

PhotoChat - Talk in pictures

En mobilapplikation för fotografisk kommunikation, Av Erik Gustafsson



Projektuppgift av i kursen

”Utveckling av mobila applikationer”

vid Umeå universitet

Innehållsförteckning

Introduktion.....	2
Beskrivning av appen.....	2
Val av projekt.....	2
Användning och målgrupp.....	3
Målbetyg för projektet.....	3
Färgval och tema.....	3
Hur man använder PhotoChat.....	4
Logga in i appen.....	4
Registrera sig.....	4
Första välkomstskärmen.....	5
Lägg till kontakt.....	5
Kontaktlista.....	5
Chatten.....	6
Skicka foto.....	6
Skicka text.....	7
Hur PhotoChat fungerar.....	7
Nätverkskommunikation.....	7
Databasmodell.....	7
Datalagring på klientsidan.....	9
Etik och säkerhet.....	10
Internetmobbing och annat missbruk av tjänsten.....	10
Rapportering av olämpligt material.....	10
Appens behörigheter.....	11
Krypterad trafik.....	11
Tekniska utmaningar under projektet.....	11
Trådar.....	11
Uppladdning av filer via HTTP.....	12
Rotation av bilder.....	12
Utvecklingsmöjligheter.....	12
Källförteckning.....	13

Introduktion

Beskrivning av appen

PhotoChat är ett kommunikationsverktyg för att enkelt ta fotografier i nuet och omedelbart dela med sig av fotografierna till vänner och bekanta. Appen fungerar som en chatt, där bilder som skickats från en och samma person hamnar i samma chattkonversation. Det är lätt att se bilderna användaren skickat eller tagit emot i efterhand då bilderna sparas i chatthistoriken. Förutom möjlighet att skicka nytagna fotografier finns även möjlighet att skicka vanliga textmeddelanden. Med dessa kan användaren exempelvis kommentera en bild denna har fått, fokuset på appen ligger dock på bilderna.

Val av projekt

När jag började utveckla slutprojektet för kursen så ville jag utveckla något som använde sig av internet. Detta eftersom en stor anledningen till att smartphones är så användbara är att de ger möjlighet att använda olika internetjänster, som tex Facebook, Twitter och Gmail direkt i telefonen. Jag ville lära mig bygga en app som kunde agera som klient kopplad mot en webbserver. Jag tyckte att en chatt var en bra utmaning eftersom jag där skulle behöva få användare att kommunicera med varandra, alltså en relativt avancerad internetapp. Det blev en kamerafokuserad chattapp vid namn

PhotoChat.

Användning och målgrupp

Appens grundidé är någorlunda lik Snapchat. En användare tar ett kort och skickar det på en gång till en annan användare. En sak som skiljer PhotoChat och Snapchat åt är att Snapchat på en gång tar bort bilder som anlänt. Detta gör inte PhotoChat, vilket särskiljer användningsområdet från Snapchat en del. PhotoChat är inte lika mycket till för att skicka pinsamma bilder som man kanske inte vill att någon annan ska se på samma sätt som i Snapchat.

Användningsområdet för PhotoChat är mer att dela bilder av det mindre pinsamma slaget än Snapchat. Ett användningsfall på hur appen är tänkt att användas kommer här:

Jag har en kompis Gustaf som gillar Norrlands Guld. När jag var på semester i Gräsö så såg jag ett stort flak med Norrlands Guld på den lokala butiken. Med PhotoChat kan jag snabbt ta ett kort på det stora flaket Norrlands Guld och skicka över direkt från kameran till min kompis Gustaf via PhotoChat.

Målgruppen för appen är vana vid vardagsteknik, och gillar att fotografera och dela med sig av de de ser till kompisar. En stor del av målgruppen är troligen unga men inte nödvändigtvis alla. Detta antagandet är bland annat baserat på att användare av den liknande appen Snapchat. Bland SnapChats användare är 71% under 25 år (Time, 2014).



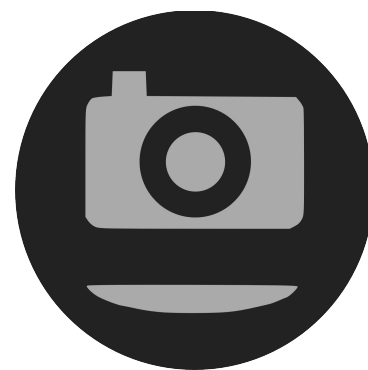
Figur 1: Hur en konversation i PhotoChat typiskt kan se ut

Målbetyg för projektet

Jag siktar på VG och tycker att PhotoChat är en så pass komplicerad app, med mycket arbete bakom att den förtjänar att få VG. Behövs något kompletteras för att kunna få det så kompletterar jag gärna med det.

