BuddyShare: La millor manera per compartir despeses

Sergio Guisado Villanueva

Resum— Les despeses són uns elements que estan molt presents en la nostre vida diària. Hi ha despeses que son individuals i altres de col·lectives. En les despeses col·lectives s'han de gestionar els participants, la quantitat que paga cadascú (no sempre és proporcional) i qui ha realitzat el pagament.

Aquest projecte neix de la necessitat de gestionar les despeses col·lectives, una aplicació per dispositius Android on es gestionen el grups, la quantitat a pagar i el pagament de les despeses.

Paraules clau— Despeses Col·lectives, Pagament Despeses, Aplicació Android, Gestió Despeses.

Abstract— Expenses are some elements that are very present in our daily lives. There are individual expenses and other collective. In collective expenses should manage participants, the amount pays each person (not always proportional) and who has made the payment.

This project born from the need to manage collective expenses, an application for Android which managed groups, the amount payable and the payment of expenses.

Index Terms— Collective expenses, Payment Expenses, Android Application, Management Expenses.

1 INTRODUCCIÓ

BUDDYSHARE és un sistema amb el qual es gestionen les despeses d'un grup d'usuaris. El sistema té una lògica implementada amb la qual es pot saber quant paga cadascú d'una despesa i si aquesta ha estat pagada, tot això en temps real i sense concurrència.
És un sistema que suporta petits grups i grans grups, això vol dir que tant un grup d'amics com un col·lectiu de qualsevol àmbit pot beneficiar-se de BuddyShare.

1.1 Objectius

L' objectiu principal és fer un sistema amb el qual dues o més persones pugin coordinar i gestionar les despeses que generen entre ells. Aquest sistema ha de permetre que qualsevol usuari d'un grup pugui afegir una despesa i marcar aquesta com pagada quan calgui. També ha de tenir la possibilitat de calcular els deutes globalment, és a dir, fent la diferència entre el que li han de pagar al usuari amb el que ha de pagar. Caldrà que l'alta d'usuari en el sistema es faci integrant una xarxa social per tal de facilitar la seva utilització.
Com a objectius més secundaris, es vol que en el sistema hi hagi una manera de justificar que s'ha fet el pagament d'una despesa. Per una altre banda, l'usuari podrà afegir una despesa i aquesta sotmetre-la a una votació per tal de saber si els altres membres del grup accepten o no aquesta despesa.
Una vegada assolits aquests objectius es vol introduir el concepte de grup privat i públic, és a dir, qualsevol usuari es pot afegir a un grup públic.
Com aquest projecte té la durada limitada, al haver un canvi en la planificació perquè s'ha allargat la durada d'unes tasques, fa que aquests objectius més secundaris, no es pugin complir. Per tant hi ha una reformulació dels objectius.
Degut a que són objectius secundaris i sense els quals, l'objectiu principal no es veu afectat, he decidit eliminar aquestes funcionalitats més secundàries de l'aplicació.
Si el projecte no tingués una data de finalització marcada el que es faria és allargar la planificació no pas eliminar un objectiu.

- *E-mail de contacte: sr.guisado@gmail.com*
- *Menció realitzada: Enginyeria del Software*
- *Treball tutoritzat per: Ramon Baldrich*
- *Curs 2014/15*

2 ESTAT DEL ART

La gestió de les despeses d'un grup és un problema molt comú a la societat, tant és així que existeixen algunes solucions ja implementades.

2.1 SETTLE UP

Una de les solucions més completes és SETTLE UP. És una aplicació per dispositius Android i iPhone, permet crear grups, anar afegint despeses i la quantitat de pagament de cada participant. Un dels inconvenients d'aquesta aplicació és que a cada despesa la informació que es mostra no és clara i costa d'entendre el que passa amb la despesa.

2.2 KIDOIKOIAKI

Aquesta aplicació també ens pot ajudar a tenir els comptes clars. És molt senzilla de gestionar. Només cal escollir qui ha fet cada despesa i el programa calcula qui ha de pagar a qui. Està disponible per Android.

2.3 PAYBACK

Aquesta app només està disponible a iPhone. Es pot crear un viatge i afegir a les persones que participaran. Així, es pot anotar cada despesa i la persona que l'ha pagat. Quan finalitza el viatge, només cal consultar l'estat dels deutes i Payback ens mostrarà una llista amb la despesa total i la quantitat de diners que ha deu cada un.

