Universitat Oberta de Catalunya

Projecte Fi de Carrera

AppRecommender:
disseny i desenvolupament d'una aplicació per a Android

Sergi Miranda Sánchez
Maig del 2011

Consultor: Jordi Ceballos Villach

Aquest projecte està dedicat a la Beth,
el far que m'ajuda a trobar el camí quan estic perdut.

I que, per altra banda,
és la millor beta-tester que podria imaginar.

Índex

1. Introducció

1.1. Context general

L'aparició de l'Iphone d'Apple, el juny del 2007, va suposar una redefinició absoluta del concepte d'*smartphone*, o telèfon intel·ligent. La revolució que va suposar l'Iphone no es va limitar a les seves característiques de maquinari (com la supressió del teclat físic o una pantalla tàctil capacitiva multitàctil), les quals, d'una manera o altra, ja havien estat implementades per dispositius d'altres fabricants amb anterioritat, sinó que fonamentalment es va basar en el seu sistema operatiu i, en gran mesura, en el mètode que utilitzava per a la comercialització d'aplicacions que en permetien extendre les funcionalitats fins a límits insospitats: l'AppStore. El seu èxit ha estat tan gran (i, en gran part, imprevisible) que no és gaire agosarat afirmar que ha suposat una fita en el món de la informàtica només comparable, potser, a l'aparició de l'ordinador personal.

Aquest èxit majúscul, evidentment, ha provocat una reacció en cadena en alguns dels pesos pesants de la indústria: alguns han estat capaços d'aprofitar l'embranzida del sistema d'Apple per a presentar alternatives equivalents que, en alguns aspectes, el milloren, mentre que d'altres han vist reduïr-se dramàticament la seva quota de mercat. En el primer grup hi ha un nom propi: Google i el seu sistema operatiu per a dispositius mòbils, l'Android.

Per tal de lluitar amb l'èxit imparable de l'Iphone, Google va decidir emprendre un camí radicalment diferent al de la companyia de la poma: enlloc de desenvolupar un programari propietari lligat a una plataforma de maquinari també propietària, Google va utilitzar el sistema operatiu de codi font obert més important del món (Linux) com a base per a oferir a qualsevol fabricant de maquinari un sistema operatiu amb unes característiques similars de les de l'Iphone a cost 0 i amb infinites possibilitats d'adaptació. D'aquesta manera, fabricants com Samsung, LG, HTC o Motorola han vist la oportunitat de crear productes que poden competir amb l'Iphone, però alhora disposar de la possibilitat de diferenciar-se respecte als dispositius de la competència. Tot això ha provocat que, en un temps rècord, l'Android s'hagi convertit en el sistema operatiu més utilitzat (pel que fa a unitats venudes) en el grup dels telèfons intel·ligents.

Això no hagués estat possible sense la creació d'un gran ecosistema de desenvolupadors entusiastes que han fet créixer ràpidament la oferta d'aplicacions (tant gratuïtes com de pagament) per a aquest sistema operatiu, que es distribueixen utilitzant el mateix paradigma de l'AppStore. Aquest repositori centralitzat de programari s'anomena Android Market.

Malgrat els grans avantatges que suposa aquest mètode de distribució de programari, tant per als desenvolupadors (que troben un canal fàcil i súmmament econòmic per a fer arribar el seu treball a un nombre ingent de clients potencials) com per als usuaris (que disposen d'un mecanisme ràpid, còmode i barat per a ampliar les possibilitats dels seus

dispositius amb noves solucions de programari), el Market, sota el meu punt de vista (i crec que succeeix el mateix amb l'AppStore) presenta un problema crucial que, a llarg termini, pot representar més que un simple inconvenient: l'enorme quantitat d'aplicacions disponibles fa que, en nombroses ocasions, sigui molt difícil per a un usuari trobar aquella aplicació que soluciona les seves necessitats. Dit d'altra manera, el Market podria morir precisament a causa del seu propi èxit.

Com a usuari d'un *smartphone* amb Android, m'he trobat freqüentment amb aquest problema en intentar cercar aplicacions al Market: pots buscar-les per nom o per categoria, però donat que n'hi ha milers, trobar aquella aplicació que realment pot solucionar-te un problema, o aquell joc que s'ajusta als teus gustos personals, de vegades esdevé una mica complicat.

L'objectiu d'aquest projecte és desenvolupar una aplicació que ajudi a resoldre, si més no en part, aquest problema. A aquesta solució de programari l'anomenarem AppRecommender.

1.2. Descripció del projecte

Com he comentat en l'apartat anterior, l'ingent nombre d'aplicacions disponibles al Market fa que sigui complicat realitzar una cerca ràpida i fàcil d'aquelles que realment poden interessar a un usuari en un noment determinat. L'AppRecommender utilitzarà un paradigma diferent per a resoldre aquest problema, recomanant als usuaris programes o jocs que poden ser del seu interès, en funció de les aplicacions descarregades per altres usuaris amb els que comparteixen una afinitat. Aquesta afinitat entre usuaris s'establirà per un mètode estadístic, no pas subjectiu: si dos usuaris han descarregat la mateixa aplicació, podem suposar que tenen una mateixa problemàtica que aquesta els ha ajudat a resoldre. En la mesura que aquests dos usuaris utilitzin un nombre més elevat d'aplicacions en comú, sembla evident que les seves necessitats o preferències personals són més similars. Segons aquesta lògica, és molt probable que un d'ells hagi trobat una aplicació fantàstica que resol una necessitat que l'altre usuari també té, però que encara no ha aconseguit resoldre, i a la inversa.

L'AppRecommender desenvoluparà aquesta idea tot creant una base de dades automatitzada d'usuaris i aplicacions, en una espècie de xarxa social, que permeti mesurar numèricament l'afinitat dels diferents usuaris, així com establir objectivament quines són aquelles aplicacions que, des d'un punt de vista estadístic, és més probable que puguin interessar un determinat usuari.

Això s'aconseguirà mitjançant una arquitectura client-servidor que disposarà de dos components diferents:

- Una aplicació per a Android que cada usuari instal·larà al seu telèfon mòbil i que portarà un registre d'aplicacions instal·lades i aplicacions esborrades.

- Un component servidor (resident al *núvol*) que s'encarregarà de guardar la informació de cada usuari i les seves aplicacions, i calcular les estadístiques

necessàries sobre afinitats entre usuaris i aplicacions recomanades.

D'aquesta manera, en el moment que un usuari descarregui l'AppRecommender en el seu telèfon mòbil, de manera automàtica entrarà a formar part del registre d'usuaris que s'emmagatzemarà al servidor. Cada vegada que l'usuari descarregui una nova aplicació, l'AppRecommender comunicarà al servidor aquest fet (per tal d'actualitzar la base de dades); el mateix succeirà quan elimini una aplicació del seu telèfon.

Quan l'usuari desitgi realitzar una cerca d'aplicacions que poden ser del seu interès, accedirà a la interfície de l'AppRecommender, que realitzarà una petició al servidor per tal de recuperar la informació estadística necessària, tot generant una llista amb les aplicacions més descarregades per altres usuaris amb els quals tingui una gran afinitat. Aquesta llista serà navegable, bé per categoria, bé realitzant una cerca per nom o descripció i, mitjançant una simple selecció, podrà descarregar-la automàticament del Market.

1.3. Objectius

Amb la realització del present projecte es persegueix l'assoliment de tot un seguit d'objectius:

- Desenvolupar, amb un cas pràctic, els coneixements adquirits al llarg de la titulació, especialment pel que fa a la gestió d'un projecte de certa envergadura (planificació, gestió d'incidències, etc.).

- Aprofundir en els coneixements de la programació orientada a l'objecte, utilitzant els llenguatges Java (per al component client) i PHP (per al component servidor).

- Aprendre el funcionament del *framework* i APIs que proporciona Android per al desenvolupament d'aplicacions, així com els mètodes de distribució d'aplicacions mitjançant el Market.

- Desplegar el component servidor al *núvol*, utilitzant el sistema operatiu Linux i les eines *open-source* Apache, PHP i MySQL.

- Considerar l'escalabilitat de la solució tenint en compte un escenari en el que poguem trobar centenars de milers d'usuaris registrats simultàniament.

2. Estructuració del projecte

2.1. Introducció

Per tal d'escometre amb èxit un projecte d'aquesta envergadura, s'ha realitzat una descomposició del treball en activitats i tasques, establint una temporització per a cadascuna d'elles segons una estimació preliminar del cost que pot suposar la seva realització. Donat que durant la realització del projecte és previsible que es produeixin incidències que poden afectar a la planificació prevista, serà necessari realitzar un seguiment constant per tal de minimitzar l'efecte d'aquestes incidències sobre el projecte. En aquest sentit, cal tenir sempre present que, en tractar-se d'un projecte d'una única persona, serà difícil poder paral·lelitzar activitats i, per tant, qualsevol desviació en una activitat probablement obligarà a replantejar-se la planificació prevista per a la resta.

En aquest sentit, i com en qualsevol altre projecte, la definició de l'abast és absolutament fonamental. Donat que el cost (econòmic) del projecte no té incidència en aquest cas, els recursos que s'hi poden dedicar estan prefixats, i la dimensió temps és també inamovible (es disposa d'una data límit inamovible), les úniques dimensions del projecte que podran assumir les desviacions seran l'abast, per una banda, i la qualitat, per l'altra. Considerant que la qualitat del projecte ha de ser molt elevada tenint en compte els usuaris potencials del producte, caldrà afinar en la definició de l'abast per evitar marcar-se objectius excessivament ambiciosos, i definir una gestió de riscos adequada, amb un pla de contingència que inevitablement hauria de suposar una disminució controlada de l'abast per tal de garantir la finalització del projecte amb la temporització prevista i un mínim de funcionalitats que ens permetin considerar que aquest s'ha completat amb èxit.

2.2. Estructura de distribució del treball

Dividirem el projecte en les següents activitats i tasques:

1. Planificació del projecte
    1. Selecció del projecte
    2. Definició de requisits
    3. Planificació temporal del projecte
    4. Redacció del pla de treball
    5. Entrega de la PAC 1

2. Disseny
    1. Disseny general de l'arquitectura
    2. Disseny de la base de dades (component servidor)
    3. Disseny de l'arquitectura del component servidor
    4. Disseny de l'arquitectura del component client
    5. Disseny de la IGU (component client)

6. Creació d'un prototip
7. Entrega de la PAC 2

3. Implementació
   1. Implementació de la base de dades
   2. Implementació del component servidor
   3. Implementació del component client
   4. Entrega de la PAC 3

4. Proves
   1. Desplegament del component servidor en un entorn de proves
   2. Proves unitàries de funcionament del component servidor
   3. Proves unitàries de funcionament del component client
   4. Proves de comunicació client-servidor en un entorn de proves
   5. Proves de funcionament sobre dispositius reals
   6. Proves de càrrega

5. Tancament del projecte
   1. Redacció de la memòria
   2. Realització de la presentació virtual
   3. Entrega final

2.3. Descripció d'activitats

A continuació es mostra una descripció detallada del contingut de cadascuna de les activitats en les que s'ha dividit el projecte, així com les seves tasques.

1. Planificació del projecte

Aquesta és la primera activitat del projecte, en la qual es definiran els requisits que ha de tenir l'aplicació a desenvolupar, es definirà l'abast d'aquesta i es realitzarà una planificació temporal de cadascuna de les tasques i activitats que permeti assolir els objectius del projecte amb èxit. Com a resultat d'aquesta fase, obtindrem aquest pla de treball, que conformarà la primera prova d'avaluació continuada.

Tasca	Descripció
1.1. Selecció del projecte	Decidir, a grans trets, el projecte que es realitzarà.
1.2. Definició de requisits	Definir el context d'ús del projecte. Definició de l'abast.
1.3. Planificació temporal del projecte	Definir es tasques i activitats. Establir una temporització per a cada activitat. Plasmar la temporització en un diagrama de Gantt.
1.4. Redacció del pla de treball	Redactar el pla de projecte a partir de la informació generada en les activitats precedents.
1.5. Entrega de la PAC 1	Fita: entregar el pla de treball obtingut com a resultat d'aquesta activitat.

2. Disseny

La fase de disseny és una de les més importants i costoses del projecte. En aquesta fase es dissenyarà l'arquitectura dels diferents components de l'aplicació, el model de dades subjacent i la interfície gràfica de l'usuari.

Com a resultat final d'aquesta activitat obtindrem un primer prototip que permetrà una primera avaluació del lliurable del projecte i que conformarà la segona prova d'avaluació continuada.

Tasca	Descripció
2.1. Disseny general de l'arquitectura	Realitzar el disseny de l'arquitectura global de l'aplicació: - Definir les responsabilitats del component servidor. - Definir les responsabilitats del component client. - Dissenyar la interfície entre els components client i servidor, establint els mecanismes de comunicació entre ambdós mòduls (llenguatge d'intercanvi de dades, casos d'ús, etc.)
2.2. Disseny de la base de dades (component servidor)	Dissenyar l'estructura de la base de dades que emmagatzemarà la informació (al component servidor). Disseny de taules, procediments emmagatzemats, disparadors, etc.
2.3. Disseny de l'arquitectura del component servidor	Dissenyar l'arquitectura del component servidor: - Disseny de les capes: model de dades, lògica de negoci, presentació. - Disseny de classes.
2.4. Disseny de l'arquitectura del component client	Dissenyar l'arquitectura del component client: - Disseny de les capes: model de dades, lògica de negoci, presentació. - Disseny de classes.
2.5. Disseny de la IGU (component client)	Dissenyar la interfície gràfica d'usuari del component client (el component servidor, per les seves característiques, no disposarà de cap interfície d'usuari).
2.6. Creació d'un prototip	A partir del disseny sorgit de les tasques precedents, crear un prototip que permeti avaluar la usabilitat i viabilitat de l'aplicació.
2.7. Entrega de la PAC 2	Fita: Entregar els diagrames de disseny i el prototip com a segona prova d'avaluació continuada.

3. Implementació

En la fase d'implementació realitzarem la codificació dels diferents components (model de dades, component client i component servidor), l'arquitectura dels quals s'ha definit clarament a la fase anterior.

Com a resultat d'aquesta activitat obtindrem una primera versió funcional de l'aplicació, que conformarà la tercera prova d'avaluació continuada. Tot i que, en aquest estadi del projecte, l'aplicació hauria d'acomplir els seus objectius, encara li mancarà superar les proves necessàries que n'assegurin la qualitat.

Tasca	Descripció
3.1. Implementació de la base de dades	Realitzar la implementació física de la base de dades a partir del model generat en l'activitat anterior: - Elecció de l'SGBD concret sobre el que es realitzarà la implementació. - Creació de taules, claus primàries, claus foranes, etc. - Desenvolupament dels (possibles) procediments emmagatzemats. - Implementació dels (possibles) disparadors.
3.2. Implementació del component servidor	Realitzar el desenvolupament del component servidor: - Elecció del llenguatge de programació amb el que es desenvoluparà el component servidor. - Implementació de la funcionalitat del component, a partir del disseny de classes generat en l'activitat anterior. - Implementació de la interfície, en la part que és responsabilitat del component servidor.
3.3. Implementació del component client	Realitzar el desenvolupament del component client: - Elecció del llenguatge de programació i entorn de desenvolupament que s'utilitzarà per a la implementació. - Implementació de la funcionalitat del component, a partir del disseny realitzat en l'activitat anterior. - Implementació de la interfície, en la part responsabilitat del component client.
3.4. Entrega de la PAC 3	Fita: Com a resultat de les tasques anteriors, es disposarà d'una primera versió funcional de l'aplicació que serà lliurada com a part fonamental de la tercera prova d'avaluació continuada.

4. Proves

En la fase d'implementació haurem obtingut una aplicació funcional, però abans de realitzar el llançament del producte, cal garantir que aquest compleix els estàndards de qualitats marcats. Per fer-ho, serà necessari realitzar proves extensives, tant unitàries (per a cada component, classe i mètode) com d'integració (comunicació entre els components client i servidor). Per fer-ho, serà necessari crear uns entorns de proves, primer, i realizar el desplegament sobre dispositius reals, després. Per últim, i a causa de les característiques específiques d'aquest projecte, és de vital importància realitzar proves de càrrega sobre el component servidor, per assegurar que el funcionament no es veu degradat en augmentar la quantitat d'usuaris del sistema.

Com a resultat d'aquesta activitat, obtindrem un lliurable que acompleix els estàndards de qualitat definits.

Tasca	Descripció
4.1. Desplegament del component servidor en un entorn de proves	Crear un entorn de proves per al desplegament del component servidor (per exemple, una màquina virtual). Realitzar un script d'instal·lació que permeti instal·lar i configurar el component.
4.2. Proves unitàries de funcionament del component servidor	Realitzar proves unitàries sobre el component servidor, per verificar que cada classe realitza correctament les seves tasques.
4.3. Proves unitàries de funcionament del component client	Realitzar proves unitàries sobre el component client, per verificar que cada classe realitza correctament les seves tasques.
4.4. Proves de comunicació client-servidor en un entorn de proves	Realitzar proves de funcionament en un entorn de proves: emulador d'un dispositiu Android per al component client, i el servidor de proves desplegat a la primera tasca, per al component servidor.
4.5. Proves de funcionament sobre dispositius reals	Realitzar proves de funcionament sobre dispositius Android reals (en funció de la disponibilitat d'aquests dispositius, seria interessant poder realitzar proves sobre diversos *smartphones* de fabricants i característiques diferents).
4.6. Proves de càrrega	Realitzar proves de càrrega per a garantir l'escalabilitat del sistema: - Desenvolupar un script que carregui grans volums de dades sobre el component client (generant milers d'usuaris, amb desenes d'aplicacions per a cada usuari). - Comprovar la velocitat de funcionament del sistema amb aquest volum de dades.

5. Tancament del projecte

Com a resultat de la fase anterior, haurem obtingut un lliurable que acompleix els requisits i abast definits en la primera fase, i que compleix els estàndards de qualitat definits (és a dir, està aparentment lliure d'errors). En aquesta última fase caldrà realitzar la redacció de la memòria definitiva del projecte, així com la presentació virtual. Com a resultat d'aquesta activitat, obtindrem aquests dos documents, que juntament amb el lliurable (codi font, codi objecte i *scripts* de creació de la base de dades) conformaran l'entrega final.

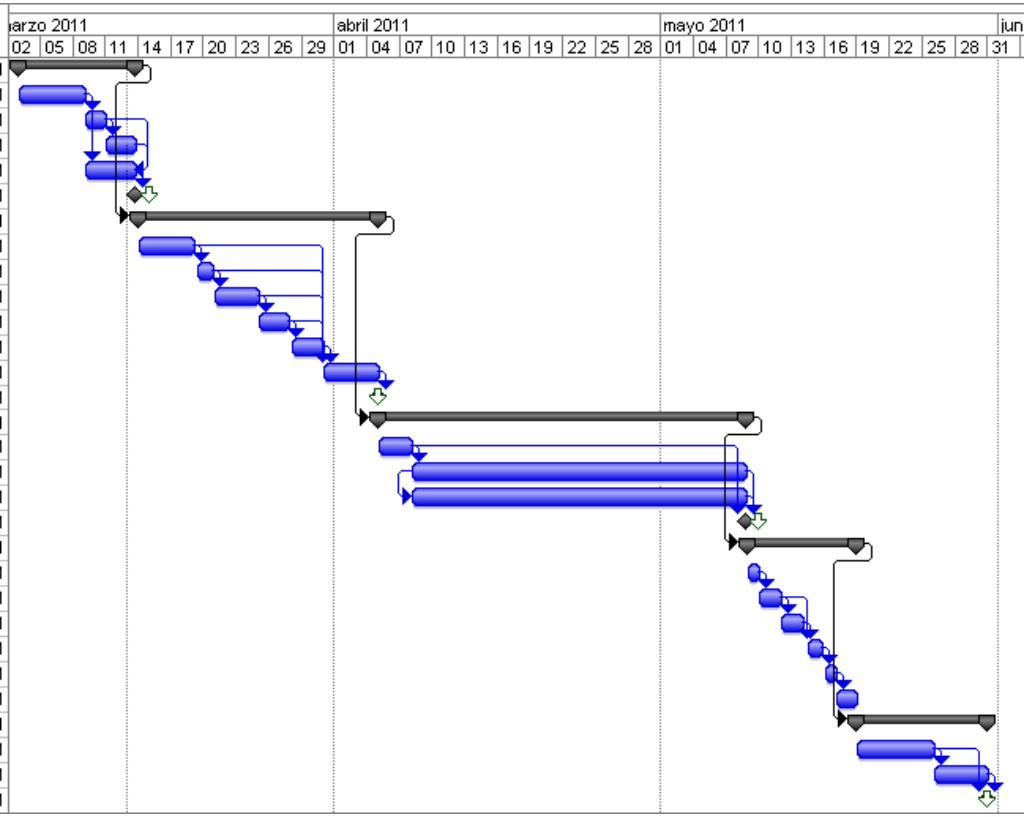
Tasca	Descripció
5.1. Redacció de la memòria del projecte	Redactar la memòria del projecte, que inclourà tota la documentació generada al llarg del projecte, captures de pantalla, procediment d'instal·lació, explicació de l'arquitectura, etc.
5.2. Realització de la presentació virtual	Realitzar la presentació virtual.
5.3. Entrega final	Fita: lliurament del projecte (memòria del projecte, lliurable obtingut i presentació virtual).

