Danmarks Tekniske Universitet

Nicolai Guldager, s082722
Oliver Egeris Groth, s083472

Hvordan Facebook kan udnytte Bluetooth i en smartphone

Bachelorprojekt, juni 2011

IMM-B.Sc.-2011-17

DTU Informatik
Institut for Informatik og Matematisk Modellering

Nicolai Guldager, s082722
Oliver Egeris Groth, s083472

Hvordan Facebook kan udnytte Bluetooth i en smartphone

Bachelorprojekt, juni 2011

IMM-B.Sc.-2011-17

Hvordan Facebook kan udnytte Bluetooth i en smartphone

Denne rapport er skrevet af:

Nicolai Guldager, s082722
Oliver Egeris Groth, s083472

Vejledere:

Jakob Eg Larsen
Sune Lehmann Jørgensen

Udgivelsesdato:	27. juni 2011
Kategori:	1 (offentlig)
Udgave:	Første
Kommentarer:	Denne rapport er skrevet ved IMM – Institut for Informatik og Matematisk Modellering, Danmarks Tekniske Universitet i perioden 10. marts 2011 til 27. juni 2011. Dette er det afsluttende projekt i B.Sc. uddannelsen. Projektet svarer til 15 ECTS point pr. deltager.

Institut for Informatik og Matematisk Modellering
Danmarks Tekniske Universitet
Bygning 321
DK-2800 Kgs. Lyngby
Danmark

www.milab.imm.dtu.dk
Tel: +45 45 25 33 51 / Fax: +45 45 88 26 73
E-mail: milab@imm.dtu.dk

Abstract

In this thesis we describe a new type of mobile application. The application is uniting the social elements of Facebook and the mobility of a smartphone, together with the opportunities provided by the utilization of Bluetooth.

The outcome of the analysis in this thesis is a mobile application, which by scanning the nearby area of the user and using data from Facebook presents an overview of the people nearby.

After a scan of nearby people, the application will sort the result by the number of mutual friends. For each person a profile is shown where the user can see the likes and mutual friends. Furthermore it is possible to post a message on that person's wall.

To store the information about the user and their smartphone the application uses an online database. Because of this it is possible for the application to identify other users without them having the application running. The only requirement is that the device has Bluetooth turned on.

The application has been developed with the purpose of demonstrating the concept. Ultimately, a minor user survey has been conducted, in which several great ideas have given a direction for the further development of the application.

Resumé

Denne rapport beskriver en ny type mobilapplikation, som forener de sociale elementer i Facebook med mobiliteten i en smartphone og de muligheder, som brugen af Bluetooth åbner op for.

Ud fra analysen i rapporten er der blevet udviklet en mobilapplikation, der skanner brugerens nærområde og med data fra Facebook præsenterer en oversigt over, hvem der befinder sig i nærheden. På oversigten vises profilbillede, navn og antallet af fælles venner.

Applikationen sorterer resultatet af en skanning efter antallet af fælles venner. For hver person vises desuden en profilside, hvor det kan ses hvilke "Synes godt om" personen har, samt hvilke venner der er fælles. Endelig er det muligt at skrive en besked på personens væg.

Applikationen benytter en online database til at gemme oplysninger om brugeren og vedkommendes smartphone. Dette har muliggjort, at applikationen kan identificere andre brugere, uden at de behøver have applikationen kørende på samme tid. Eneste forudsætning er, at enheden er synlig via Bluetooth.

Applikationen er udviklet med det formål at demonstrere selve konceptet. En mindre brugerundersøgelse er afslutningsvis udført, hvilket har givet gode idéer til, i hvilken retning applikationen fremover kan udvikles i.

Forord

Denne rapport er skrevet ved IMM – Institut for Informatik og Matematisk Modellering, ved Danmarks Tekniske Universitet i perioden 10. marts 2011 til 27. juni 2011. Dette er det afsluttende projekt i B.Sc. uddannelsen. Projektet svarer til 15 ECTS point pr. deltager.

Rapporten omhandler udviklingen af funktionalitet til mobilapplikationen for det sociale netværk Facebook. Det færdige produkt er en mobilapplikation til Android-baserede telefoner og enheder, som demonstrerer konceptet for vores idé.

Vejledere for projektet er Jakob Eg Larsen og Sune Lehmann Jørgensen, Institut for Informatik og Matematisk Modellering, Danmarks Tekniske Universitet.

Nicolai Guldager Oliver Egeris Groth

Indhold

Kapitel 1

Indledning

1.1 Motivation

Internettet og de forskellige sociale netværk har ændret måden hvorpå vi kommunikerer og interagerer med andre mennesker. Særligt Facebook har formået at tiltrække mange brugere – omtrent hver 14. menneske på kloden bruger Facebook mindst én gang om måneden [1] – og det er da også her vi har vores fokus.

Med Facebook blev det muligt at opbygge sin egen online identitet. Facebook er meget bredt funderet, og henvender sig ikke til en specifik gruppe, og er ikke som MySpace fokuseret på musikverdenen eller som LinkedIn der henvender sig til folk indenfor erhvervslivet. Disse tjenester har deres helt egne styrker, men når det gælder popularitet, antallet af brugere og disses aktivitet, er Facebook førende [1].

Samtidig med den enorme popularitet for de sociale netværk, er sket en stor stigning i salget af telefoner med mulighed for installation af software – smartphones[1]. Dette har motiveret de sociale netværk til også at rykke ind på denne, relativt nye platform.

Det er her vores projekt tager sin begyndelse.

Mobilapplikationen for Facebook er i bund og grund en mobil udgave af hjemmesiden. Der er ikke noget nyt i den, og der er ingen udnyttelse af mobiltelefonernes mange muligheder. For nylig[2] er Facebook Places[3] blevet rullet ud, men der er i vores øjne stadig en uudforsket funktionalitet, som samler enderne mellem Facebooks sociale styrker og mobiltelefonens mobile muligheder. Analysefirmaet Gartner har udarbejdet en liste over de 10 typer mobilapplikationer, der i deres optik er værd at holde øje med

[1] http://blog.nielsen.com/nielsenwire/consumer/smartphones-to-overtake-feature-phones-in-u-s-by-2011/
[2] Udrulning startede den 18. August 2010, se http://www.facebook.com/blog.php?post=418175202130
[3] Facebook Places udnytter telefonens GPS modtager, og deler ens position med andre.

i 2012. På denne liste nævnes netop lokations-bestemte services og sociale netværk. Ligeledes understreges det, at *"Winning mobile apps will have unique features that cater to the mobile environment rather than act as a mobile extension of their online peers."*[2].

1.2 Problemformulering

Vi vil udvikle en mobilapplikation til smartphones. Den primære funktionalitet i applikationen vil være brugen af enhedens Bluetooth radio sammen med en kobling til Facebook Graph. Således kan der modtages data fra andre brugere af applikationen, som befinder sig i umiddelbar nærhed, og på den måde er det muligt at generere en oversigt over de Facebook profiler, som brugeren befinder sig tæt ved. Vi vil fokusere på at udvikle en helt grundlæggende "proof of concept" applikation i dette projekt, men samtidig undersøge hvilke muligheder for yderligere funktionalitet der senere kan implementeres.

1.3 Afgrænsning

Vi vil udvikle en mobilapplikation, der ved hjælp af enhedens Bluetooth radio lader brugeren se, hvem der befinder sig i umiddelbar nærhed, samt hvilke relationer de har til fælles. Formålet med applikationen er at demonstrere konceptet, og fokus er derfor på helt grundlæggende elementer. Med en mindre brugerundersøgelse er det afslutningsvis ønsket at få respons på konceptet og modtage ideer til yderligere funktionalitet.

Kapitel 2

Analyse

Problemformuleringen indeholder formålet med projektet, men ikke vejen dertil eller det eksakte indhold i applikationen. For at kunne udarbejde en række passende krav, ser vi i dette kapitel på sociale netværk og mobilapplikationer som har beskæftiget sig med lignende problemstilling. Endelig analyseres og sammenlignes flere forskellige platforme til udvikling af mobilapplikationer, og vi ser på hvordan vi får adgang til data fra Facebook, verificerer brugerne og på, hvad der gør Bluetooth til en egnet teknologi.

2.1 Sociale netværk

Et socialt netværk udgøres af flere individer, som sammen udgør en gruppe. Der er grundlæggende to typer af sociale netværk, offline og online. At være social offline er som vi altid har kendt det, når man møder andre i skole, supermarkedet, sportshallen, på jobbet eller til fest. Når flere individer samles i en gruppe, udgør de et socialt netværk. I daglig tale vil de fleste imidlertid forbinde begrebet sociale netværk med det at være ”online”. I takt med computerens og i særdeleshed internettets fremmarch, er der opstået nye og enestående muligheder for social interaktion på tværs af landegrænser, kulturer, religioner og sprog. At interagere socialt online er et vidt begreb, som dækker hele paletten af muligheder: finde nye venner, opstøve gamle klassekammerater, diskutere alverdens emner med ligesindede, holde videokonferencer med forretningspartnere eller sågar kontakte og bestille en russisk husholderske. Mængden af muligheder og typer af sociale netværk begrænses kun af de fælles behov og ønsker, som alle verdens brugere af internettet tilsammen definerer. Denne sum er enorm, og det er derfor kun naturligt at der er opstået en mængde forskellige online tilbud, som hver i sær henvender sig til forskellige grupper og søger at samle forskellige typer af individer. Nogle henvender sig bredt og inviterer alle, mens andre udgør en niche. Som eksempel kan nævnes Facebook, der med sin

halve milliard brugere [5] er den største spiller på markedet og henvender sig til alle, til forskel fra f.eks. LinkedIn, som søger at samle individer om deres professionelle liv og uddannelse og skabe netværk indenfor disse områder.

Sideløbende med den hastige udvikling af sociale netværk er sket en stor udbredelse af mobilt internet. I store dele af verden er telekommunikationsnettet opgraderet til at kunne transmittere store datamængder til mobile enheder – ud af verdens 5.3 milliard mobiltelefonabonnenter har næsten 1 milliard adgang til 3G netværk [3]. Dette giver mulighed for, at de sociale netværk kan rykke ind på den mobile platform i en kvalitet, og med svarhastigheder, som kan tilfredsstille selv krævende brugere. Samtidig sælges stadigt flere smartphones – telefoner, som understøtter at brugeren selv installerer software og således kan benytte applikationer. I Danmark anslås det, at 33 % af befolkningen ejer en smartphone [4]. Alle disse elementer gør, at det er helt essentielt for udbyderne af sociale netværk, at de tilbyder og tilpasser deres service til brugere med smartphones.

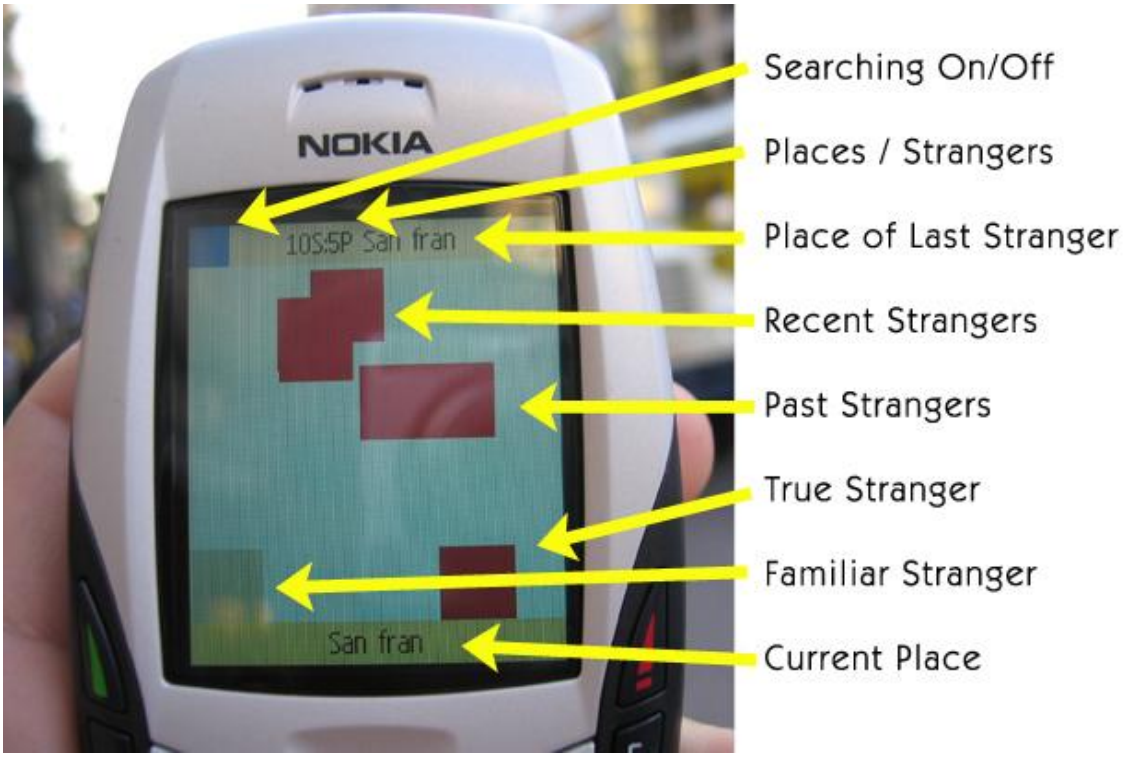
Facebook selv anslår, at mere end 250 millioner brugere tilgår Facebook via en mobil enhed [5].

2.2 Tilsvarende applikationer

Når det gælder selve ideen om at kunne præsentere, hvem der befinder sig i nærheden af hinanden, er dette projekt første skud på stammen. Så vidt vores søgninger rækker, eksisterer der ikke andre applikationer, som kombinerer Bluetooth med Facebook.

Gruppen "Urban Atmospheres", som består af folk fra bl.a. "Intel Research Berkeley"[4], har dog gang i en samling forskellige projekter som undersøger de nye muligheder indenfor mobilitet og socialisering. De har bl.a. udviklet applikationen Jabberwocky, som ved hjælp af både GPS og Bluetooth indsamler data om enheder i nærheden. Brugeren præsenteres for resultatet ved, at den indsamlede data visualiseres ud fra flere parametre – møder man fx samme fremmede hver morgen i bussen, indikeres dette med en bestemt farve og position på displayet (se Figur 2.1).

[4] http://berkeley.intel-research.net/



Figur 2.1: Jabberwocky GUI[5]

Konceptet er således set tidligere, men endnu ikke med profilvisning af selve personen, i stedet for blot firkantede kasser.

Umiddelbart ville det være oplagt også at integrere telefonens GPS antenne i vores applikation. Det vil give mulighed for yderligere implementering af smarte funktioner, fx vil det være interessant at hente inspiration fra netop Jabberwocky, og vise brugeren hvor og hvornår man tidligere er stødt på en person.

Facebook Places (se Figur 2.2) er et andet bud på en eksisterende funktionalitet, som minder om det vi sigter efter – blot benyttes GPS, og ikke Bluetooth. Facebook Places eksisterer i dag som en del af den officielle Facebook applikation. Hver bruger kan selv angive, hvor (og med hvem) han eller hun befinder sig. Andre brugere kan herefter få vist vedkommendes "check in" [6].

[5] Billede lånt fra http://www.urban-atmospheres.net/Jabberwocky/demo.htm

Figur 2.2: Screenshot af Facebook Places

Den store styrke ved Facebook Places er den kombination af mobilitet og Facebook, som vi sigter efter i dette projekt. Den store svaghed er, efter vores mening, nøjagtigheden – hvor Bluetooth er så geografisk afgrænset, at en søgning kun bør returnere andre brugere, som rent fysik er indenfor synsvidde, kan et "check in" på fx Roskilde Festival betyde, at andre "check ins" befinder sig et par kilometer væk.

I samme boldgade findes også Google Latitude[6] og foursquare[7].

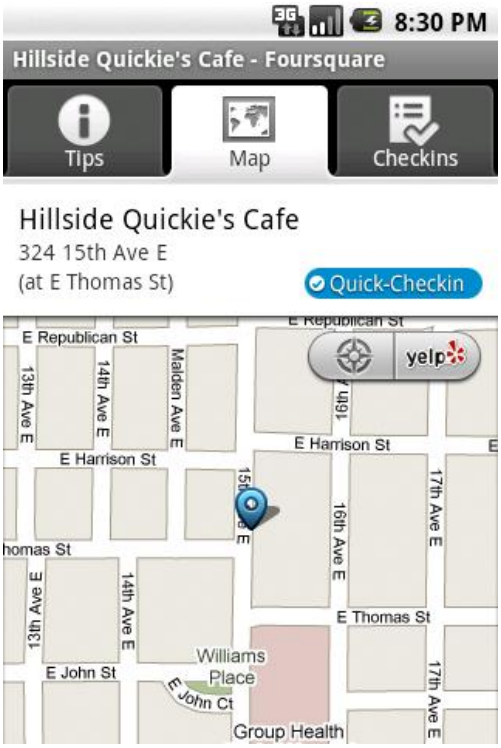
Latitude startede i 2009[8], og er en tjeneste som lader brugere med en Google konto registrere deres position på et kort (se Figur 2.3). Tjenesten er udviklet som en service til korttjenesten Google Maps. Det er muligt at tjekke ind både fra browser og fra smartphone [7].