En resum, aquestes aplicacions, totes són aplicacions mòbils, excepte Settle Up, que està pensada per dispositius mòbils però té suport Web.
Un dels inconvenients d'aquestes aplicacions, és que no es mostra la informació de manera clara i ordenada.
En aquestes aplicacions falta la funcionalitat de gestionar els pagaments de les despeses, un punt important en la gestió de les despeses.

4 METODOLOGIA

La metodologia utilitzada per el desenvolupament d'aquest projecte és una metodologia en cascada, degut a que per aquest projecte els objectius i requeriments estan definits al principi de projectes i no s'esperen canvis significatius en aquests.
Concretament, el que s'ha fet en el desenvolupament es fer primer un disseny de la interfície d'usuari utilitzant wireframes, seguidament els diagrames de casos d'ús, els dissenys de classes i de seqüència. Una vegada fet el disseny, es fa el desenvolupament de les tasques i per últims es testeja el codi desenvolupat.
Cal dir, que en els dissenys de classes i de seqüencia s'han agut de anar canviant per la falta de coneixements de les tecnologies, sobretot en els dissenys d'Android, que si no es té un coneixement alt de les llibreries és molt difícil fer un disseny correcte. Però a mida que anava avançant el desenvolupament els dissenys eren més correctes.

5 ARQUITECTURA

Encara que aquest projecte és una aplicació Android, s'ha decidit utilitzar una arquitectura FrontEnd BackEnd, per tal d'obtenir un producte escalable, el qual faciliti el desenvolupament amb altres sistemes.
Pel que fa al FrontEnd s'ha implementat una aplicació amb Android 5.0 [1][2][3][4][5][6][7], incorporant Material Design, el nou estil de disseny de les aplicacions Android.

Material Design rep el seu nom per estar basat en objectes materials. Peces col·locades en un espai i amb un temps determinat. És un disseny on la profunditat, les superfícies, les ombres i els colors juguen un paper principal. Uns dels elements claus són la llum i les ombres, una il·luminació realista que proporciona indicis de com es comportarà un element i en quin nivell es troba.
Uns altres dels elements més característics són el moviment i les animacions, incorpora una llibreria que fa que el desenvolupador pugui dissenyar i programar un interfície d'usuari més dinàmica.

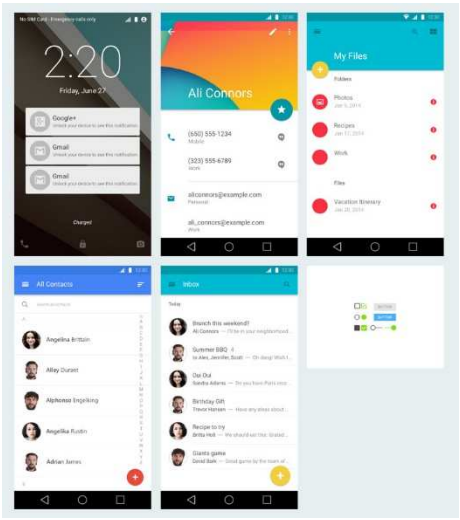

Fig.1 Exemple Material Design

En el BackEnd, s'ha implementat una API REST[8] utilitzant la llibreria Jersey[9] per a Java.
Una API REST és un estil d'arquitectura per a sistemes distribuïts, es centra en l'ús dels estàndards HTTP per a la transmissió de dades sense la necessitat de comptar amb una capa addicional. Les operacions (o funcions) es sol·liciten mitjançant GET, POST, PUT i DELETE , pel que no requereix d' implementacions especials per consumir aquests serveis. A més es pot utilitzar JSON en comptes de XML com a contenidor de la informació, que facilita el desenvolupament en el llenguatge Java, ja que existeixen llibreries que pasen de JSON a objecte Java directament.
Per tal de que l'usuari tingui la informació de l'aplicació en temps real, s'ha utilitzat el servei Google Cloud Messagin per implementar les notificacions.
Google Cloud Messaging[10] és un servei que permet enviar dades des del servidor a dispositius Android, implementa tots els aspectes de gestió de cues de missatges i

de lliurament a l'aplicació. Com he integrat aquest servei amb l'aplicació està explicat a la part de desenvolupament.
Per poder utilitzar aquest servei, s'ha registrat l'aplicació a la pàgina de Google Service, que proporciona un identificador d'aplicació, per tal de saber on enviar els missatges.

Fig2. Arquitectura servei GCM.

Per tal de que les dades que es generes siguin persistents, s'ha utilitzat MongoDB [11][12][13] com a base de dades de l'API.
MongoDB és una base de dades NoSQL, està orientat a documents i no pas a registres. Aquests documents són guardats en format BSON, això facilita la seva implementació, ja que passar un objecte java a BSON és senzill.
Encara que per les dades que es generen en l'aplicació sería més convenient utilitzar una base de dades SQL, ja que les dades están molt estructurades, he decidit utilitzar MongoDB per tal d'apendre base de dades NoSQL.