2.3. Temporització del projecte

A partir de la definició d'estructura de distribució del treball definida als apartats anteriors s'ha creat la temporització de cadascuna de les tasques i activitats per tal de poder assolir els objectius del projecte en els terminis establerts per a cadascuna de les proves d'avaluació continuada i l'entrega final.

En aquesta temporització s'han paral·lelitzat dues activitas (la implementació dels components servidor i client), donat que es preveu que es realitzaran de manera simultània. En el cost temporal d'aquestes dues activitats (31 dies) ja s'ha tingut en compte aquesta paral·lelització (és a dir, el cost conjunt de les dues tasques seran 31 dies; la distribució exacta de temps que es dedicarà a cadascuna d'elles serà variable, éssent aproximadament d'un 50% per a cada tasca).

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**Planificació del projecte**	**11 días**	**jue 03/03/11**	**dom 13/03/11**
2	Selecció del projecte	6 días	jue 03/03/11	mar 08/03/11
3	Definició de requisits	2 días	mié 09/03/11	jue 10/03/11
4	Planificació temporal del projecte	3 días	vie 11/03/11	dom 13/03/11
5	Redacció del pla de treball	5 días	mié 09/03/11	dom 13/03/11
6	Entrega de la PAC 1	0 días	dom 13/03/11	dom 13/03/11
7	**Disseny**	**22 días**	**lun 14/03/11**	**lun 04/04/11**
8	Disseny general de l'arquitectura	5 días	lun 14/03/11	vie 18/03/11
9	Disseny de la base de dades (component servid	2 días	sáb 19/03/11	dom 20/03/11
10	Disseny de l'arquitectura del component servidor	4 días	lun 21/03/11	jue 24/03/11
11	Disseny de l'arquitectura del component client	3 días	vie 25/03/11	dom 27/03/11
12	Disseny de la IGU (component client)	3 días	lun 28/03/11	mié 30/03/11
13	Creació d'un prototip	5 días	jue 31/03/11	lun 04/04/11
14	Entrega de la PAC 2	0 días	lun 04/04/11	lun 04/04/11
15	**Implementació**	**34 días**	**mar 05/04/11**	**dom 08/05/11**
16	Implementació de la base de dades	3 días	mar 05/04/11	jue 07/04/11
17	Implementació del component servidor	31 días	vie 08/04/11	dom 08/05/11
18	Implementació del component client	31 días	vie 08/04/11	dom 08/05/11
19	Entrega de la PAC 3	0 días	dom 08/05/11	dom 08/05/11
20	**Proves**	**10 días**	**lun 09/05/11**	**mié 18/05/11**
21	Desplegament del component servidor en un ent	1 día	lun 09/05/11	lun 09/05/11
22	Proves unitàries de funcionament del component	2 días	mar 10/05/11	mié 11/05/11
23	Proves unitàries de funcionament del component	2 días	jue 12/05/11	vie 13/05/11
24	Proves de comunicació client-servidor en un ent	2 días	sáb 14/05/11	dom 15/05/11
25	Proves de funcionament sobre dispositius reals	1 día	lun 16/05/11	lun 16/05/11
26	Proves de càrrega	2 días	mar 17/05/11	mié 18/05/11
27	**Tancament del projecte**	**12 días**	**jue 19/05/11**	**lun 30/05/11**
28	Redacció de la memòria	7 días	jue 19/05/11	mié 25/05/11
29	Realització de la presentació virtual	5 días	jue 26/05/11	lun 30/05/11
30	Entrega final	0 días	lun 30/05/11	lun 30/05/11



3. Disseny del projecte

En primer lloc, voldria indicar que en tot el disseny (tant pel que fa a la base de dades com pel que fa a les classes, components, etc.) s'ha utilitzat la llengua anglesa per als identificadors, noms de les funcions, taules, camps, etc. Aquesta ha estat una decisió motivada pel fet que crec que aquest projecte podria tenir viabilitat (fins i tot econòmica) més enllà dels objectius purament acadèmics. Per aquest motiu, i preveient que en un futur potser es podria utilitzar tant el disseny com el propi codi font com a base d'un projecte més gran, en el que podrien intervenir persones d'àmbit internacional, he cregut convenient aplicar aquest enfoc.

3.1. Descripció general de l'arquitectura

Segons els requisits establerts a la primera fase del desenvolupament del projecte, l'aplicació tindrà dos components ben diferenciats:

- **Component servidor:**

  Aquest component estarà instal·lat en un servidor (una màquina física o bé en un servei de computació en grid com ara Amazon EC2). En un moment determinat trobarem un únic component servidor desplegat, que serà administrat per la persona o companyia responsable del servei. Els usuaris del sistema no tindran accés directe a aquest component, sinó que hi interactuaran indirectament a través del component client instal·lat al seu dispositiu Android.

  El disseny del component servidor serà agnòstic pel que fa a la plataforma del component client; és a dir, tindrem un únic component servidor que serà capaç de gestionar la informació d'usuaris que disposin d'un component client de diverses plataformes. D'aquesta manera, podrem trobar usuaris d'Android (inicialment), però en un futur podríem desenvolupar components clients per a iPhone o altres plataformes, que podran compartir el mateix component servidor.

  Les responsabilitats del component servidor seran:

  - Mantenir actualitzada la base de dades d'usuaris i aplicacions instal·lades per cada usuari.

  - Proporcionar una interfície amb els components client (que es dissenyarà com un conjunt de serveis web). Aquesta interfície permetrà als clients realitzar les següents accions:

    - Notificar la instal·lació del component client en un nou dispositiu (alta d'usuaris).

- Notificar la instal·lació o desinstal·lació d'una aplicació per part d'un usuari. Aquesta acció provocarà el recàlcul de les estadístiques dels usuaris afins.

- Realitzar consultes sobre la base de dades d'usuaris i aplicacions. Concretament, permetrà realitzar els següents tipus de consultes:
  - Consulta de les aplicacions instal·lades d'un usuari.
  - Consulta dels usuaris afins a un usuari determinat.
  - Consulta de les aplicacions recomanades per a un usuari (en funció dels usuaris afins i les aplicacions instal·lades per part d'aquests usuaris).

- **Component client:**

  Aquest component estarà instal·lat físicament al dispositiu de cada usuari. Al seu torn, es dividirà en dos subcomponents:

  - Servei en segon plànol:

    Aquest component estarà sempre en execució al dispositiu del client. Tindrà una única responsabilitat: la notificació al component servidor, mitjançant la interfície d'aquest, la instal·lació de o desinstal·lació d'una aplicació al dispositiu del client. Això permetrà actualitzar la base de dades d'aplicacions al servidor.

  - Component principal:

    Aquest component disposarà d'una interfície gràfica d'usuari (IGU), i tindrà les següents responsabilitats:

    - En la primera execució, notificar al servidor (a través de la seva interfície) la instal·lació de l'aplicació al component client (això provocarà la creació de l'usuari a la base de dades del servidor).

    - També en la primera execució, recopilar la llista d'aplicacions instal·lades al dispositiu i transmetre-la al servidor.

    - Proporcionar la interfície gràfica amb la qual interactuarà l'usuari. Aquesta interfície ha de permetre:

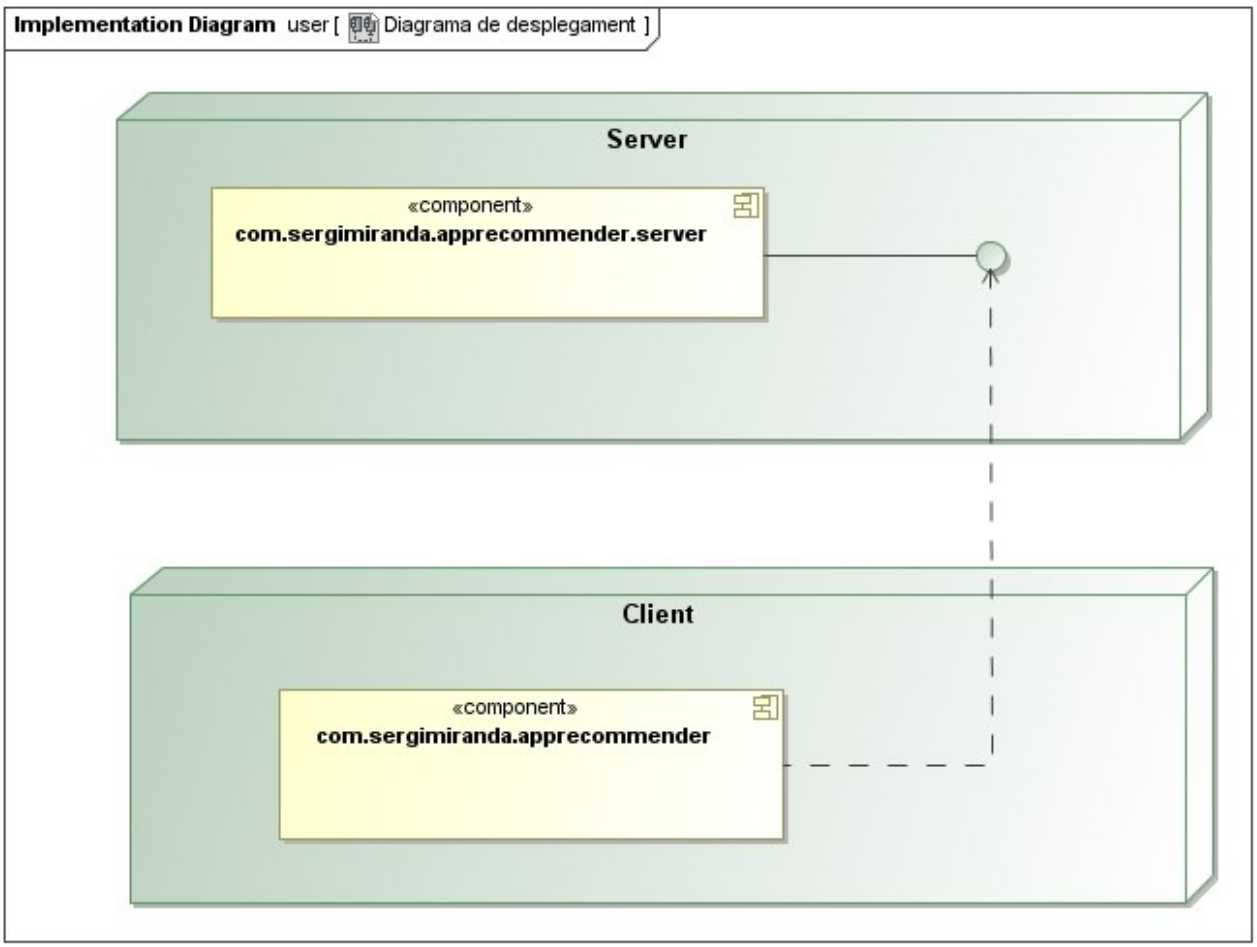
      - Obtenir una llista de les aplicacions recomanades. Per a cada aplicació, es mostrarà el nom, categoria, puntuació (que s'haurà calculat automàticament en funció del nombre d'usuaris afins que la tenen instal·lada) i número de vots positius (un usuari pot marcar una aplicació com a “m'agrada”).
        L'usuari ha de poder realitzar una cerca d'aplicacions recomanades per nom, categoria, puntuació mínima, vots mínims i/o usuari.
        També disposarà de la opció d'ordenar els resultats (inicialment s'ordenaran per puntuació).

      - Obtenir una llista d'usuaris afins. Els usuaris afins són aquells amb els

quals es comparteix un nombre mínim d'aplicacions. Addicionalment, l'usuari ha de poder marcar a un usuari afí com a *amic.* Marcar un usuari com a *amic* tindrà com a conseqüència que els vots d'aquest usuari ponderaran més que els que no ho són.
A la llista d'usuaris afins apareixerà l'àlies de l'usuari, número d'aplicacions compartides amb ell i una icona per veure si l'usuari és amic o no i per poder canviar l'estat.
Fent click sobre un usuari, s'accedirà a la llista d'aplicacions instal·lades per part d'aquest usuari (amb la seva puntuació corresponent).

- Seleccionant una aplicació, l'usuari ha de poder:
  - Veure el nom, descripció i categoria de l'aplicació.
  - Veure la puntuació.
  - Descarregar l'aplicació directament del Market.

Per últim, es mostra a continuació el diagrama de desplegament de l'arquitectura dissenyada, on apareixen els dos components (servidor i client) i la seva interrelació mitjançant la interfície oferta pel component servidor:



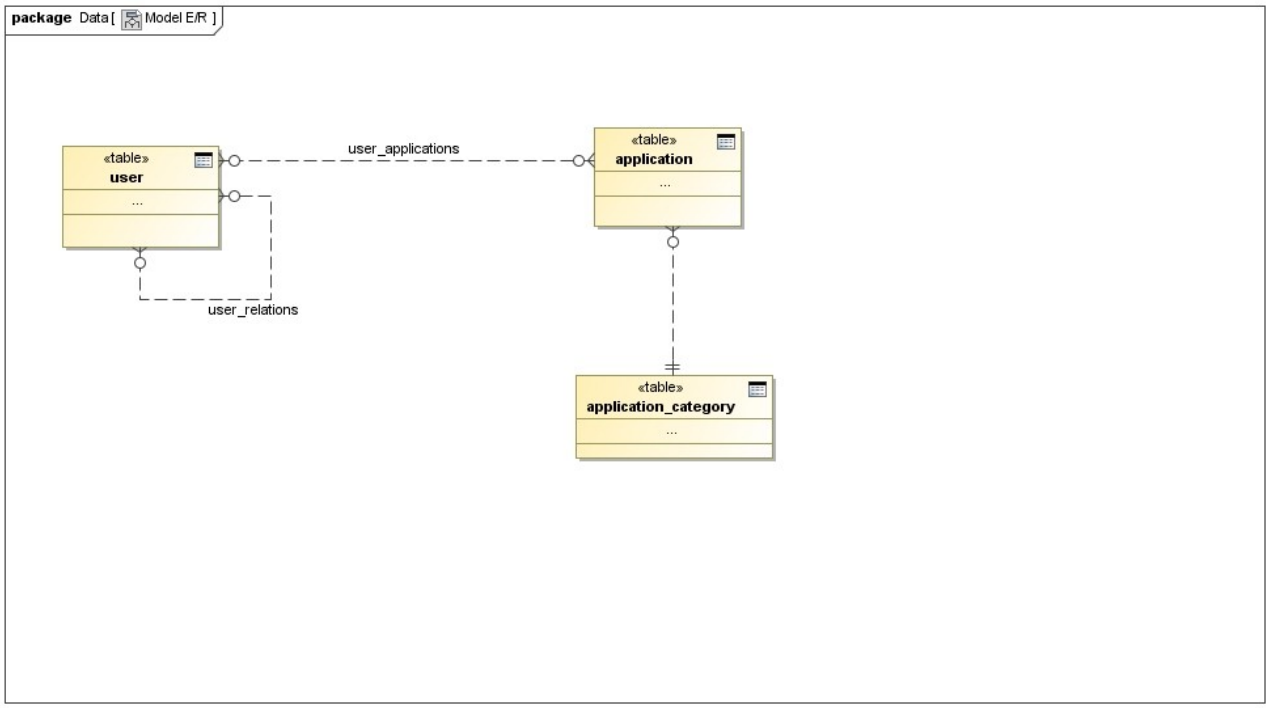
3.2. Disseny de la base de dades

Com s'ha indicat a la descripció general de l'arquitectura, tota la informació que necessita el programa per a acomplir les seves tasques es guardarà en una base de dades del servidor. Aquesta base de dades ha d'emmagatzemar les dades dels usuaris i de les aplicacions instal·lades per part d'aquests usuaris.

Amb aquests requisits, el disseny de la base de dades és força simple. Concretament, crearem les següents entitats:

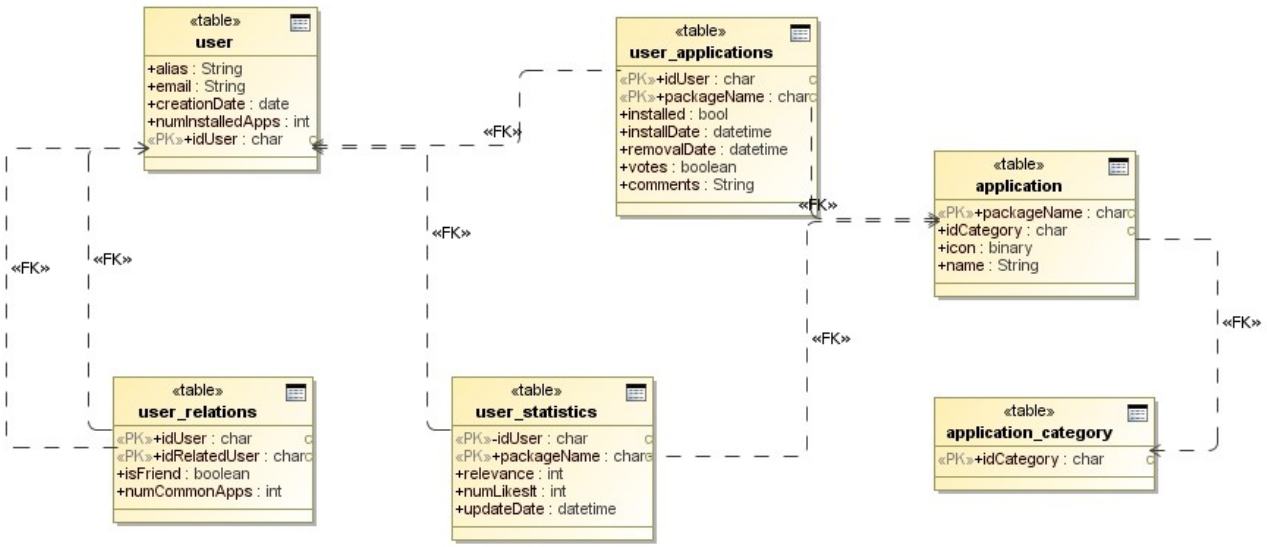
- user: Modela un usuari del sistema.
- application: Modela una aplicació qualssevol (que pot estar instal·lada, o no, per part d'algun dels usuaris).
- application_category: Modela una categoria d'aplicació (jocs, aplicacions de productivitat, aplicacions educatives, etc.).

El model E/R que permetrà assolir els requisits amb aquestes 3 entitats i les seves relacions corresponents és el que es mostra a la figura següent:



Com pot observar-se al diagrama, l'entitat *user* té una relació amb multiplicitat 0..* amb ella mateixa (relació anomenada *user_relations* i que modela la relació entre usuaris afins) i una altra relació també amb multiplicitat 0..* amb l'entitat *application* (aquesta relació permet establir les aplicacions relacionades amb aquell usuari). Per altra banda, entre les entitats *application* i *application_category* s'estableix una relació amb multiplicitat 1 de la banda d'*application_category* i 0..* de la banda d'*application.*

Transformant el model E/R al model relacional i afegint els atributs necessaris a cada entitat, obtenim el diagrama que es mostra a la figura següent:



En el pas del model E/R al model relacional hem convertit les relacions *user_relations* i *user_applications* en taules (a causa de la multiplicitat 0..* als dos cantons de cadascuna d'aquestes relacions).

Hem afegit els atributs necessaris a cadascuna de les entitats, i hem definit les claus primàries i foranes per a cadascuna d'elles. Per últim, hem creat una entitat no contemplada al model E/R: *user_statistics*. Aquesta entitat mantindrà informació agregada de les aplicacions recomanades per a cada usuari; d'aquesta manera, no serà necessari calcular aquesta informació constantment. Això permetrà un augment significatiu del rendiment i una millor escalabilitat del sistema.

Definirem cada entitat amb els seus atributs:

- **application_category**: Modela una categoria d'aplicacions.
  idCategory: Clau primària. Serà l'identificador de la categoria (serà un valor de tipus cadena). Els valors possibles estan definits a les especificacions del sistema Android.

- **application**: Modela una aplicació (que pot estar instal·lada per algun usuari, però no forçosament ha de ser així).
  packageName: Clau primària. És un valor de tipus cadena que identifica unívocament l'aplicació. Qualsevol aplicació per a Android ha de tenir un nom de paquet únic, que és el que utilitzarem per a identificar les diferents aplicacions.
  idCategory: Identificador de la categoria a la que pertany l'aplicació.
  icon: Valor binari. Guarda la icona de l'aplicació.
  name: Valor de tipus cadena. Guarda el nom de l'aplicació.

user: Modela un usuari del sistema.
idUser: Clau primària. Serà un valor de tipus cadena. Cada dispositiu Android té un identificador únic, que serà el que utilitzarem per a identificar els usuaris.
alias: Valor de tipus cadena. Serà el nom amb el qual l'usuari vol ser identificat en l'àmbit de l'aplicació.
email: Valor de tipus string. En aquest camp s'emmagatzemarà l'adreça de correu electrònica de l'usuari (que per motius de protecció de dades no serà visible per als altres usuaris, però sí que podrà ser utilitzada per a que un usuari pugui buscar-ne a un altre).
creationDate: Valor de tipus data i hora. Indica el dia i la hora en la que s'ha creat l'usuari a la base de dades.
numInstalledApps: Valor enter. Indica el número total d'aplicacions que té instal·lades aquest usuari al seu dispositiu Android.

- **user_relations**: Modela la relació entre dos usuaris (usuaris afins, perquè comparteixen una o més aplicacions).
idUser: Clau primària.
idRelatedUser: Clau primària.
isFriend: Valor de tipus booleà. El seu valor serà *cert* quan un usuari hagi indicat que l'altre usuari és amic seu.