[6] www.google.com/latitude
[7] https://foursquare.com
[8] http://en.wikipedia.org/wiki/Google_Latitude

Figur 2.3: Screenshot af Google Latitude[9]

foursquare (se Figur 2.4) blev ligeledes udviklet i 2009. Manden bag, Dennis Crowley, har tidligere udviklet Dodgeball, som senere blev købt af Google og blev til Google Latitude[10]. foursquare lader brugeren indlæse venner fra bl.a. telefonbogen, Facebook og Twitter. Desuden er implementeret et system, hvor brugere kan tjene "badges", en slags point, hvilket er med til at fastholde brugernes interesse, da det således er attraktivt at tjekke ind ofte [8].

[9] http://www.google.com/mobile/latitude/
[10] http://en.wikipedia.org/wiki/Dodgeball_%28service%29#Dodgeball



Figur 2.4: Screenshot af foursquare[11]

Facebook Places, Latitude og foursquare bygger altså på samme ide, men med fokus på forskellige elementer:

- Facebook Places fokuserer på at vise hvilke af brugerens venner, som befinder sig i nærheden (og hvor). Er efter vores mening bedst egnet til listevisning.
- Google Latitude viser kun brugere med Google konto. Den grafiske visning er meget udbygget, og der gives et godt overblik med kortvisningen.
- foursquare lader brugerne indlæse venner fra forskellige medier og sociale netværk. Desuden belønnes brugere som ofte tjekker ind.

[11] Billede lånt fra http://areacellphone.com/2010/05/htc-evo-incredible-download-10-top-android-apps-htc-evo/

2.3 Mobile platforme

I forbindelse med applikationsudvikling til smartphones, er det nødvendigt at gøre sig overvejelser omkring, hvilken platform man vil udvikle til, alternativt udvikle til flere samtidig vha. cross-platform værktøjer. I dette projekt har vi valgt at begrænse os til at udvikle til én platform. I det følgende er fokus derfor på de udviklingsværktøjer, som producenterne selv anbefaler til udviklere, og ikke på udviklingsværktøjer til flere platforme.

Der er mange faktorer, der spiller ind på valget af platform. Vi beskriver både fordele og ulemper ved nogle af de store mobilplatforme på markedet såsom Android, BlackBerry, iOS og Mobile Phone 7. Vi har valgt ikke at medtage Nokias Symbian, da denne platform er ved at blive udfaset af Nokia til fordel for Windows Phone 7[12].

For hver platform gennemgås følgende:

- Bluetooth
- Programmeringssprog
- Udviklingsmiljø
- Adgang til markedet
- JSON understøttelse (Facebook Graph returnerer JSON)

Derudover vil vi også komme ind på, hvordan man kan tilgå Facebook Graph (se afsnit 2.4 Facebook Graph) på de forskellige platforme, i og med at det er en af grundstenene i vores projekt og dermed af stor betydning for valget af platform.

2.3.1 Android

Den største mobile platform på markedet i øjeblikket er Android [9]. Udvikling til Android foregår primært i Java, dog kan også C bruges, vha. Androids eget Native Development Kit (NDK). Udvikling til Android kan ske på både Windows, Mac og Linux, og i flere forskellige udviklingsmiljøer. Eclipse er imidlertid det eneste Google selv understøtter. Udover Eclipse er det også nødvendigt at bruge Androids Application Programming Interface (API) samt forskellige udviklingsværktøjer, fx Android Virtual Device (AVD), som kan benyttes til at teste applikationer under udvikling, uden

[12] http://press.nokia.com/2011/05/31/nokia-lowers-devices-services-second-quarter-2011-outlook-and-updates-full-year-2011-outlook/

nødvendigvis at investere i fysiske Android-enheder. En gratis guide forefindes på Androids udviklerside[13]. Dokumentationen er således på plads og let tilgængelig. Med Android API kan man både tilgå enhedens Bluetooth [10] og konvertere de JSON objekter som Facebook Graph returnerer [11].

Princippet med Android er, at det skal være et åbent styresystem. Dette giver den fordel, at man med Android API til Java kan tilgå de fleste af enhedens funktioner. Dem der ikke umiddelbart er tilgængelige, er der adgang til vha. NDK og en root (superbrugeradgang) af enheden. Dette ophæver dog enhedens garanti.

At Android er på markedet i mange forskellige versioner, er til dels en ulempe for udviklere og brugere. Der eksisterer ingen regler om, hvorvidt producenterne skal blive ved med at opdatere enhederne, efter de er blevet udgivet. Dette giver udviklere en udfordring, i forhold til at få deres applikationer ud til et stort publikum, i kraft af, at det er nødvendigt at sørge for kompatibilitet med flere forskellige versioner.

Det er med Android ikke påkrævet, at brugerne henter sine applikationer fra det dedikerede Android Market. Det betyder, at udviklere kan have forskellige måder at udbyde deres applikationer på, og dermed at Android ikke har kontrol med, hvilke typer applikationer der bliver udviklet, og hvilken kvalitet de har. Det er på sin vis positivt for udbredelsen af applikationer, men dog på bekostning af konsistens applikationer imellem.

For at udgive applikationer på Android Market, skal man registreres som udvikler og betale et registreringsgebyr. Herefter er der adgang til at uploade applikationer på Android Market, uden nogen yderligere godkendelsesproces. Brugerne kan således ikke vide sig sikre på, om applikationerne er kompatible med netop deres enhed.

[13] http://developer.android.com/index.html

2.3.2 BlackBerry

Udvikling til BlackBerry kan foregå i blandt andet Java og web-udvikling (HTML, JavaScript osv.), og kan derfor udvikles i både Eclipse og Visual Studio vha. enten Research In Motions (RIM) plug-in til Eclipse, eller BlackBerrys eget WebWorks plug-in til begge miljøer. RIM har valgt at begrænse dele af styresystemet, hvilket kun gør det muligt at teste applikationer "On Device" ved at få en signing key fra RIM, for at tilgå disse dele af systemet. Alternativt kan en simulator bruges til test. BlackBerry understøtter, som Android, også både Bluetooth[14] og JSON konvertering[15].

For at udgive applikationer til App World, skal udviklere registreres og indsende applikationer til RIM, som tester, at de fastlagte retningslinjer til publicering følges[16]. Dette er gratis og gør det simpelt for udviklere at udgive applikationer. Samtidig er der en sikkerhed for brugerne, idet at applikationerne er godkendte og dermed må forventes at være sikre at hente og benytte.

2.3.3 iOS

Tredjestørst på smartphonemarkedet er iPhone med iOS [9]. Udviklingen til iOS sker i Objective-C og med frameworket Cocoa Touch. Udvikling til iPhone er udelukkende muligt ved hjælp af Apples egne værktøjer, bl.a. benyttes Xcode til kodning og Interface Builder til design af GUI. Begge værktøjer eksisterer kun til Mac. Der findes alternative løsninger ved at benytte VirtualBox i Windows[17], men dette er forbundet med flere besværligheder, bl.a. understøttes kun Intel CPU'er.

For at teste On Device koster det et årligt gebyr på $99, som også giver adgang til at udgive på App Store. iOS understøtter, modsat Android og BlackBerry, ikke JSON som standard. Det er imidlertid muligt med tredjeparts frameworks som "json-framework"[18] at få iOS til at understøtte dette. Derimod understøtter iOS Bluetooth via Apples egen Game Kit framework[19].

[14] http://www.blackberry.com/developers/docs/4.3.0api/index.html
[15] http://www.blackberry.com/developers/docs/6.0.0api/org/json/me/package-summary.html
[16] https://appworld.blackberry.com/isvportal/home/guidelines.seam
[17] http://www.virtualbox.org/manual/ch03.html#intro-macosxguests
[18] https://github.com/stig/json-framework/
[19] http://developer.apple.com/technologies/ios/networking.html

Apple har valgt at begrænse adgangen til visse af telefonens funktioner. Det har den ulempe for udviklere, at de kun kan tilgå de funktioner, som Apple giver adgang til. For at udgive applikationer på App Store, skal den færdige applikation igennem en godkendelsesproces hos Apple, hvor det kontrolleres om den er bygget op efter Apples designretningslinjer, og om den potentiel er skadelig for brugeren eller enheden. Denne lidt bureaukratiske tilgang giver imidlertid en stor fordel for brugerne, som dermed kan vide sig sikre på, at alle applikationer fra App Store er uskadelige for systemet [12].

At godkendelsesprocessen er omstændelig opvejes af, at det kun er Apple der producerer enheder med iOS, og endda med lang tid mellem hver ny model. Der er således et overskueligt antal forskellige versioner som udviklerne skal supportere, og applikationer kan have en lang levetid og en vis grad af fremtidssikring.

2.3.4 Windows Phone 7

Windows Phone 7 er den nye version af Windows Mobile. Udvikling til Windows Phone 7 foregår i Silverlight og XNA som begge kan bruge C#.

Silverlight bruges sædvanligvis til XAML-baserede, eventdrevne applikationer med standard Windows Phone layout, mens XNA primært bruges til spil eller speciallavet design af brugerinterfaces. Disse kan også benyttes sammen, hvor udviklere kan gøre brug af det nødvendige fra de enkelte frameworks. Udvikling til Windows Phone foregår i Visual Studio (Professionel eller Express), og foregår dermed kun på Windows. I nuværende version understøtter Windows Phone 7 ikke Bluetooth[20], men vi kan ikke forestille os andet end at dette kommer i en senere version. JSON konvertering er understøttet som standard[21].

Microsoft håndterer applikationsudgivelser lidt lig Apple og RIM. For at få en applikation på Windows Phone Marketplace skal den igennem en godkendelses procedure hos Microsoft, før den kan blive udgivet[22]. Dette giver de samme fordele og ulemper som hos Apple. Det skal igen understreges, at systemet er forholdsvis nyt, og at det derfor er sandsynligt at de nævnte forhold ændrer sig løbende.

[20] http://forums.create.msdn.com/forums/p/77644/471237.aspx
[21] http://msdn.microsoft.com/en-us/library/system.runtime.serialization.datacontractjsonserializer%28v=VS.90%29.aspx?appId=Dev10IDEF1&l=EN-US&k=k%28DATACONTRACTJSONSERIALIZER%29&rd=true#Y29
[22] http://msdn.microsoft.com/en-us/library/hh184843(v=vs.92).aspx

Det er svært at konkludere, hvad der vil være muligt og ikke muligt for udviklere, i og med at systemet er forholdsvis nyt.

2.3.5 Facebook adgang

Adgang til Facebook Graph kan ske på to måder, enten via biblioteket Facebook SDK eller browserbaseret via HTTP. Alle systemer kan derfor få adgang til Facebook via deres browser. Android og iPhone understøtter også adgang til Facebooks eget bibliotek[23], og er derfor langt mere simpel at benytte.

Den ene metode, som kun Android og iPhone udnytter, fungerer ved at det ekstra bibliotek Facebook SDK tilføjes udviklerens platform. Med SDK'et er der mulighed for at få adgang til et væld af oplysninger, bl.a. billede og navn [13].

Fælles for alle systemer er, at der er mulighed for at kalde Facebooks Graph API (den som Facebook via browseren selv bruger) vha. HTTP requests. Dette kræver dog, at brugeren enten har givet adgang til applikationen inden, og dermed har adgang til brugerens access token[24], eller at applikationen kører via den mobile browser og er webbaseret.

[23] http://developers.facebook.com/docs/guides/mobile/
[24] En access token indeholder info om login sessionen, bl.a. brugerens rettigheder

2.3.6 Valg af platform

En opsummering af egenskaberne for de undersøgte platforme kan ses i Tabel 2.1.

Tabel 2.1: Sammenligning af platforme

	iOS	Android	BlackBerry	Windows Phone 7
Adgang til Facebook	✓	✓	✓	✓
Dedikeret Facebook API	✓	✓	-	-
Bluetooth adgang	✓	✓	✓	-
JSON understøttelse	✓	✓	✓	✓
Sprog	Objective-C	Java	Java	C#

Da Windows Phone 7 ikke understøtter Bluetooth er denne platform ikke interessant i dette projekt. BlackBerry har adgang til Facebook, men kun via http, og vælges derfor også fra. Tilbage er Android og iOS, som begge tilbyder den funktionalitet og de muligheder, vi har brug for.

Det endelig valg falder på Android, grundet ønsket om at lære mere om Android platformen samt tidligere programmeringserfaring med Java.

2.4 Facebook Graph

Facebook er baseret på udtræk af data til visning af relationer mellem mennesker, begivenheder, billeder og interesser. Kernen i dette er Facebook Graph. Hvert eneste objekt i Facebook Graph er tildelt et unikt ID [13]– fx kan siden for Danmarks Tekniske Universitet tilgås med Facebook Graph på URL'en http://graph.facebook.com/178930892125092, hvilket browseren vil returnere et JSON objekt til:


```
{
   "id": "178930892125092",
   "name": "Danmarks Tekniske Universitet",
   "picture": "http://profile.ak.fbcdn.net/static-
ak/rsrc.php/v1/ym/r/wh4xTXo3Vvh.png",
   "link": "http://www.facebook.com/pages/Danmarks-Tekniske-
Universitet/178930892125092",
   "likes": 68,
   "category": "Local business",
   "website": "http://www.dtu.dk",
   "is_community_page": true,
   "location": {
      "street": "Anker Engelundsvej 1",
      "city": "Kongens Lyngby",
      "country": "Denmark",
      "zip": "2800",
      "latitude": 55.78546,
      "longitude": 12.52327
   },
   "phone": "45 25 25 25",
   "checkins": 467
}
```


På samme måde vil en forespørgsel på personen med ID = 4 (Facebook stifteren Mark Zuckerberg) returnere følgende efter et kald af http://graph.facebook.com/4:


```
{
   "id": "4",
   "name": "Mark Zuckerberg",
   "first_name": "Mark",
   "last_name": "Zuckerberg",
   "link": "http://www.facebook.com/zuck",
   "username": "zuck",
   "gender": "male",
   "locale": "en_US"
}
```

Samme data vil i øvrigt kunne hentes med http://graph.facebook.com/zuck, da både personer og sider kan tilgås med deres unikke ID og unikke navn, hvis dette er valgt af brugeren.

Med så enorme datamængder tilgængeligt på en relativt struktureret facon via Facebook Graph, er behovet for kontrol af, hvem der kan se hvad, ganske naturligt stort. Denne kontrol har Facebook delvist ladet være op til den enkelte bruger - kun fornavn, efternavn, køn, netværk og profilbillede er det ikke muligt at holde skjult[25]. I praksis viser det sig dog, at det godt kan lade sig gøre at skjule kønnet, selvom Facebook ikke skilter med det.

De viste eksempler på adgangen til Facebook Graph foregår via en browser. Adgangen foregår noget anderledes fra en mobilapplikation. Først og fremmest er det kun den officielle Facebook applikation, der kan tilgå alt data. Applikationer som vores skal have tilladelse fra brugeren, før der er adgang til brugerens data. Dette gælder både det altid offentlige data og alt udover dette, bl.a. adgang til venneliste, "Synes godt om" og mailadresse.

Adgangen til Facebook Graph fra en mobilapplikation kræver en verifikation af både brugerens login og af selve applikationen. Denne verifikation håndteres af Facebook SDK.

Der er to typer af login: OAuth 2.0 User Agent [14] og Single Sign-On (SSO) [16]. Pointen i SSO er at gøre login på flere forskellige services nemmere for brugeren. Efter at have benyttet sit login ét sted, vil brugeren ikke blive bedt om brugernavn og kode efterfølgende. Facebook SDK prioriterer login med SSO, og benytter kun OAuth 2.0 hvis SSO ikke er muligt. I bund og grund gælder det, at hvis den officielle Facebook Applikation er installeret, er muligheden for SSO til stede. Selve login proceduren sker ved, at verifikationen routes gennem den officielle Facebook Applikation, som ved gyldigt login returnerer den access token, der er nødvendig for at foretage kald til Facebook Graph[26].