6 BUDDYSHARE

Primer de tot, es va fer un anàlisi de les solucions existents i una captura dels requeriments que es poden extreure dels objectius.
Una vegada fet aquest anàlisi, es va arribar a la conclusió que els punts més importants de l'aplicació eren la personalització de quant paga cadascú, una manera d'identificar ràpidament quines despeses són les generades per l'usuari i quines ha de pagar l'usuari i, implementar notificacions per tal de que l'usuari estigui al tant dels canvis d'un grup.

En aquest anàlisi també es va veure que existeixen diferents tipologies de grups, en la taula s'esmenten aquestes tipologies i quines s'han implementat.

Tipologia	Descripció	Implementat
Grup Estàndard	Grup on es poden afegir despeses, sense modificacions.	Sí
Grup Pagament Per Participant	No es paguen les despeses, sinó el balanç entre els participants.	Sí
Grup amb opció afegir amic	Grup en el qual es permet afegir un amic una vegada s'ha creat el grup. S'ha de tenir en compte les despeses ja generades (si el nou membre les ha de pagar o no).	No
Grup amb opció a sortir del grup	Grup on es permet sortir del grup a qualsevol participant. S'ha de tenir en compte les despeses generades per aquest participant i les que no ha pagat.	No

A continuació s'explicarà les funcionalitats desenvolupades i els problemes trobats al desenvolupar l'aplicació.

6.1 Iniciar Sessió

Per iniciar sessió es farà a través de Facebook [Fig 5.], ja que així serà més còmode per a l'usuari fer ús de l'aplicació.
Facebook té una llibreria (Facebook SDK), amb la qual es pot iniciar sessió a través d'un dispositiu Android. Aquesta llibreria proporciona un botó amb la funcionalitat ja implementada per anar a la pàgina de Facebook i fer login. Una vegada l'usuari ha iniciat sessió a Facebook, aquest retorna un accesToken i un objecte Profile amb les dades principals del usuari. Aquest accesToken és important, ja que per interactuar amb l'Api de Facebook cal utilitzar el accesToken com identificador.
Pel que fa a la implementació d'aquesta funcionalitat del botó de login del Facebook, el que he fet és una vegada retorna el accesToken, guardar-lo com a propietat de l'aplicació per tal d'interactuar més endavant amb l'Api de Facebook. Amb el Profile retornat, es crea un objecte Usuari i s'envia a l'Api de BuddyShare per guardar-lo a la base de dades com a nou usuari. Si aquest usuari ja existeix a la base de dades (es verifica mirant el facebookId)

retornarà l'usuari existent, sinó retornarà el usuari creat. Aquest usuari es guarda en l'aplicació mòbil per tal de poder operar sense haver d'iniciar sessió cada vegada que s'obre l'aplicació.
Per tal de que l'aplicació no es quedi penjada, s'utilitza la classe AsyncTask, que proporciona la llibreria d'Android per crear un nou thread, per fer la comunicació amb les APIs.

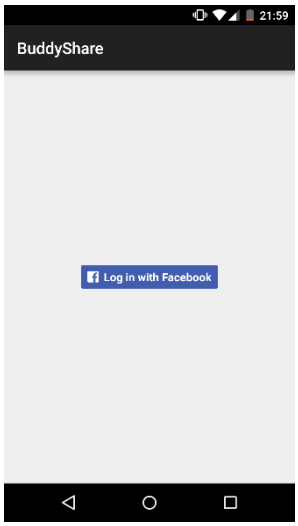

Fig5. ScreenShot de la pantalla d'iniciar sessió a un mòbil Android

6.2 Afegir Amics

L'aplicació ha de permetre afegir amics per tal de poder interactuar amb altres usuaris. Per fer això s'utilitza Facebook, és a dir, els amics que es poden afegir són els amics que tenen Facebook i estan utilitzant l'aplicació.

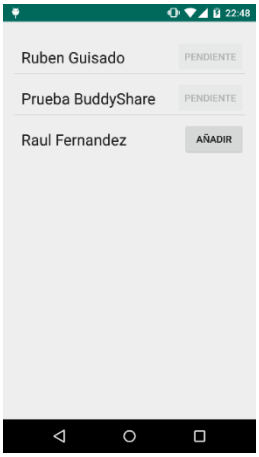

Fig6. ScreenShot de la pantalla afegir amics en un mòbil Android.

