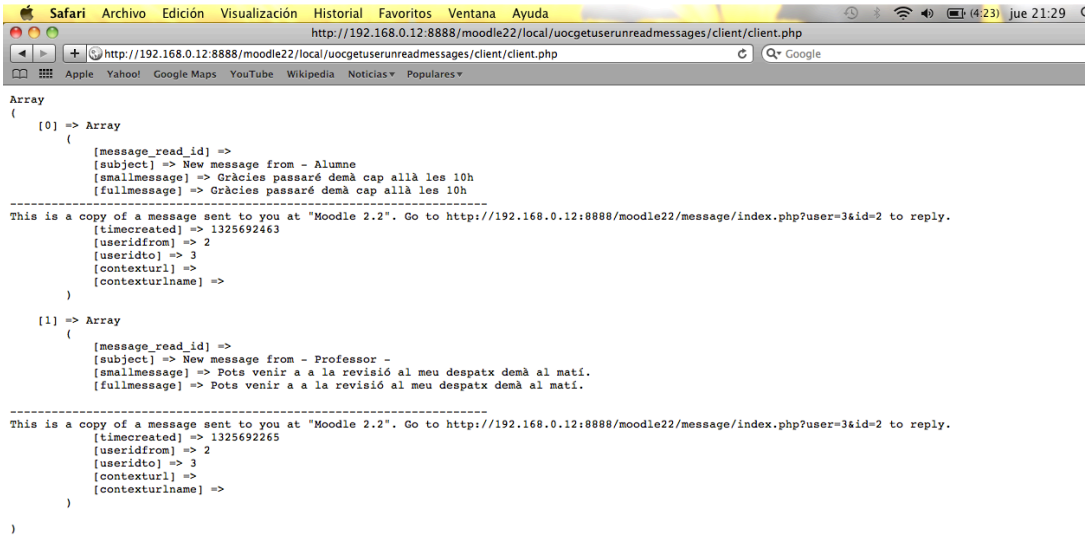
Inici de l'aplicació:
Login -> Introduir un usuari i una contrasenya:

Menú: Veure els blocs de notificacions disponibles:

Missatges: Escollim l'opció Missatges:



On veiem que els primers registres pintats en color taronja més claret seran els que agafarem del webservice.
Si observem el client del webservice, podem veure les dades que retorna el webservice:

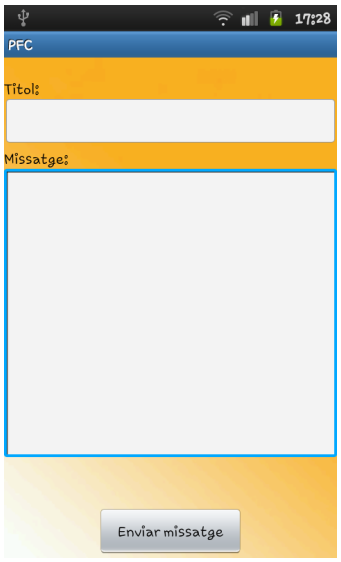


Els que tenen un color taronja més fosc, són els registres que hi han guardats a la Base de dades que veurem a continuació dels més recents.

Detall del Missatge: veurem el detall de l'alerta

On apareixen els caps de títol, de i missatge que corresponen al missatge sencer. Si es vol respondre a l'alerta, caldrà fer un clic al botó Responder:

Crea alerta: formulari que ens permetrà crear una nova alerta



Una vegada omplert els camps, al clicarà el botó Enviar missatge, s'enviarà al webservice de Moodle per que processi l'alerta com una nova notificació.

10. Possibles millores

Possibles futures línies que es podrien desenvolupar per millorar l'aplicació:

- Connexió al servidor amb mode push. Ara per ara hem implementat un tipus de connexió via pull. En un futur podria ser millorable i que fos el mateix servidor que envies les alertes quan es generessin.

- Veure les alertes a l'apartat de Notificació del mòbil. Ha estat un dels punts que ha quedat per desenvolupar. Que quan entrin les alertes vagin al menú de notificacions propi del mòbil.

- Millorar l'estètica de l'aplicació. Com ja vam comentar, dissenyar una aplicació atractiva i funcional amb Android no és una tasca senzilla. Les eines no permeten jugar massa amb els colors i fins i tot el color de la lletra d'una listView per defecte és blanc i no es pot canviar. També he pogut comprovar que visualitzar l'aplicació en diferents dispositius mòbils amb Android fa que no sempre tingui la mateixa estètica. Depenen del tamany de la pantalla a vegades els menús canvien lleugerament de lloc. Caldria treballar doncs més aquest aspecte.

- Es podria afegir un menú propi de l'aplicació que permetés configurar l'aplicació d'Android. Ara per ara només funciona el botó de retrocedir que ens portaria a la pàgina anterior.

11. Conclusions

Amb aquesta entrega farem la memòria de tot el procés que ha comportat el treball final de carrera. Han estat tres etapes que ens han ajudat a plantejar-nos la feina pas a pas, diferenciant les etapes d'un projecte. Y ara amb aquest escrit volem recollir tota la feina realitzada.

El projecte l'he trobat molt engrescador. Per una banda treballar en equip i reaprofitar parts que un altre alumne ha realitzat ho trobo molt interessant. Quan acabes l'etapa com estudiant i comences la laboral, veus que mai seràs tu qui realitzarà una aplicació al 100%, o al menys no normalment, sinó que la feina sempre és en equip. I que no sempre es senzill i fàcil entendres amb altre gent, o tenir la paciència de poder entendre allò que necessites o allò que ha fet l'altre. Per això crec que un ha d'aprendre a treballar en equip i a compartir la feina.

També he disfrutat descobrint el mon d'Android. No m'havia enfrontat mai a un projecte d'aquest tipus ni desenvolupat mai en aquesta tecnologia. I un dels aspecte que me va atreure d'agafar aquest projecte era el repte de conèixer com desenvolupar aplicacions per aquests dispositius.
L'experiència ha estat molt bona, encara que després de treballar amb llenguatges més complexos, veus que aquest no és gaire complicat. El fet de tenir activitats i pantalles fa que sigui molt intuïtiu.

Amb aquest treball espero poder acabar una etapa docent de la meva vida. I aconseguir el projecte final de carrera de l'Enginyera Informàtica.

12. Bibliografia

- Web SDK Android: http://developer.android.com/index.html
- SQLite: http://www.sqlite.org/docs.html
- Desenvolupar Blackberry: http://docs.blackberry.com/es-es/developers/deliverables/16511/Java_ME_and_Java_APIs_for_BB_446980_11.jsp
- Web de desenvolupador d'Android: http://developer.android.com