Hvis den officielle Facebook applikation ikke er installeret, benyttes OAuth 2.0 i stedet. Dette indebærer, at et WebView indlejres ovenpå applikationens skærmbillede med en dialogboks (se Figur 2.5). Verifikationen foretages dermed ikke af klient applikationen, men mellem det indlejrede WebView og Facebook. Access token bliver modtaget af dialogboksen i form af en URL fra Facebook[26].

[25] http://www.facebook.com/privacy/explanation.php#basicinfo
[26] Informationerne står i kommentarer til kildekoden for Facebook SDK, i filen Facebook.java.

Figur 2.5: Login med OAuth 2.0 i WebView

2.5 Bluetooth

Helt essentielt for applikationen er brugen af enhedens Bluetooth radio. Bluetooth benyttes i vores applikation til at søge efter andre enheder i nærheden, og til at gøre enheden synlig for andre enheder. Teknologien understøtter herudover bl.a. filoverførsler mellem enheder og trådløs kommunikation mellem telefon og headset.

De største fordele ved Bluetooth er, at det er en trådløs og billig teknologi som samtidig har et lavt strømforbrug. Det samme er gældende for en infrarød forbindelse, men her er man begrænset af at enhederne skal befinde sig umiddelbart overfor hinanden – med Bluetooth undgås dette, og således opnås en højere grad af mobilitet [15].

Det der gør Bluetooth til en passende teknologi i dette projekt, er at rækkevidden på samme tid er tilstrækkelig lille og tilstrækkelig stor. Bluetooths signalstyrke er holdt

nede på 1 milliwatt (hvor en kraftig mobiltelefon kan benytte op til 3 watt til sin cellulære antenne)[27], og udover at kræve et minimum af strøm, resulterer det i en begrænset rækkevidde på omkring 10 meter. I vores projekt er det ikke interessant at få præsenteret personer fra nabobyen, men netop kun dem, som befinder sig i umiddelbar nærhed. At strømforbruget samtidig er lavt er kun et plus, i en verden hvor strømkapaciteten i lithium-ion batterier, en teknologi fra 1970-erne[28], til stadighed er et irritationsmoment for de fleste smartphone ejere.

[27] http://electronics.howstuffworks.com/bluetooth2.htm
[28] http://en.wikipedia.org/wiki/Lithium-ion_battery

Kapitel 3

Design

I dette kapitel beskrives opbygningen af applikationen samt designet.

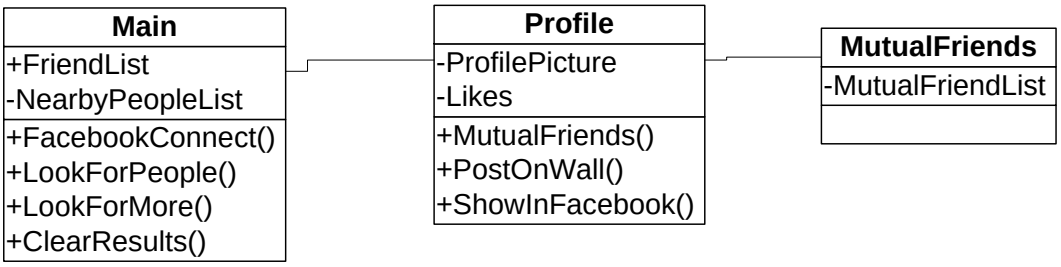
3.1 Kravspecifikation

Med udgangspunkt i analysen er følgende kravspecifikation udarbejdet.:

1. Applikationen skal kunne vise hvem der befinder sig i umiddelbar nærhed af brugeren.
2. Al brugerdata bliver udtrukket fra Facebook.
3. Ved første kørsel af applikationen skal brugeren give tilladelse til, at applikationen får adgang til data fra brugerens Facebook konto.
4. For hver bruger skal vises enkelte oplysninger fra deres Facebook konto, fx. navn, billede og antal fælles venner.
5. Efter en skanning af brugerens nærområde skal resultatet vises i sorteret rækkefølge vægtet efter antallet af fælles venner.
6. Det skal være muligt at få vist en detaljeret profil på brugere, fx. med visning af stort profilbillede, en liste over fælles venner og hvilke ”Synes godt om”, en bruger har.
7. Andre brugere skal ikke nødvendigvis også have applikationen åben. Det skal være tilstrækkeligt at enhedens Bluetooth forbindelse er tændt.
8. Ved visning af en specifik brugerprofil skal det være muligt at sende en besked til vedkommendes væg.
9. Ved visning af en specifik brugerprofil skal være et link til vedkommendes Facebook profil.

3.2 Arkitektur

Applikationen er bygget op af 3 dele. Main, når du starter applikationen, Profile og MutualFriends (se Figur 3.1). Det vil i det følgende blive beskrevet hvordan disse dele fungerer samt navigations flowet i applikationen.



Figur 3.1: Programarkitekturen

3.2.1 Main

Main aktiviteten indeholder det brugerne skal kunne, når de starter programmet. Når Main aktiviteten bliver startet, vil brugeren blive bedt om at logge på Facebook, og serveren vil herefter blive synkroniseret med brugerens data.

Desuden gemmes brugerens egen venneliste, så denne kan tilgås fra hele programmet. Dette er nødvendigt for senere at kunne udregne fælles venner. Brugeren kan vælge, om resultatet af en skanning skal tilføje eller erstatte tidligere fundne. Efter hver skanning vil resultatet blive vist i en liste med et profilbillede i lille format, samt navn og antallet af fælles venner. Listen bliver løbende sorteret efter antallet af fælles venner.

3.2.2 Profile

Ved at vælge en person fra listen i Main aktiviteten, bliver brugeren ført til Profile aktiviteten. Her ses den valgte persons profilbillede i en større version, og brugeren vil derudover kunne se en liste over personens "Synes godt om".

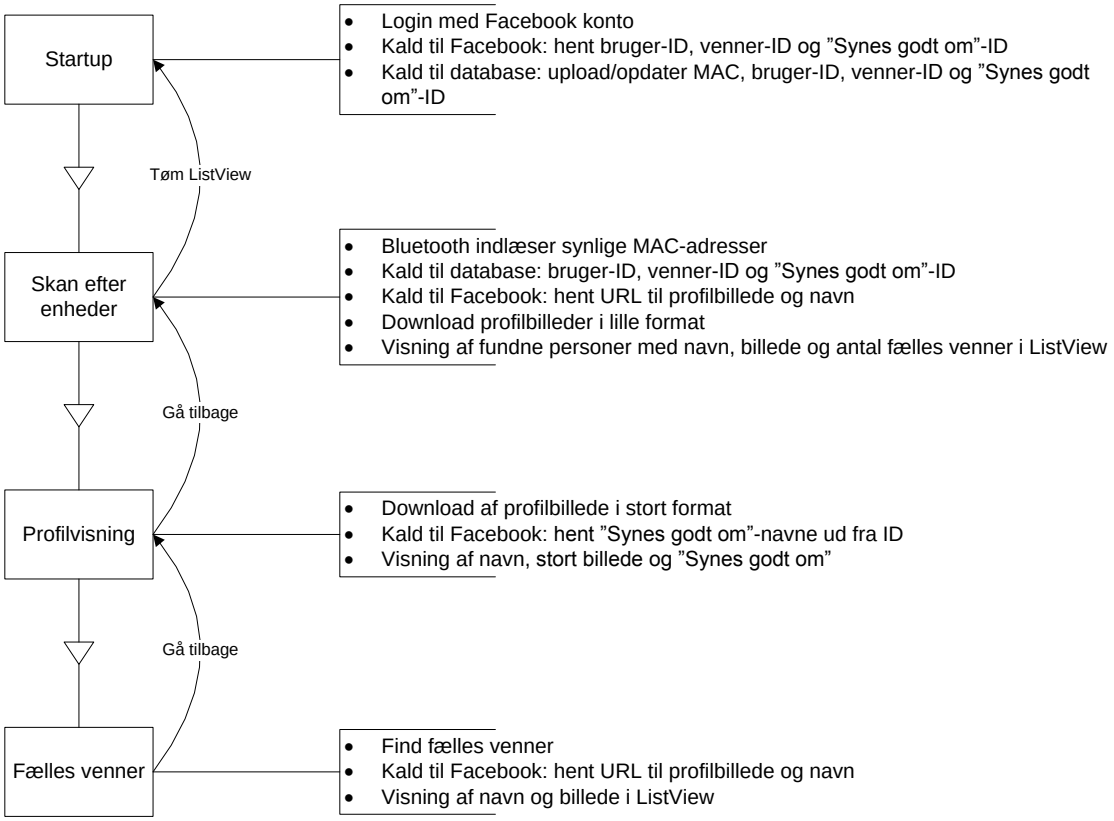
Det er desuden muligt at skrive en besked på personens væg, se vedkommendes Facebook profil i den officielle Facebook applikation eller se en liste over de venner, der er til fælles. Vælges sidstnævnte føres brugeren til MutualFriends aktiviteten.

3.2.3 MutualFriends

I MutualFriends aktiviteten vises en oversigt over fælles venner, med den valgte person. Listen er opbygget på samme måde som i Main, med et billede af personerne samt navn.

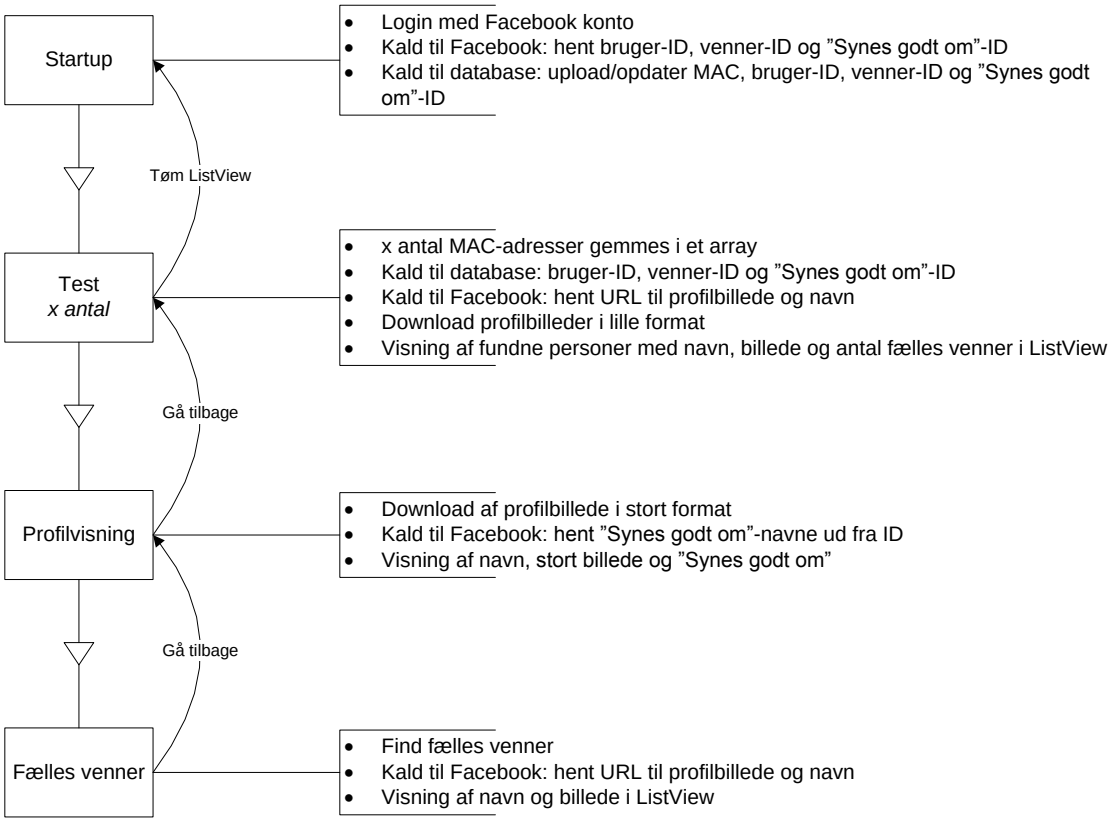
3.2.4 Flow

I det følgende gennemgås det naturlige flow i applikationen. Figur 3.2 viser helt normal brug, hvor brugeren åbner applikationen, foretager en skanning og får vist profil og fælles venner på en funden person. Til venstre er vist brugerens valg, til højre er vist de handlinger applikationen udfører undervejs. Figuren læses oppefra og ned.



Figur 3.2: Flow ved almindelig skanning

Figur 3.3 viser flowet ved kørsel af testfunktionen. Forskellen ligger i, at en Bluetooth skanning ikke sættes i gang. I stedet er indkodet en række MAC adresser i applikationen, som indlæses det antal gange, brugeren definerer ved kørsel af testfunktionen. Flowet er herudover uændret.

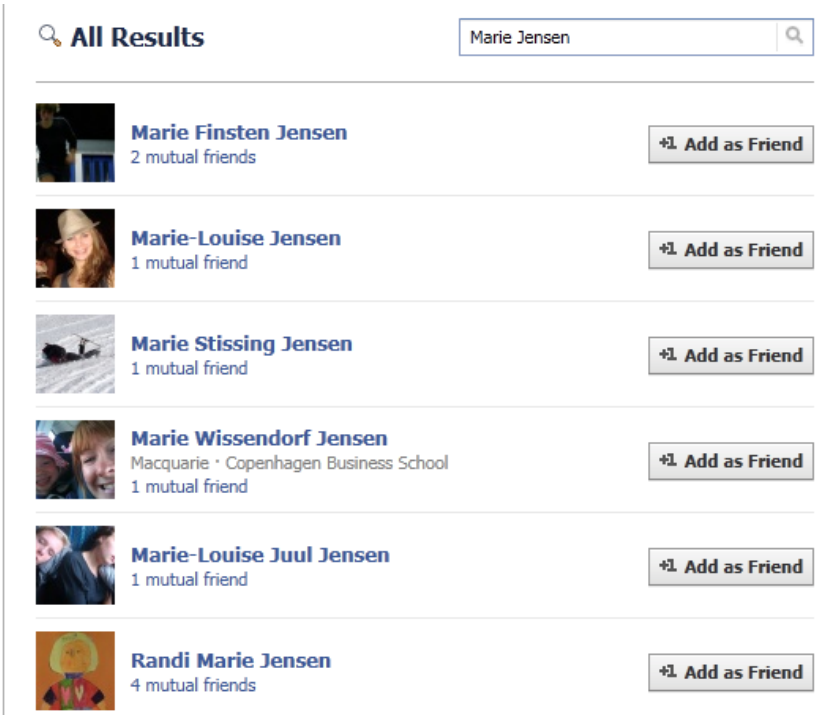


Figur 3.3: Flow ved kørsel af test-funktionen

3.3 Bruger interface

Applikationens GUI er holdt i et simpelt design, og med få og enkelte funktioner:

- Ved opstart vises en menu med de funktioner der er tilgængelige. Menuen skjules efter valg af funktion, men kan til hver en tid vises ved at trykke på ”Menu”-knappen på enheden.
- Resultatet af en skanning samt visning af fælles venner vises i logisk opbyggede lister, som de er kendt fra søgeresultater på Facebook (se Figur 3.4).
- Profilvisninger er opbygget efter samme model som den nuværende officielle Facebook Applikation. Visningen holder skærmens bredde og udbygges nedefter. Vores implementering viser blot navn, billede og ”Synes godt om”, men nyt indhold vil let kunne tilføjes senere. Det kan fx. være kontaktoplysninger, by, job og uddannelse.
- Applikationen består alt i alt af tre skærmbilleder: startside, profil og fælles venner.