- **user_applications**: Modela la relació entre els usuaris i les seves aplicacions instal·lades o eliminades (quan un usuari esborra una aplicació del seu dispositiu, no esborrarem el registre, per poder mantenir un històric de les aplicacions que ha eliminat).
idUser: Clau primària. Apunta a l'usuari.
packageName: Clau primària. Apunta a l'aplicació.
installed: Valor booleà. Indica si l'usuari té actualment instal·lada l'aplicació al seu dispositiu.
installDate: Valor de tipus data i hora. Indica el dia i hora en els que l'usuari va instal·lar l'aplicació al seu dispositiu.
removalDate: Valor de tipus data i hora. Indica el dia i hora en els que l'usuari va esborrar l'aplicació al seu dispositiu.
likesIt: Valor de tipus booleà. Serà cert quan un usuari hagi indicat que li agrada aquesta aplicació.
comments: Valor de tipus cadena. En aquest camp s'emmagatzemaran els comentaris que un usuari vulgui publicar al respecte de l'aplicació, perquè els altres usuaris puguin conèixer la seva opinió al respecte.

- **user_statistics**: Emmagatzema les estadístiques de les aplicacions recomanades de l'usuari (és a dir, les aplicacions que els seus usuaris afins tenen instal·lades als seus dispositius Android). Això permet no haver de refer els càlculs constantment, sinó únicament quan l'usuari o un dels seus usuaris afins instal·la o elimina una aplicació del seu dispositiu.
  idUser: Clau primària. Apunta a l'usuari.
  packageName: Clau primària. Apunta a l'aplicació.
  relevance: Valor enter. Indica la puntuació de l'aplicació (calculada en funció del nombre d'usuaris afins que la tenen instal·lada i el nombre d'usuaris afins que l'han eliminat del seu dispositiu).
  votes: Valor de tipus enter. Indica el nombre d'usuaris que han votat a favor d'aquesta aplicació.
  updateDate: Valor de tipus data i hora. Indica el dia i hora de la última actualització de les estadístiques.

Resumint, l'estructura de la base de dades serà la següent:

application_cartegory (idCategory)

application (packageName, idCategory, icon, name)
on {idCategory} referencia application_category

user (idUser, alias, email, creationDate, numInstalledApps)

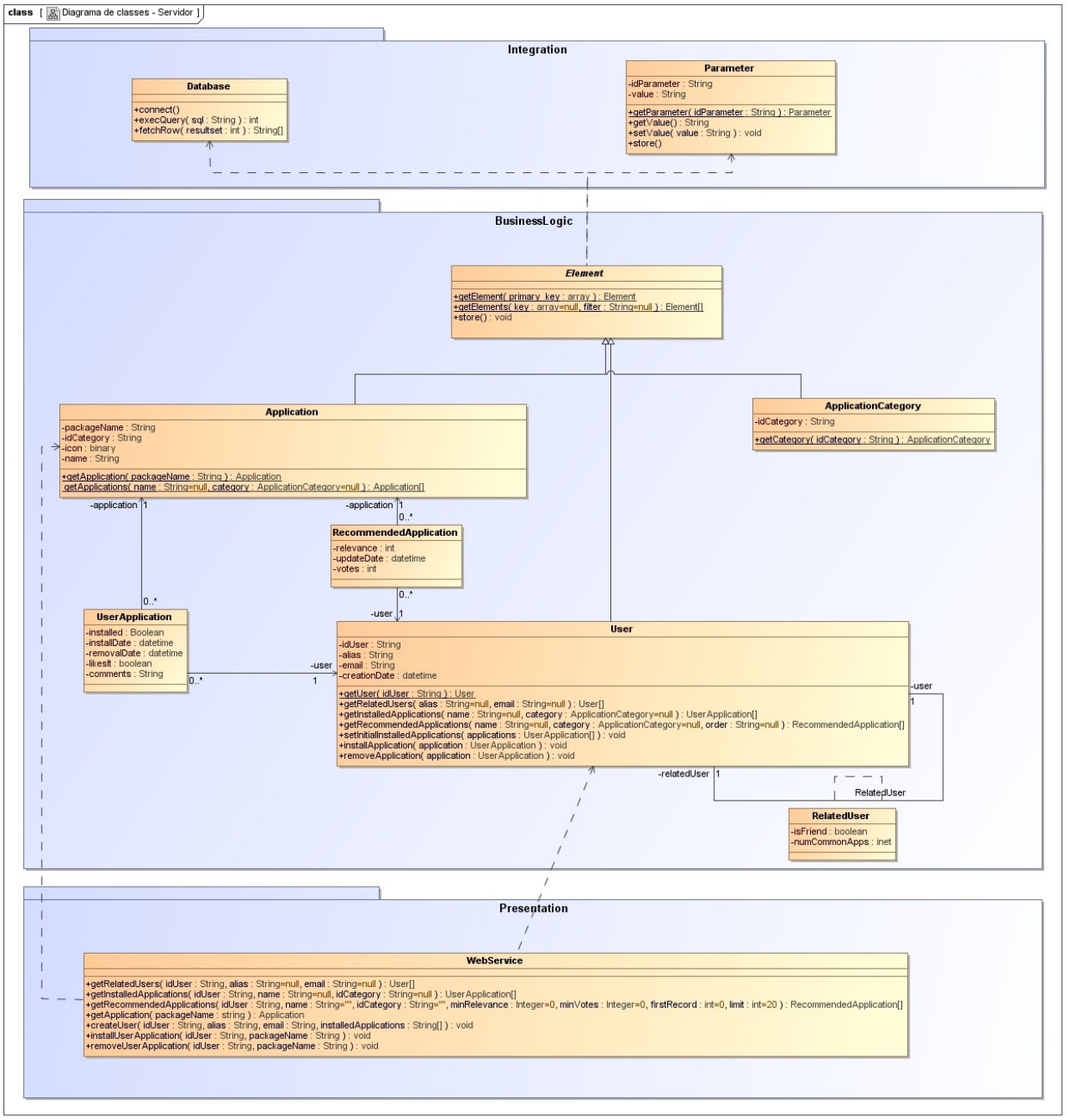
user_relations (idUser, idRelatedUser, isFriend)
on {idUser} referencia user
i {idRelatedUser} referencia user

user_applications (idUser, packageName, installed, installDate, removalDate, likesIt, comments)
on {idUser} referencia user
i {packageName} referencia application

user_statistics (idUser, packageName, relevance, votes, updateDate)
on {idUser} referencia user
i {packageName} referencia application

3.3. Disseny de l'arquitectura del component servidor

A continuació es mostra el diagrama de classes que s'ha dissenyat per al component servidor:



Com queda palès al diagrama, s'ha dissenyat una arquitectura de 3 capes:

- **Integration:** La capa d'integració comprèn les classes Database i Parameter, que són les que depenen de la implementació concreta que es realitzarà (per exemple, el motor de base de dades utilitzat).
- **Business Logic:** Aquesta capa comprèn les classes on es desenvoluparà la lògica de negoci de l'aplicació.
- **Presentation:** Tot i que el component servidor no disposa d'una capa de presentació pròpiament dita (ja que els usuaris no hi interactuaran directament), s'ha creat una capa de presentació que inclou la classe WebService, que és la interfície amb la qual interactuarà el component client.

A continuació, s'ofereix una breu descripció de cadascuna de les classes:

- **Capa d'integració:**

  - **Parameter:**
    Aquesta classe encapsula un paràmetre de l'aplicació (seran parametritzables, per exemple, les credencials d'accés a l'SGBD). Mitjançant els mètodes públics d'aquesta classe es podrà recuperar el valor d'un paràmetre determinat o modificar aquest valor.

  - **Database:**
    Aquesta classe encapsula els mètodes d'accés a l'SGBD. Mitjançant aquests mètodes, es podrà connectar a la base de dades, realitzar consultes SQL i recuperar els registres d'aquestes consultes. Mitjançant una implementació diferent d'aquesta classe es podria canviar fàcilment d'SGBD sense afectar la capa de la lògica del negoci.

- **Capa de la lògica del negoci:**

  - **Element:**
    Aquesta classe abstracta és la base sobre la que es construeixen les altres classes d'aquesta capa, representant un element qualssevol de l'aplicació. Els mètodes d'aquesta classe permeten llegir un element (o col·lecció d'elements), establir-ne els valors dels seus atributs i guardar-lo a la base de dades.

  - **ApplicationCategory:**
    Modela una categoria d'aplicació.

  - **Application:**
    Modela una aplicació.

  - **UserApplication:**
    Modela una aplicació instal·lada per un usuari.

  - **RecommendedApplication:**
    Modela una aplicació recomanada.

- **User:**
  Modela un usuari. Aquesta classe és la que encapsularà les funcionalitats més importants de la lògica del negoci. Entre d'altres, ofereix els mètodes públics de recuperació de les aplicacions instal·lades per part d'aquest usuari, les aplicacions recomanades per a l'usuari i l'actualització de les aplicacions instal·lades (i/o esborrades) per part de l'usuari.

- **RelatedUser:**
  Classe de l'associació User – User. Modela un usuari afí a un altre, indicant si l'usuari ha indicat que l'altre és amic seu.

- **Capa de presentació:**

  - **WebService:**
    Aquesta classe representa el servei web que actuarà d'interfície amb el component client. Els mètodes públics d'aquesta classe permeten d'oferir al component client els serveis que s'han definit a les especificacions (obtenir la llista d'usuaris afins, obtenir les aplicacions instal·lades o recomanades per a un usuari, obtenir una aplicació a partir del seu identificador, crear un nou usuari i notificar la instal·lació o desinstal·lació d'una aplicació per part d'un usuari.

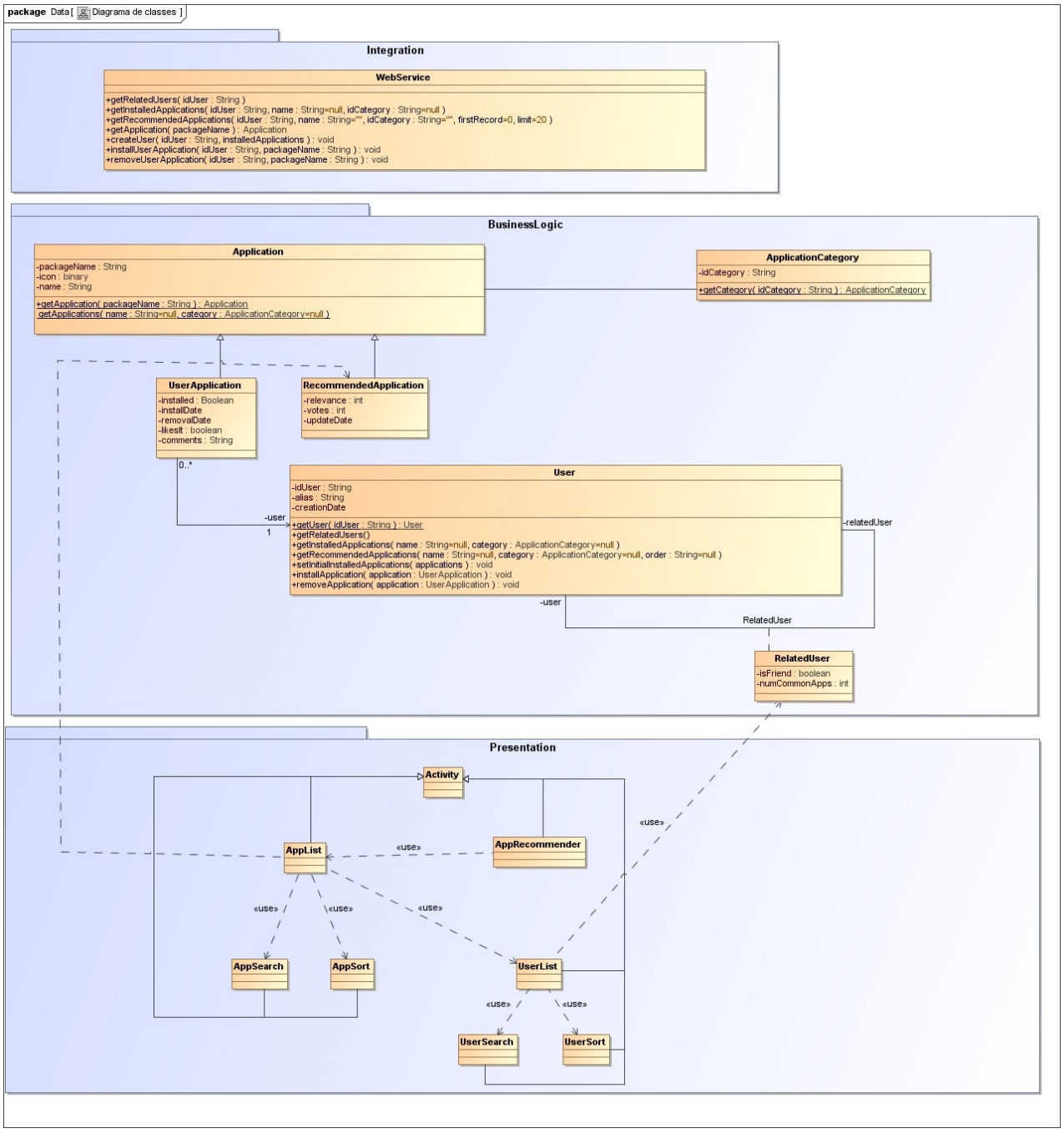
En el disseny d'aquesta arquitectura s'ha intentat seguir els principis de disseny bàsics, com ara:

- Baix acoblament.
- Alta cohesió.
- La llei de Demeter (limitació dels objectes que han d'utilitzar les operacions d'un objecte).
- Substitució de Liskov (les instàncies d'una subclasse C han de ser subsituïbles per instàncies de les superclasses de C).
- Segregació d'interfícies (els objectes no han de dependre d'operacions que no utilitzen).

També s'han tingut en compte determinats patrons de disseny, com ara el patró *mètode factoria* a la classe Element (que crearà instàncies de les subclasses determinant-les en temps d'execució) o Expert (assignant les responsabilitats de les operacions a la classe que disposa de la informació necessària per a realitzar aquestes operacions, com en el cas de la classe User, que disposa de la informació dels usuaris afins).

3.4. Disseny de l'arquitectura del component client

A continuació es mostra el diagrama estàtic de classes del component client:



El component client ha estat dissenyat, com en el cas del component servidor, amb 3 capes:

- **Integration:** Aquesta capa té delegada la responsabilitat de la comunicació amb el component servidor. Per aquest motiu, hem dissenyat una classe WebService que proporciona els mètodes necessaris per a recuperar les dades del servidor. Aquests mètodes tenen la mateixa signatura que els mètodes homònims de la classe WebService que hem definit a la capa de presentació del component servidor.
- **BusinessLogic:** En aquesta capa hi trobarem la lògica del negoci. Hi trobem classes anàlogues a les de la capa de la lògica del negoci del component servidor (tot i que no tenen tots els atributs i/o mètodes d'aquelles, ja que les seves responsabilitats són diferents).
- **Presentation:** Els usuaris interactuaran amb el component client; per tant, aquest sí que ha de disposar d'una capa de presentació *clàssica.* En aquesta capa trobem les classes que tenen la responsabilitat de definir la IGU; és a dir, presentar les dades a l'usuari i gestionar la interacció amb aquest.
  A la capa de presentació s'ha definit una classe per a cadascuna de les pantalles que se li mostraran a l'usuari. En Android, cada *pantalla* ha de ser una subclasse d'Activity; al diagrama de classes s'ha representat aquesta relació, així com les relacions d'ús entre les diferents classes. Per simplicitat, no s'han detallat tots els atributs i mètodes d'aquestes classes.

3.5. Disseny de la IGU

En una aplicació per a dispositius mòbils, un bon disseny de la interfície d'usuari representa la diferència entre l'èxit i el fracàs. Quan es dissenya la IGU d'aquestes aplicacions, cal tenir en compte una sèrie de factors que les diferencien de les aplicacions d'escriptori:

- La IGU ha de ser completament intuïtiva. En aquestes aplicacions no hi ha lloc per a manuals d'usuari ni llargues aplicacions, i els usuaris perdran l'interès en l'aplicació si no l'entenen després de treballar-hi durant 1 o 2 minuts.
- Cal seguir estrictament les normes estilístiques de la plataforma. Aquest fet està relacionat amb el punt anterior: si l'aplicació té el comportament que l'usuari espera (és a dir, el seu *look and feel* és el mateix que el de la resta d'aplicacions que té instal·lades al seu dispositiu), l'usuari tindrà moltes menys dificultats per saber-la utilitzar.
- L'aplicació ha de ser completament localitzable. El mercat d'aquests dispositius és global i, per tant, cal tenir molt presents els aspectes de localització ja no només a nivell de l'idioma de la IGU, sinó també en altres aspectes com el format de les dates, moneda, etc.
- L'aplicació ha d'estar preparada per passar a segon plànol en qualsevol moment, i recuperar el seu estat anterior quan l'usuari la torna a executar. En aquest tipus de dispositius, hi ha un gran nombre d'events que poden provocar que l'aplicació passi a segon plànol (o sigui apagada pel sistema): l'entrada d'una trucada telefònica, l'esgotament de la bateria, etc. Per tant, quan es dissenya la IGU cal tenir cura en guardar tota la informació que l'usuari ha entrat en aquell mateix moment, per evitar la pèrdua de dades.

A banda d'aquests criteris bàsics de disseny, cal tenir en compte els criteris fonamentals comuns a qualsevol aplicació informàtica, com ara la limitació de la visibilitat, maximització de la consistència, seguir el principi d'agrupació, etc.

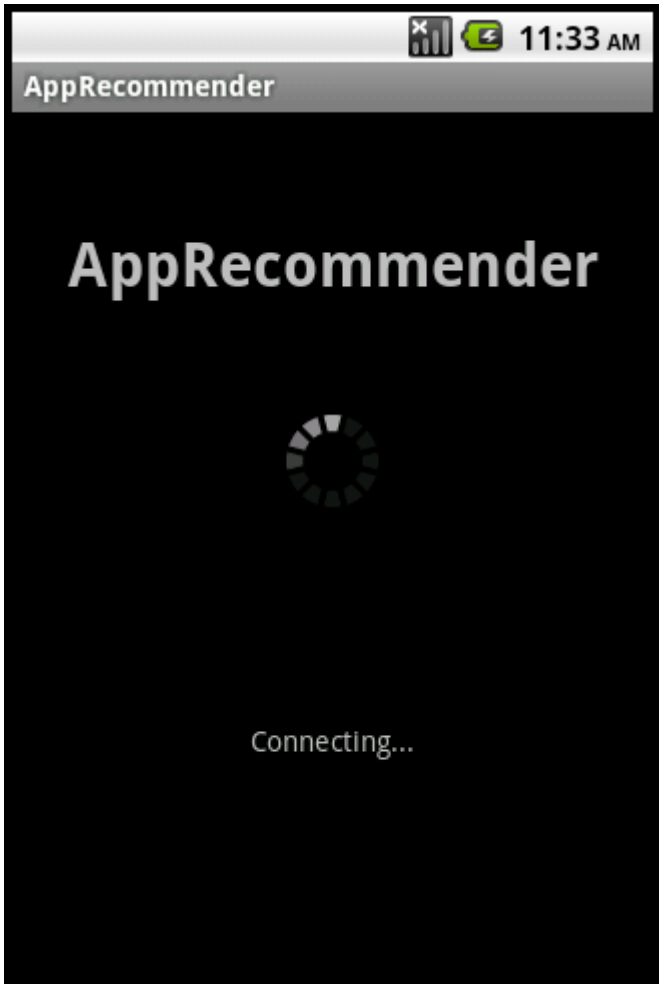
En el disseny de la IGU per a AppRecommender s'ha intentat seguir tots aquests principis. Per fer-ho, en primer lloc s'han dissenyat les pantalles, a partir dels requeriments definits en les fases prèvies, sobre paper. Aquest disseny ha estat contrastat amb altres aplicacions de rellevància dins de la plataforma (Android) per comprovar que és coherent amb el que l'usuari esperarà de l'aplicació. La fase següent ha estat plasmar aquest disseny en un prototip, utilitzant l'SDK d'Android. Aquest prototip serà la base del desenvolupament definitiu del component client.

Pels motius ja exposats anteriorment en aquest document, s'ha decidit de realitzar el desenvolupament nativament en llengua anglesa (és a dir, per defecte la IGU mostra tota la informació en anglès). En la versió definitiva de l'aplicació, aquesta estarà localitzada també al castellà i el català.

A continuació, es mostren les captures de cadascuna de les pantalles (o activitats, seguint l'argot de la plataforma Android).