Färgval och tema

Eftersom det är bilderna som ska vara i fokus i denna app så är allting som inte är bilder svart, vitt eller grått. Bilderna är dock i färg, vilket gör att dessa sticker ut från resten av appen. Detta ska ge lite samma känsla som ett vitt papper med svart tryck på, och på det finns det fastklitrade fotografier. Logotypen för applikationen som syns i figur 2 är ett glatt ansikte i svartvitt gråskala där ögonen är utbytta mot en kamera.



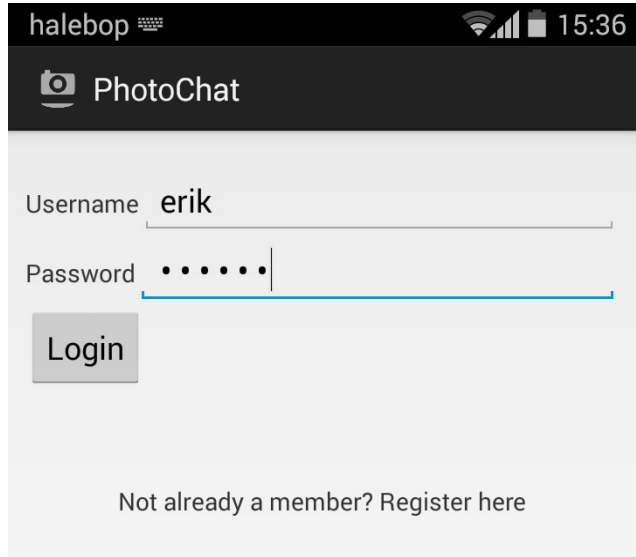
Figur 2: PhotoChats logotyp

Hur man använder PhotoChat

Här följer en genomgång på de skärmar och funktioner som finns i PhotoChat. Eftersom alla skärmar har en tillhörande bild på dig och en tillhörande text så har jag när jag författat artikeln valt att lägga texten med tillhörande bild sida vid sida.

Logga in i appen

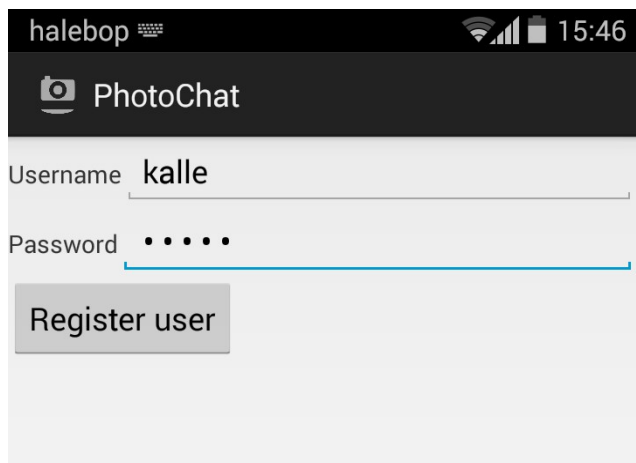
Det första som möter användaren när denne startar appen är en inloggningsskärm. Här kan användaren antingen logga in med användarnamn och lösenord, eller registrera ett nytt konto om användaren inte redan har ett.



Figur 3: Inloggningsskärmen innehåller fält för användarnamn och lösenord, samt möjlighet att registrera nytt konto

Registrera sig

På skärmen för att registrera användare är det inte mycket som behöver fyllas i. Användarnamn och lösenord räcker för att komma igång med tjänsten vilket ger en låg ingångströskel för att bli medlem på tjänsten. Många tycker att det är jobbigt att fylla i många fält vid registrering, och dessutom kanske en del inte vill lämna ifrån sig personuppgifter om sig själv till främmande aktörer på internet som användaren inte vet så mycket om. Skulle PhotoChat släppas för en bredare publik skulle det även ha fält för mailadress, så att användaren kan återställa lösenordet ifall användaren glömmer detta.



Figur 4: Om användaren går in på att registrera nytt konto så kommer denna till ett väldigt litet formulär som ej ska vara jobbigt att fylla i. På sikt skulle återställnings-mailadress för bortglömda lösenord också kunna vara med.