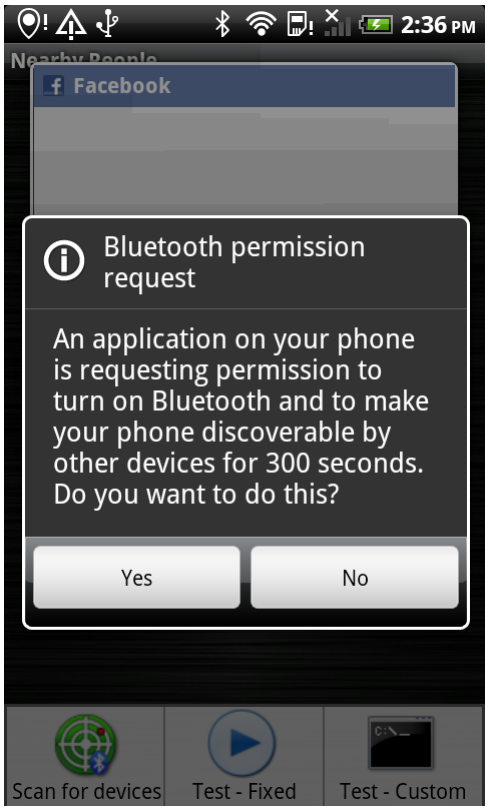
Figur 3.4: Visning af søgeresultat på Facebook[29]

[29] Screenshot fra www.facebook.com, 24.05.2011

I applikationen har vi gjort to test-funktioner tilgængelige. Dette er gjort både for selv at kunne foretage en simpel test af større eller mindre ændringer i koden, men også for at gøre applikationen anvendelig i sin nuværende form. Antallet af brugere med applikationen installeret kan tælles på én hånd, og det er derfor ikke sandsynligt, at en skanning med Bluetooth vil returnere nogle svar. Med den indbyggede testfunktion er det muligt at indlæse et valgfrit antal foruddefinerede MAC adresser, og derved efterligne at man befinder sig i et rum fyldt med andre brugere af applikationen.

En endelig version af applikationen vil dermed kunne være endnu mere enkel, da den kan begrænses til at indeholde én knap synlig ved opstart: ”Skan”.

Når applikationen åbnes præsenteres brugeren for en popup, som beder om godkendelse til, at enheden er synlig for andre i nærheden i 300 sekunder (se Figur 3.5). Herefter bliver brugeren bedt om at logge på Facebook (se Figur 3.6), og hvis det er første gang brugeren logger på, skal der gives tilladelse til at applikationen henter oplysninger fra Facebook.

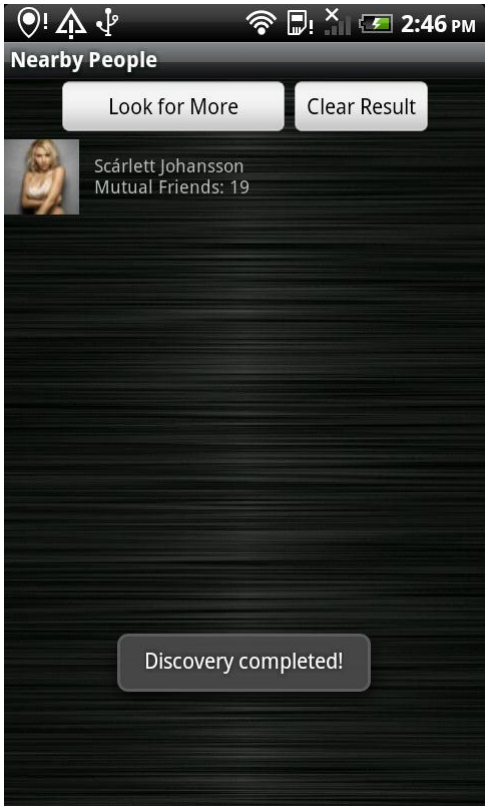


Figur 3.5: Applikationen åbnes

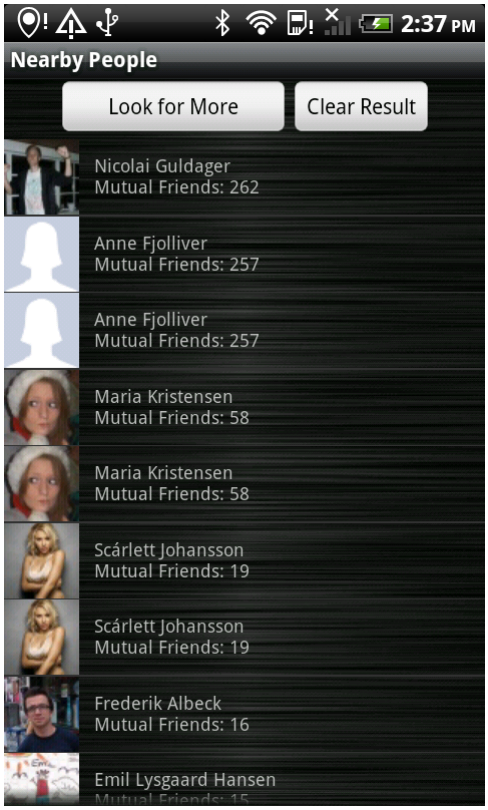
Figur 3.6: Facebook login

Herefter logger applikationen på serveren, og brugeren bliver nu mødt af velkomstskærmen (se Figur 3.7). Her kan man i den nuværende version vælge at skanne, eller at benytte de 2 testmetoder – som tidligere nævnt er disse medtaget, da projektet er ”proof of concept”. Når skanningen for enheder i nærheden er fuldført, vil en liste med de fundne personer blive vist (se Figur 3.8 for én funden og Figur 3.9 for flere fundne), og brugeren kan herefter vælge at finde flere, rydde resultatet eller vælge at få vist en detaljeret profil for en af personerne.

Figur 3.7: Velkomstskærm



Figur 3.8: Fundet én person

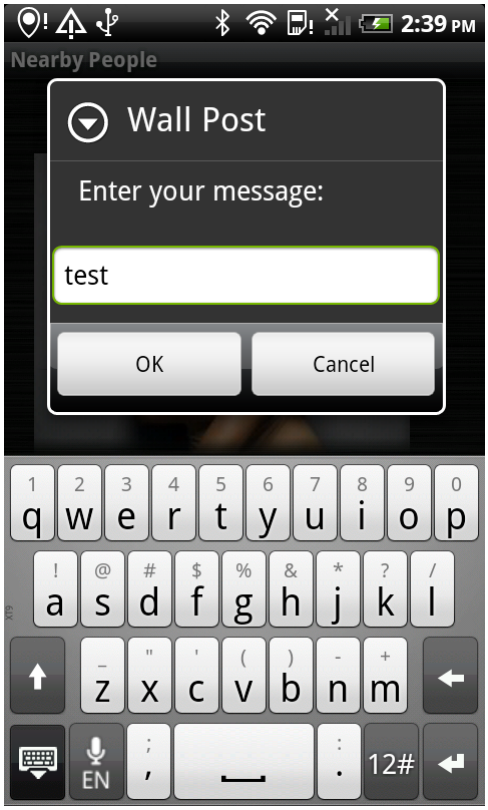


Figur 3.9: Fundet flere personer

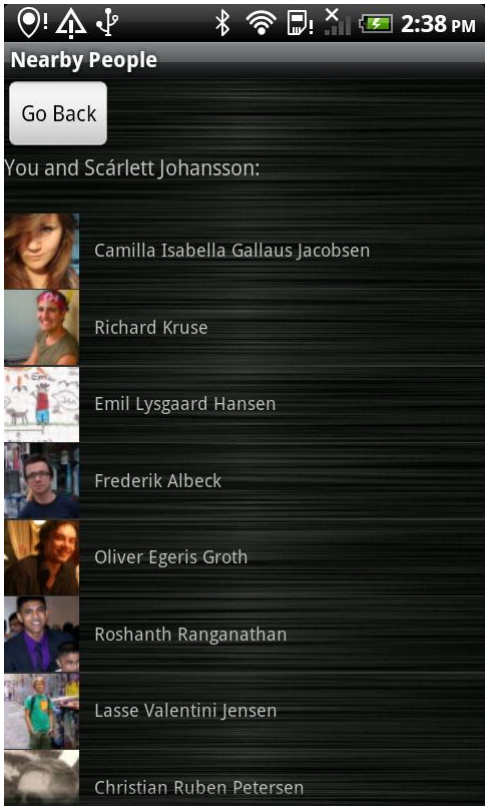
Når brugeren har valgt en person fra søgeresultatet, vil Profile aktiviteten blive vist. Her er der mulighed for at gå tilbage til søgeresultatet, se fælles venner, skrive på personens væg eller at få vist vedkommendes Facebook profil i den indbyggede Facebook applikation (se Figur 3.10).

Figur 3.10: Profil for valgt person

Hvis brugeren vælger at skrive på personens væg, vises en popup hvor brugeren kan indtaste sin meddelelse og sende den (se Figur 3.11). Hvis brugeren vælger at få vist deres fælles venner, vil MutualFriends aktiviteten blive vist (se Figur 3.12).



Figur 3.11: Skrive på en persons væg



Figur 3.12: Fælles venner med valgt person

Kapitel 4

Implementering

I dette kapitel beskrives implementeringen af applikationen.

Alle dele af kravspecifikationen er blevet implementeret og i dette kapitel gennemgås de løsningsmodeller vi har valgt.

Som beskrevet i problemformuleringen, har vi valgt ikke at gå efter en færdigudviklet applikation. I stedet har vi fokuseret på at udvikle en applikation, som kan demonstrere og afprøve konceptet. Tillige har vi valgt ikke at inkorporere brug af GPS, og holde os til den helt grundlæggende Bluetooth funktionalitet. Således kan vi få vished for, hvilke elementer af selve konceptet der fungerer godt, og hvilke der fungerer mindre godt. Efterfølgende er der dermed dannet et grundlag for at arbejde videre med applikationen i den rigtige retning.

Ultimativt er det ønsket og forventningen, at de færdigimplementerede funktioner kan tilføjes den kommende officielle DTU mobilapplikation, og ad den kanal komme ud til en stor gruppe brugere.

4.1 Udviklingsmiljø

Applikationen er udviklet i Eclipse IDE version 3.6.1. For at gøre Eclipse egnet til at udvikle til Android benyttes Android SDK, som indeholder de nødvendige Android-biblioteker, en emulator, flere eksempler med kildekode og udviklingsværktøjer. Der medfølger desuden en ”Android SDK and AVD Manager”, hvorfra der kan installeres den eller de udgaver af Android platformen man sigter efter samt virtuelle enheder at køre software på. Det er også her en USB driver kan hentes, for understøttelse af

kørsel af software direkte på en Android enhed. Ydermere har vi benyttet et ADT[30] 10.0.0 plug-in, som er udviklet specifik til Eclipse og bl.a. tilføjer en simpel GUI[31]-editor, debugging funktionalitet og mulighed for at generere .apk[32] filer.

Forbindelse og kommunikation til Facebook Graph skabes med Facebook SDK, som er udviklet og bliver vedligeholdt af Facebook selv, og hentes fra et GitHub repositorie[33]. Herefter oprettes et nyt Android projekt med Facebook SDK som kilde, og vores applikation refererer til dette projekt ved at tilføje det som bibliotek i Eclipse.

For at have adgang til Facebook Graph er det nødvendigt at oprette sig som "Facebook Developer" og oprette en applikationsside for at få et applikations ID. Dette ID er påkrævet for at foretage kald til Facebook Graph fra applikationen. Desuden skal Facebook applikationssiden verificeres ved at linke den sammen med Android-applikation, som beskrevet på Facebooks introduktionssite for udviklere:

> *"We now need to export the signature for your app so that Facebook can use to ensure users are only communicating with your app on the Android"*[34]

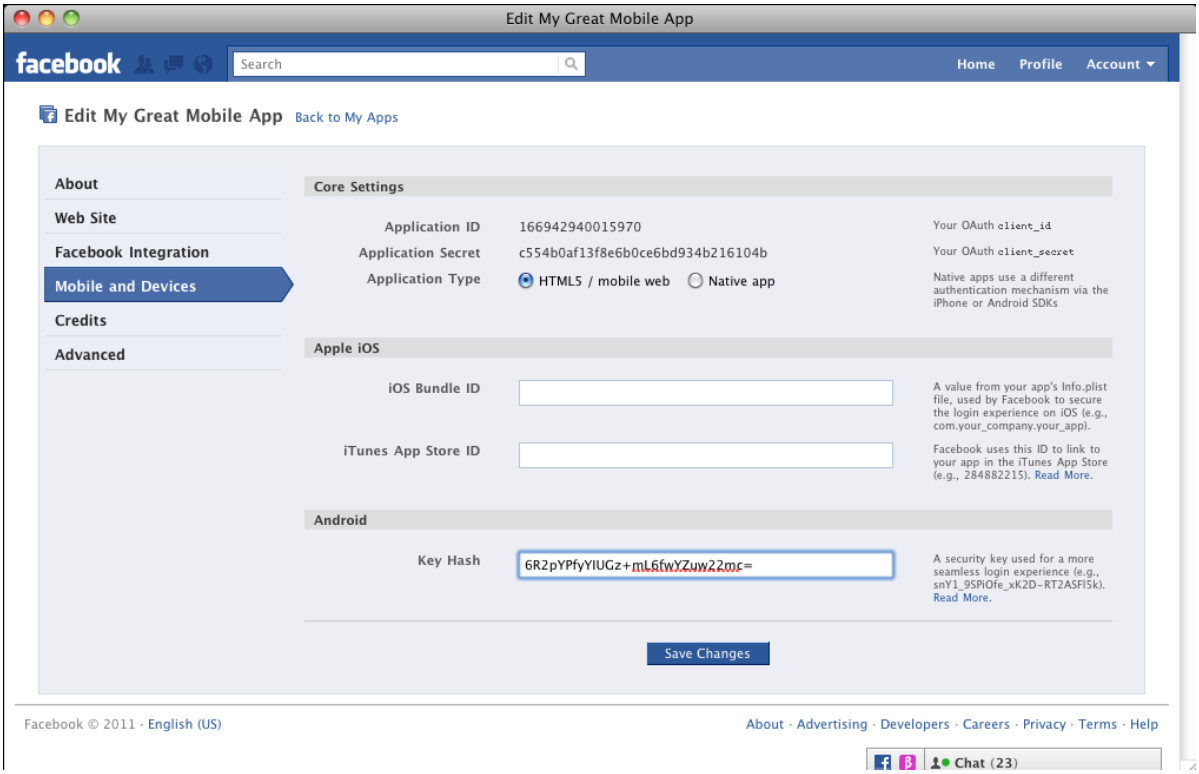
Dette gøres med det medfølgende *keytool*, som køres med følgende kommando:

```
keytool -exportcert -alias androiddebugkey -keystore
~/.android/debug.keystore
| openssl sha1 -binary
| openssl base64
```

Kommandoen skal køres i Linux, men da vi arbejder i Windows har det været nødvendigt at bruge et værktøj der kan emulere et Linux-miljø. Cygwin[35] er et udmærket valg til dette formål.

Den genererede kode indtastes i feltet "Android Key Hash" på indstillingssiden for Facebook applikationen (se Figur 4.1).

[30] Android Development Tools
[31] Graphical User Interface
[32] .apk-filer kan installeres direkte på Android-enheder, udenom Android Market
[33] git://github.com/facebook/facebook-android-sdk.git
[34] http://developers.facebook.com/docs/guides/mobile/#android
[35] Linux-miljø til Windows – se http://www.cygwin.com



Figur 4.1: Facebook applikationsside[34]

Installationsrækkefølgen er således:

1. Eclipse
2. Android SDK
3. ADT plug-in
4. Android SDK Manager
    a. SDK Platform Android 2.3.3
    b. SDK Tools
    c. Google USB Driver Package
5. Facebook SDK
    a. Indtast Application ID i applikationen
    b. Generér ”Key Hash” og indtast på Facebook applikationssiden

4.2 Debugging

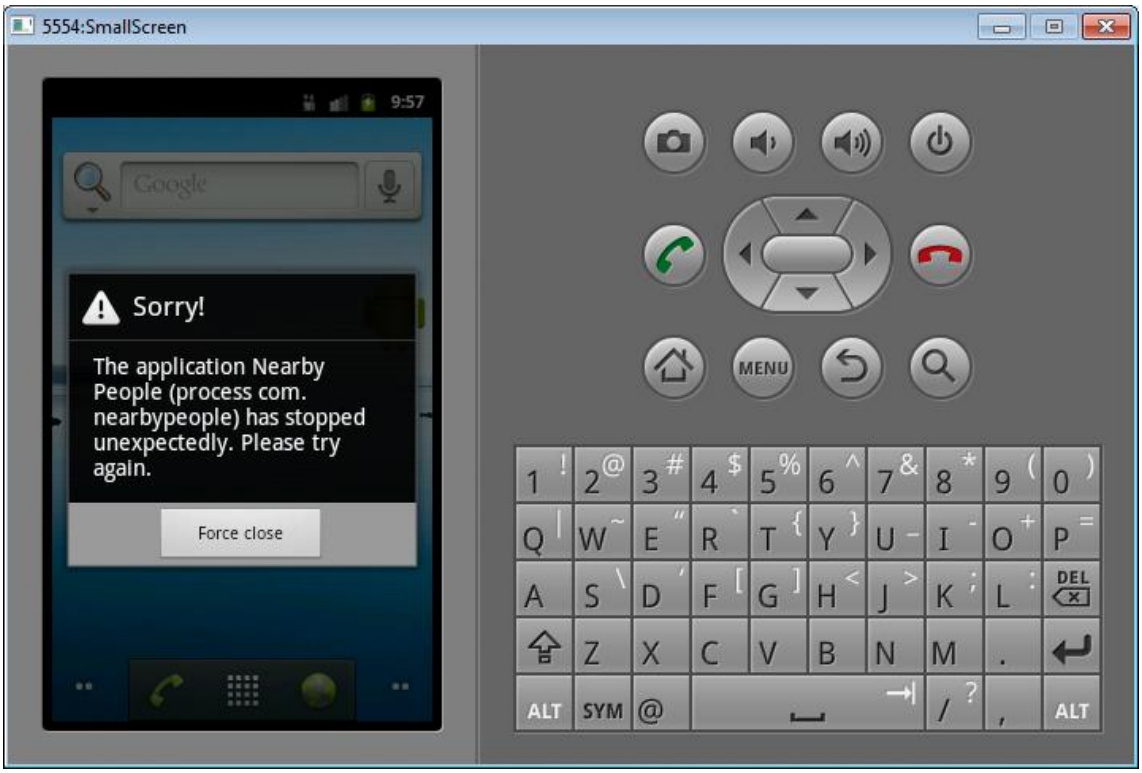
Til debugging har vi benyttet to forskellige værktøjer: Android Virtual Device (se Figur 4.2: AVD) og mobiltelefonen HTC Desire (Android version 2.2). De har hver især styrker og svagheder, og fungerer i virkeligheden bedst i fællesskab.