Per fer això s'utilitzarà Graph v2.3. Graph és l'API de Facebook que permet interactuar amb la plataforma de Facebook.
El que es fa és una petició HTTP a l'API de Graph, amb els paràmetres accesToken obtingut al iniciar sessió i la url de destí, en aquest cas /friends. Per poder accedir als amics d'un usuari, aquest usuari ha d'haver donat permís per tal de que la aplicació pugi accedir.
Aquesta petició HTTP retorna una llista de d'amics (facebookId i nom) en format JSON, per tal de mostrar aquesta llista es mapeja aquests JSON a la classe FacebookAmic.
Si l'usuari pressiona a afegir amic, s'envia a l'API de BuddyShare l'objecte amic seleccionat i s'afegeix a la llista d'amics de l'usuari, amb estat pendent d'acceptació. Una vegada afegit a la llista d'amics s'envia una notificació al usuari afegit per tal de que accepti la sol·licitud d'amic.
Aquesta crida retorna l'usuari actualitzat amb l'amic agregat a la llista d'amics. Una vegada obtingut l'objecte usuari, es guarda en local per tal de poder consultar l'aplicació sense connexió a internet.
Aquesta llista d'amics que es mostra a l'usuari, com ja he dit són els amics que tenen instal·lada l'aplicació, per tant si l'amic esta afegit es mostrarà "Añadido", si està en pendent d'acceptació es mostrarà "Pendiente" i si no està afegit es mostrarà el botó "Añadir".

6.3 Crear Grups

L'acció de crear un grup es fa pressionant el botó de la pantalla principal [Fig 7.]. Aquesta pantalla és bastant senzilla, un formulari on omplir les dades del grup, seleccionar els amics que participaran al grup i un botó crear. Al fer clic en aquest botó es crida a l'Api de BuddyShare i s'envia el grup creat, l'Api el guarda a la base de dades, actualitza la llista de grups del usuari i notifica al altres usuaris que se'ls ha afegit a un nou grup. Per enviar aquesta notificació fa servir el servei GCM explicat anteriorment. Una vegada fet això, retorna el grup creat amb l'id corresponent.
Al crear un grup, es selecciona el mode de pagament del grup, hi han dos:
- Pagament per participant: En aquesta modalitat la quantitat de diners a pagar per cada membre del grup a un altre membre, es calcularà amb el conjunt de despeses entre aquests dos membres del grup.
- Pagament per despesa: En aquesta modalitat es paga la despesa, sense tenir en compte altres deutes en altres despeses.

Una vegada fet això, per tal d'estalviar una crida a l'Api, s'insereix el grup en local així es mostrarà a la llista de grups sense haver de fer una altre petició HTTP.

Fig.7 Botó per afegir un grup

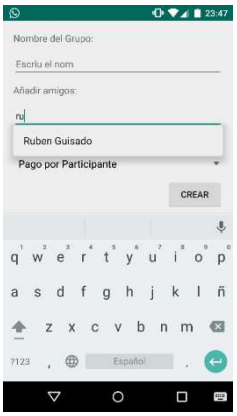

Fig8. ScreenShot crear grup en un mòbil Android.

6.4 Mostrar Grups

La pantalla principal de l'aplicació és una llista dels grups del usuari i una altre llista d'amics, tot això implementat amb pestanyes.

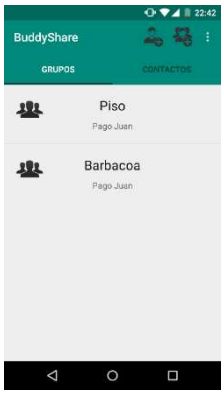

Fig9. Pantalla principal de l'aplicació, on es mostren els grups. ScreenShot d'un mòbil Android

Per mostrar la llista de grups, el que es fa és primer de tot fer una petició HTTP a l'API de BuddyShare, per tal d'obtenir una llista de grups del usuari, la qual es mostrarà per pantalla. Si aquesta petició no retorna resposta, o retorna un error es mostrarà la llista de grups guardada al mòbil.