Första välkomstkärmen

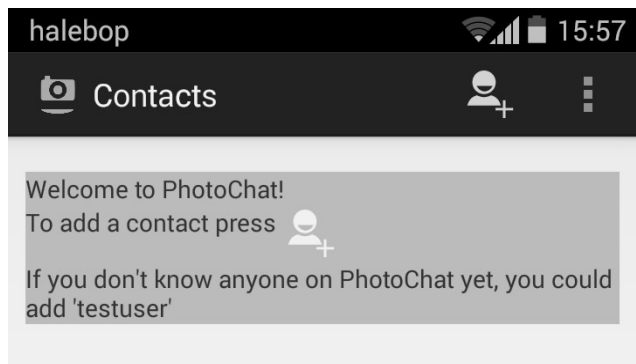
När användaren loggar in första gången möts denna av en välkomsthälsning. Denna berättar för användaren hur denne kan komma igång med appen. Detta genom att lägga till en kontakt med hjälp av knappen som ser ut som ett gubbe med ett plus på. Så fort användaren har minst en kontakt i kontaktlistan försvinner välkomstkärmen vid inloggning.

Lägg till kontakt

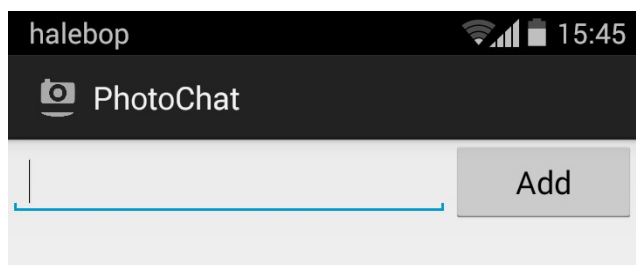
Trycker användaren på lägg till kontakt så kommer användaren till en ny skärm där användaren kan skriva in namnet på den kontakt denne vill lägga till. Detta skulle i framtiden kunna vidareutvecklas genom att till exempel automatiskt hitta kontakter via epostadress och telefonens adressbok. I dagsläget måste användaren dock veta användarnamnet på kontakten som användaren vill lägga till.

Kontaktlista

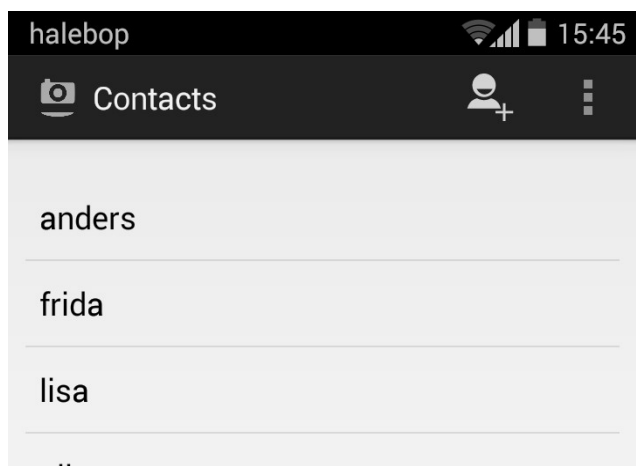
Kontaktlistan innehåller de kontakter som användaren har lagt till i bokstavsordning. Detta för att kontakterna ska gå snabbare att hitta bland. Om användaren trycker på en av kontakterna så kommer denne till en chattskärm mellan sig och den valda kontakten. Det går också att ta bort kontakter från kontaktlistan genom att trycka på kontakten med lång fingertryckning och välja ta bort i popup-menyn.



Figur 5: Välkomstkärm som möter användaren fram tills att denna har lagt till en kontakt



Figur 6: Trycker användaren på lägg till kontakt-knappen så kommer användaren till en ny skärm med ett fält för att skriva användarnamnet på kontakten som ska läggas till.

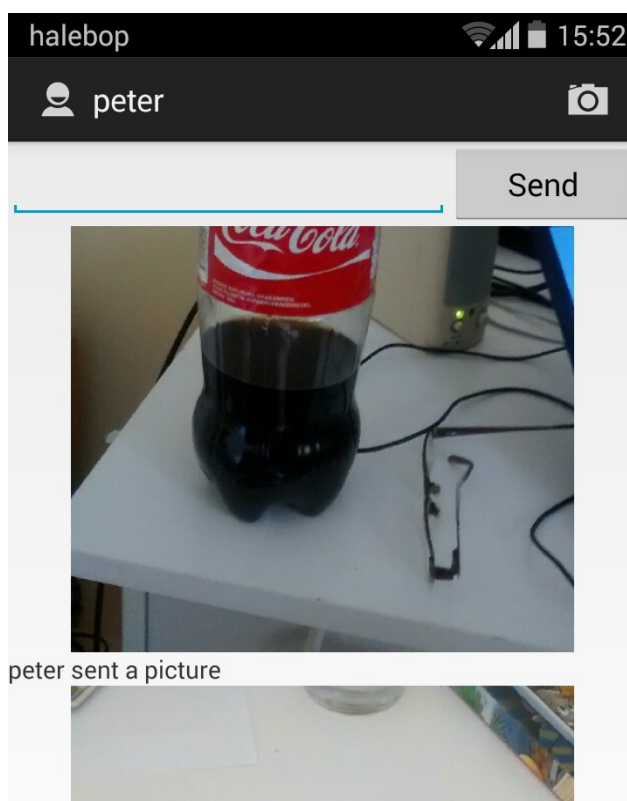


Figur 7: Kontaktlistan innehåller kontakterna i bokstavsordning

Chatten

Chatten fungerar som i många andra chattprogram med ett scrollande flöde med alla meddelanden.