Figur 4.2: AVD

Android Virtual Device er et værktøj, som følger med Android SDK, og kan køres direkte fra Eclipse. Det er valgfrit, hvilken version af Android der ønskes installeret, og der er ligeledes mulighed for at ændre bl.a. skærmopløsning og størrelse på hukommelseskort. På trods af disse gode muligheder for at udvikle en applikation, der kan køre på flere forskellige udgaver af Android, blev vi i dette projekt afskåret fra at benytte AVD, da vi startede på at implementere Bluetooth funktionaliteten. AVD understøtter ganske simpelt ikke Bluetooth (se Figur 4.3)[36].

[36] http://stackoverflow.com/questions/2175076/how-to-use-bluetooth-in-android-emulator

Det er muligt at foretage nogle ændringer i AVD'ens konfiguration, der tillader at kompilere[37], men dette er stadig ikke tilstrækkeligt til vores behov.



Figur 4.3: AVD med Bluetooth

Det betyder at det kun giver mening, og kun er muligt, at teste dele af applikationen i AVD. For størstedelen af projektet har vi derfor benyttet en ganske almindelig HTC Desire, venligst udlånt af DTU.

Denne enhed understøtter Bluetooth, og arbejder desuden dobbelt så hurtigt som AVD. Ulempen er, at det besværliggør at sikre konsistens af applikationen på tværs af Android versioner.

[37] https://sites.google.com/a/android.com/opensource/projects/bluetooth-faq

4.3 Applikation

I det følgende beskrives i detaljer hvilke løsninger der benyttes i applikationen.

4.3.1 Kald til Facebook

Facebook SDK tilbyder en mængde metoder, som gør det simpelt at foretage kald til Facebook grafen [16]:

```
// get information about the currently logged in user
facebook.request("me");
//get the logged-in user's friends
facebook.request("me/friends");
```

For hver af ovenstående forespørgsler skal blot implementeres en `onComplete`-metode, som tager imod den JSON-formatterede tekststreng, der bliver returneret med svaret på kaldet til Facebook Graph.

AsyncFacebookRunner er tilgængelig via Facebook API, og benyttes til at foretage kald til Facebook Graph med syntaksen:

```
AsyncFacebookRunner.request(Bundle parameters, RequestListener
listener)
```

Formålet er at undgå at blokere UI tråden indtil svaret kommer retur. Det er implementeret i Facebooks API ved at lade hver forespørgsel kører i sin egen tråd.

Essentielt for applikationen er visning af profilbilleder. Facebook har gjort alle profilbilleder offentligt tilgængelige, og det er derfor muligt at finde URL'en direkte til billedet ved at kalde

```
request(picture, new PersonRequestListener());
```

Dette returnerer som vanligt et JSONObject, hvorfra en String med URL'en findes ved:

```
url = json.getString("picture");
```

Som standard har man imidlertid kun mulighed for at hente en nedskaleret udgave af profilbilledet på 50 x 50 pixels, og vores ønske er at vise profilbilledet i stort format på 250 x 250 pixels. Dette kan løses med en simpel manøvre.

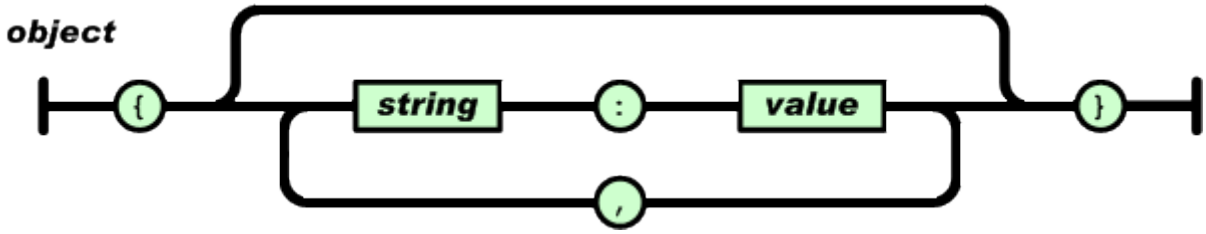
Det viser sig, at den direkte URL til profilbilleder er logisk opbygget – det kan aflæses direkte, hvilken Facebook server billedet er gemt på, hvilket ID billedet har samt hvilket Facebook bruger ID der har uploadet billedet. Desuden ender alle adresserne med "_q". Ved i stedet at slutte adressen med "_n", vises billedet i stort format (varierende, men cirka 200 x 200 pixels). Således kan det alligevel lade sig gøre at undgå en uhensigtsmæssig opskalering af profilbilledet, med dette lille stykke kode:

```
f.url_large = f.url_small.replace( "_q", "_n" );
```

, hvor vi blot udskifter "_q" med "_n". Det virker som ønsket, men er selvfølgelig sårbart overfor eventuelle ændringer i opbygningen af URL'en i fremtiden. Vi håber derfor på, at Facebook på et tidspunkt vil tillade flere størrelser af profilbilleder via Facebook API.

4.3.2 Konvertering af JSON

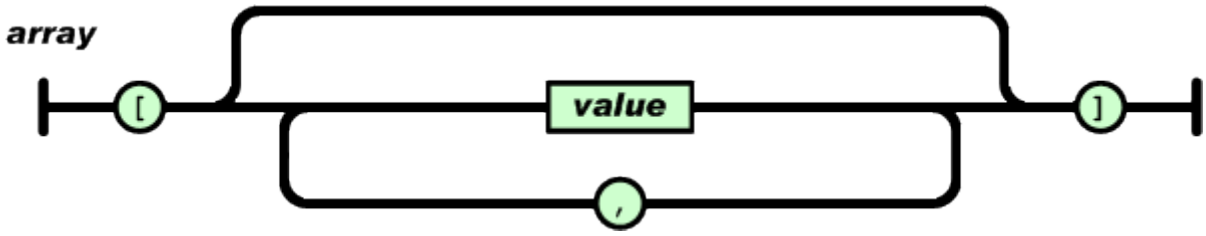
Svar fra Facebook returneres som et JSON objekt. Et JSON objekt er i princippet en String opbygget efter et strengt regelsæt, og er derfor både let for mennesker at læse, men også for maskiner. Et JSON objekt starter med en { (venstre tuborgklamme) og slutter på } (højre tuborgklamme), hvorimellem der er et arbitrært sæt af navne/værdi par som er adskilt af kommaer. Opsætningen af et JSON objekt kan ses på Figur 4.4.



Figur 4.4: JSON objekt[38]

En værdi i JSON kan være en String, et tal, true, false, null eller et objekt eller Array. Et JSON Array er opbygget af en [(start firkantparentes), et arbitrært antal af værdier adskilt af kommaer og en] (slut firkantparentes), som vist på Figur 4.5.

[38] Billede lånt fra http://www.json.org/



Figur 4.5: JSON Array[39]

For at konvertere et JSON objekt fra Facebook, importeres Javas indbyggede JSON bibliotek. Indledningsvist initialiseres et JSON objekt ud fra svaret fra Facebook Graph:

```
final JSONObject json = new JSONObject(response);
```

Derefter trækkes de ønskede oplysninger ud af objektet og gemmes dem i en String:

```
String name = json.getString("name");
String id = json.getString("id");
String picture = json.getString("picture");
```

I dette eksempel bliver indholdet gemt i et Person objekt sammen med brugerens MAC adresse, ”Synes godt om” og venner, og endelig tilføjet til listen over personer i nærheden.

Når vi henter brugerens ”Synes godt om” fra Facebook ankommer de i et JSON objekt indeholdende et JSON Array af JSON objekter:

```
{
   "id": "1198288396",
   "name": "Nicolai Guldager",
   "likes": {
      "data": [
         {
            "name": "CoolStuff.dk",
            "category": "Company",
            "id": "123231958645",
            "created_time": "2011-06-10T12:53:38+0000"
         },
         {
            "name": "PowerSkin",
            "category": "Product/service",
            "id": "142218355835349",
            "created_time": "2011-06-02T18:51:32+0000"
         }
      ]
   }
}
```

[39] Billede lånt fra http://www.json.org/

Dette er efter vores mening en ulogisk opsætning. Det vil give bedre mening at ændre navn/værdi parret til, at navnet indeholder ”likes” og værdien indeholder JSON Array’et med de ”Synes godt om” brugeren har, altså:

```
{
   "id": "1198288396",
   "name": "Nicolai Guldager",
   "likes": [
         {
         "name": "Facebook",
         "category": "Product/service",
         "id": "20531316728",
         "created_time": "2010-09-26T18:36:54+0000"
      },
         {
         "name": "Hegnet",
         "category": "Bar",
         "id": "27992501059",
         "created_time": "2010-05-24T18:50:16+0000"
      },
   ]
}
```

Omend en smule besværligt, er det sådan Facebook har valgt at opsætte sit svar, og vi må derfor forholde os til dette. Derfor kommer konverteringen fra JSON objektet til en String af typen ”Like1ID;Like2ID;…;” til at foregå lidt klodset:

```
final JSONObject json = new JSONObject(response);
JSONObject likes = json.getJSONObject("likes");
JSONArray data_array = likes.getJSONArray("data");

int l = (data_array != null ? data_array.length() : 0);
for (int i=0; i<l; i++) {
           JSONObject data_obj = data_array.getJSONObject(i);
           String like_id = data_obj.getString("id");
           myLikelist.append(like_id + ";");
}

myName = json.getString("name");
myFbId = json.getString("id");
```

På samme vis foregår konvertering af brugerens venneliste.

4.3.3 Sprog

Applikationens sprog er sat til engelsk som standard. Vi har dog lavet en dansk oversættelse, som aktiveres hvis brugeren i forvejen har sat deres enheds sprog til at være dansk.

Alle synlige elementer i brugerinterfacet indlæser deres tekstlabel fra en XML fil, og selve oprettelsen af flere sprog kræver derfor blot at denne fil oversættes og kopieres ind i en ny mappe. I Eclipse-projektet forefindes to mapper: res/values/ og res/values-da/. Ønskes det at tilbyde russisk, oversættes filen strings.xml til russisk og kopieres ind i mappen res/values-ru/. Android leder ved opstart af applikationen efter en mappe med navnet på enhedens sprog – findes denne ikke vælges blot mappen res/values/, som i vores implementering indeholder standardsproget engelsk.

4.3.4 Indlæsning af billeder

Billeder bliver vist flere steder i applikationen: visning af søgeresultat, visning af profil og visning af fælles venner. Til disse visninger benytter vi to forskellige funktioner til at hente billederne fra Facebook. Profilbilledet er et enkelt billede, og her er derfor ikke behov for andet end blot at downloade og vise billedet. Dette håndteres ved et par linjers simpel kode:

```
private Drawable download_img(String url, String src_name)
            {
            return Drawable.createFromStream(((java.io.InputStream)
                        new java.net.URL(url).getContent()), src_name);
            }
```

Dette er tilstrækkeligt til enkelte billeder og statiske visninger. Vi har testet koden også på visning i ListView (søgeresultat og fælles venner), og det går sådan set fornuftigt og som forventet i forhold til at downloade billederne. Problemet opstår, når brugeren scroller gennem listen. Da ingen cache benyttes, er hvert billede kun tilgængeligt så længe det bliver vist. Scrolles ned og op, vil billedet skulle hentes igen. Dette forårsager en træg fornemmelse af scrollingen, fordi der konstant bliver indlæst data, og et unødvendigt dataforbrug.

I stedet for at bruge tid på at implementere de ønskede forbedringer selv, har vi valgt at benytte et stykke komplet kode, der håndterer download af større mængder billeder for os. Filen imageLoader.java er således ikke noget, vi selv kan tage æren for,

men derimod brugeren Fedor på Stackoverflow[40]. Koden er frit tilgængelig[41], dog har vi foretaget enkelte modifikationer for at tilpasse koden til vores behov. Af smarte funktioner skal fremhæves, at billeder nu skaleres ned for at spare hukommelse, cachen kan benytte både intern hukommelse og SD-kort og endelig single threading – i stedet for at køre en tråd per billede og dermed blokere enhedens dataforbindelse en periode, bliver hver URL lagt i en kø og afviklet i én tråd, et billede ad gangen.

4.3.5 Database opsætning

Applikationen benytter en MySql[42] database med en enkelt tabel. På sigt er det meningen at DTU skal stille en server og database til rådighed, men dette er ved projektets afslutning endnu ikke på plads. Tidligt i forløbet opsatte vi vores egen database på et privat domæne, til brug i udviklingsforløbet, og det er denne der stadig er i brug. Databasen er hostet på en server hos firmaet Gigahost[43], som stiller 15 GB diskplads til rådighed via en forbindelse på 2 x 1000 Mbit[44], hvilket må siges at være rigeligt til at dække vores behov, så længe applikationen er beregnet til ”proof of concept” og ikke decideret produktion.

Tabellen er opsat som vist i Tabel 4.1. Felterne ”fb_id”, ”mac”, ”friendlist” og ”likes” oprettes eller opdateres fra applikationen ved hver kørsel. Felterne ”id” og ”time” bliver automatisk sat af serveren, første gang en given MAC adresse bliver indlæst, og vil herefter forblive uændret. Netop feltet ”mac” er indstillet til at være unikt, således at vi kan identificere allerede oprettede enheder, og efter første kørsel blot opdaterer felterne i pågældende række.

Feltet ”name” er manuelt indtastet og er kun interessant i udviklingsforløbet. Formålet har været at give et overblik over de enheder vi har benyttet til at teste applikationen, og holde styr på hvilke personer vi har tildelt hvilke MAC adresser.

I og med at ”mac” er unik, er feltet ”id” ikke nødvendigt til andet end at holde styr på antallet af brugere. ”time” er kun beregnet til at se, hvornår en enhed første gang har kørt applikationen. Den endelige database vil således kunne skæres ned til at bestå af ”fb_id”, ”mac”, ”friendlist” og ”likes”.