6.5 Afegir Despesa

L'acció d'afegir despesa es bastant similar a crear grup en quant a funcionalitat. Una despesa es crea pressionant el botó de la pantalla de grup. Aquesta pantalla és bastant senzilla, un formulari on omplir les dades de la despesa i un botó crear.
Aquest formulari permet personalitzar la quantitat de diners que pagarà cada membre del grup, per defecte es reparteix proporcionalment.
Al fer clic en el botó de crear, es crida a l'Api de BuddyShare i s'envia la despesa creada amb l'id del grup al que pertany la despesa, l'Api el guarda a la base de dades, actualitza la llista de despeses del grup i retorna el grup actualitzat. Abans de retornar el grup actualitzat, envia una notificació a la resta d'usuaris mitjançant el servei de GCM.
Una vegada fet això, per tal d'estalviar una crida a l'Api, s'insereix el grup en local així es mostrarà a la llista de grups sense haver de fer una altre petició HTTP.

Fig10. ScreenShoot de la pantalla afegir despesa en un mòbil Android.

6.6 Detalls Grup

Si l'usuari fa clic en el nom del grup, anirà a la pantalla de detalls del grup. En aquesta pantalla es mostrarà una llista amb els participants del grup.
Si el mode del grup és per participant, en aquesta pantalla serà on els usuaris tindran la informació de l'estat dels seus deutes. Al costat de cada participant es mostra la diferencia entre les despeses que l'ha de pagar l'usuari i les que ha de pagar a aquest participant. Si la diferència és positiva, vol dir que el participant li deu aquesta quantitat, en canvi si es negativa, l'usuari l'ha de pagar aquesta quantitat.
La metodologia de pagament és la mateixa que amb les despeses, el primer usuari indica que l'ha pagat i l'altre usuari ha de confirmar aquest pagament per poder marcar el deute a 0.
Per implementar aquesta funcionalitat el que es fa és una crida a l'Api que retorna les depeses d'aquest grup i amb aquestes despeses, per cada membre del grup, es va calculant el balanç amb la quantitat a pagar de cada despesa introduïda per l'usuari, amb la quantitat a pagar de les despeses introduïdes per l'altre membre del grup.
Quan es confirma el pagament d'un deute, es marquen totes les despeses entre aquests dos usuaris com pagades, així la següent vegada no les tindrà en compte per fer el càlcul del balanç.

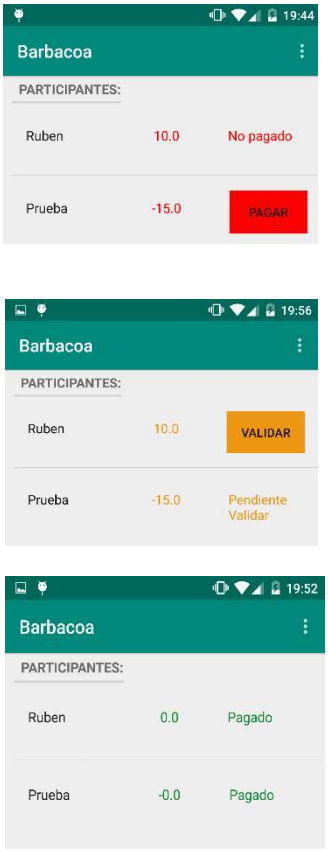

Fig11. ScreenShot transicions del estat d'una despesa

6.7 Mostrar les meves despeses

Una vegada l'usuari entra en la pantalla principal del grup, es fa una petició a l'API BuddyShare amb la informació del grup. Quan es rep el grup es creen dues llistes, una amb les despeses creades per el usuari i una altre amb les despeses creades per la resta dels usuaris.
En la pestanya de les meves despeses, es mostra una llista amb el nom de la despesa, el total de la despesa i si esta pagada o no per tots.
Per veure els detalls de qui l'ha pagat i qui no, es fa fent clic en la despesa que es vol consultar.
Si el grup s'ha creat indicant el mode de pagament per participant, no apareixerà cap informació de qui l'ha pagat o no.

Fig12. ScreenShot llista de les meves despeses en un mòbil Android.

6.8 Detalls les meves despeses

En aquesta pantalla es mostren els detalls de la despesa seleccionada, per mostrar els detalls no es fa una petició a l'API, sinó que es passa l'objecte despesa des de la pantalla anterior.
En aquesta pantalla es mostra un llistat dels altres membres del grup, detallant la quantitat a pagar i si està pagada o no. Per tal de pagar una despesa el receptor l'ha de marcar com a pagada i l'emissor des de aquesta pantalla ha de confirmar el pagament.
Si el grup s'ha creat indicant el mode de pagament per participant, no apareixerà cap informació de qui l'ha pagat o no.

Fig13. ScreenShot detalls de les meves despeses en un mòbil Android.