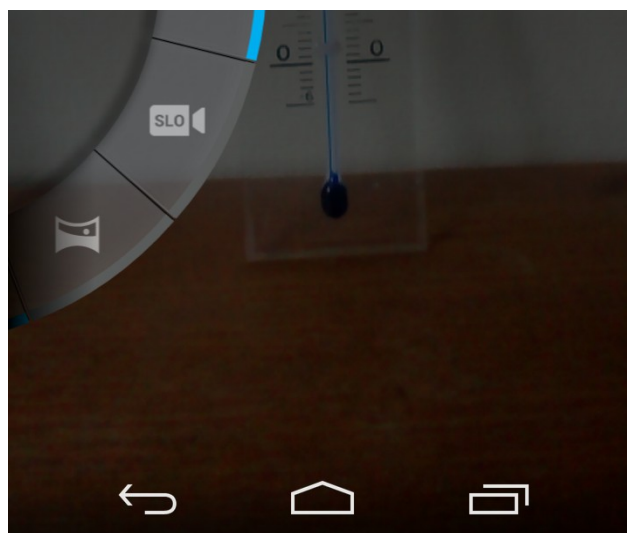
En sak som skiljer PhotoChat från många andra chatter är dock att PhotoChat scrollar neråt för äldre inlägg och har det senaste inlägget högst upp. Syftet med detta är dels att det är naturligt att börja högst upp när man kollar igenom ett dokument, och det som har skickats senast borde därför rimligtvis ligga högst upp då användaren vill läsa det som är aktuellt först. Chatten kan innehålla både text och bilder, men bilderna är stora och tar troligen upp den dominerande ytan i en fotochattkonversation.



Figur 8: Chatten innehåller ett scrollbart flöde, som dock till skillnad från många andra chattar går uppåt istället för neråt

Skicka foto

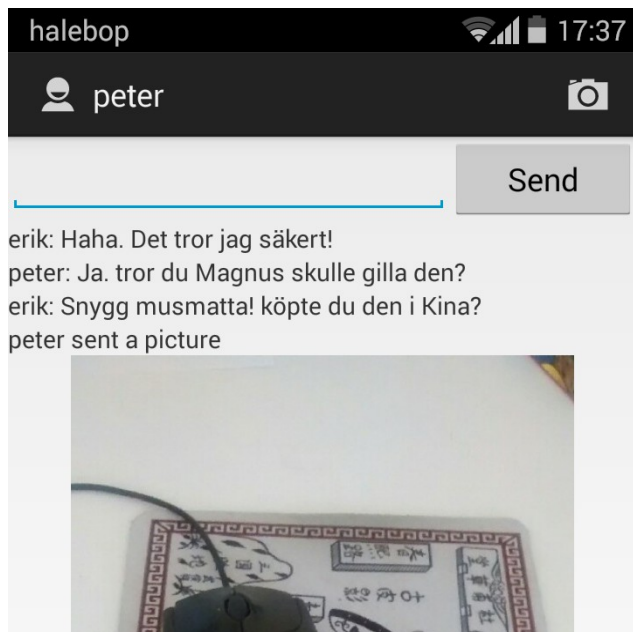
Den viktigaste funktionen i PhotoChat är som namnet antyder att skicka fotografier. För att göra detta använder appen telefonens inbyggda kameraapp för att ta fotografierna, och sedan skickar resultatet från denna app. Valet av den inbyggda kameraappen istället för att bygga en egen är så att användaren ska känna igen sig i användargränssnittet för kameran. Det ger också fördelen att användaren kan använda alla specialfunktioner som tex slå och av blix eller zooma in och ut om dessa finns i användarens kameraapp. Skicka-foto funktionen kommer användaren till genom att trycka på kameraknappen i actionbaren inne i på en chattskärmen som syns i figur 8.



Figur 9: För att ta kort anropar PhotoChat telefonens inbyggda kameraapp. Detta ska ge bästa igenkänningsfaktorn för användaren

Skicka text

Även om detta inte är själva huvudfunktionen för PhotoChat så finns fortfarande möjligheten att skicka textmeddelanden, vilket kan användas t.ex. för att diskutera bilder som skickats i appen. Text skickas från skickafältet som ligger högst upp i chattfönstret, precis under actionbaren. Se figur 10.