[40] http://stackoverflow.com/questions/541966/android-how-do-i-do-a-lazy-load-of-images-in-listview
[41] https://github.com/thest1/LazyList
[42] Open source database, se http://www.mysql.com/
[43] https://gigahost.dk/
[44] https://gigahost.dk/about

Tabel 4.1: Oversigt over database tabellen

Field	Type	Eksempel på data	Note
id	int(11)	96	Autogenereres af serveren
time	timestamp	2011-05-18 10:45:26	Autogenereres af serveren
name	varchar(50)	Nicolai - Olivers iPhone	Manuelt indtastet for at holde styr på vores testenheder og tilhørende Facebook konto
fb_id	varchar(20)	100002207924006	Opdateres ved hver kørsel af applikationen
mac (unik)	varchar(17)	CC:08:E0:98:70:AC	Opdateres ved hver kørsel af applikationen
friendlist	mediumtext	631135040;…;…;1198288396;	Opdateres ved hver kørsel af applikationen
likes	text	126793310719591;…;…;178930892125092;	Opdateres ved hver kørsel af applikationen

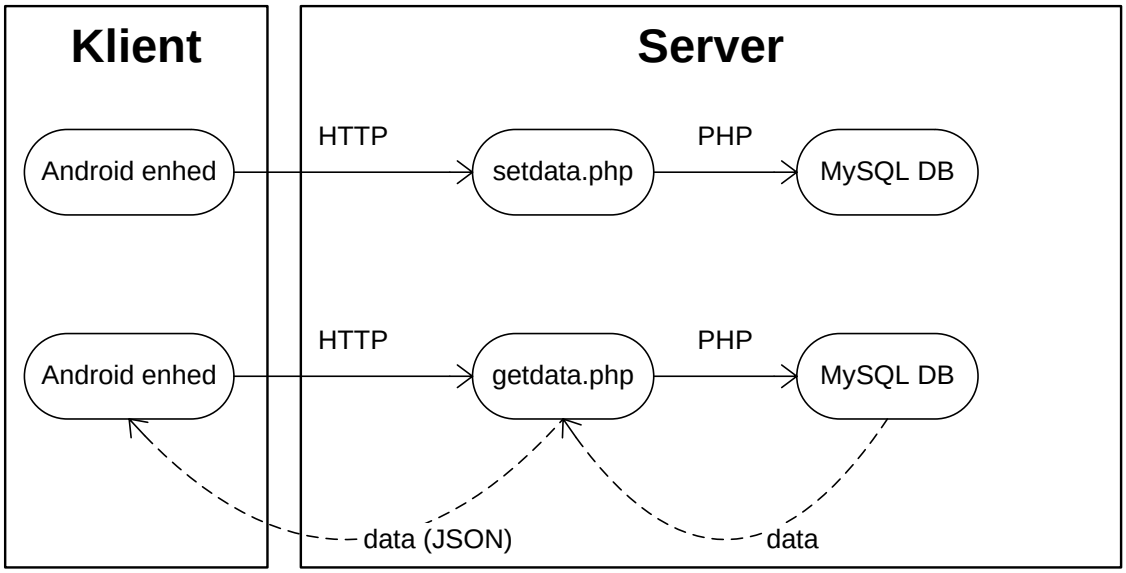
Som det ses, vil felterne ”friendlist” og ”likes” for mange brugere skulle indeholde tekststrenge af en vis størrelse. Ved udregning af fælles venner mellem to brugere, som i nuværende udgave af applikationen sker på enheden, vil alt indhold i ”friendlist” skulle hentes ned. Det er planen, at vi foretager denne udregning på serveren, når DTU får opsat server og database. Således vil det blot være nødvendigt at hente selve listen over fælles venner.

Den logiske fremgangsmåde ville være at bruge en JDBC driver til at forbinde til databasen, som det kendes fra almindelige Java-programmer. Dette er dog ikke umiddelbart muligt, da Android ikke understøtter JavaSE, som er nødvendig for JDBC[45]. Desuden ville dette kræve at login oplysninger til databasen gemmes lokalt i applikationen, hvilket ikke er sikkerhedsmæssigt forsvarligt.

Løsningen er, at kommunikationen mellem enheden og databasen sker med HTTP via et PHP-script, som beskrevet i [17]. Således kalder applikationen setdata.php med parametre for de felter der ønskes opdateret og dertilhørende data. Ved download af data kaldes getdata.php, som sørger for at returnere den ønskede data i JSON format

[45] http://stackoverflow.com/questions/1728476/does-android-support-jdbc

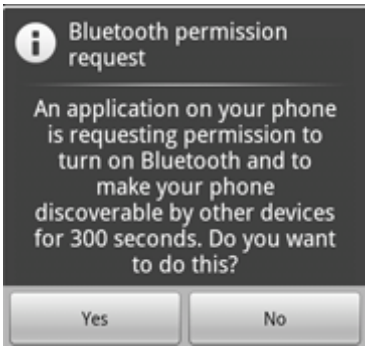
(se Figur 4.6). Den sikkerhedsmæssige fordel er, at koderne til databasen kun opbevares i PHP-filerne, og ikke kan ses fra applikationen.



Figur 4.6: Database flow

4.3.6 Håndtering af Bluetooth

At benytte Bluetooth i en Android-applikation er relativt simpelt. Den nødvendige tilladelse indskrives i Android manifestet[46] og med "`import android.bluetooth`", er der adgang til enhedens Bluetooth radio. Brugeren skal give tilladelse, hvilket håndteres som vist i Figur 4.7 nedenfor.



Figur 4.7: Bluetooth tilladelse

[46] AndroidManifest.xml er en vigtig konfigurationsfil for alle Android applikationer

Helt generelt har vi brug for to ting:

- At tænde Bluetooth radioen
- At indlæse MAC adresse på synlige enheder

Ingen af disse er som sådan problematiske at implementere. Men så alligevel – der er indbygget en timeout, så Bluetooth radioen automatisk lukker ned for "discoverability" en given tidsperiode efter den er blevet tændt. Det er således ikke umiddelbart muligt at holde enheden synlig konstant, og enheden er dermed kun synlig for andre i en begrænset periode. Dette er på sin vis underminerende for selve konceptet. I praksis vil dette betyde, at alle brugere i samme rum skal åbne applikationen samtidig, hvis alle skal kunne opdage hinanden.

Dette har været en udfordring at finde en løsning på. Vi er ikke de eneste der savner denne mulighed, så Google har i deres eget Android-forum tilkendegivet, at det vil være implementeret i en kommende version[47]. Allerede nu er muligheden til stede i Honeycomb 3.1, men dette er endnu kun tilgængeligt for tablets[48].

En alternativ løsning er at root'e enheden og opdatere med fx firmwaren "CyanogenMod", hvilket der er indikationer på skulle give flere rettigheder til at ændre i kildekoden[49], og ad den vej ændre den nuværende timeout på maximalt 300 sekunder. Dette vil dog samtidig bryde enhedens garanti. Vi valgte derfor den sikre kurs, mens vi venter på en opdatering af Android OS.

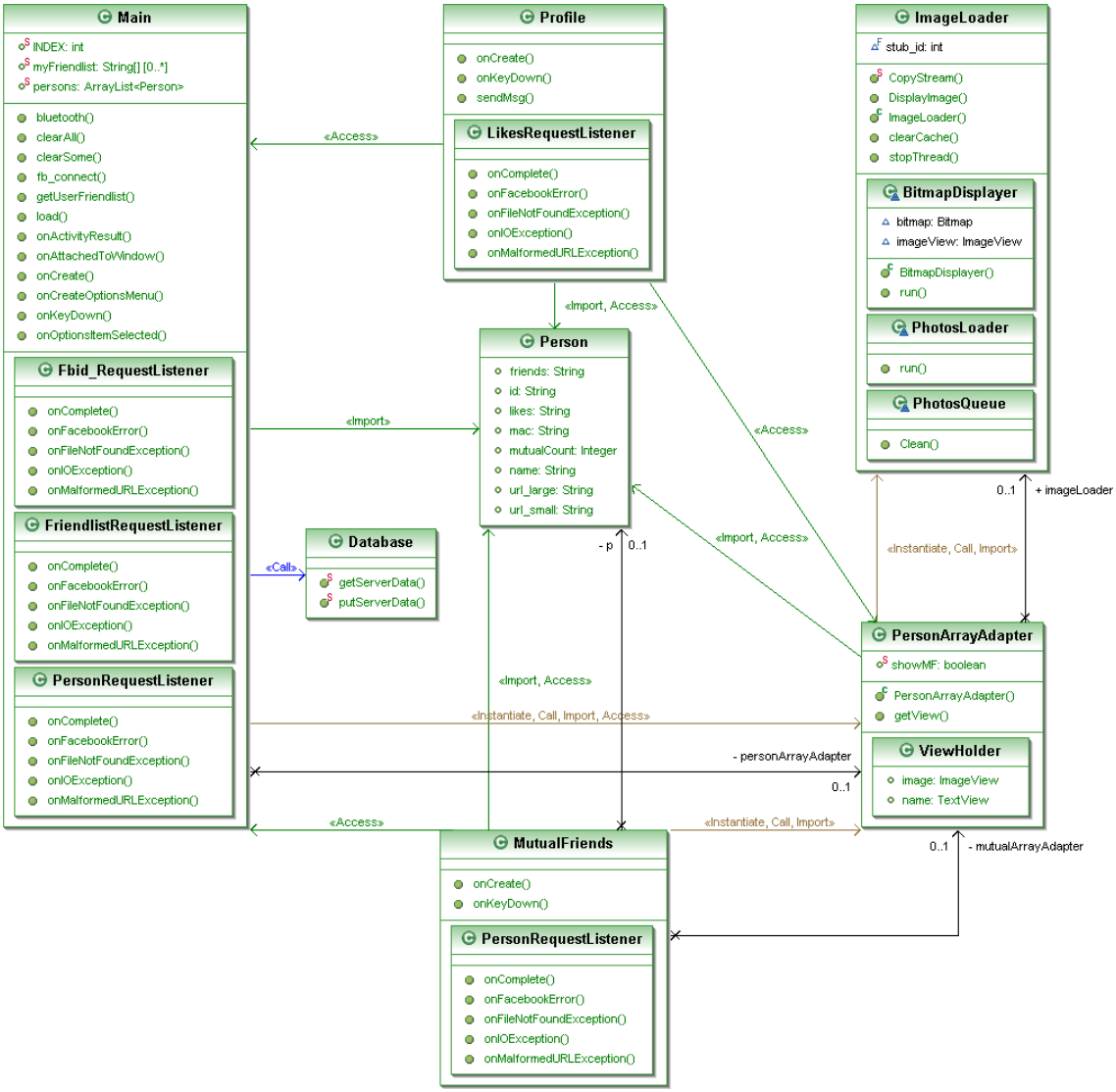
Først og fremmest sætter vi, ved opstart af applikationen, den periode Bluetooth radioen er tændt, til højest mulige værdi, 300 sekunder – standardværdien er 120 sekunder [10]. Desuden kontrolleres det før hver skanning, om Bluetooth radioen er tændt – hvis ikke, tændes den. Omend dette fungerer, har vi måtte give afkald på en del brugervenlighed, da brugeren skal vente på at enheden har tændt Bluetooth hver gang applikationen startes, hvilket kan tage et par sekunder.

[47] http://code.google.com/p/android/issues/detail?id=6348
[48] http://www.anandtech.com/show/4150/googles-android-event-analysis/2
[49] http://stackoverflow.com/questions/3190623/make-bluetooth-on-android-2-1-discoverable-indefinitely

4.3.7 Klassediagram

Applikationen indeholder tre aktiviteter og fire hjælpeklasser. Klassediagrammet kan ses på Figur 4.8. Der henvises i øvrigt til Appendiks 1 – Klassediagram, hvor en større version kan ses.

De tre aktiviteter er Main, Profile og MutualFriends. Det er disse som brugeren interagerer med. Person, PersonArrayAdapter, ImageLoader og Database bliver brugt af de tre aktiviteter til enten at håndtere, vise, hente eller sende data.



Figur 4.8: Klassediagram

I det følgende beskrives aktiviteterne og hjælpeklasserne i detaljer.

4.3.8 Aktiviteter

Applikationen er opdelt i tre aktiviteter – en for hvert skærmbillede. Hver aktivitet foretager operationer i en eller flere tråde. Således er det muligt at navigere videre fra et skærmbillede, inden alt data er indlæst. Den pågældende tråd vil da fortsætte sin handling, og brugeren vil opleve at vende tilbage til et skærmbillede med det forespurgte data indlæst.

Vi benytter to typer tråde. Almindelige arbejder-tråde samt den Android-specifikke ”runOnUiThread” (betegnet med (UI) i Tabel 4.2 nedenfor). Den mest interessante er sidstnævnte, som benyttes til at opdatere indhold på skærmen. Disse tråde afvikles med det samme, og det er derfor anbefalet ikke at køre tidskrævende kode heri. I praksis ville dette betyde at tråden blokerer for opdateringer og indtastning i den viste aktivitet på skærmen, og deraf at brugeren ikke kan betjene applikationen. I vores applikation er de tidskrævende handlinger indlæsning af data og kald til Facebook Graph. Dette afvikler vi i stedet i arbejder-tråde, som i de fleste tilfælde ikke vil genere brugeren [18].

Tabel 4.2: Aktiviteter og tråde

Aktivitet	Skærmbillede	Tråde	Handling
Main	Søgebillede	load() sort (UI) showToast (UI)	Hent data, kør FB kald. Løbende sortering. Beskeder til brugeren.
Profile	Profilvisning	getLikeName Likes (UI) showToast (UI)	Hent navn på ”Synes godt om” fra Facebook Graph. Listen med ”Synes godt om” opdateres løbende. Beskeder til brugeren.
MutualFriends	Fælles venner	loadMutuals	Hent venneliste fra database og hent navn på fælles venner fra Facebook Graph.

4.3.9 Person objekt

Til at repræsentere de fundne personer, som vises i både Main og MutualFriends, har det været mest hensigtsmæssigt at lade objektet ”Person” holde styr på alle de data, der er nødvendige i forhold til hver enkelt person, såsom Facebook ID og URL til profilbillede. Dette gør det lettere at få fat i alt data tilhørende en person fra alle aktiviteter, da man blot henter det ønskede Person objekt fra ArrayListen og dermed har adgang til alle oplysninger. Denne løsning er samtidig simpel at udbygge med flere oplysninger senere.

4.3.10 Main

Når Main aktiviteten startes, initialiseres hovedmenuen med adgang til skan og 2 testmuligheder. Desuden initialiseres begge knapper i topmenuen, men de holdes skjult indtil deres funktion er nødvendig. Derudover initialiseres OnClickListeners på det ListView, der står for den grafiske præsentation af et søgeresultat. Endelig startes login proceduren.

Login proceduren kalder først facebook.autorize med krav om at have tilladelse til `"user_about_me", "user_likes", "publish_stream".` Når brugeren har logget ind, og hvis det er første gang den benyttes, givet adgang til vores applikation, hentes brugerens navn, ”Synes godt om” og id vha. funktionskaldet:

```
Bundle fieldsToLoad = new Bundle();
String whomToLoad = "me";
fieldsToLoad.putString("fields", "name,id,likes");
asyncRunner.request(whomToLoad,fieldsToLoad, new
Fbid_RequestListener());
```

Når Facebook har svaret hentes JSON Array’et med ”Synes godt om” og konverteres til en semikolonsepareret String på formen ”Like1ID;Like2ID;…;” , som sendes til serveren sammen med brugerens Facebook ID og enhedens Bluetooth MAC adresse. Når dette er gjort hentes brugerens venneliste på lignende vis, konverteres til en String på samme form som tidligere og sendes til serveren. Dog gemmes vennelisten også i et Array af Strings lokalt i applikationen, hvor hvert element er en String indeholdende vennens ID.

I menuen er der 3 muligheder: Skan folk i nærheden, test med 10 personer og test på valgfrit antal personer. Alle 3 funktioner afslutter på samme måde - forskellen ligger i

måden hvorpå String Array'et med MAC adresser bliver genereret. Til testmuligheden hvor brugeren selv kan vælge antallet af personer der skal indlæses, åbnes en dialogboks, hvori antallet vælges. Dialogboksen godtager kun tal, som bliver parset til en integer, hvorefter det valgte antal MAC adresser tilføjes til et String Array. Vi benytter et antal test adresser, som vi manuelt har tilknyttet data til.

Applikationens egentlige funktion – at skanne efter folk i nærheden og tilføje dem hvis Bluetooth er synlig – udføres ved at igangsætte en Bluetooth skanning. Findes en enhed i nærheden, tjekkes det om denne enhed allerede er indlæst (sommetider findes samme enhed flere gange ved hver skanning), og MAC adressen for hver ny enhed gemmes.

For hver funden enhed hentes herefter personernes informationer fra serveren, og ud fra vennelisten for hver person udregnes antallet af fælles venner. Derefter hentes informationer for hver person fra Facebook Graph og gemmes i et Person objekt. Endelig vises personerne i et ListView, som er sorteret efter antallet af fælles venner.

Efter en fuldendt skanning vises to usynlige knapper og brugeren har nu også valget mellem at finde flere, som tilføjes de allerede fundne, eller rydde de nuværende og foretage en "ren" søgning.

4.3.11 Profile

Når aktiviteten Profile startes, initialiseres alle fire knapper, den valgte person findes i listen fra Main, profilbilledet downloades ud fra den URL der er gemt i Person objektet, og al data vises for brugeren. Til sidst hentes navnene på vedkommendes "Synes godt om" fra Facebook ud fra den String der er gemt i Person objektet, og vises i et TextView.