6.9 Despeses del Grup

En aquesta pantalla es mostren una llista amb les despeses a pagar, cada despesa conté qui l'ha creat, el nom de la despesa, la quantitat a pagar i si està pagada per l'usuari.
Per pagar una despesa s'ha der fer clic en la despesa i anar a la pantalla de detalls.
Si el grup s'ha creat indicant el mode de pagament per participant, no apareixerà cap informació de qui l'ha pagat o no.

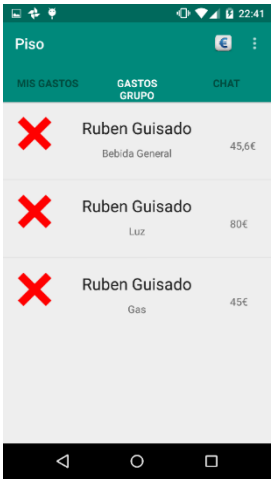

Fig14. ScreenShot llista de les despeses en un mòbil Android.

6.10 Detalls despeses del Grup

En aquesta pantalla es mostra un llistat dels altres membres del grup, detallant la quantitat a pagar i si està pagada o no. És bastant similar a la pantalla de detalls de les meves despeses, amb la diferència que en aquesta hi ha un botó per tal d'indicar que l'usuari ha pagat la despesa.
Si el grup s'ha creat indicant el mode de pagament per participant, no apareixerà cap informació de qui l'ha pagat o no.

Fig15. ScreenShot detalls de les despeses en un mòbil Android.

6.11 Chat

Dintre de la pantalla de grup hi ha implementat un chat on els integrants del grup poden intercanviar missatges.
El chat conté una llista de missatges i un camp per escriure un missatge i enviar-ho. Quan l'usuari envia un missatge, aquest s'envia a l'API, que no guarda cap registre del missatge, sinó que envia una notificació als destinataris amb el missatge. L'aplicació quan rep la notificació la afegeix a la llista de missatges que té guardada en local.
Per mostrar els missatges es llegeix un arxiu guardat en local amb els missatges. El servidor no té necessitat de guardar aquesta informació, per tant només es guarda als dispositius mòbils.

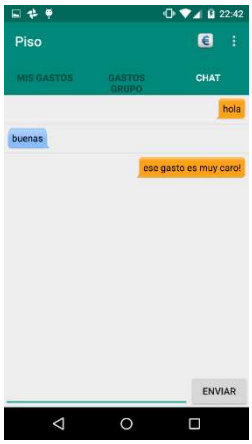

Fig.16 ScreenShot del chat d'un grup.

6.12 Notificacions

Per tal d'avisar a l'usuari de novetats i canvis en els grups s'ha implementat una sèrie de notificacions, les quals s'envien utilitzant el GCM.
- Notificació de Creació de Grup:
Aquesta notificació s'envia quan l'usuari es inclòs en un nou grup. Quan l'usuari fa clic en la notificació, s'obre l'aplicació amb la pantalla principal d'aquest grup.

Fig17. ScreenShot d'una notificació de creació del grup.

- Notificació de despesa afegida:
Aquesta notificació s'envia quan un usuari afegeix una despesa. . Quan l'usuari fa clic en la notificació, s'obre l'aplicació amb la pantalla principal d'aquest grup.

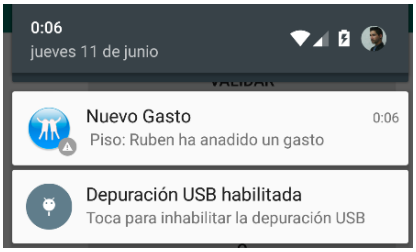

Fig18. ScreenShot d'una notificació de creació d'una despesa.

- Notificació sol·licitud d'amistat:
Aquesta notificació s'envia quan afegeixen a l'usuari com amic. Com es pot veure a la imatge, des de la notificació és on l'usuari acceptarà o no aquesta sol·licitud. Si l'usuari accepta la sol·licitud s'envia un missatge a l'Api que marca a l'usuari com afegit i introdueix l'usuari a la seva llista d'usuaris.

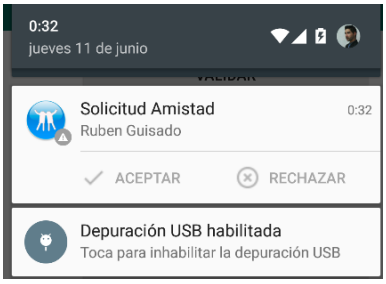

Fig19. ScreenShot d'una notificació d'afegir amic.

- Notificació avís de pagament de despesa:
Quan hi ha un canvi en l'estat de pagament d'una despe-

sa que té relació amb l'usuari, s'envia una notificació per informar a l'usuari d'aquest canvi. Si l'usuari fa clic en la notificació s'obre l'aplicació i va a la pantalla de detalls de la despesa.