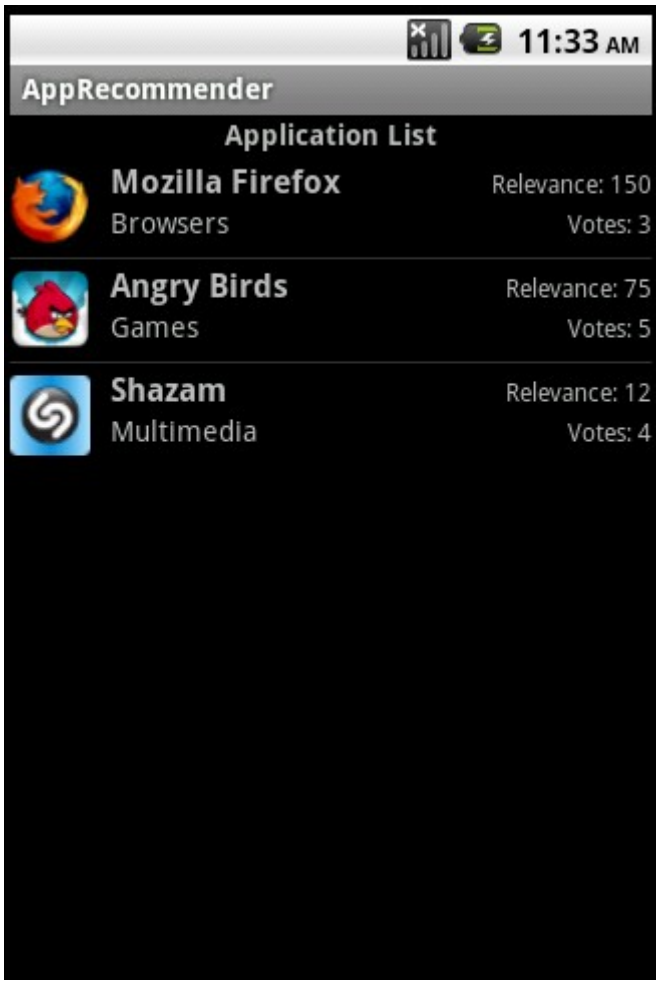
- Pantalla principal:

Aquesta pantalla es mostra quan arrenca l'aplicació en la seva primera execució. La seva utilitat és mantenir informat l'usuari en el procés de transmissió de dades cap al servidor (en el qual es crearà l'usuari a la base de dades i es transmetrà la llista d'aplicacions instal·lades per part d'aquest):



- Llista d'aplicacions recomanades:

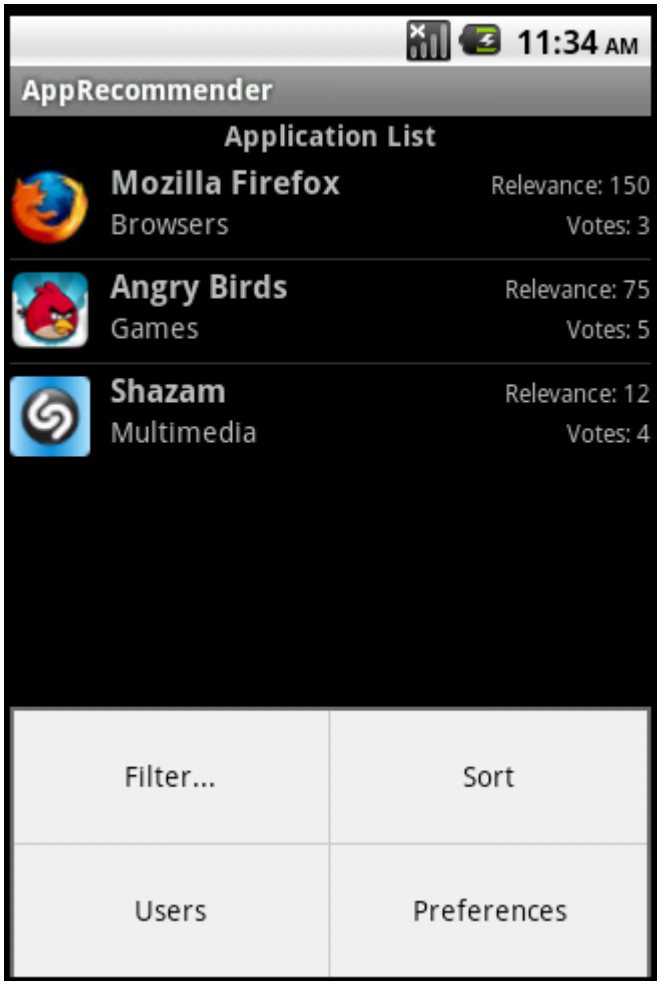
Una vegada finalitzada la transmissió de les dades en la primera execució, i immediatament quan arrenca l'aplicació a partir d'aquell moment, l'usuari obtindrà una llista amb les aplicacions recomanades. Per a cada aplicació es mostra la seva icona, el seu nom, categoria, rellevància (és a dir, puntuació que ha obtingut l'aplicació a partir dels usuaris afins) i vots (número d'usuaris afins que han votat a favor de l'aplicació):



- Llista d'aplicacions recomanades (menú)

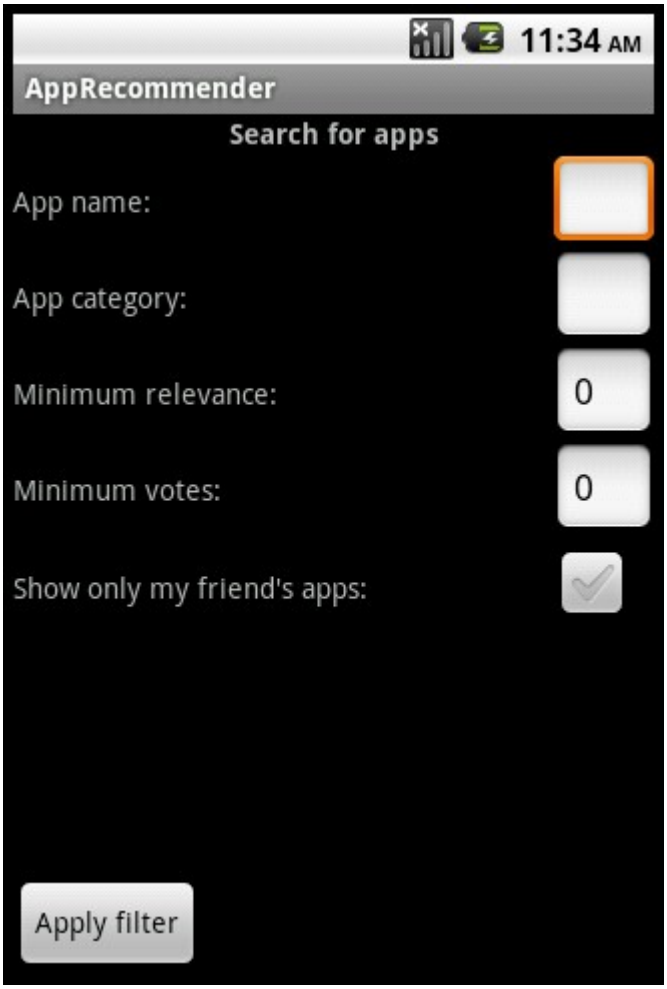
En la pantalla de llista d'aplicacions, l'usuari pot pitjar la tecla MENU per a accedir a un menú desplegable en el que es mostren les següents opcions:

- Filtrar: Permet filtrar la llista per diferents criteris.
- Ordenar: Permet definir l'ordre en el que apareixen les aplicacions.
- Usuaris: Permet obtenir una llista d'usuaris afins.
- Preferències: Permet definir els paràmetres de l'aplicació.



- Cerca d'aplicacions:

Aquesta pantalla es mostra quan l'usuari vol buscar una aplicació. Es pot aplicar un filtre per nom de l'aplicació, categoria, rellevància mínima, quantitat mínima de vots o definir que només es volen visualitzar les aplicacions recomanades pels usuaris afins que estan marcats com a amics.



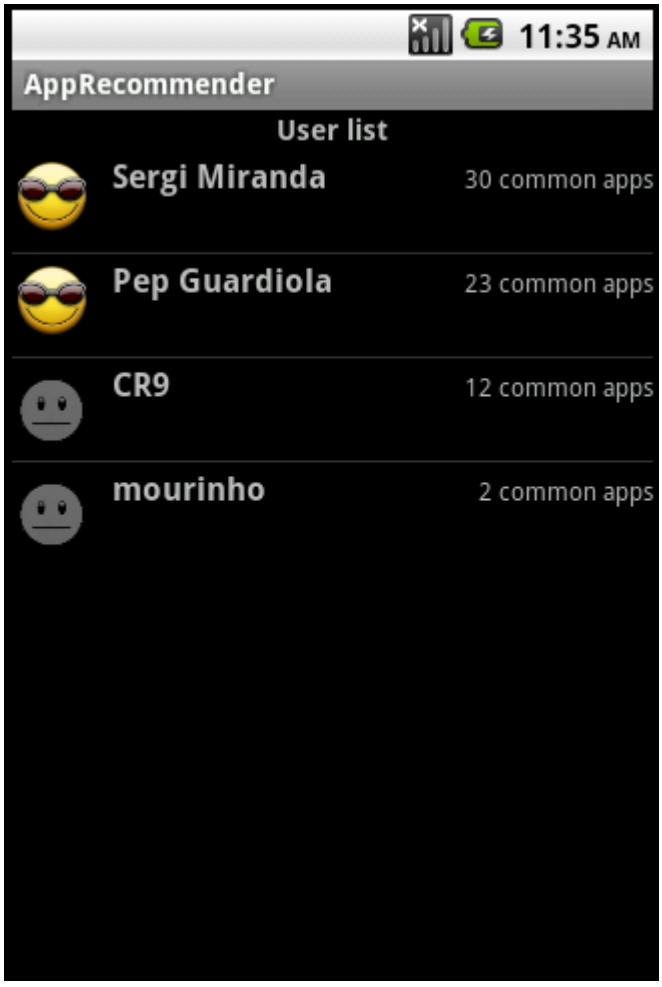
- Ordenar la llista d'aplicacions:

Si l'usuari vol canviar l'ordre de les aplicacions (que per defecte serà en funció de la seva rellevància), podrà utilitzar aquesta pantalla:

- Llista d'usuaris afins:

Aquesta pantalla (accessible mitjançant la opció *Usuaris* del menú de la llista d'aplicacions) mostra una llista amb els usuaris afins. Els usuaris es divideixen en dues categories: amics i usuaris afins. Per defecte, a la llista apareixen primer els amics, i a continuació la resta d'usuaris, ordenats segons l'afinitat (és a dir, segons el nombre d'aplicacions comunes).

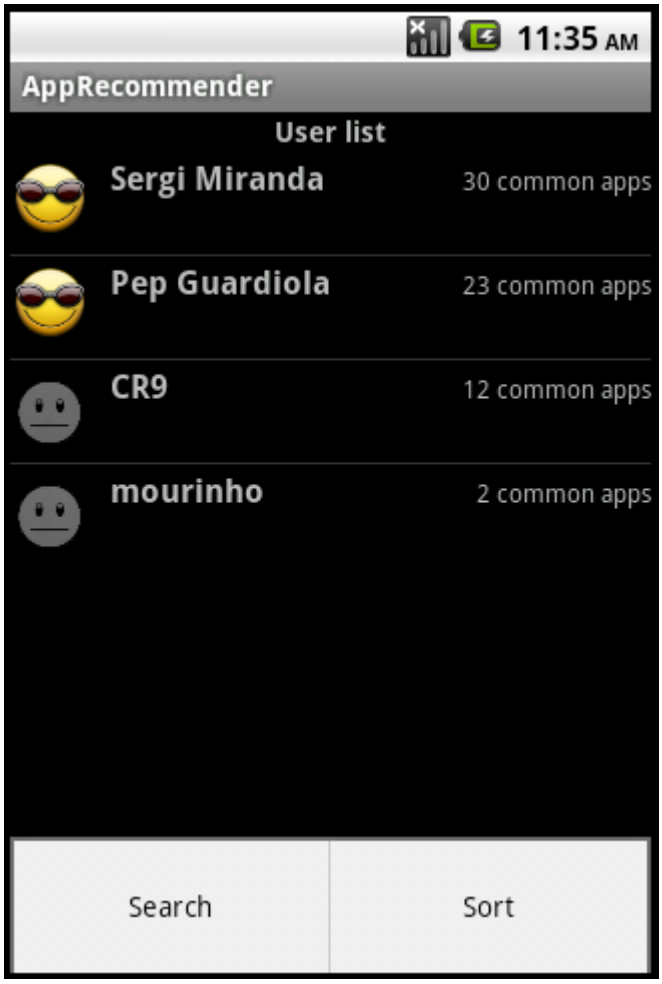
Per a cada usuari es mostra el seu àlias i el nombre d'aplicacions comunes. La icona de l'esquerra de cada usuari indica si l'usuari és amic o és un usuari afí. Pitjant sobre la icona, es canviarà l'estat de l'usuari.



- Llista d'usuaris afins (menú):

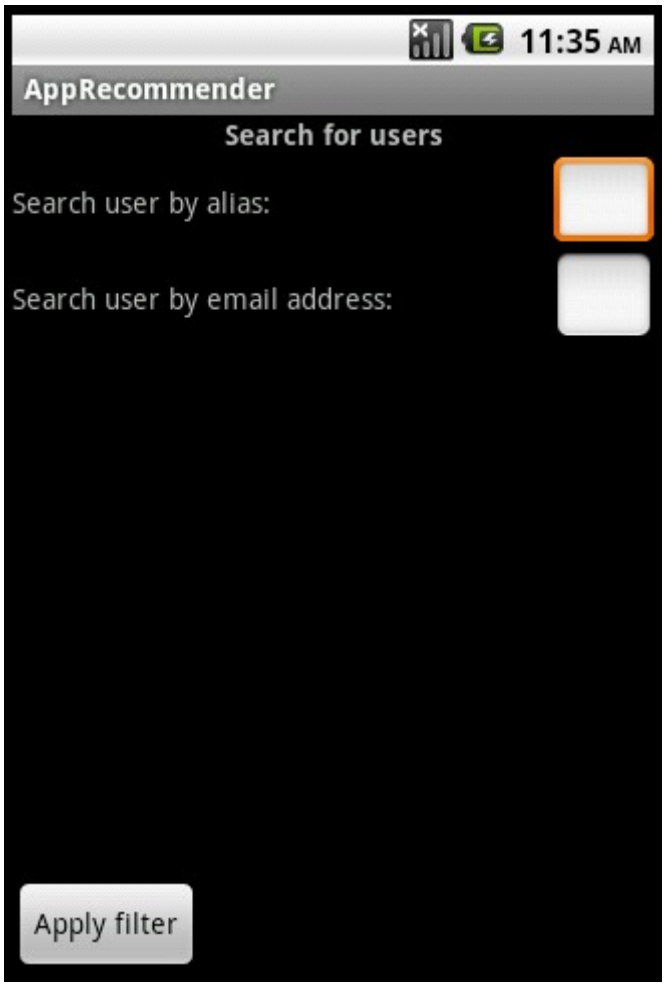
Si l'usuari pitja la tecla MENÚ dins de la llista d'usuaris, se li obrirà un menú desplegable amb les opcions:

- Buscar: Permet filtrar la llista d'usuaris per diferents criteris.
- Ordenar: Permet d'ordenar la llista d'usuaris.



- Cerca d'usuaris:

Aquesta pantalla (accessible mitjançant la opció *Buscar* del menú de la llista d'usuaris) permet definir uns filtres sobre els usuaris afins. Concretament, permet buscar un usuari a partir del seu àlies o a partir de la seva adreça de correu electrònic.



- Ordenar la llista d'usuaris:

Aquesta pantalla (accessible mitjançant la opció Ordenar del menú de la llista d'usuaris) permet definir l'ordre en el que l'usuari vol obtenir la llista d'usuaris (per defecte, es mostren primer els amics, i a continuació la resta d'usuaris, ordenats segons el nombre d'aplicacions comunes):

3.6. Desenvolupament d'un prototip

En el procés de disseny de la IGU, com ja s'ha comentat a l'apartat anterior, s'han dissenyat les pantalles mitjançant l'SDK d'Android. Això ha permès la creació d'un prototip que pot executar-se sobre l'emulador d'Android proporcionat per l'SDK.

Adjunt a aquest document podem trobar aquest prototip. En executar l'aplicació, es mostra la pantalla inicial; fent click al text “Connecting” s'emularà la finalització d'aquest procés i es mostrarà la pantalla amb la llista d'aplicacions recomanades (s'han codificat algunes aplicacions de prova per comprovar-ne l'efecte). Es pot utilitzar la tecla MENÚ de l'emulador per accedir a les altres pantalles.

Les característiques del projecte no han fet necessari realitzar un prototip del component servidor ni de la interfície entre els dos components.

4. Implementació

En aquesta fase realitzarem la implementació del projecte, seguint el disseny desenvolupat en les fases anteriors. Per fer-ho, utilitzarem els llenguatges i eines següents:

- **Component client:**
  El component client es desenvoluparà en Java, mitjançant l'IDE Eclipse, ja que aquest és el mètode recomanat per Google per al desenvolupament d'aplicacions per a Android.

- **Component servidor:**
  El component servidor serà desenvolupat en PHP, un dels llenguatges més populars per al desenvolupament d'aplicacions web. A partir de la versió 5 del llenguatge, aquest va ser redissenyat per tal de millorar les seves capacitats d'orientació a l'objecte, augmentant molt significativament les seves possibilitats per al disseny d'aplicacions complexes i més fàcilment mantenibles. Tant el PHP com el servidor web que utilitzarem (Apache 2, el més utilitzat del mercat) són eines de codi font lliure i multi plataforma. La versió de PHP que utilitzarem és la 5.2.

  Com a SGBD utilitzarem MySQL, en la versió 5. Tot i que darrerament hi ha hagut certa controvèrsia respecte a la continuïtat del projecte d'ençà la seva adquisició per part de Sun i, posteriorment, la compra d'aquesta última companyia per part d'Oracle, MySQL continua sent un dels SGBDs més utilitzats. MySQL té una estructura modular que permet utilitzar diversos *back-ends* per a emmagatzemar la informació, amb diferents possibilitats. Nosaltres utilitzarem InnoDB, que és el més potent d'ells i, de fet, esdevindrà el back-end per defecte a partir de la versió 6 de MySQL.

  La unió d'aquestes tres eines de codi font lliure formen el que s'acostuma a anomenar LAMP (quan es despleguen sobre plataformes Linux) o WAMP (sobre Windows).

4.1. Implementació de la base de dades

A continuació, es mostra l'script de creació de la base de dades i les taules que utilitzarà el component servidor.