De fire knapper i Profile aktiviteten har funktionerne: "Tilbage", "Fælles venner", "Skriv på væg" og "Vis Facebook profil". Når brugeren trykker på "Tilbage"-knappen, lukkes den nuværende aktivitet og fjernes fra stakken af aktiviteter, og Main aktiviteten kommer derfor i front. Dette sker også hvis man trykker på telefonens fysiske "Tilbage"-knap. Når funktionen "Fælles venner" vælges, åbnes aktiviteten MutualFriends.

Hvis brugeren i stedet vælger at trykke på ”Skriv på væg” knappen åbnes en tekstboks, hvor en hilsen kan indtastes og sendes. ”Vis Facebook profil” knappen åbner profilen for den valgte person i den officielle Facebook applikation.

4.3.12 MutualFriends

Aktiviteten MutualFriends henter personens venner fra databasen og finder de fælles venner der måtte findes. Dette gøres ved brug af frameworket Java Collections[50]. Frameworket indeholder mange funktioner til mange formål, dog benytter vi kun ListUtils.intersection til at finde skæringen af de 2 vennelister. Således returneres en ny liste indeholdende de venner der er fælles. I det følgende kan ses hvordan personens venneliste fra databasen konverteres til et Array af Strings indeholdende de enkelte venner på hvert index, hvorefter skæringen findes:

```
private List<String> mutualFriends(String friendlist, int
friend_count) {
     String[] friendlist_temp = new String[friend_count];
     int i;
     int pos_start = 0;
     int pos_end = 0;

     for (i = 0; pos_end+1<=friendlist.length(); i++) {
          pos_end = friendlist.indexOf(";", pos_end);
          friendlist_temp[i] = friendlist.substring(pos_start,
               pos_end);
          pos_start = pos_end+1;
          pos_end = pos_end+1;
     }
     List<String> mutuals =
          ListUtils.intersection(Arrays.asList(friendlist_temp),
          Arrays.asList(Main.myFriendlist));
     return mutuals;
}
```

[50] http://commons.apache.org/collections/

4.3.13 PersonArrayAdapter

En ArrayAdapter [19] kan bl.a. benyttes til at vise data i et TextView. Den indbyggede getView-metode benyttes til at opdatere netop det indhold, brugeren har scrollet hen til. Således bliver indlæsningen af større mængder data prioriteret.

Ved at override getView-metoden kan den returnere andre typer af layout – i vores implementering et View med både billede og tekst indlejret. En ArrayAdapter initialiseres med:

```
personArrayAdapter = new PersonArrayAdapter(Main.this,
R.layout.friendlist, persons);
```

Parametrene er henholdsvis datakilden (i dette eksempel Main), et ID for det layout der styrer fremstillingen og formateringen af data og den ArrayList der indeholder data. Efterhånden som ny data indlæses, opdateres ArrayAdapteren med kaldet `personArrayAdapter.notifyDataSetChanged();` , hvilket brugeren vil opleve, som at fundne personer vises efterhånden som deres data er downloadet. Det er i denne sammenhæng vigtigt, at opdateringen sker i UI-tråden, da applikationen ellers vil lukke ned.

4.4 Problemer

Applikationen demonstrerer konceptet fuldt ud, men lider stadig under nogle ”børnesygdomme”.

Hvis enheden er koblet til WiFi samtidig med at Bluetooth er tændt, vil WiFi i ny og næ lukke ned. Problemet opstår altså, hvis begge radioer benyttes samtidigt, og er oplevet af en del andre Android-ejere, dog uden udsigt til en forklaring eller løsning [20]. Dette problem har vi prøvet at løse ved at sikre, at Bluetooth skanningen er færdig og radioen slukket, før vi starter med at kontakte serveren via Wifi. Problemet dukker dog stadig op en gang i mellem. Samme problem har ikke vist sig, når brugeren er på det mobile net.

Derudover er der lidt forhindringer forbundet med Facebook API, da det ikke er muligt at skrive på personers væg, med mindre man er venner med dem. Dette kunne vi godt ønske at løse ved at skrive privatbeskeder til personer i stedet, men dette er heller ikke understøttet for andre end Facebooks officielle applikation.

Endelig er der et problem med at få vist personers Facebook profil via linket fra profilvisningen. Dette kan ikke lade sig gøre, med mindre brugeren har været logget på i den officielle Facebook applikation tidligere, eller har tilknyttet en Facebook konto i sin Android enhed. Dette problem kan undgås ved at bruge internetversionen af Facebook, men vi har valgt at det skulle ske i Facebooks egen applikation, og går derfor ud fra at brugerne har tilknyttet en Facebook konto til deres enhed.

4.5 Forbedringer

Overordnet set er vi tilfredse med den applikation vi har udviklet. Den er designet til at kunne demonstrere et koncept, og set i det lys opfylder applikationen vores mål. Ikke desto mindre er visionen, at konceptet skal udvikles yderligere, og i det følgende vil vi komme med vores bud på, hvilke forbedringer der vil være interessante.

4.5.1 Implementeret i DTUs officielle applikation

Dette har været meningen fra projektets begyndelse. Ventetid på svar på juridiske spørgsmål i forhold til DTUs applikation har gjort, at vi ved projektets afslutning endnu ikke har fået adgang til kodebasen. Når dette sker, er det forhåbningen at vores projekt kan inkorporeres og ad den kanal blive afprøvet af en stor gruppe brugere.

4.5.2 DTU database

Det er selvfølgelig uholdbart at vi er kørende på et domæne, hvortil vi har begrænsede rettigheder i forhold til databasen. Nogle medstuderende kører et sideløbende projekt, med nogle ideer til en cloud baseret serveropsætning hos Amazon, som ser spændende ud. Det er planen, at dette skal udgøre en samlende platform for flere forskellige projekter hos IMM, som ad den kanal vil kunne drage fordel af hinandens data.

4.5.3 CampusNet

Ved at integrere vores funktionalitet i DTUs applikation, åbnes op for en ny verden, hvor vi pludselig har adgang til bl.a. hvilke kurser brugerne følger. Det er oplagt at dette skal indgå som vægt i sorteringen af søgeresultatet. Således vil øverste resultat i søgevinduet være den person, brugeren har mest til fælles med.

4.5.4 Sortering efter flere kriterier

I forlængelse af ovenstående, vil vi gerne gøre sorteringen mere reel. Jo flere parametre, jo mere interessant vil søgeresultatet være for brugeren. Fx kan det medregnes, om man har nogle tidligere ”check ins” til fælles, har boet i samme by, er tilhænger af samme politiker eller fodboldklub osv.

4.5.5 Simplificeret GUI

Som tidligere nævnt er testfunktionerne udelukkende medtaget, for at applikationen skal være brugbar i sin nuværende form. En endelig version vil sandsynligvis indgå som en del af en større applikation, og GUI’en bør derfor genovervejes til den tid.

Kapitel 5

Test

I dette kapitel beskrives den brugerundersøgelse, vi har foretaget.

Test er en vigtig del af ethvert softwareudviklingsprojekt. Et færdigt stykke software, som ikke har været testet, vil sandsynligvis skuffe kunden og brugerne, efterhånden som de ikke registrerede fejl dukker op til overfladen. I udviklingsprojekter i dag er test derfor blevet en essentiel part i hele forløbet.

En test kan afsløre fejl i kodningen, ved fx at foretage kørsler med forskellige datamængder, negative værdier og grænsetilfælde. Simulationstest af mange brugeres samtidige forespørgsel til en database kan afsløre problemer med performance, flaskehalse på netværket eller en uhensigtsmæssigt opsat server. For hver type software der eksisterer, findes mange forskellige former for test og meninger omkring hvem, hvornår og hvad der skal testes.

Den applikation, der er udviklet i dette projekt har til hensigt at demonstrere et koncept. Applikationen har således lang vej endnu, før den vil være færdigudviklet og klar til at blive sendt i produktion. Af denne grund anser vi det for overflødigt, at foretage en gennemgribende softwaretest på et så tidligt stadie – applikationen vil senere hen blive justeret og tilrettet, og en ny test vil være påkrævet uanset hvad.

Ikke desto mindre har vi testet og presset applikationen gennem hele forløbet. Pointen i ”proof of concept” er jo netop, at applikationen kan fremvises og demonstrere hvad den har at byde på. Antallet af fejl i en sådan demonstration, skal selvfølgelig tilsigtes at være så lavt som muligt.

Mens en omfattende softwaretest af de nævnte grunde ikke har været prioriteret, er vi derimod interesseret i, hvordan brugere modtager konceptet. Det er på dette stadie i forløbet interessant, at få forskellige meninger omkring, hvordan og i hvilke situationer

applikationen tænkes benyttet. Til at belyse dette, har vi foretaget en kvalitativ test, hvor vi har ladet fem personer afprøve applikationen, og efterfølgende stillet spørgsmål til deres oplevelse af den. Testpersonerne er udvalgt af forskellige aldersgrupper med den eneste betingelse at de har en Facebook profil. Ingen af dem har tidligere haft applikationen i hånden, men de har på forhånd haft et kendskab til projektet. Da applikation er ment som ”proof of concept” er det ikke en test af brugergrænsefladen, men i stedet en test af idéen og flowet i applikationen.

Testen er foretaget ved, at alle har fået samme introduktion til formålet med applikationen og formålet med testen. Herefter har vi bedt testpersonerne udføre tre opgaver i applikationen. På forhånd har vi indlæst nogle af testpersonens Facebook venner i databasen, for at sikre at ”Fælles venner”-menuen kunne testes. Slutteligt blev testpersonerne stillet en række uddybende spørgsmål til deres oplevelse. Testen er konstrueret efter principperne beskrevet i [21] og kan ses i afsnittet Appendiks 2 – Køreplan til test.

Kapitel 6

Evaluering

De adspurgtes svar kan findes i afsnittet Appendiks 3 – Resultater fra test.

Undersøgelsen har givet os et godt indblik i hvordan en gennemsnitlig bruger vil imødekomme vores applikation. Alle dem vi har interviewet havde ingen problemer med at benytte sig af vores applikation, hvilket peger på at det flow vi har valgt til applikationen fungerer.

Alle testpersonerne kunne godt lide idéen om, at man kan få vist folk der er i nærheden. I forhold til beskyttelse af private informationer er testpersonerne enige om, at så længe man selv vælger om man vil bruge applikationen samt hvad man vil dele, er der ikke noget problem. Det nævnes som positivt, at brugerne selv styrer, om de ønsker at være synlige, i kraft af at man skal aktivere Bluetooth for at gøre sig synlig. Dog blev det nævnt, at vi bør overveje at lade brugeren have en styring af, hvilke informationer der vises for hvem. Fx at vi henter ”Synes godt om” fra vores database og lader alle se dem – også personer, som ikke har adgang til dette via Facebook. Det er klart, at vi skal implementere en kontrol af dette, og som minimum give samme kontrol af informationsdeling, som Facebook selv tilbyder. Dette kunne gøres enten ved at hente indstillingerne fra Facebook, eller vise tydeligt hvilke informationer vi gør tilgængelige og give brugerne en mulighed for at afvise alle eller nogle af disse vilkår eller melde sig ud senere.

Alle testpersoner udtrykker, at de godt kan se sig selv bruge applikationen som den er, dog udtrykker flere, at de vil bruge den oftere hvis den er en del af noget større. Allerede inden vi har spurgt ind til Facebook nævner en testperson, at Facebooks officielle applikation er et oplagt sted at implementere vores koncept, hvilket bekræfter at denne idé er oplagt. Andre testpersoner tilkendegiver efterfølgende det samme.

Til sidst blev testpersonerne spurgt om eventuelle idéer til at videreudvikle applikationen, og her dukkede flere gode idéer op. En af personerne foreslår at der integreres en form for chatklient i applikationen, så man kan kommunikere med dem i nærheden uden at skulle igennem den integrerede Facebook applikation.

En anden synes det vil være smart at få forskellige virksomheder med i applikationen, så når man befinder sig i fx et indkøbscenter kan man få information om de butikker der er i nærheden, og eventuelt modtage tilbud via applikationen.

En tredje testperson synes applikationen bør fokusere meget mere på brugernes interesser og "Synes godt om". For eksempel kan brugerne inddeles i grupper efter hvilke sportsgrene de kan lide eller deres politiske overbevisning, og udbygge dette med flere måder at søge efter folk i nærheden på: almindelig søgning som den er nu, søgning specifikt efter personer med samme interesser og en intelligent søgning, der udvælger og viser de personer i nærheden, som vil være mest interessante for brugeren at se, beregnet ud fra forskellige parametre - altså en søgning hvor applikationen vælger hvilke krav der skal søges efter. Denne testperson nævner også, vi viser forbindelsen mellem venner, i de tilfælde hvor brugerne ingen fælles venner har. Således kan det udregnes, i hvilket led personer hænger sammen. I denne sammenhæng vil det være interessant også at se, hvilke af ens venner, venners venner osv., som er interesseret i det samme som en selv.

Flere af de idéer som testpersonerne bidrager med, kan umiddelbart implementeres i en senere iteration af applikationen. Fx vil en måde at kommunikere med personer i nærheden, uden at skulle bruge andre applikationer, være en god videreudvikling. Derudover passer nogle af vores egne forbedringsforslag godt sammen med testpersonernes ønske om at kunne søge efter specielle kriterier, såsom sportsinteresser eller politisk overbevisning. Dermed er det muligt at vælge enten at sorterer efter personer i nærheden eller blot vise de personer, der opfylder præferencer sat af brugeren.

Idéen om at kunne finde korteste vej mellem brugeren og en person uden fælles venner, kan blive en regnekraftig udvidelse, særligt ved et højt søgeresultat. Det vil derfor være mest fordelagtigt at implementere denne udregning på serveren, for ikke at belaste enheden unødvendigt.

Alt i alt har det været en udbytterig test. Vi har fået mange gode forslag til forbedringer af applikationen og alle testpersoner har tilkendegivet, at det er en applikation med et stort potentiale, som de godt kan se sig selv bruge, både i den nuværende udformning, men i endnu højere grad hvis den bliver en del af en større applikation. Her vil den officielle Facebook applikation være oplagt, men også den kommende DTU applikation vil være en interessant platform at få afprøvet konceptet på i større skala.

Kapitel 7

Diskussion

Indledningsvis beskrev vi visionen om en applikation, der kan vise hvem der befinder sig i umiddelbar nærhed af hinanden. Efter en analyse valgte vi at implementere en sådan applikation på Android platformen.

Vi besluttede tidligt i projektet, at målet skulle være en applikation, der demonstrerer selve konceptet. Denne beslutning blev taget dels for at holde projektet på et passende omfang, dels for at få mulighed for at afprøve konceptet og modtage respons og gode idéer til fremtidig videreudvikling.

Resultatet er blevet en funktionsdygtig applikation, beregnet til demonstration. Med applikationen i hånden er det muligt at foretage en skanning og få præsenteret hvem der befinder sig i nærheden. Det er dog kun muligt for applikationen at genkende enheder som tidligere har installeret applikationen. I praksis er applikationens udbredelse meget begrænset, så for at sikre mulighed for at kunne afprøve en situation hvor mange brugere befinder sig samme sted, har vi implementeret en test funktion. Således kan brugeren selv styre hvor mange andre enheder der skal indlæses.

Resultatet af en skanning præsenteres i sorteret rækkefølge, prioriteret efter antallet af fælles venner. Denne sortering er det oplagt at arbejde videre med. Hvis en søgning returnerer mange svar, er det essentielt at resultatet præsenteres med mest mulig relevans for brugeren. I afsnit 4.5 Forbedringer nævner vi flere forslag til, hvilke parametre der kan medtages. Bl.a. vil det være interessant at medregne fælles ”Synes godt om”.

Applikationen viser søgeresultatet med navn og billede. For hver person kan der vises en detaljeret profilside, hvor en stor version af personens profilbillede samt ”Synes godt om” og fælles venner vises. Det har været vigtigt for os, at applikationen ikke er begrænset til blot at præsentere en mængde navne og billeder, men også giver mulighed for at tage kontakt til andre brugere. Som nævnt i kravspecifikationen er det derfor muligt at skrive en besked på andre brugeres væg. Desværre er vi her underlagt

de generelle restriktioner fra Facebook, og det kan dermed kun lade sig gøre at skrive til de personer brugeren selv i forvejen er ven med, eller dem som tillader kontakt fra fremmede.

Facebook har desuden valgt at undgå en potentiel eksplosion i mængden af spam ved ikke at give adgang til at sende private beskeder, hvilket ellers er oplagt i en applikation af denne type.

Endelig tillader Facebooks API heller ikke muligheden for at tilføje nye venner. Dette har vi imødekommet ved at indsætte et link, der aktiverer den officielle Facebook applikation, hvis den er tilgængelig på enheden. Ad denne omvej er det således muligt alligevel at tilføje nye venner.