Figur 10: Förutom att skicka foton så går det fortfarande att skicka text med PhotoChat. Detta kan vara användbart för att t.ex. kommentera en bild

Hur PhotoChat fungerar

Nätverkss kommunikation

PhotoChat använder HTTP eller HTTPS för att kommunicera med sin server där chattmeddelandena och bilderna lagras innan dom når sitt mål. Det fungerar så att PhotoChat-appen skickar olika förfrågningar till servern för att utföra olika handlingar. Exempelvis skickar den inloggningsuppgifter om användaren ska logga in, eller med en bild om användaren ska dela med sig av en bild till tjänsten. För att hålla reda på att det är rätt användare som utför en viss handling så sparar servern sessionscookies som identifierar den unika användaren medan denne är inloggad. Svaren från HTTP-servern är oftast i JSON-format där detta är lämpligt(ej för t.ex. bilder), vilket gör det relativt hanterbart för mobilapplikationen att tolka datat och göra olika saker beroende på innehållet i informationen.

Databasmodell

Jag funderade först på att dokumentera datat i appen med ett klassdiagram i UML för att beskriva hur applikationen fungerar. Då applikationen är klient-server baserad så är det mycket av informationen som bara finns i databasen på servern, men inte i själva mobilapplikationen. Speciellt när det kommer till databehandlingen som inloggning, leverans av meddelanden etc. Eftersom mycket av datat bara lagras i databasen, men inte i mobilapplikationen så var ett UML klassdiagram av mobilapplikationen inte den bästa lösningen för att beskriva datat i applikationen som helhet.

Den mesta databehandlingen görs genom att klienten anropar PHP-skript på servern. PHP-skripten ställer sedan frågor till en MySQL-databas, och svaret från dessa frågor returneras till mobilapplikationen i form av JSON-data. I mobilapplikationen lagras bara minsta nödvändiga data som ska visas på skärmen. Jag har därför valt att göra ett databas-diagram istället för ett UML-klass diagram.

Data som finns i databasen men ej i mobilapplikationen:

Lösenordshash

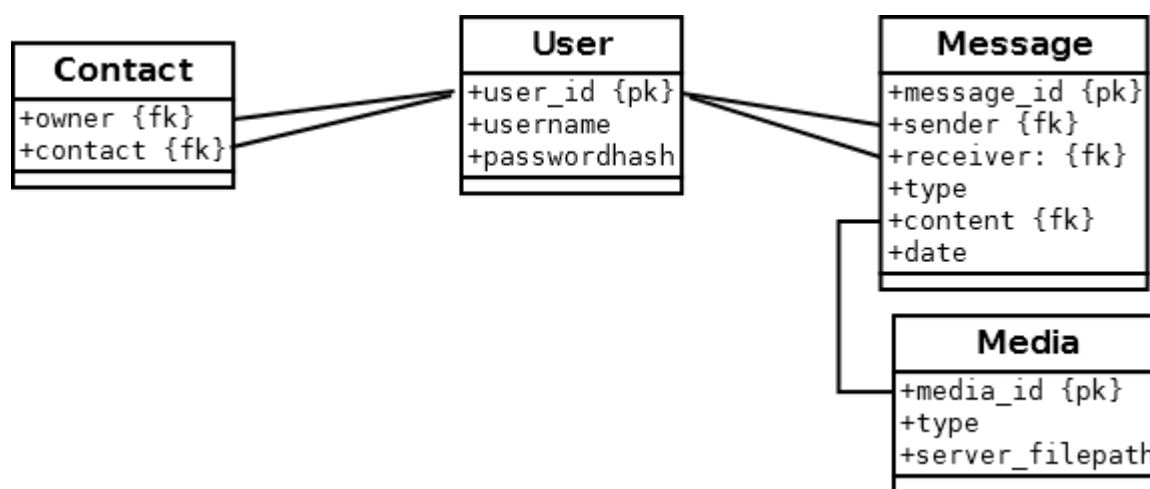
Eftersom lösenordshasharna ska vara hemliga såklart

Kontaktlistor till fler än en användare

På servern finns information om kontaktklistor för samtliga användare. På klienten finns detta bara för den just nu inloggade användaren

Id-nummer på användare

Eftersom jag vill ha möjlighet att senare byta ut den bakomliggande serverimplementationen utan att byta mobilapplikationen så är användarnas id-nummer helt osynliga för mobilapplikationen. Mobilapplikationen refererar istället till användare den känner till med hjälp användarnamn för olika anrop till servern. Det är enbart på serversidan som applikationen arbetar med id-nummer på användarna för kunna ställa rätt databasfrågor.



Figur 11: En översiktlig beskrivning av databasen

Här kommer en beskrivning på de tabeller som finns i databasen.