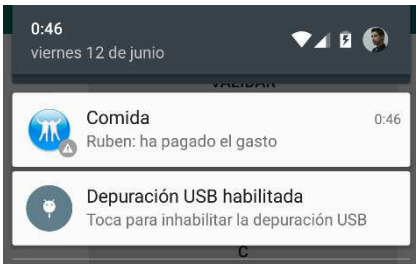

Fig.20 ScreenShot d'una notificació de pagament de despesa.

- Notificació nou missatge:

Fig21. ScreenShot d'una notificació d'un nou missatge.

Per implementar les notificacions s'utilitza el servei GCM. Com ja he explicat abans, GCM és un servei per fer notificacions push, en aquest projecte s'utilitza per enviar les notificacions als usuaris.
El que he fet per poder utilitzar el servei GCM, és primer de tot registrar l'aplicació en la web de Google service, que proporciona una AppID que identifica la aplicació.
Una vegada l'aplicació esta registrada, el que cal fer és registrar cada dispositiu com a client d'aquesta aplicació. Per fer això la llibreria de GCM proporciona uns mètodes per obtenir el registrationID. Aquesta petició es fa, una vegada l'usuari s'ha loggejat a l'aplicació.
Aquest registrationID es guarda coma atribut de l'usuari i s'utilitzarà per indicar al servei de Google a quin dispositiu ha d'enviar el missatge.
Els missatges de notificació els envia l'Api de BuddyShare, que utilitzant la classe GCMNotification, envia els missatges a cada registrationID.
L'aplicació quan rep els missatges del servei, construeix la notificacions amb els paràmetres obtinguts del missatges.

6.13 Problemes Trobats

La primera tasca a desenvolupar va ser iniciar sessió utilitzant l' Api de Facebook, aquí em vaig trobar amb els primers problemes, ja que seguint la documentació de Facebook[14] no vaig poder fer funcionar el iniciar sessió i els exemples que trobava per internet eren amb una versió inferior a l'actual. Finalment vaig trobar el problema, era degut la versió de Android que estava utilitzant, vaig passar de la versió 2.3 a la 4.1 i em va funcionar correctament.
Aquesta tasca a la planificació tenia de durada dos dies, però vaig tardar 5 dies per poder iniciar sessió amb Facebook. Una vegada fet això va ser relativament fàcil accedir als amics del usuari.

Un altre problema que ha afectat a la planificació és que una vegada fet la pantalla principal, on estan els grups i els contactes [Fig.3], vaig haver de canviar-la completament, utilitzant altres classes de la llibreria d'Android i una altre implementació. Vaig haver de canviar aquesta pantalla degut a que, tal i com l'havia fet des d'un principi no es podia navegar per les pestanyes desplaçant la pantalla amb els dits. Em vaig adonar d'això, utilitzant una altre aplicació que tenia aquest mateix problema, que per navegar per les pestanyes s'havia de clicar en les pestanyes.

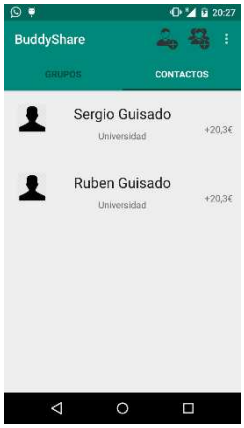

Fig3. Pantalla principal del sistema, mostra la pestanya seleccionada de contactes

Aquest problema no l'havia detectat perquè utilitzava un emulador al PC per provar l'aplicació i, en l'emulador s'utilitza el ratolí per navegar per les pantalles. Per tal d'implementar això, he utilitzat el nou concepte d'Android Material Design explicat anteriorment.
El fet de canviar tot això ha implicat una setmana de treball extra que no estava contemplat a la planificació.

Un altre problema que he tingut és degut a que el meu ordinador on treballo té una memòria RAM de 4GB i tenir el Android Studio amb l'emulador del mòbil, el servidor Tomcat pel Back-End i la base de dades MongoDB, tot arrancat, feia que l'ordinador es quedés congelat. Per tant he hagut de provar la aplicació directament amb el telèfon i només engegar el servidor i la base de dades.
L'emulador d'Android consumeix molta RAM i en el meu ordinador no era viable treballar i vaig canviar a fer-ho directament amb el meu mòbil. Això m'ha generat molts problemes per poder provar l'aplicació, ja que no sempre

el meu ordinador era capaç d'identificar el mòbil i per provar una petita funcionalitat em podia passar hores.
Pel que fa a la utilització del servei Google Cloud Messa-ge, vaig tenir problemes per enviar un primer missatge, ja que jo estava enviant un objecte json i només accepta objectes String. El problema està en que el servei només llança un error però no indica que el error és degut a això, tal i com diu a la documentació d'Android[], els valors han de ser String sinó són ignorats.
Per últim, per desenvolupar el chat, he hagut d'implementar la vista de les bombolles que contenen el missatge amb el 9-patch.