El nom que donarem a la base de dades serà AppRecommender, tot i que és possible utilitzar-ne qualsevol altre, canviant la configuració del component servidor.

```
create database AppRecommender;

CREATE TABLE `application` ( `packageName` varchar(100) COLLATE utf8_unicode_ci
NOT NULL, `idCategory` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL, `icon`
blob, `name` varchar(150) COLLATE utf8_unicode_ci DEFAULT NULL, PRIMARY KEY
(`packageName`) ) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

CREATE TABLE `application_category` ( `idCategory` varchar(100) COLLATE
utf8_unicode_ci NOT NULL, PRIMARY KEY (`idCategory`) ) ENGINE=InnoDB DEFAULT
CHARSET=utf8 COLLATE=utf8_unicode_ci;

CREATE TABLE `user` ( `idUser` varchar(150) COLLATE utf8_unicode_ci NOT NULL,
`alias` varchar(150) COLLATE utf8_unicode_ci DEFAULT NULL, `email` varchar(150)
COLLATE utf8_unicode_ci DEFAULT NULL, `creationDate` datetime DEFAULT NULL,
`numInstalledApps` int(11) DEFAULT NULL, PRIMARY KEY (`idUser`) ) ENGINE=InnoDB
DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

CREATE TABLE `user_applications` ( `idUser` varchar(150) COLLATE utf8_unicode_ci
NOT NULL, `packageName` varchar(100) COLLATE utf8_unicode_ci NOT NULL,
`installed` tinyint(1) DEFAULT NULL, `installDate` datetime DEFAULT NULL,
`removalDate` datetime DEFAULT NULL, `likesIt` int(11) DEFAULT NULL, `comments`
text COLLATE utf8_unicode_ci, PRIMARY KEY (`idUser`,`packageName`) )
ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

CREATE TABLE `user_relations` ( `idUser` varchar(150) COLLATE utf8_unicode_ci
NOT NULL, `idRelatedUser` varchar(150) COLLATE utf8_unicode_ci NOT NULL,
`isFriend` int(11) DEFAULT NULL, `numCommonApps` int(11) DEFAULT NULL, PRIMARY
KEY (`idUser`,`idRelatedUser`) ) ENGINE=InnoDB DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;

CREATE TABLE `user_statistics` ( `idUser` varchar(150) COLLATE utf8_unicode_ci
NOT NULL, `packageName` varchar(100) COLLATE utf8_unicode_ci NOT NULL,
`relevance` int(11) DEFAULT NULL, `numLikesIt` int(11) DEFAULT NULL,
`updateDate` datetime DEFAULT NULL, PRIMARY KEY (`idUser`,`packageName`) )
ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

4.2. Implementació del component servidor

La implementació del component servidor s'ha realitzat seguint l'arquitectura desenvolupada en les fases precedents. Tanmateix, durant el procés d'implementació s'han trobat determinades limitacions que han fet necessàries algunes modificacions.

4.2.1. Estructura de directoris:

El component servidor presenta la següent estructura de directoris:

- **class:**
  Inclou totes les classes PHP del component. Es divideix en 3 subdirectoris:
  - **BusinessLogic:**
    Inclou les classes que formen la lògica del negoci del component. Concretament, són les següents:
    - **Application:**
      Modela una aplicació.
    - **ApplicationCategory:**
      Modela una categoria d'aplicacions.
    - **Element:**
      Classe base que encapsula la funcionalitat de lectura i gravació dels models a la base de dades. Totes les altres classes de la lògica del negoci són subclasses d'Element.
      Element proporciona mètodes per llegir un objecte de la base de dades, llegir-ne o escriure'n els atributs i guardar-lo a l'SGBD.
    - **RecommendedApplication:**
      Modela una aplicació recomanada per a un usuari.
    - **RelatedUser:**
      Modela la relació entre dos usuaris.
    - **User:**
      Modela un usuari. Aquesta classe també proporciona la majoria de mètodes que s'utilitzaran per a gestionar la instal·lació/desinstal·lació d'una aplicació per part de l'usuari, càlcul de les estadístiques associades, etc.
    - **UserApplication:**
      Modela una aplicació instal·lada (o desinstal·lada) per un usuari.

- **Integration:**
  La capa d'integració inclou les classes següents:

  - **Database:**
    Encapsula les funcions bàsiques d'accés a l'SGBD: connexió, desconnexió, realització de consultes SQL, etc.

  - **MarketGrabber:**
    Encapsula les funcions d'accés a la API del Market d'Android. Mitjançant aquesta classe es recupera la informació de les aplicacions la primera vegada que un usuari en notifica la instal·lació: descripció de l'aplicació, categoria i icona.

  - **MarketSession:**
    Encapsula una sessió amb l'Android Market. Aquesta classe ha estat desenvolupada per Niklas Nilsson i està disponible a: http://code.google.com/p/android-market-api/.

  - **Parameter:**
    Modela un paràmetre del component servidor. Encapsula els mètodes i funcions necessaris per tal d'accedir a l'arxiu de configuració, llegir-ne els paràmetres i gravar-los.

- **Presentation:**
  El component servidor no disposa d'una interfície d'usuari; per tant, en aquesta capa s'han inclòs les classes necessàries per a la gestió del servei web que actuarà d'interfície amb el component client.

  - **Webservice:**
    Encapsula els mètodes i funcions del servei web. Rep les peticions del component client, gestiona les crides als objectes que han de processar-les i en formata la resposta per tal que aquesta pugui ser processada correctament per part del client.

  - **WebserviceException:**
    Permet transmetre un error al component client. En cas de que es produeixi un problema en una de les operacions del servei web, es transmetrà un objecte WebserviceException indicant-ne els detalls.

  - **WebserviceOperation:**
    El component client esperarà una resposta a tots els mètodes i funcions del servei web. Aquells mètodes que, per les seves característiques, només hagin de retornar un missatge indicant l'èxit o el fracàs de la operació, retornaran un objecte de tipus WebserviceOperation.

- **docs:**
  En aquesta carpeta trobarem la documentació de totes les classes del component servidor, en format PHPDoc. Totes les classes i mètodes han estat documentades (en anglès). PHPDoc processa els comentaris (realitzats amb una sintaxi molt similar a la de Javadoc) i genera una documentació fàcilment llegible.

- **etc:**
  En aquest directori hi ha un únic arxiu, *config.xml*. Aquest arxiu conté els paràmetres de configuració del component servidor, concretament:
  - Paràmetres d'accés a la base de dades: adreça IP de l'SGBD, usuari, contrassenya i nom de la base de dades.
  - Paràmetres d'accés al Market: L'API d'accés al Market necessita d'un nom d'usuari i contrassenya vàlids de Google (s'ha creat un compte específic per a l'aplicació, però en podríem utilitzar un altre qualsevol) i un identificador de dispositiu.

  Els valors de configuració que es proporcionen per defecte haurien de funcionar en qualsevol desplegament del component (en plataformes Linux o Windows), sempre i quan l'usuari d'accés a la base de dades sigui *root* sense contrassenya, i el nom de la base de dades sigui *AppRecommender*.

- **imgs:**
  En aquesta carpeta es guardaran les icones de l'aplicació de manera temporal, quan es realitza un test de l'aplicació (en un entorn de producció, no s'utilitzarà).

- **lib:**
  En aquesta carpeta trobem els arxius de suport que necessita el component servidor per tal de funcionar correctament. En concret:

  - **AppRecommender.lib.php:** Estableix una variable global amb la ruta de la carpeta on es troba instal·lat el component i l'autocàrrega de classes.

  - **proto:** En aquesta carpeta hi ha els arxius de suport necessaris per a accedir a la API del Market d'Android.

- **test:**
  En aquesta carpeta trobarem l'arxiu *test.php* que es pot utilitzar per a realitzar algunes proves sobre l'aplicació (crear/esborrar usuaris i aplicacions, així com consultar els usuaris creats i les seves aplicacions i estadístiques).

- **webservice.php:**
  Aquest és el punt d'entrada al servei web (actua com a controlador). La seva funció és analitzar els paràmetres d'entrada (rebuts en una petició POST per part del component client) i realitzar la crida apropiada a la classe Webservice.

Aquest component ha estat desenvolupat en Linux mitjançant l'IDE Netbeans. Tot i que no hauria d'haver cap problema en obrir el projecte en una altra plataforma i/o IDE, és recomanable fer-ho amb aquest, ja que això ens permetrà *navegar* pel codi font d'una

manera molt més senzilla, gràcies als comentaris en format PHPDoc, que el Netbeans reconeix.

4.2.2. Flux de treball:

Com ja ha estat explicat anteriorment, el component servidor no disposa d'una interfície d'usuari (tret de la pàgina que permet realitzar proves sobre l'aplicació). Per tant, presenta un únic punt d'entrada, que és l'arxiu **webservice.php**.

Per simplicitat, la comunicació entre els components client i servidor s'ha implementat en forma de servei web REST. Aquest estàndard és una mica més simple i lleuger que l'XML-RPC o el SOAP, i per aquest motiu ha estat triat. L'enviament de paràmetres del component client al servidor es realitza mitjançant una petició HTTP POST. L'enviament de la resposta del component servidor al client es realitzarà mitjançant un objecte en format JSON, que és fàcilment codificable i descodificable tant en PHP com en Java. Això fa que aquest format sigui molt adient per a aquesta comunicació.

L'arxiu **webservice.php** (que actua com a controlador) ha de rebre, com a mínim, un paràmetre pel mètode HTTP POST, anomenat **function**. Aquest paràmetre correspondrà a una de les funcions de la classe **Webservice**. Addicionalment, rebrà un nombre indeterminat de paràmetres (també per GET), que correspondran (en l'ordre adient) als paràmetres de la funció que es vol cridar.

El controlador delega en la classe **Webservice** l'execució de la funció. Aquesta, al seu torn, delega en les classes de la lògica del negoci, que li retornen el resultat (normalment, en forma d'un *array* d'objectes de la lògica del negoci o un resultat booleà). Per a evitar malbaratar ample de banda, la classe **Webservice** analitza la resposta i genera *arrays* associatius únicament amb els camps que són necessaris per al component client. Això comporta una reducció de la complexitat dels objectes que es transmeten per part del servidor, i permeten una anàlisi més ràpida d'aquests en el component client.

4.2.3. Conclusions:

El component ha estat desenvolupat seguint una filosofia d'orientació a l'objecte que permet estructurar el projecte d'una manera força racional i fàcil de mantenir. S'han utilitzat les característiques més avançades que proporciona el llenguatge PHP (autocàrrega de classes, mètodes màgics, mètodes estàtics, etc.) per tal de permetre una abstracció al màxim nivell. Tanmateix, en alguns casos ha estat necessari disminuir aquest nivell d'abstracció per tal d'augmentar el rendiment de l'aplicació (per exemple, en la lectura de les aplicacions recomanades per a l'usuari).

El projecte s'entrega lliure d'errors de funcionament fins a on ha estat possible testejar-lo. S'han realitzat proves amb diversos usuaris (tant amb l'emulador d'Android com amb dispositius reals) sense que s'hagi detectat cap error aparent.

La documentació en format PHPDoc (que ha estat realitzada en anglès) proporciona una

visió clara de l'estructura del component. Així mateix, el codi font es troba també comentat en aquells punts que poden presentar més confusió (els comentaris interns del codi han estat realitzats en català).

Pel que fa a l'estat de desenvolupament, s'ha implementat el 100% dels requeriments, tot i que la manca de temps ha impedit afinar alguns aspectes que hauran de ser millorats en versions posteriors del producte (per exemple, suport per a la localització pel que fa a les categories i noms de les aplicacions, ja que ara aquesta informació s'obté únicament en anglès).

4.3. Implementació del component client

El component client ha estat desenvolupat en Java, utilitzant l'IDE Eclipse i l'SDK proporcionat per Google. Per tant, el mètode més recomanable per a *analitzar* el codi font és utilitzar aquest entorn.

Com en el cas del component servidor, s'ha intentat seguir al màxim el disseny realitzat en les fases precedents, tot i que ha estat necessari realitzar algunes modificacions conforme s'ha anat avançant en la implementació.

En tot moment s'ha intentat seguir les recomanacions de bones pràctiques de desenvolupament de Google per a la plataforma (http://developer.android.com/index.html). Per exemple, totes les cadenes es troben al seu arxiu de recursos (es proporcionen traduïdes al castellà i a l'anglès). També s'han creat tasques asíncrones per a la realització de les operacions més feixugues amb el servidor, per tal de no bloquejar la interfície d'usuari mentre s'espera la resposta d'aquest.

4.3.1: Estructura del component

Com s'ha comentat, s'ha intentat mantenir l'estructura de classes dissenyada en les fases anteriors. En part, aquesta estructura és similar a la del component servidor, tot i que, per motius de rendiment i les característiques pròpies de cada plataforma, s'han modificat en alguns aspectes.

Les classes del component són, doncs:

- **Application:**
  Modela una aplicació. A banda dels atributs de l'aplicació (que són equivalents als del component servidor) i els mètodes accessors, proporciona un mètode per a recuperar la icona de l'aplicació del servidor (mitjançant una crida al servei web).

  Aquesta classe té un atribut estàtic de tipus HashMap que guarda una llista de les aplicacions que s'han rebut del servidor (en una mateixa execució de l'aplicació). D'aquesta manera, no és necessari llegir dues vegades la informació de la mateixa aplicació (especialment la icona), sinó que aquesta informació es reutilitza.

- **AppRecommender:**
  Activitat principal de l'aplicació. Aquesta classe es limita a comprovar si l'usuari ha estat creat prèviament i, en cas negatiu, llança l'activitat **UserPreferences** per tal de demanar a l'usuari un àlies i la seva adreça d'email, procedint tot seguit a la transmissió de les dades necessàries al servidor per tal de crear l'usuari.
  En cas de que l'usuari ja hagi estat creat prèviament (és a dir, no es tracta de la primera execució de l'aplicació), la classe AppRecommender simplement llança l'activitat **RecommendedAppList**, que és l'encarregada de mostrar la llista d'aplicacions recomanades per a l'usuari.

- **AppRecommenderService:**
  Aquesta classe (derivada de BroadcastReceiver) és un servei d'Android que està vinculat als *intents* PACKAGE_ADDED i PACKAGE_REMOVED. D'aquesta manera, cada vegada que un usuari instal·la o desinstal·la una aplicació al seu dispositiu, la classe AppRecommenderService rep una notificació. A partir de les dades de la notificació (bàsicament, amb el nom del paquet instal·lat), aquesta classe realitza una petició al servei web del component servidor (mitjançant REST) per tal de notificar-li la instal·lació o desinstal·lació de l'aplicació i, d'aquesta manera, procedir al càlcul de les estadístiques.

- **AppSearch:**
  Aquesta activitat mostra una pantalla a l'usuari per tal que aquest pugui definir uns criteris de filtre sobre la llista d'aplicacions recomanades.

- **AppSort:**
  Aquesta activitat mostra una pantalla a l'usuari per tal que aquest pugui definir uns criteris d'ordenació sobre la llista d'aplicacions recomanades.

- **Category:**
  Modela una categoria d'aplicacions.

- **RecommendedApplication:**
  Modela una aplicació recomanada per a un usuari. Tot i que el disseny inicial definia aquesta classe com a derivada d'Application, finalment s'ha decidit crear una relació de dependència entre ambdues classes, ja que això permet una optimització del rendiment (sobretot pel que fa a la descàrrega de la icona de l'aplicació).

- **RecommendedAppList:**
  Aquesta és l'activitat més important del component, ja que és l'encarregada de mostrar la llista d'aplicacions recomanades.
  Aquesta classe recupera la llista d'aplicacions del servidor (aplicant els filtres i criteris d'ordenació definits per l'usuari) i gestiona la visualització de les dades en pantalla. També crea el menú (al qual es pot accedir pitjant el botó MENÚ del terminal) i gestiona la resposta a les accions de l'usuari (tant quan selecciona una opció del menú com quan pitja sobre una aplicació de la llista).

- **RelatedUser:**
  Modela la relació entre l'usuari del terminal i un altre usuari. És una classe derivada d'User. Com en el cas d'Application, s'ha definit un atribut de tipus HashMap que permet la reutilització de les dades (per evitar realitzar peticions innecessàries al servidor).

- **RequestMethod:**
  Enumeració que permet definir quin mètode HTTP s'utilitzarà per a la petició REST al servidor.

- **RequestResponseType:**
  Enumeració que permet definir quin tipus de resposta s'espera del servidor: STRING (quan el servidor ha de retornar un objecte JSON) o DRAWABLE (quan el servidor ha de retornar una imatge).

- **RestClient:**
  Encapsula els mètodes de comunicació amb el servidor (mitjançant REST). La base del codi d'aquesta classe ha estat obtingut de:
  http://lukencode.com/2010/04/27/calling-web-services-in-android-using-httpclient/
  Sobre aquest codi, s'han realitzat modificacions per tal de permetre la recepció d'imatges del servidor.

- **User:**
  Modela un usuari. Com en el cas d'Application, s'ha definit un atribut de tipus HashMap que permet la reutilització de les dades (per evitar realitzar peticions innecessàries al servidor).

- **UserList:**
  Aquesta activitat és l'encarregada de gestionar la visualització de la llista d'usuaris relacionats. S'encarrega de recuperar els usuaris relacionats del servidor, crear la llista i realitzar la presentació de les dades en pantalla. També és l'encarregada de crear els menús (un, contextual, que és accessible fent un *click llarg* sobre un usuari, i un altre que és accessible prement la tecla MENÚ del terminal), així com cridar a les activitats que permeten buscar usuaris i canviar l'ordre de presentació de les dades.

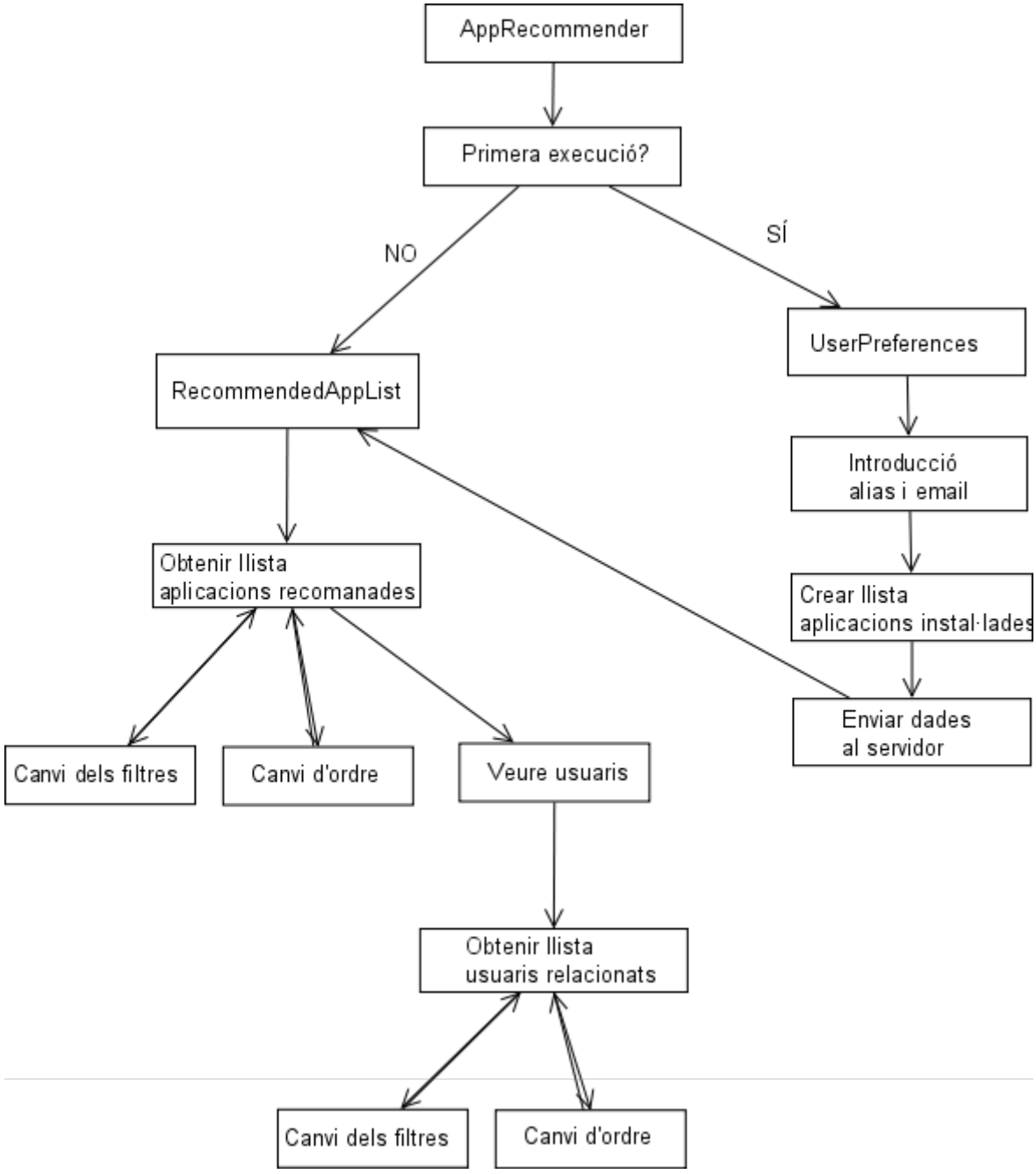
- **UserPreferences:**
  Aquesta activitat té dues missions:
  - Creació de l'usuari:
    En la primera execució de l'aplicació, aquesta activitat mostra a l'usuari una pantalla de benvinguda on pot introduir un àlies i la seva adreça d'email. A continuació, llegeix la llista d'aplicacions instal·lades al terminal i transmet tota aquesta informació al component servidor. Una vegada l'usuari ha estat creat amb èxit, inicia l'activitat **RecommendedAppList** per mostrar a l'usuari la llista d'aplicacions recomanades.
  - Modificació de les dades de l'usuari:
    Mitjançant la opció **Preferences** del menú, l'usuari pot modificar les seves dades (àlies i email). Aquesta classe s'encarrega de mostrar la pantalla on l'usuari pot fer-ho, i gestionar la petició al servidor per tal de guardar els canvis.

- **UserSearch:**
  Aquesta activitat mostra una pantalla on l'usuari pot introduir uns criteris de cerca (per àlies i adreça d'email) per tal de buscar usuaris (dins de la llista d'usuaris relacionats).

- **UserSort:**
  Aquesta activitat permet canviar l'ordre en el que es presenten els usuaris relacionats.

4.3.2: Flux d'execució

A continuació, es mostra el diagrama simplificat d'execució del component client:

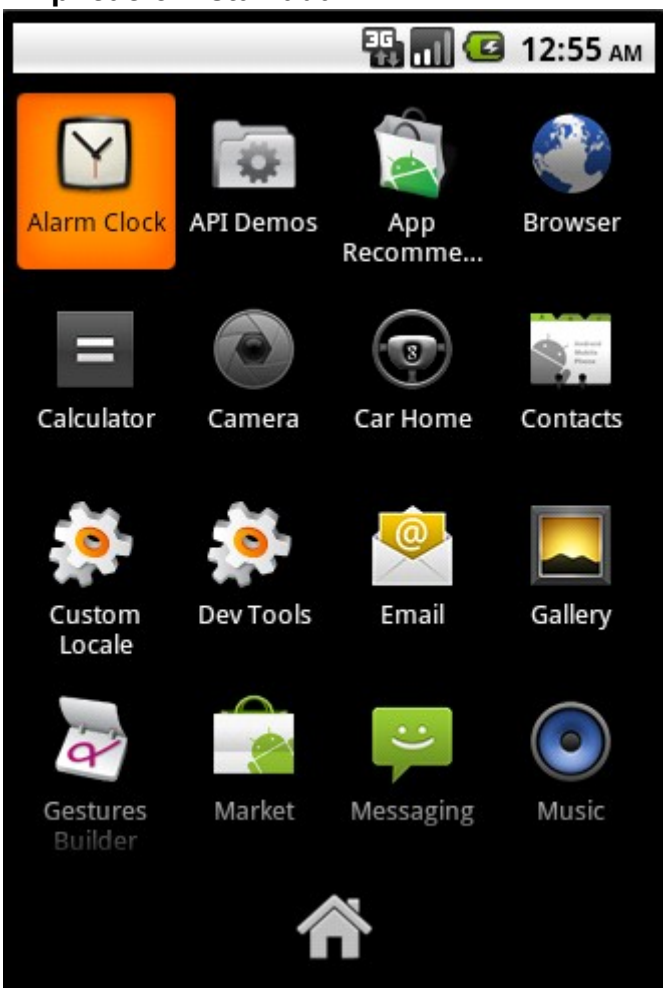


4.3.3: Proves d'execució

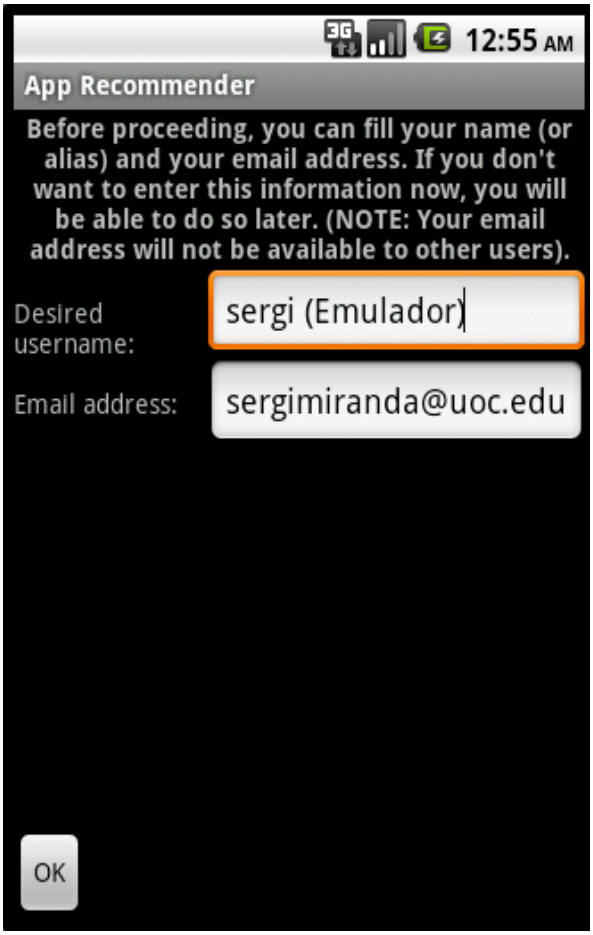
El component ha estat provat tant amb l'emulador d'Android proporcionat amb l'SDK com en diversos dispositius reals (concretament, amb un Nexus One, un Nexus S i un HTC Desire).

Tot seguit, es mostren les captures de pantalla de l'aplicació funcionant sobre l'emulador.

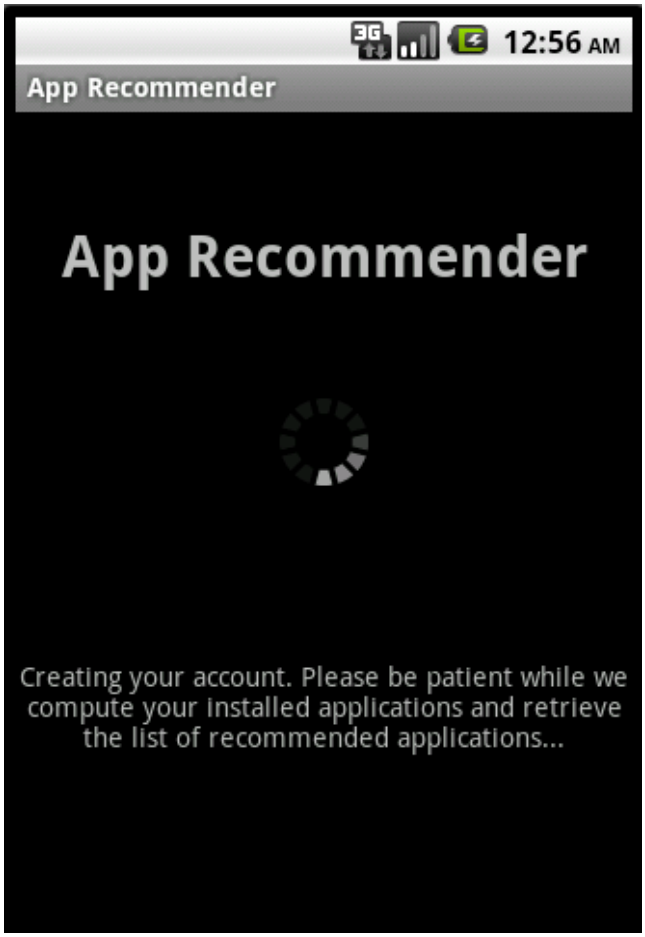
1. Aplicació instal·lada:



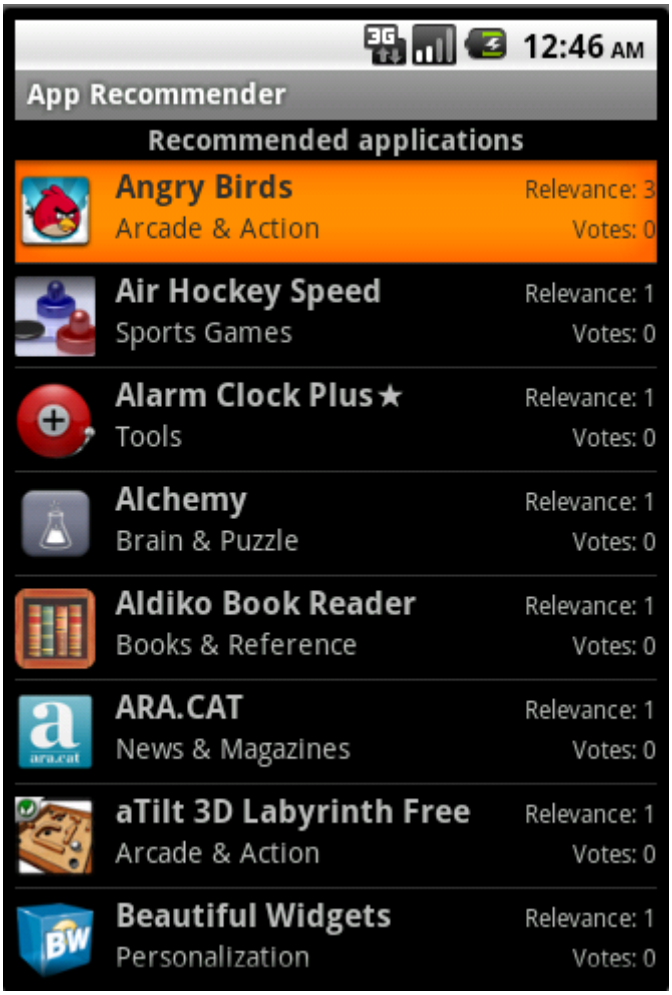
2. **Primera execució:**



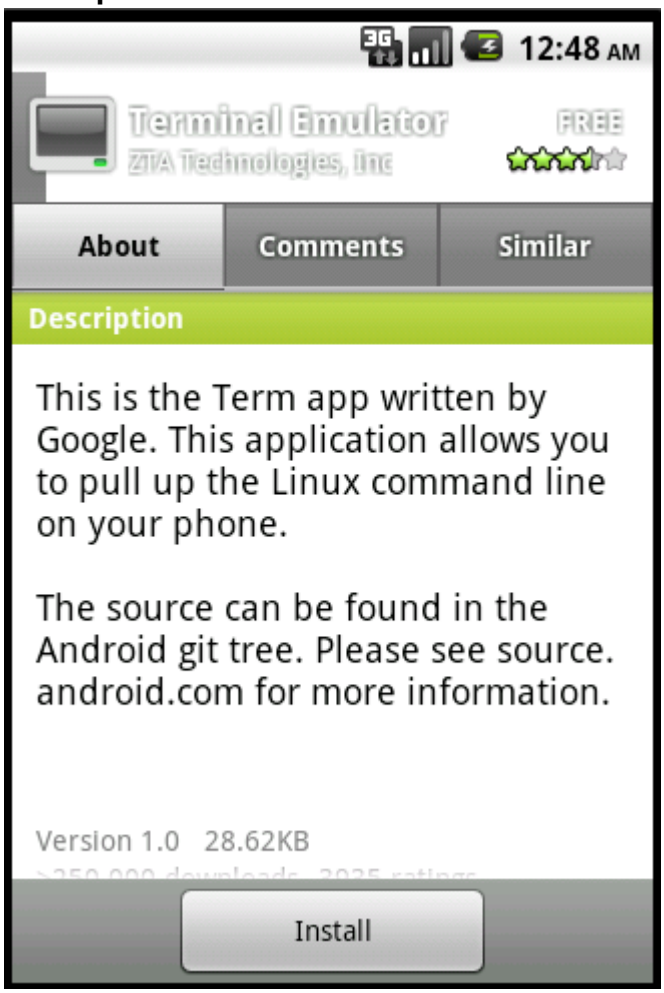
3. **Creació de l'usuari:**



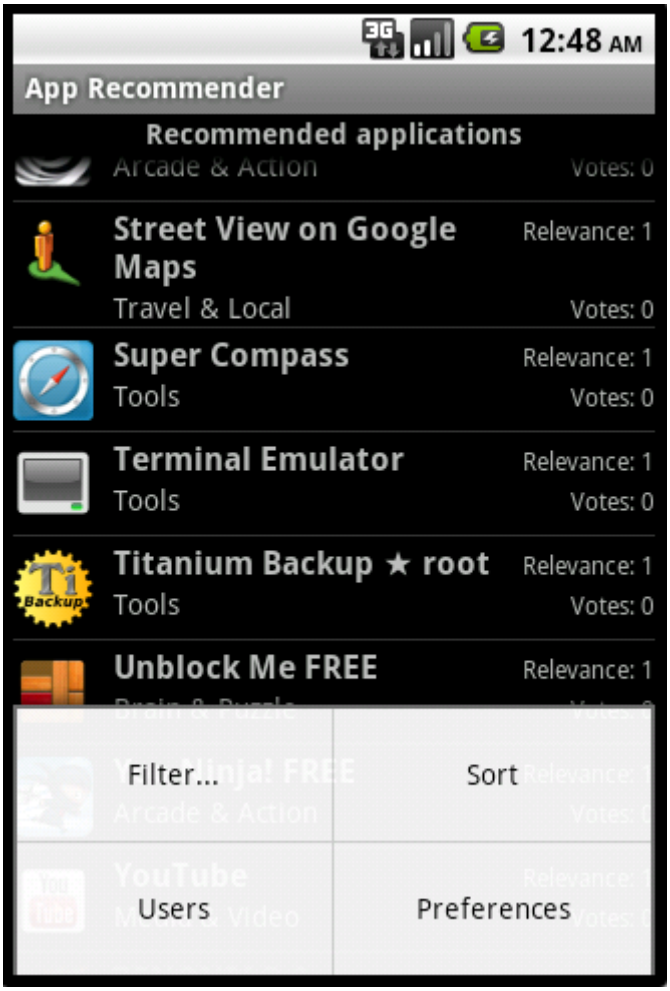
4. Llista d'aplicacions recomanades:



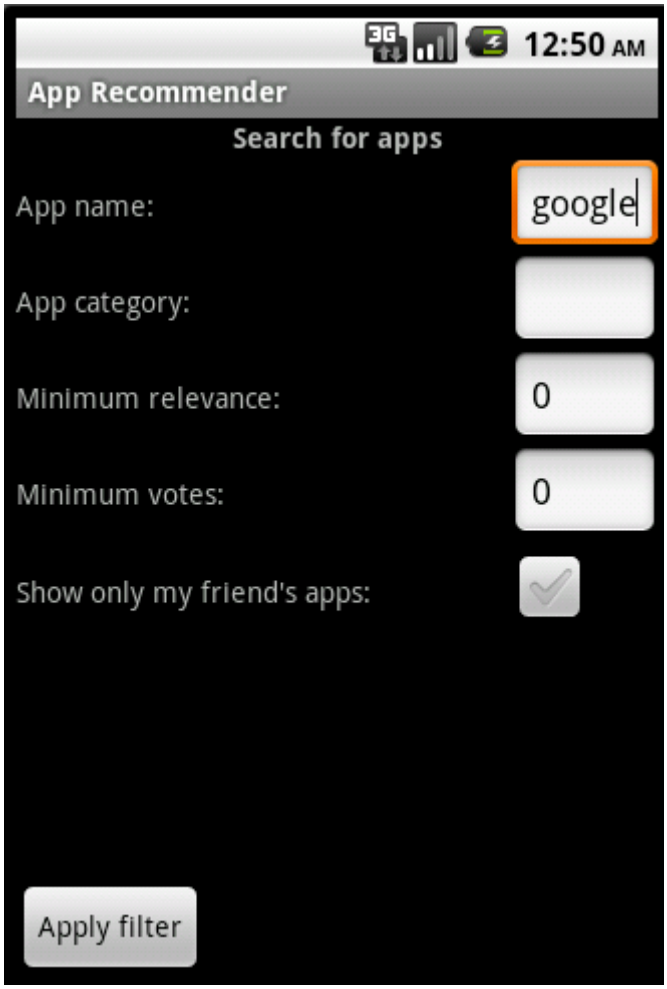
5. Visualització de la informació d'una aplicació al Market en fer-hi click:



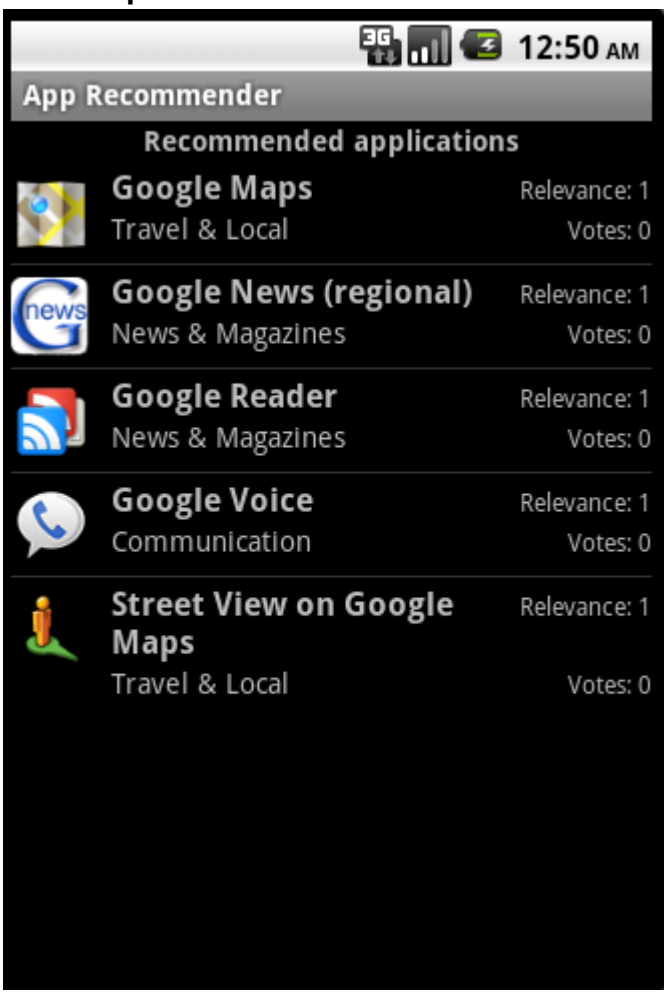
6. Menú de la llista d'aplicacions:



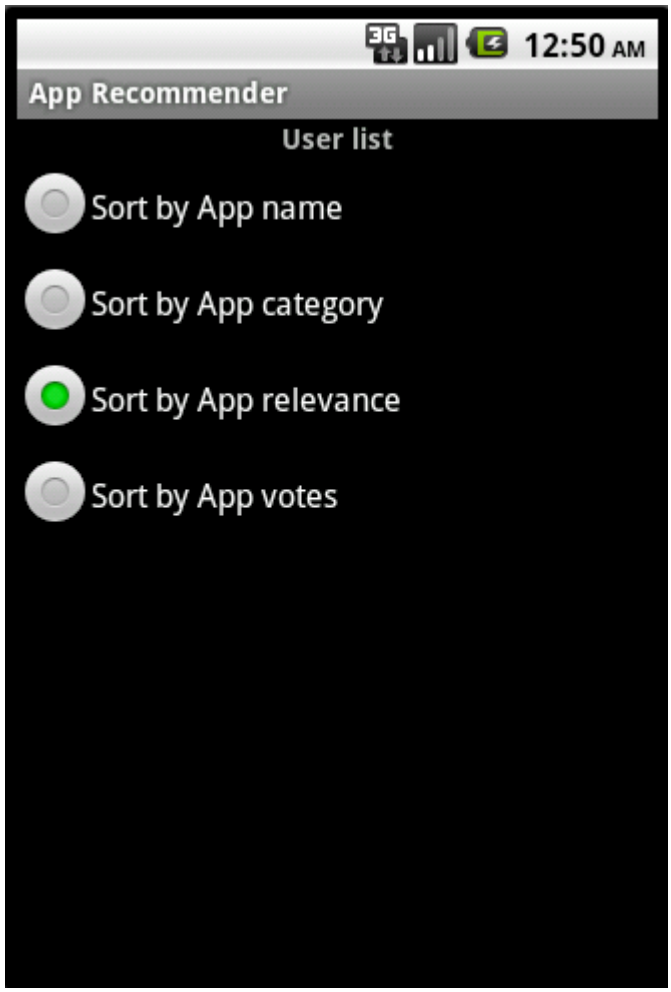
7. Definició dels criteris de cerca:



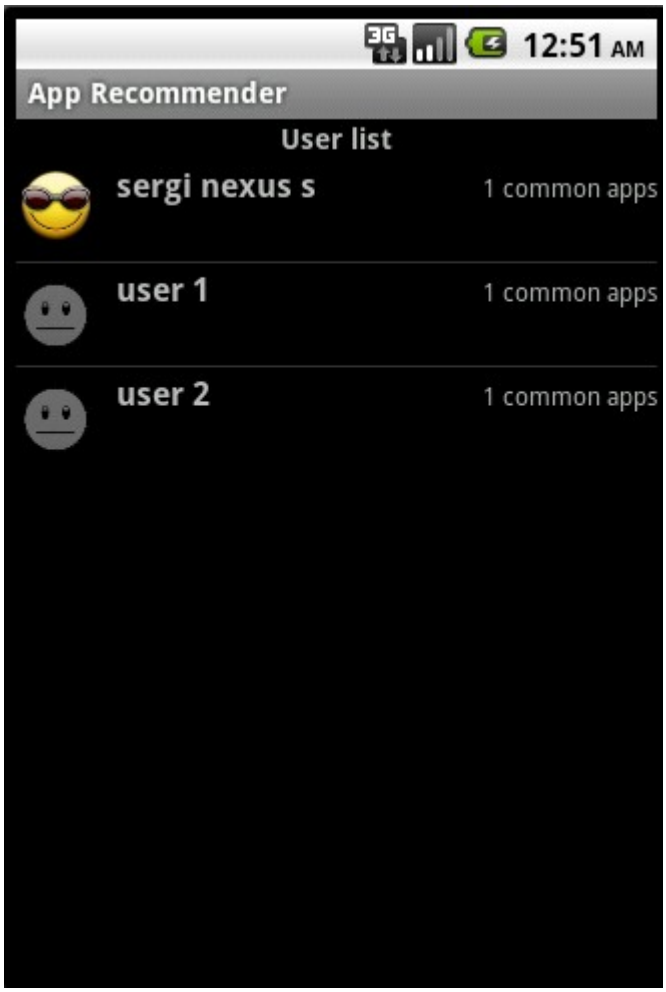
8. Filtres aplicats sobre la llista d'aplicacions:



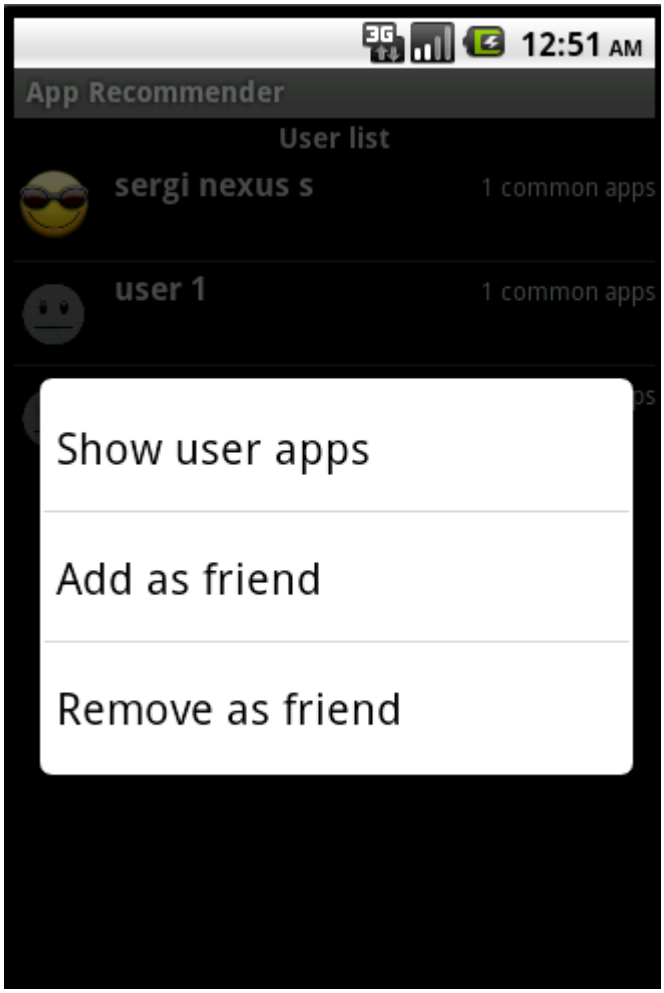
9. Definició dels criteris d'ordenació:



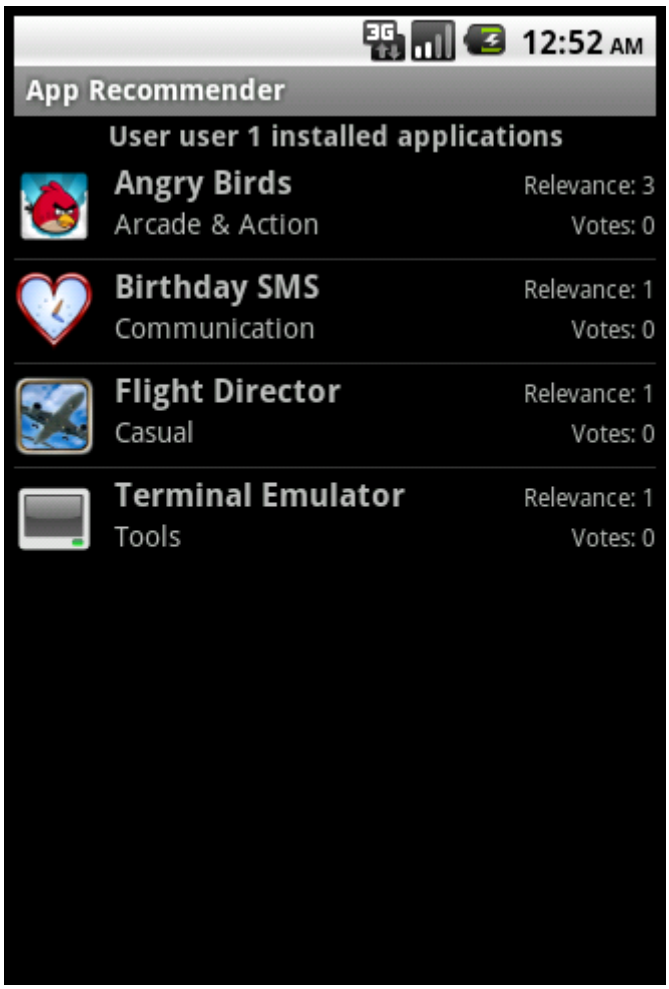
10. Llista d'usuaris relacionats:



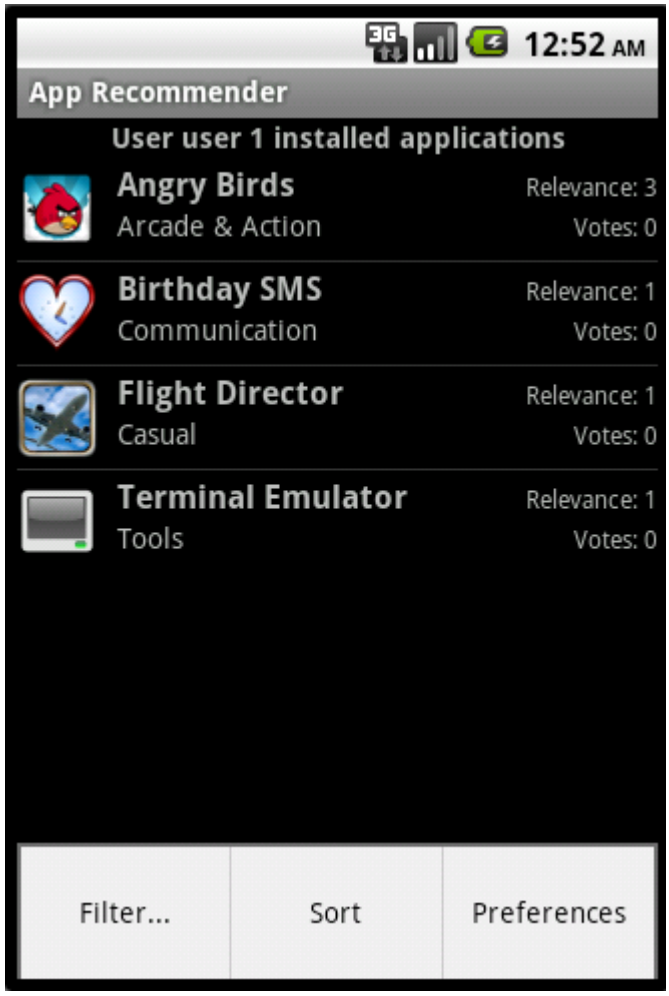
11. Menú contextual a la llista d'usuaris:



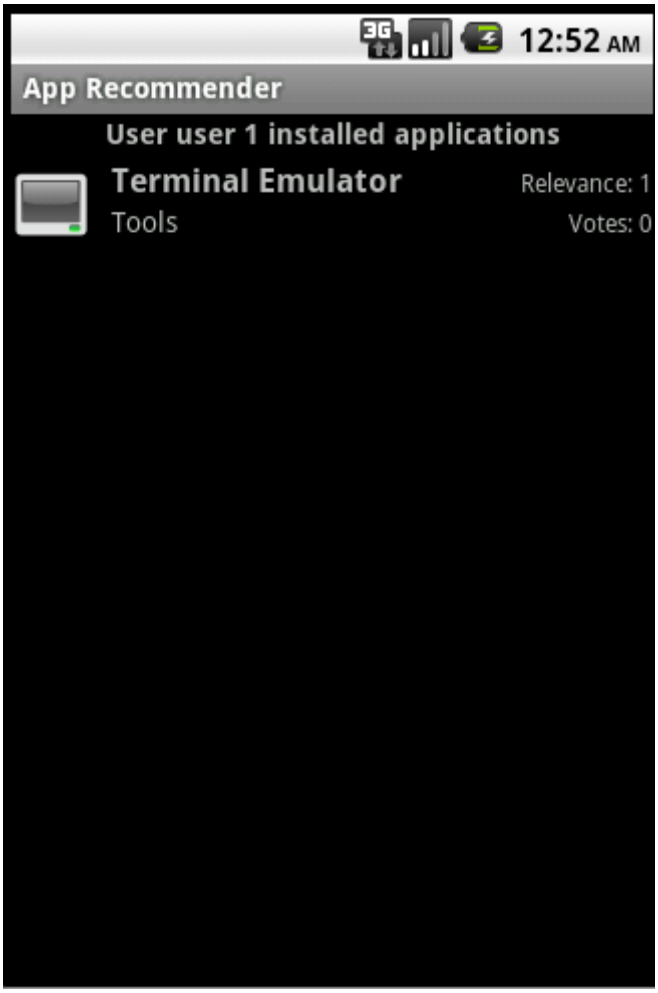
12. Llista de les aplicacions instal·lades d'un usuari:



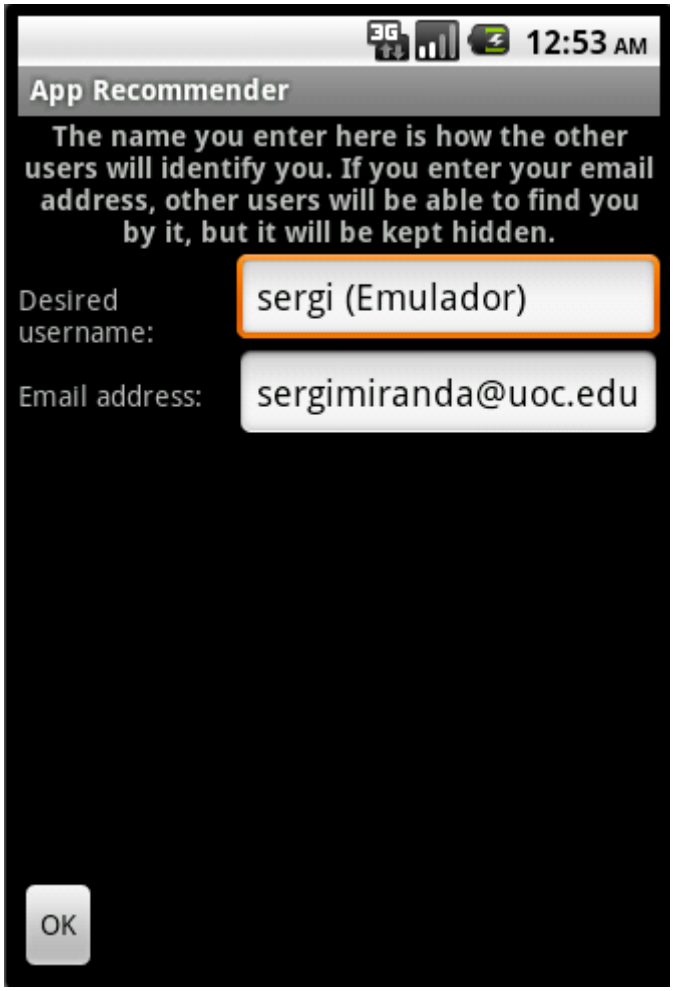
13. Menú a la llista d'aplicacions instal·lades d'un usuari:



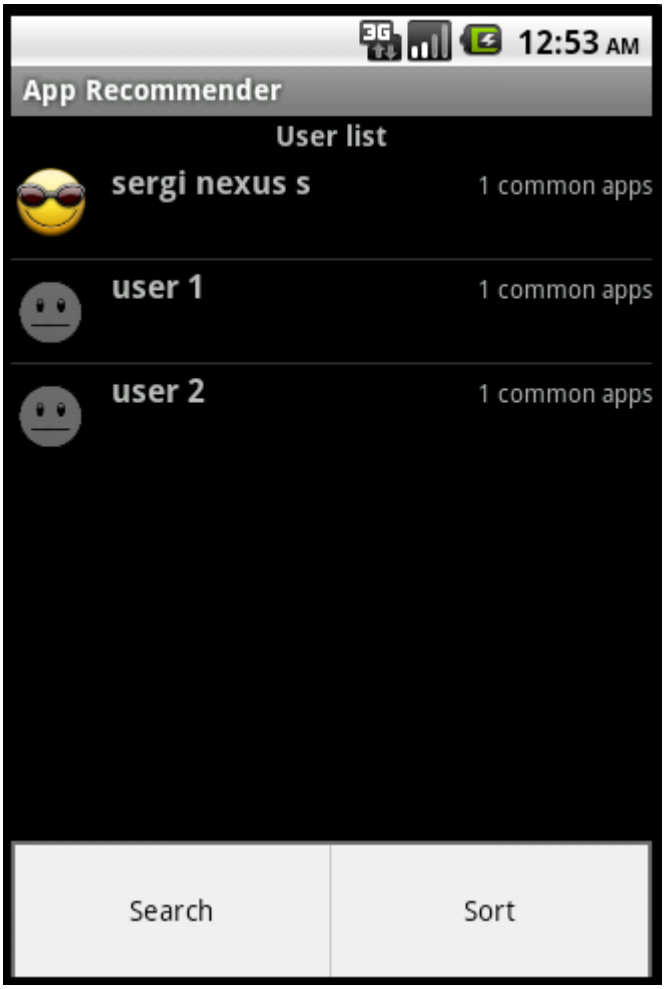
14. Filtrant a la llista d'aplicacions instal·lades d'un usuari:



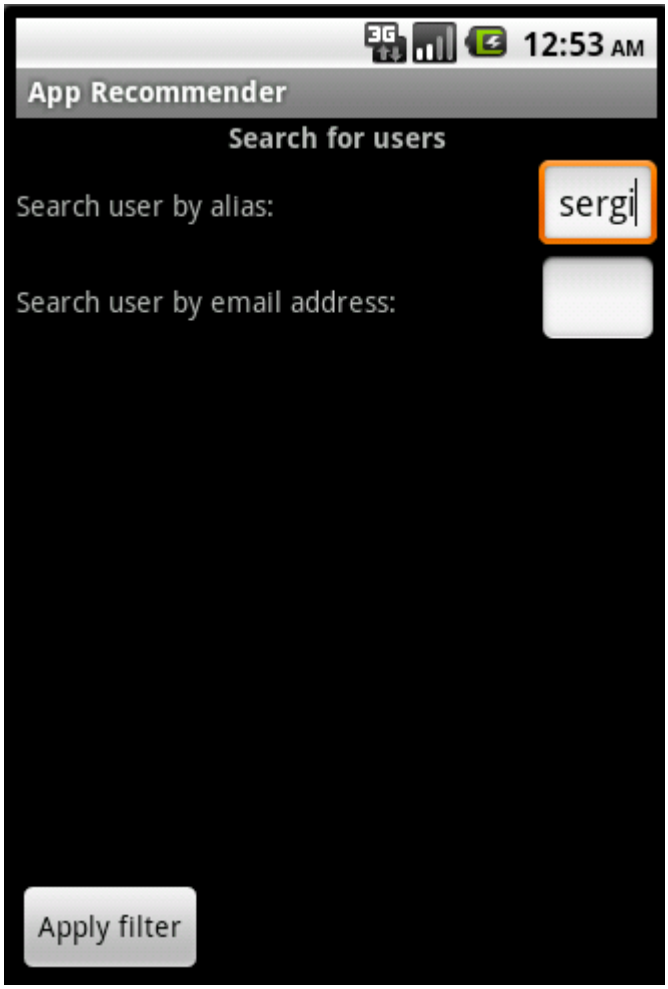
15. Pantalla de preferències:



16. Menú de la llista d'usuaris:



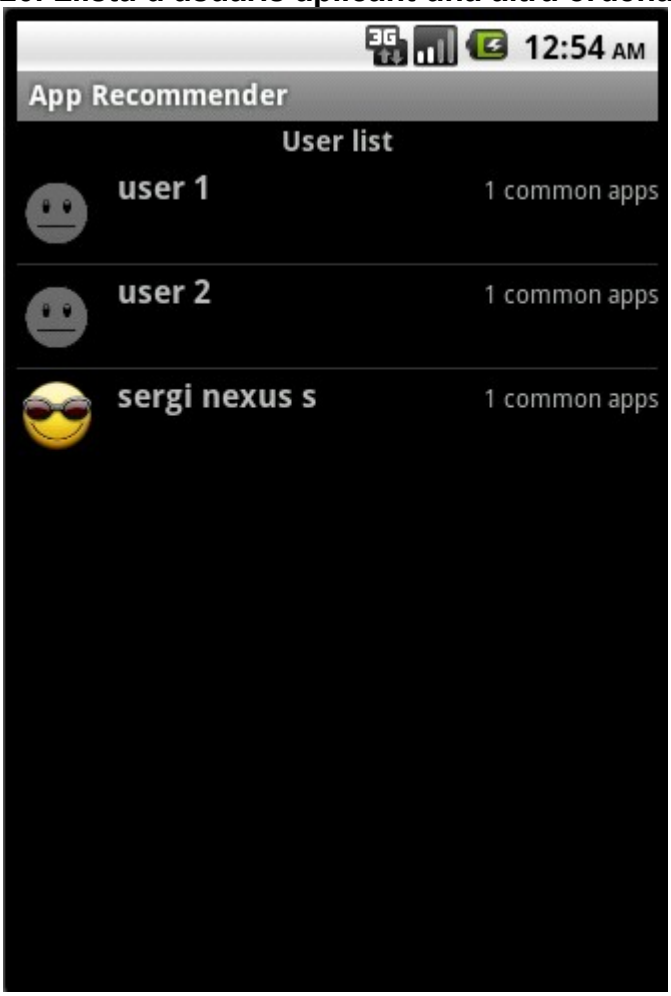
17. Definició dels criteris de cerca d'usuaris:



18. Llista d'usuaris amb un filtre per nom aplicat:

19. Definició dels criteris d'ordenació de la llista d'usuaris:

20. Llista d'usuaris aplicant una altra ordenació de resultats:



4.3.4. Conclusions

Tant el llenguatge (Java) com la pròpia arquitectura de la API d'Android estan fortament orientats a l'objecte i a l'aplicació de bones pràctiques de desenvolupament. En tot moment s'ha procurat seguir aquestes bones pràctiques, així com les directrius de desenvolupament de la plataforma.

El producte s'entrega sense errors aparents, tot i que la fragmentació del mercat de dispositius Android (on conviuen terminals amb característiques molt diferents tant pel que fa al seu maquinari com a la versió del seu sistema operatiu) i la manca de temps no permeten garantir un funcionament perfecte en tots ells.

Com s'ha comentat anteriorment, les proves s'han realitzat principalment sobre l'emulador, però també en dispositius reals, tot i que tots ells tenen unes característiques de maquinari molt similars (pel que fa a CPU i dimensions de la pantalla) i tenen una versió avançada del sistema operatiu (2.2 i 2.3). La fase següent del projecte ha de permetre realitzar una bateria de tests més àmplia, amb versions anteriors de la plataforma i altres dimensions de pantalla.

Pel que fa a l'estat de desenvolupament del projecte, s'ha acomplert la pràctica totalitat dels requisits, tret de la opció de votar les aplicacions (on un usuari podrà escollir una de les aplicacions que té instal·lades i marcar-la conforme *li agrada*). Això permetrà d'afinar encara més els resultats de la cerca d'aplicacions recomanades, donant preferència a aquelles que tenen un nombre més gran de vots. Tot i que la base de dades ha estat dissenyada per tal de suportar aquesta funcionalitat, i tant el component servidor com el component client s'intercanvien aquesta informació, la manca de temps ha impedit el desenvolupament de l'activitat que hauria de permetre a l'usuari visualitzar les seves aplicacions instal·lades i modificar l'estat del seu vot (positiu o negatiu).

5. Proves

Tot i que al llarg del procés d'implementació s'han anat realitzant proves tant sobre els components individuals (servidor i client) com sobre tot el conjunt, en aquesta fase aprofundirem en aquest aspecte, realitzant una bateria de tests que ens permeti assegurar que la implementació realitzada acompleix tots els requisits establerts a l'abast del projecte.

El primer pas per a la realització de les proves és realitzar el desplegament de la solució en un entorn controlat. En els següents apartats detallarem el procés de preparació d'aquest entorn de proves (requisits i instal·lació del programari necessari, incloent el propi producte).

5.1. Descripció de l'entorn i els lliurables

El desenvolupament de l'aplicació s'ha realitzat en entorn Linux, tot i que les característiques de l'aplicació fan que sigui perfectament possible desplegar-lo en un entorn Windows o OSX.

- Lliurables:

El producte resultant d'aquest projecte són dues aplicacions complementàries: el component servidor (desenvolupat en PHP i que funciona sota una plataforma LAMP o WAMP) i el component client (desenvolupat en JAVA i que funciona en terminals Android).

Juntament amb aquest document, es lliura un arxiu comprimit que conté les següents carpetes:

- **Client:**

  En aquesta carpeta trobarem el lliurable del component client. Concretament, hi trobarem el codi font del projecte Eclipse, així com el producte compilat, preparat per a ser instal·lat en qualsevol dispositiu amb Android:

  - **AppRecommender.apk:**
    Aquest és el paquet compilat de l'aplicació.

  - **AppRecommender:**
    Aquesta carpeta conté el codi font del component client. Donat que ha estat desenvolupat amb Eclipse, cal copiar-la a la carpeta *workspace* de l'usuari i obrir el projecte amb aquest IDE.
    Dins d'aquesta carpeta hi trobarem una subcarpeta **docs**, on hi trobarem la documentació generada amb Javadoc de totes les classes del projecte.

- **Servidor:**

  En aquesta carpeta hi trobarem el lliurable corresponent al component servidor. Concretament:

  - **AppRecommender.sql:**
    Script de creació de la base de dades. Aquest script conté dades de proves, amb diversos usuaris, aplicacions, etc.

  - **AppRecommender:**
    Aquesta carpeta conté el codi font del component servidor. Per tal de desplegar-lo, només cal copiar-lo a la carpeta arrel del nostre servidor Apache.

5.2. Preparació de l'emulador d'Android

Les característiques del projecte fan que, per tal de realitzar proves sobre l'emulador, sigui necessari realitzar una preparació prèvia d'aquest per tal de poder executar l'Android Market (que, per defecte, no està instal·lat a les imatges de l'emulador proporcionades a l'SDK d'Android).

El procediment que s'exposa a continuació ha estat obtingut de "http://www.tech-recipes.com/rx/10004/accessing-android-market-from-android-sdk/"; m'agradaria agrair al seu autor, Lê Hoàng, la seva tasca en la preparació d'aquestes instruccions, que m'han estat de gran utilitat.

Suposarem que ja tenim instal·lat l'SDK d'Android i hem creat un nou emulador amb la versió 2.2 de l'SDK. Anomenarem "Android_2.2" a aquest emulador.

Hem de tenir en compte que, a les instruccions que es detallen a continuació, pot ser necessari canviar la ruta de la carpeta on hem instal·lat l'SDK d'Android. Partirem de la base de que aquesta carpeta és:

```
/opt/android-sdk-linux_x86/platform-tools/
```

En cas de que es tracti d'una altra carpeta (per exemple, si estem realitzant les proves en entorn Windows), només caldrà canviar les instruccions per tal de reflectir la carpeta adient.

1. Necessitarem els arxius **Vending.apk** i **GoogleServicesFramework.apk** (que es troben a la carpeta Client\Emulador de l'arxiu adjunt a aquest document). Copiarem aquests dos arxius a **/opt/android-sdk-linux_x86/platform-tools/**.

2. Des de la línia de comandes, canviem a la carpeta "platform-tools" de l'SDK. Per exemple:

   ```
   cd /opt/android-sdk-linux_x86/platform-tools/
   ```

3. Arrenquem l'emulador amb la següent instrucció (pot ser necessari canviar el nom de l'emulador si no hem utilitzat "Android_2.2":

   ```
   ./emulator -avd Android_2.2 -partition-size 100
   ```

4. Després d'uns segons, ens arrencarà l'emulador. En general, el número que apareixerà a la barra de títol serà 5554; en cas de que sigui un altre número, caldrà ajustar la instrucció següent al nombre que ens hagi aparegut.

   Obrim un nou terminal, canviem de nou a la carpeta "platform-tools" de l'SDK i executem la següent instrucció:

   ```
   ./adb -s emulator-5554 shell
   ```

5. Ara estarem en un shell de l'emulador. Escriurem les següents instruccions per tal d'obtenir permisos d'escriptura sobre la carpeta /system:

```
mount -o remount,rw -t yaffs2 /dev/block/mtdblock0 /system
chmod 777 /system/app
exit
```

6. Ara copiarem els arxius Vending.apk i GoogleServicesFramework.apk a l'emulador:
```
./adb -s emulator-5554 push Vending.apk /system/app/.
./adb -s emulator-5554 push GoogleServicesFramework.apk /system/app/.
```

7. Esborrem l'arxiu SdkSetup.apk de la carpeta /system/app de l'emulador:
```
adb shell rm /system/app/SdkSetup.apk
```

8. Ara podem tancar l'emulador. Anem a la nostra carpeta personal. Dins trobarem una carpeta **.android** (comença amb un punt), amb una subcarpeta **avd**. Dins d'aquesta segona carpeta en trobarem una anomenada **Android_2.2.avd** (el nom serà diferent si el nostre emulador no s'anomena "Android_2.2"). En aquesta carpeta trobarem dos arxius que ens cal eliminar: **userdata-qemu.img** i **cache.img**. Hem d'esborrar aquests dos arxius.

9. Ja hem finalitzat. Podem iniciar l'emulador normalment i trobarem el Market disponible per a utilitzar-lo (serà necessari utilitzar un compte de Google per tal d'accedir-hi).

Tot i que aquest procés permet d'executar el Market a l'emulador, i instal·lar i desinstal·lar paquets, en el procés d'implementació he trobat dos problemes:

1. Moltes de les aplicacions no estan disponibles. Probablement això és degut a un conjunt de factors (versió de l'API, configuració regional, etc.), que limita el nombre d'aplicacions que apareixen.
2. En instal·lar (i també en desinstal·lar) aplicacions, el Market dóna un error i es tanca. Tanmateix, això succeeix després de que s'hagi fet el procés, així que no suposa cap problema per tal de provar el funcionament de l'aplicació.

5.3. Desplegament del component servidor

Per simplificar el procés de prova, he desplegat el component servidor, en la versió actual, en un servidor a Internet propietat de la meva empresa. L'adreça d'aquest servidor és:

http://apprecommender.icssolution.com/AppRecommender/

El component client està compilat per comunicar-se amb aquest servidor; per tant, no és necessari desplegar el component servidor en local per tal de provar el funcionament del producte.

En cas de desitjar realitzar el desplegament en local, tanmateix, serà necessari prèviament haver instal·lat LAMP o WAMP; és a dir, necessitarem una instal·lació funcionant del programari següent:

- Apache 2.2 o superior
- PHP 5.2 o superior
- MySQL 4.1 o superior

La instal·lació i configuració d'aquest programari està fora de l'abast d'aquest document; en qualsevol cas, no hauria de ser necessari realitzar cap tipus de configuració especial, ja que els valors predeterminats de les tres aplicacions haurien de permetre un funcionament correcte del component servidor.

Una vegada instal·lat i configurat el servidor web i l'SGBD, haurem de copiar la carpeta AppRecommender (del component client) a la carpeta arrel de l'Apache (establerta a la directiva DOCUMENT_ROOT).

Per tal de crear la base de dades, cal anar a la línia de comandes i, des de la carpeta bin de MySQL, executar la següent instrucció:

```
mysql < AppRecommender.sql
```

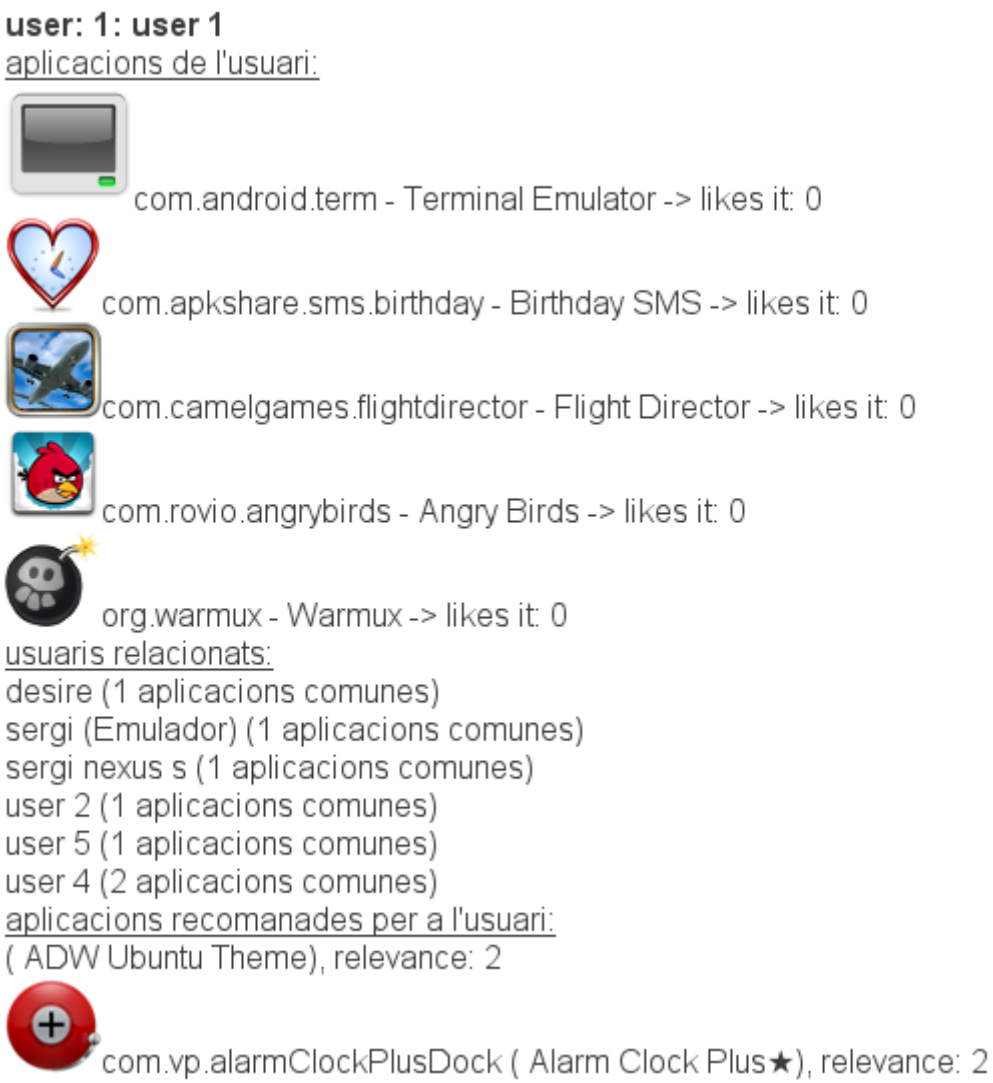
On caldrà substituir AppRecommender.sql per la ruta completa a la ubicació on tenim l'arxiu. Òbviament, també és possible crear la base de dades mitjançant qualsevol client de MySQL avançat, com ara phpMyAdmin.

En cas de que el servidor MySQL tingui establerta una contrasenya per a l'usuari *root* (o desitgem utilitzar un usuari amb menys privilegis), caldrà editar l'arxiu **config.xml** (ubicat a la carpeta *AppRecommender\etc*) i introduir les credencials d'accés correctes (només cal tenir en compte que aquest usuari ha de poder tenir accés complert a la base de dades AppRecommender).

Per últim, podem comprovar que tot està funcionant correctament obrint la URL de test del component amb el navegador:

http://127.0.0.1/AppRecommender/test/test.php

Si tot és correcte, hauríem d'obtenir una pantalla com la següent:



5.4. Desplegament del component client

El component client pot ser executat a l'emulador d'Android o en un terminal real.

Per a executar-lo a l'emulador, el més senzill és obrir el projecte amb l'Eclipse i simplement arrencar-lo.

Per a executar-lo en un terminal real, només cal copiar l'arxiu AppRecommender.apk a la tarja de memòria del dispositiu i executar-lo.

Cal tenir en compte que, per tal d'instal·lar aplicacions no provinents del Market, cal anar a la pantalla de configuració del dispositiu, seleccionar *Aplicacions* i marcar la casella *Orígens desconeguts*.

En la primera execució del programa, ens demanarà un àlies i una adreça de correu electrònic. No és necessari introduir cap de les dues dades, tot i que sí és recomanable. El procés d'actualització de la informació a la base de dades del servidor pot trigar una mica, sobretot si el nombre d'aplicacions és molt gran. Quan finalitzi el procés, ens apareixerà la llista d'aplicacions recomanades. En cas que no aparegui cap, és perquè al terminal (o emulador) no hi ha cap aplicació coincident amb les dels usuaris de la base de dades de prova. Podem instal·lar-ne una qualsevol (per exemple, “Warmux”, que apareix al Market a l'emulador, o “Angry birds” si estem fent la prova en un terminal físic). En finalitzar la instal·lació de l'aplicació, automàticament aquesta informació haurà estat enviada al servidor i ja tindrem algun usuari relacionat (i, per tant, una llista d'aplicacions recomanades).

En cas que hàgim desplegat el component servidor en local, serà necessari modificar la configuració del component client per indicar quina és l'adreça IP del servidor (haurà de ser l'adreça de la xarxa local (LAN) o wi-fi (WLAN) de l'ordinador on hàgim desplegat el component servidor). Per a canviar aquesta configuració, haurem d'obrir el projecte Eclipse i editar els arxius:

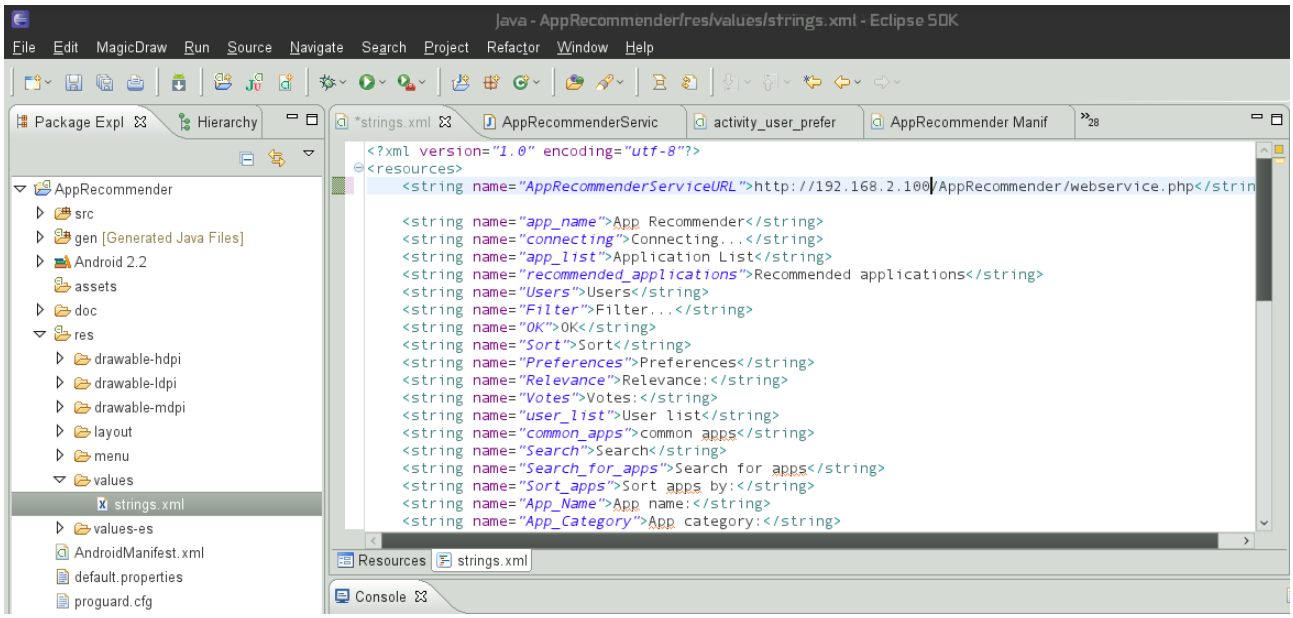
\res\values\strings.xml

i

\res\values-es\strings.xml

És necessari editar tots dos arxius perquè, en funció de la configuració d'idioma del nostre dispositiu, s'utilitzarà un o altre (*\res\values\strings.xml* en el cas de que el dispositiu estigui en anglès -com, per exemple, és el cas de l'emulador- i *\res\values-es\strings.xml* si està en castellà).

El primer node de l'arxiu XML és AppRecommenderServiceURL. Canviarem apprecommender.icssolution.com per la nostra IP. A continuació, podem executar el projecte a l'emulador o generar un nou paquet per tal de desplegar-lo en un terminal físic. A la captura de pantalla següent pot observar-se el canvi que caldria realitzar si el desplegament l'hem realitzat a un ordinador amb adreça IP 192.168.2.100:



6. Línies obertes

Com ja s'ha exposat prèviament, el projecte en el seu estat actual assoleix la pràctica totalitat dels objectius marcats a l'abast inicial. Tot i que penso que aquest producte pot tenir un recorregut més llarg, deixant de ser un simple exercici acadèmic per tenir una aplicació real, probablement serà necessari realitzar algunes millores que en permetin un ús més massiu.

La meva intenció és continuar amb el desenvolupament de l'aplicació, afegint-hi noves funcionalitats i oferint un servei gratuït però de qualitat a tots els seus usuaris. Algunes de les millores que tinc previst realitzar a l'aplicació són:

- **Component client:**
  - Afegir una nova funcionalitat que permeti als usuaris votar les seves aplicacions preferides.
  - Implementació d'un sistema de càrrega progressiva d'aplicacions i usuaris a les llistes, per optimitzar la velocitat de presentació.
  - Afegir una llista que mostri les últimes aplicacions instal·lades pels usuaris afins.
- **Component servidor:**
  - Implementació de la localització, per a retornar les dades (noms de les categories i de les aplicacions) en l'idioma de l'usuari.
  - Millora dels algorismes de càlcul d'usuaris afins i aplicacions recomanades, per tal de fer-los més ràpids i escalables.

7. Conclusions

En els últims tres anys hem assistit a una (nova) revolució en l'àmbit de les tecnologies de la informació adreçades al gran públic: l'adopció massiva dels telèfons intel·ligents. L'aparició de l'iPhone d'Apple va suposar una redefinició del concepte d'smartphone, suposant un autèntic fenomen de vendes. D'entre les múltiples qualitats d'aquest aparell, el sistema de distribució d'aplicacions (l'AppStore) n'és potser la més destacada, aconseguint un èxit sense precedents i totalment inesperat.

Aquest nou paradigma de distribució de programari, basat en la distribució massiva d'aplicacions de molt baix cost, ha atret milers de desenvolupadors, alguns dels quals han esdevingut milionaris en trobar un mercat enorme de clients àvids d'expandir les funcionalitats dels seus telèfons intel·ligents. Així, en només tres anys, Apple ha aconseguit superar les 200.000 aplicacions disponibles, i més de mil milions de descàrregues, generant un negoci de milers de milions de dòlars.

Davant de l'èxit aclaparador de l'iPhone, altres pesos pesants de la indústria han intentat reaccionar, presentant plataformes alternatives que puguin plantar cara a Apple. D'aquestes, l'Android de Google és, sense cap mena de dubte, la més reeixida, fins al punt que, actualment, ja supera a l'iOS pel que fa a vendes i a nombre d'aplicacions disponibles (tant gratuïtes com de pagament) al seu sistema de distribució de programari, el Market.

Tot i els indubtables beneficis que suposen les botigues d'aplicacions per a Apple i Google, així com per als desenvolupadors, la ingent quantitat d'aplicacions disponibles provoca dos problemes que comencen a aflorar amb força:

- Els usuaris es troben amb creixents dificultats per a trobar allò que busquen d'entre tantes alternatives. En alguns àmbits hi ha desenes d'aplicacions similars, i sovint és molt difícil trobar la que millor s'ajusta a allò que busquem. D'aquesta manera, molts usuaris acaben descarregant (i pagant) les aplicacions més conegudes, que possiblement no són les que millor resolen les seves necessitats.

- Per als desenvolupadors, cada vegada és més complicat tenir visibilitat per a les noves creacions. Així, moltes aplicacions que demostren una gran creativitat i aporten solucions innovadores a problemes coneguts acaben oblidades o, encara pitjor, copiades per les grans companyies que, cada vegada amb més força, es queden amb un tros més gran del pastís.

L'AppRecommender neix amb l'esperit de proporcionar una solució parcial a aquests dos problemes. Mitjançant la creació dinàmica de relacions entre els usuaris de l'aplicació (basant-se en el nombre d'aplicacions en comú), l'AppRecommender ajuda els usuaris a trobar les aplicacions que més es poden ajustar al seu perfil. La possibilitat de marcar altres usuaris com a amics, i veure fàcilment quines aplicacions tenen instal·lades aquests usuaris, és també una via ràpida de trobar aplicacions que poden ser interessants per a l'usuari.

Per altra banda, això proporciona una nova manera d'aconseguir visibilitat per als desenvolupadors d'aplicacions dirigides a un tipus de públic en concret: en la mesura que aquestes aplicacions donin una bona solució a les necessitats d'aquest perfil d'usuaris, aquesta esdevindrà ràpidament popular en el seu nínxol de mercat.

Per a implementar aquesta idea, s'ha dissenyat una arquitectura client-servidor, basada en eines de codi font lliure àmpliament utilitzades:

- El component client ha estat desenvolupat amb la plataforma LAMP (Linux, Apache, MySQL, PHP).
- El component client ha estat desenvolupat per a Android, mitjançant l'SDK disponible per a aquesta plataforma.
- La comunicació entre els dos components s'ha desenvolupat com un servei REST, utilitzant l'estàndard JSON per a l'intercanvi d'informació entre els dos llenguatges (PHP i JAVA).

Tot al llarg del projecte, des de les primeres fases (recollida de requisits, definició de l'abast del projecte, planificació de les diferents fases del desenvolupament, etc.), s'han aplicat abastament els coneixements adquirits durant aquests últims anys en les diferents assignatures de l'enginyeria, especialment:

- Pel que fa a la gestió del projecte, l'assignatura Gestió de projectes m'ha proporcionat la base que m'ha permès definir els requisits, realitzar la planificació temporal, estructurar els documents d'acompanyament del projecte, etc.

- Pel que fa al disseny de l'arquitectura, les diferents assignatures de l'àmbit de les bases de dades (Bases de dades I i II, Sistemes de gestió de bases de dades) m'han donat la base teòrico-pràctica necessària per tal de dissenyar eficientment l'estructura de dades subjacent de l'aplicació.

- Pel que fa a la implementació, les diferents assignatures de programació (especialment Programació orientada a l'objecte, Tècniques de desenvolupament del programari, Enginyeria del programari orientat a l'objecte i Procés d'enginyeria del programari) m'han aportat els coneixements necessaris per tal de realitzar el disseny i la implementació seguint mètodes de desenvolupament modernes i fiables, com ara la utilització de patrons o el disseny mitjançant diagrames UML.

- Els coneixements adquirits a l'assignatura Interacció humana amb els ordinadors han estat també de gran importància per a l'elaboració de la interfície d'usuari.

- Altres assignatures, que podríem considerar *menors*, com ara Anglès o Competència comunicativa per a professionals de la informàtica, han jugat també un paper rellevant durant aquests mesos, pel que fa a l'elaboració de la documentació del codi font (que he realitzat en anglès), com pel que fa a la redacció d'aquesta memòria.

- Per últim (però en cap cas amb una importància menor), el seguiment del mètode UOC d'aprenentatge, mitjançant el lliurament progressiu de les proves d'avaluació

continuada, m'ha proporcionat una metodologia de treball que ha esdevingut fonamental per tal de poder completar amb èxit el projecte, compaginant-lo amb la meva carrera professional.

Aconseguir el títol d'enginyer informàtic és, sens dubte, la fita que es marca qualsevol estudiant el primer semestre de la carrera. Tanmateix, el títol no és, ni de bon tros, el més important que hauré obtingut després d'aquests cinc anys i escaig en els que he estat estudiant de la UOC: els coneixements tècnics (molts dels quals ja aplico en la meva vida professional com a desenvolupador), els de gestió de projectes i, sobretot, la cultura de l'esforç continuat que es potencia des de la Universitat suposaran, a mitjà i llarg termini, el premi més important a tot l'esforç realitzat durant aquests anys.

El projecte que estic presentant suposa la culminació del meu procés d'aprenentatge en aquesta titulació, sublimant tots els coneixements i metodologies adquirits en un mateix projecte.

Em permetré concloure amb unes paraules robades a un dels més grans escriptors en llengua castellana:

Caminante, son tus huellas
el camino y nada más;
Caminante, no hay camino,
se hace camino al andar.
Al andar se hace el camino,
y al volver la vista atrás
se ve la senda que nunca
se ha de volver a pisar.
Caminante no hay camino
sino estelas en la mar.

Antonio Machado