User-tabellen

Denna lagrar själva användarna. Denna har ett id-nummer för varje användare. Detta id-nummer finns inte i mobilapplikationen eftersom databasen ska vara transparent och utbytbar mot ett annat backend för mobilapplikationen. Förutom det finns ett användarnamn och en lösenordshash i tabellen. Lösenordet är hashat så att om även om någon inkräktare kommer över databasen ska det vara svårt att ta ut lösenordet. På sikt skulle applikationen även kunna ha saltade hashar för att förstärka säkerheten ytterligare

Contact-tabellen

Contacts tabellen lagrar alla användares kontaktlistor. För varje kontakt någon användare har på sin kontaktlista finns en rad i Contact-tabellen som säger vem som äger kontakten och vem kontakten är. Kontakter är envägs, vilket betyder att Olle kan ha Kalla på sin kontaktlista, utan att Kalle har Olle på sin kontaktlista. Detta designbeslut är gjort därför att om tex Olle tar bort Kalle som kontakt så vore det konstigt om Kalles kontaktlista också helt plötsligt saknar en kontakt. Jag tycker också det kan vara rimligt att en person har den andra tillagd, men den andra personen inte har den första.

På sikt är det tänkt att Olle ska behöva godkänna Kalle för att Kalle så få lägga till Olle på sin kontaktlista, men i den nuvarande prototypen använder applikationen en förenkling där Kalle

automatiskt även läggs till på Olles kontaktlista om Kalle lägger till Olle som kontakt

Message-tabellen

Message lagrar ett meddelande i systemet. Varje meddelande har en avsändare och en mottagare som båda pekar på User-tabellen. Varje meddelande har också ett unikt id där id alltid stiger med antalet skickade meddelanden. Detta gör att vi kan veta vilken ordning meddelandena kom i genom att sortera på id. Applikationen skulle i dagsläget fungera även utan unikt id för meddelanden eftersom det också går att sortera på datumet som innehåller en timetamp. Att ha ett unikt id för ett meddelande verkar dock vara en bra ide för eventuellt framtida utbyggnader när det ska gå att exempelvis anmäla ett meddelande som ej följer PhotoChats regler och unikt identifiera det. Type används för att säga vilken typ av meddelande det är. Det finns i dagsläget två typer: "text" och "image". Detta kan dock byggas vidare till flera typer i framtiden. Content fungerar olika beroende på vilken type som används. Content är en sträng som i de fall meddelandet är ett textmeddelande lagrar texten i meddelandet. I de fall där det som meddelandet innehåller en bild lagras istället ett unikt id som identifierar den specifika bilden. Att säga att Content är en foreign key till media som visas i figur 11 är därför en förenkling. Content används dock i alla fall utom när meddelandet är ett textmeddelande som foreign key.

Media-tabellen

Media innehåller referensinformation till en mediafil som finns sparade på servern. Den säger vilket id en mediafil har och vad för file-path på servern som filen har. Den säger också vilken typ av media det är. Man kan tycka att det ser lite redundant ut att "type" finns både i meddelandet och i media. Detta är för att det i efterhand ska gå att arbeta med den media som har laddats upp utan att det behöver vara ihopkopplat med ett specifikt meddelande. För att undvika kollisioner i filnamnen på servern så sparas alla filer med ett namn som är deras md5summa. Filnamnet på filen som laddas upp spelar alltså ingen roll, eftersom detta inte sparas. Detta system är till för att två filer som heter samma inte ska skriva över varandra såvida de inte är exakt samma fil.

Datalagring på klientsidan

Väldigt lite data lagras på klientsidan, vilket är varför jag istället valt att fokusera den dokumentationen av datamodellen på serversidan. Av den data som ligger hos klienten i mobilapplikationen ligger det i AppState klassen som ärver från Application. Detta eftersom det är det enda stället i mobilapplikationen där data sparas mellan aktiviteter.

Viktig data som lagras hos klienten i klassen AppState är:

private String loggedin

Denna lagrar användarnamnet på den inloggade användaren och är null om ingen användare är inloggad

private ArrayList<String> contacts;

Denna lagrar en lista på användarnamn för användarna i kontaktlistan. Det behövs ingen mer information om användarna än så på klienten eftersom all databehandling sker på serversidan.

private HashMap<String, Conversation> conversation_map;

Denna lagrar en cache på de konversationer som användaren deltar i. De riktiga konversationerna sparas på servern, men för att inte hela tiden behöva ladda om konversationerna finns detta också cashat hos klienten i en hashmap där Sträng-nyckeln säger vilken användare konversationen hör till, och Conversation objektet innehåller själva meddelandena.