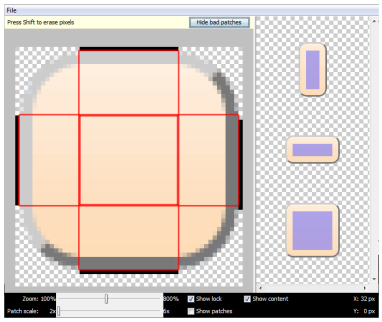

Fig4. Exemple imatge 9-Patch

9-patch és un tipus d'imatge per Android, que s'utilitza per canviar la mida de la imatge i no perdre la resolució de la imatge. Aquesta imatge 9-patch incorpora unes barres negres, que indiquen al dispositiu Android com ajustar la imatge.
Per generar aquest 9-patch correctament, he tingut alguns problemes ja que no ajustava bé les barres i la imatge es deformava.
Tot això a contribuït a que els objectius més secundaris, justificants de pagaments, votació de despesa i despeses públiques no s'hagin pogut desenvolupar.

7 Conclusions

En la realització d'aquest projecte s'ha desenvolupat una aplicació per Android en la qual es poden gestionar des-peses en grups i portar un seguiment de qui les ha paga-des i qui no.
L' arquitectura del sistema consta d'un FrontEnd amb Android i un BackEnd amb una API REST. Gràcies a aquesta API, només cal desenvolupar el FrontEnd per tenir aquesta aplicació per a altres sistemes mòbils.
He pogut veure la quantitat de tecnologies i fra-meworks que hi ha, tant de pagament com de codi lliure. He après a utilitzar alguns com Android SDK, Spring, Jersey, MongoDB SDK, encara que no en pro-funditat.
Com ja he explicat anteriorment, no he pogut complir tots els objectius marcats al principi del projecte, per tant, no dono com acaba la aplicació.

Aquestes són les tasques que queden per fer:

- Implementació de les diferents tipologies d'un grup.
- Compte d'usuari propi de l'aplicació
- Sistema de votació de despeses
- Justificant de pagament d'una despesa
- Dia màxim de pagament.
- Pàgina personal on tenir una visió general dels grups i amics.
- Implementar mecanismes de seguretat tant en l'aplicació Android com en l'Api de Buddyshare.
- Versió per a diferents plataformes mòbils.

Agraïments

M'agradaria agrair al meu tutor del projecte, pel l'ajuda al fer el paper de client i els consells per redactar als docu-ments.

Bibliografia

[1] Darwin, I. F. (2011). Android Cookbook. O'Reilly Media.
[2] Aydin, M. (2012). Android 4: New Features for Application Deve-lopment. Packt Publishing.
[3] Gargenta, M. (2011). Learning Android. O'Reilly Media.
[4] Gupta, A. (2014). Learning Pentesting for Android Devices. Packt Publishing.
[5] Wolfson, M. (2013). Android Developer Tools Essentials. O'Reilly Media.
[6] Zapata, B. C. (2013). Android Studio Application Deve-lopment. Packt Publishing.
[7] Zigurd Mednieks, G. B. (2013). Enterprise Android. Wrox.
[8] Jim Webber, S. P. (2010). REST in Practice. O'Reilly Media.
[9] Jersey RESTful Web Services in Java. (2015, 03 2). Retrieved from https://jersey.java.net/
[10] Google Cloud Messaging. (20 / May / 2015). Recollit de Google Deve-lopers: https://developers.google.com/cloud-messaging/gcm?hl=es
[11] Chodorow, K. (2011). 50 Tips and Tricks for MongoDB Deve-lopers. O'Reilly Media.
[12] Chodorow, K. (2011). Scaling MongoDB. O'Reilly Media.
[13] Chodorow, K., & Dirolf, M. (2010). MongoDB: The Definitive Guide. O'Reilly Media.
[14] Facebook SDK for Android. (2015, 03 18). Retrieved from Facebook: https://developers.facebook.com/docs/android
[15] Dahanne, A. (2013). Spring for Android Starter. Packt Publis-hing.

APÈNDIX

A1. UML API BUDDYSHARE