På klientsidan finns två klasser som är till för att lagra data:

Message-klassen

Denna lagrar meddelanden som kommer från Message tabellen. Denna klass har en ganska bra överensstämmelse med databastabellen med samma namn.

Conversation-klassen

Denna lagrar konversationen mellan den inloggade användaren och en annan användare. Denna innehåller bara i princip en array av Message och ett gränssnitt för att hantera meddelandena utifrån.

Etik och säkerhet

Internetmobbing och annat missbruk av tjänsten

Internet och social media har blivit en del av folks vardag. Något som tyvärr också blivit vardag med det är mobbing som sker över internet. Företeelsen kallas ofta internetmobbing eller cyber bullying på engelska.

En stor social plattform där internetmobbing förekommer är SnapChat(Sherri Gordon, 2014). Eftersom PhotoChat är baserat på liknande idéer som SnapChat är det av högsta vikt att även PhotoChat tar risken för internetmobbing på allvar. PhotoChat bör göra allt det PhotoChat som tjänsteleverantör kan göra för att förhindra att mobbing förekommer på tjänsten. Konkurrenten SnapChat har den egenskap att en bild automatiskt försvinner inom 10 sekunder. Det är därför vanligt att förövare av internetmobbing använder SnapChat då det är lätt att komma undan, eftersom beviset på en gång förstörs efter att det öppnats(SnapChat Bullying Tactics).

Just denna metod att internetmobba kommer troligen inte bli så utbred på PhotoChat eftersom bilderna på PhotoChat finns kvar efter att bilden har skickats. Bilderna sparas även på PhotoChats servrar efter att det bilden har skickats vilket gör att bilderna går att gå tillbaka till och använda som bevismaterial.

Rapportering av olämpligt material

PhotoChat ska när det släpps arbeta på ett liknande sett som Facebook med att bekämpa missbruk av tjänsten. Facebook använder ett system med möjlighet för användarna att rapportera missbruk av tjänsten direkt till Facebook(Facebook community standards).

När en användare på PhotoChat hittar material som denna finner olämpligt så ska det finnas ett en valmöjlighet för användaren att på ett enkelt sätt rapportera det direkt till PhotoChats administratörer. Det kommer också gå att skriva en bifogad text som förklarar vad som är fel med materialet som rapporteras.

Om administratören finner att materialet strider mot PhotoChats policy så kommer materialet att tas bort. Beroende på hur allvarlig överträdelsen är kan användaren som utfört denna antingen få en varning eller få kontot avstängt från PhotoChat. Om materialet som anmäls även är av mycket allvarlig karaktär, exempelvis barn pornografi, kommer PhotoChat polisanmäla användaren som laddat upp materialet och PhotoChat kommer i detta fall även att dela med sig av användarens IP nummer till polisen.

Information om exakt vad som ej är tillåtet kommer att framgå i PhotoChats policy som måste

accepteras för att registrera sig på tjänsten. Som grund kan sägas att allting som är olagligt i det land PhotoChats servrar befinner sig i, alltså Sverige, samt i det land där användaren befinner sig är förbjudet även på PhotoChat. Även internetmobbing och rasistiska budskap är förbjudna att förekomma på PhotoChat oavsett om det faller innanför lagens ramar för detta land eller ej.

Appens behörigheter

För att utsätta användaren för så liten risk som möjligt när denne installerar applikationen så har jag valt att försökt hålla ner mängden behörigheter som appen använder till ett minimum.

Applikationen använder tre behörigheter. Dessa är

För internet: **android.permission.INTERNET**

Eftersom appen behöver internet för att kommunicera med en server för att logga in, kontakta andra användare och skicka bilder. Dessa är applikationens huvudfunktioner.

Externt lagringsutrymme: **android.permission.WRITE_EXTERNAL_STORAGE** och **READ_EXTERNAL_STORAGE**

Appen i sig behöver inte tillgång till externt lagringsmedia, men appen interagerar med kameraapplikationen och behöver för att få säga åt denna att spara undan bilden tillgång till externa lagringsutrymmet. I Android 4.4 Kitkat har jag på test lyckats få bort kravet på externt lagringsutrymme, eftersom det där går att använda en speciell avsatt plats för att mellanlagra filer utan speciella behörigheter. Detta går dock inte i Android tidigare än 4.4. Då applikationen ska fungera på androidversioner tidigare än 4.4 Kitkat kan applikationen inte använda mellanlagring på externt minne utan behörighet.

Krypterad trafik

All trafik kör över HTTP som vid skarp release av programmet ska bytas till HTTPS för att kryptera all trafik mellan användaren och servern. För testversionen som nu är i drift så har jag ej tillgång till ett HTTPS certifikat. Demoversionen av PhotoChat kör därför över vanlig HTTP. Detta är dock väldigt lätt att byta eftersom kommunikationen i PhotoChat använder Javas standardbibliotek för HTTP, och dessa har inbyggt stöd för HTTPS redan från grunden.

Tekniska utmaningar under projektet

Trådar

När jag har utvecklat PhotoChat har jag behövt ta mig förbi ett flertal tekniska utmaningar. Den största har varit trådar. Nätverkskommunikation på Android får inte ske i huvudtråden som användargränssnittet ligger i eftersom detta skulle bromsa upp användargränssnittet medan programmet väntar på nätverkstrafik(Android Developer - Connecting to the Network). Man måste därför lägga all nätverkskommunikation i en separat tråd. I början försökte jag bygga detta med klassiska Javatrådar av typen Thread. Detta lede till en hel del svårigheter som tex att nätverkstrådarna inte fick tillgång till användargränssnittet och inte kunde presentera inkommande data för användaren. Detta gick att komma runt med metoden `runOnUiThread`, men även med denna lösning fick jag många typer av problem med och konstiga krascher. Efter mycket utforskande så kom jag i slutändan fram till att Androids egna klass `AsyncTask` var det bästa och enklaste verktyget att utveckla med multipla trådar på Android.

Uppladdning av filer via HTTP

Detta kan man tro skulle vara lätt att lösa med Javas HTTP-bibliotek, men så var ej fallet. Det finns inget inbyggt sätt i Javas standardbibliotek just för att posta filer till en webbserver. Det tog mig många dagar att läsa på hur jag skulle gå tillväga, genom att manuellt sätta diverse olika värden i HTTP-headern för att posta en fil. Till sist var det ett antal bra svar från olika användare på Stackoverflow som fick mig att få ihop en fungerande filuppladdning.

Rotation av bilder

Detta var ett problem jag inte hade tänkt på innan jag började utveckla. Saken är den att olika kameror tar bilder med olika rotation och vad som är neråt i en bild är inte alls så självklart som man kan tro. Neråt kan vara på vilken kant som helst i en bildfil som har samma rådata. Detta är väldigt förvirrande och jag fick lära mig om något som hette EXIF som används tillsammans med JPEG för att bestämma vad som är ner i en bild. Detta problem uppkom först när jag skalade bilden via Androids verktyg för att hantera Bitmapgrafik. Efter skalningen så var ofta ner åt fel håll. Genom att extrahera ut värden från EXIF kunde jag bestämma hur en bild skulle roteras för att få rätt sida neråt i bilden igen(Rotate images according to EXIF info, 2008).

Utvecklingsmöjligheter

Skulle jag utveckla detta projekt vidare skulle nästa steg vara att implementera stöd för Google Cloud Messeging, GCM som är ett ramverk för att kunna skicka notifikationer ut till appar över internet. GCM kräver inte att mottagarappen är igång i förväg när ett GCM meddelande skickas, utan GCM säger istället åt Android att starta upp rätt applikation som meddelandet tillhör, och med rätt parametrar. Detta gör att man snabbare skulle kunna nå användare som ej har PhotoChat igång för stunden, genom att skicka en push-notis om att användaren fått ett nytt meddelande till appen. Många stora appar använder i dagsläget Google Cloud Messeging för bakgrundskommunikation, som exempelvis Whatsapp, Viber och Facebook (Niroj Manandhar, 2014).

Källförteckning

Adrian Ber. (2008). *Rotate images according to EXIF info*. Available: <http://beradrian.wordpress.com/2008/11/14/rotate-exif-images/>. Last accessed 2014-08-28.

Facebook. *Facebook community standards*. Available: <https://www.facebook.com/communitystandards>. Last accessed 2014-09-12.

Google. *Connecting to the Network*. Available: <https://developer.android.com/training/basics/network-ops/connecting.html>. Last accessed 2014-08-28.

Niroj Manandhar. (2014). *List of Google Cloud Messaging (GCM) enabled apps for Greenify Pro Users*. Available: <http://www.jucktion.com/tech/mobile/android/google-cloud-messaging-gcm-enabled-apps-list/>. Last accessed 2014-08-28.

No Bullying. *SnapChat Bullying Tactics*. Available: <http://nobullying.com/snapchat-bullying-tactics/>. Last accessed 2014-09-12.

Sherri Gordon. (2014). *How Kids Are Misusing Snapchat to Sext and Cyberbully*. Available: <http://bullying.about.com/od/Cyberbullying/a/How-Kids-Are-Misusing-Snapchat-To-Sext-And-Cyberbully.htm>. Last accessed 2014-09-12.

Time. (2014). *This Chart Explains Why You're Not on Snapchat*. Available: <http://time.com/50384/snapchat-users-under-25-female/>. Last accessed 2014-08-28.