Gennem en mindre brugerundersøgelse har vi søgt at modtage respons på applikationen og konceptet. Undersøgelsen blev udført, ved at lade testpersonerne udføre tre simple opgaver i applikationen, og herefter svare på en række spørgsmål. Da applikationen ikke har til hensigt at være andet end demonstration af konceptet, har vi ikke været interesseret i at undersøge kvaliteten og brugervenligheden i det grafiske design. At vi har valgt at lade testpersonerne udføre tre opgaver, har derfor været ud fra devisen om, at personlig erfaring giver et mere reelt indtryk af applikationen, end en mundtlig introduktion fra os kan give.

Det essentielle i brugerundersøgelsen har været testpersonernes svar på en række spørgsmål. Her har vi modtaget en samling gode idéer til nye funktioner. Et interessant forslag er at opsætte Bluetooth sendere i butikkerne i fx et storcenter. Således vil vores applikation kunne sættes i stand til at kommunikere med de butikker brugeren passerer. Hvis butikken samtidig har adgang til oplysninger om brugeren, åbner der sig en helt ny verden af målrettet reklame, hvor butikken sættes i stand til at kommunikere tilbud og inspiration til netop de personer der befinder sig i og omkring butikken. Samtidig vil en dataindsamling over en periode kunne bruges til at udarbejde en analyse af hvilke typer og aldersgrupper der passerer butikken på hvilke tidspunkter, hvilket hver butik kan bruge til fx sammensætning af tilbud og planlægning af bemanding.

Et andet spændende forslag går på, at brugeren selv skal have mulighed for at målrette sin søgning. Som tidligere nævnt, har vi selv et ønske om at lade flere kriterier indgå i sorteringen af søgeresultatet. Det er i den forbindelse oplagt at lade brugeren selv have indflydelse herpå. Således kan brugeren nøjes med at få vist mænd, kvinder, danskere, amerikanere, personer med samme musiksmag og så videre.

Endelig har vi et ønske om at applikationen skal testes blandt studerende på DTU, som en del af den kommende officielle DTU applikation, som har adgang til data fra DTU CampusNet. Hvis dette føres ud i livet, er det planen at profilvisningen af fundne personer skal udvides til også at indeholde hvilke kurser vedkommende følger. Dette kan så samtidig indgå som en vægt i sorteringen af søgeresultater, sådan at studerende som følger eller har fulgt samme kurser som brugeren prioriteres.

Alt i alt fungerer applikationen rigtig godt. På trods af de nævnte begrænsninger i Facebook API og Android-relaterede udfordringer med WiFi og Bluetooth aktive samtidigt, lever applikationen op til vores krav. Vi ser desuden et stort potentiale i applikationer af denne type, og det er oplagt senere at implementere de gode forslag til forbedringer vi har modtaget.

Kapitel 8

Konklusion

Denne rapport beskriver udviklingen af en mobilapplikation til Android. Applikationen udnytter Bluetooth radioen i enheden til at lade brugeren se, hvem der befinder sig i nærheden. Denne funktionalitet kan bidrage med helt nye muligheder for den officielle Facebook applikation. Der er fra Facebooks side kun ringe udnyttelse af at deres produkt følger med brugerne rundt overalt. I denne rapport har vi set på hvordan man kan kombinere de sociale styrker i Facebook, med de muligheder en smartphone rummer.

Det primære fokus har været på at udvikle en applikation der kan demonstrere selve konceptet. I stedet for at sigte efter en færdig applikation, har vi ønsket at give vores bud på hvordan problemet kan løses.

Selvom applikationen fungerer som ønsket, er der rigt grundlag for at implementere de forslag til forbedringer vi har modtaget. En større brugerundersøgelse, foretaget med flere forskellige typer af personer og over en længere periode, vil bidrage med endnu flere idéer og forslag, som vil gøre den færdige applikation endnu bedre.

Ønsket er, at vores løsning bliver implementeret som delfunktionalitet i den officielle DTU mobilapplikation, når denne er klar. Ad denne kanal håber vi, at applikationen kan blive udbredt til en stor brugerskare og således afprøvet i stor skala, og endelig, at brugerne vil få en positiv og udbytterig oplevelse med denne nye måde at socialisere på.

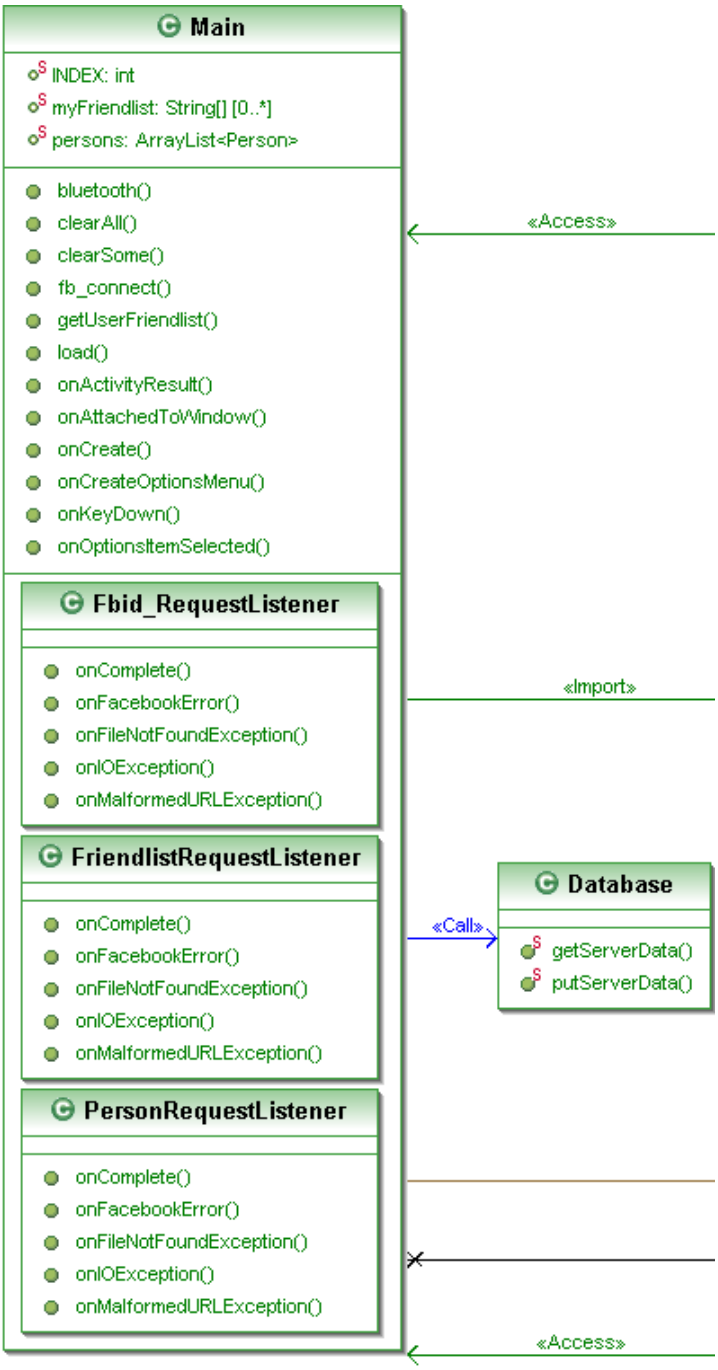
Problemformuleringen:

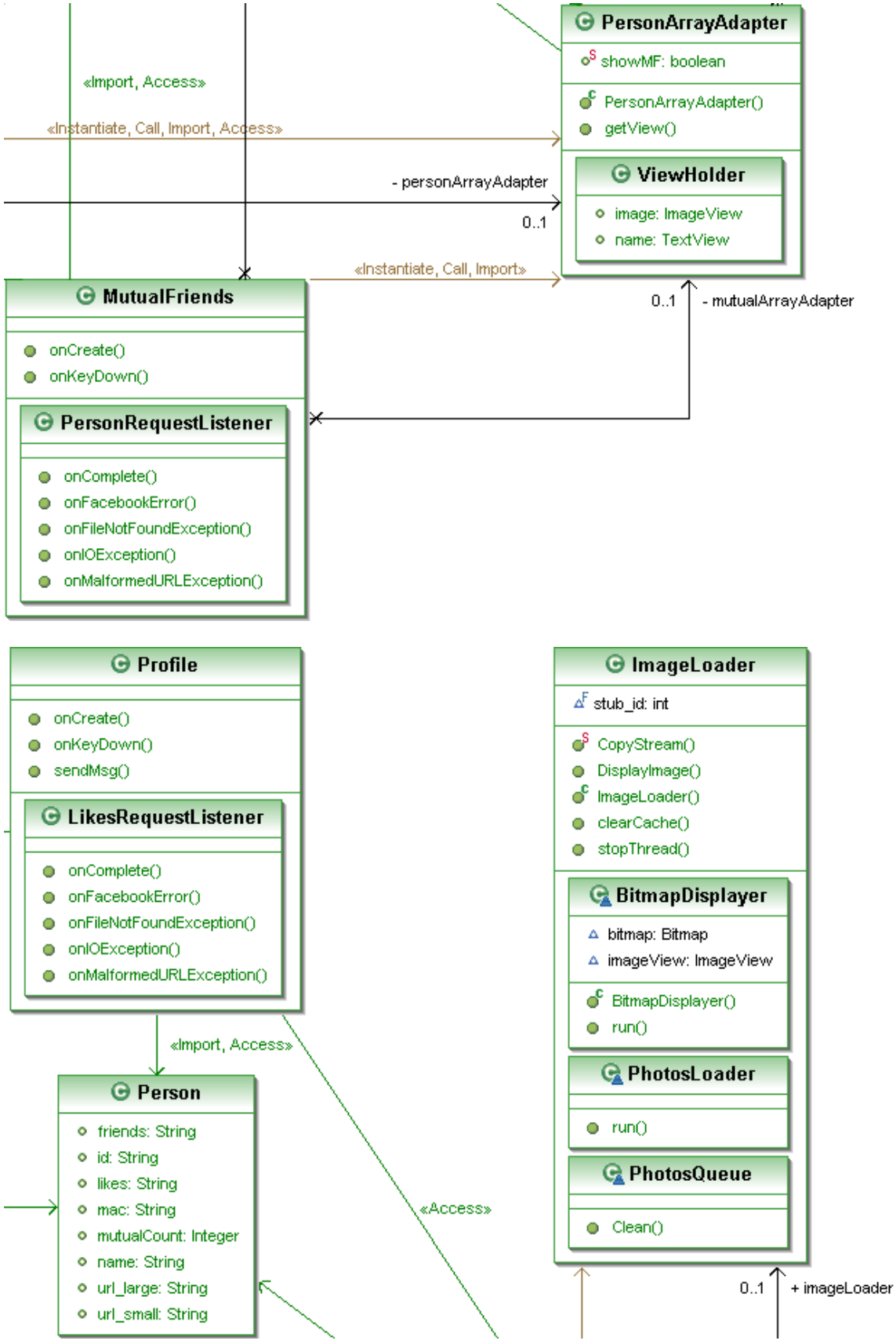
> *"Vi vil udvikle en mobilapplikation til smartphones. Den primære funktionalitet i applikationen vil være brugen af enhedens Bluetooth radio sammen med en kobling til Facebook Graph. Således kan der modtages data fra andre brugere af applikationen, som befinder sig i umiddelbar nærhed, og på den måde er det muligt at generere en oversigt over de Facebook profiler, som brugeren befinder sig tæt ved. Vi vil fokusere på at udvikle en helt grundlæggende "proof of concept" applikation i dette projekt, men samtidig undersøge hvilke muligheder for yderligere funktionalitet der senere kan implementeres."*

er blevet løst. Vi har udviklet en mobilapplikation, der ved at bruge Bluetooth og data fra Facebook, viser brugeren hvem der befinder sig i nærheden. Vi har udført en mindre brugerundersøgelse, der har bidraget med idéer til, hvilke forbedringer der vil gavne applikationen.

Appendiks

Appendiks 1 – Klassediagram




PersonArrayAdapter
showMF: boolean
PersonArrayAdapter()
getView()
ViewHolder
image: ImageView
name: TextView
«Import, Access»
«Instantiate, Call, Import, Access»
- personArrayAdapter
0..1
«Instantiate, Call, Import»
0..1
- mutualArrayAdapter
MutualFriends
onCreate()
onKeyDown()
PersonRequestListener
onComplete()
onFacebookError()
onFileNotFoundException()
onIOException()
onMalformedURLException()
Profile
onCreate()
onKeyDown()
sendMsg()
LikesRequestListener
onComplete()
onFacebookError()
onFileNotFoundException()
onIOException()
onMalformedURLException()
«Import, Access»
Person
friends: String
id: String
likes: String
mac: String
mutualCount: Integer
name: String
url_large: String
url_small: String
«Access»
ImageLoader
stub_id: int
CopyStream()
DisplayImage()
ImageLoader()
clearCache()
stopThread()
BitmapDisplayer
bitmap: Bitmap
imageView: ImageView
BitmapDisplayer()
run()
PhotosLoader
run()
PhotosQueue
Clean()
0..1
+ imageLoader

Appendiks 2 – Køreplan til test

Introduktion

- Tak fordi du vil hjælpe – det tager ca. 20 minutter
- Introduktion til applikationens formål
- Det skal ske:
    - Løse tre små opgaver så godt du kan
    - Svare på en række spørgsmål
- Det er en test af applikationen, ikke af dig
- Mens du løser opgaverne, må du meget gerne stille spørgsmål og komme med din mening og kommentarer
- Spørgsmål?
- Lad os starte

Opgaver

1. Åben applikationen og se, hvem der befinder sig i nærheden af dig.
    - *Husk at tænde Bluetooth, så testpersonen rent faktisk finder én*
    - *Testen er succesfuld, når søgningen viser et resultat*
2. Har du nogle fælles venner med ham? Hvilke?
    - *Opgaven er fuldført, når der er navigeret via profilvisning til fælles venner*
3. Send en lille hilsen til ham.
    - *Opgaven er fuldført, når der er navigeret hen til profilvisning og en besked er sendt med "Post on Wall"-funktionen.*

Spørgsmål

- Hvordan gik det med at løse opgaverne? Nemt, svært?
- Hvad synes du om, at man kan få vist hvem man er i nærheden af?
- Synes du der er nogle privatlivs aspekter, vi skal tænke over?
- I sin nuværende form, ville du være interesseret i at bruge applikationen selv?
    - Ja: Hvornår? Til hvad?
    - Nej: Hvad mangler applikationen, før du ville bruge den?
- Ville du hellere/oftere bruge applikationen, hvis den var en del af en større applikation?
- Har du den officielle Facebook applikation?
    - Kunne du forestille dig vores applikation integreret heri?
- Har du nogen ideer til, hvad applikationen mangler? Eller hvad der er overflødigt?

Appendiks 3 – Resultater fra test

Testpersonernes svar er samlet nedenfor.

- Hvordan gik det med at løse opgaverne? Nemt, svært?
    1. Opgaverne er lette nok, det er svært at gøre noget forkert.
    2. Det gik strålende med at løse opgaverne. Da jeg først fandt rundt i navigationen gik det nemt.
    3. Det var da nemt nok
    4. Det var ret let
    5. Det gik rigtig fint.
- Hvad synes du om, at man kan få vist hvem man er i nærheden af?
    1. Jeg synes det er ret smart, specielt til arrangementer med mange mennesker, ville man let kunne finde nogle man kender. Men selvfølgelig kun hvis man har installeret programmet og aktiveret Bluetooth.
    2. Jeg synes umiddelbart det er en god og for mig ny ide. Andre programmer som Facebook kan også, dog i langt mindre grad, give et billede af hvor folk er gennem check in, men denne giver et øjebliksbillede og derfor synes jeg det er en god ide, at lave et program der kan netop dette. Det er en af de ting, man ikke vidste at man manglede.
    3. Jeg synes det er fint nok, og da det er via en applikation er det jo frivilligt
    4. Det er da meget sjovt.
    5. Det er smart. Specielt hvis der nu er en, man synes ser sød ud og gerne vil snakke med, så er der en nemmere mulighed for at gå over og snakke med vedkommende.
- Synes du der er nogle privatlivs aspekter, vi skal tænke over?
    1. Nej egentlig ikke, for brugerne vælger vel selv at de har installeret programmet og aktiveret Bluetooth.
    2. Så længe man selv aktivt skal gå ind og aktivere Bluetooth osv., kan jeg ikke se problemer omkring privatlivet.
    3. Nej, det synes jeg ikke rigtigt da det jo er via en applikation som folk selv har hentet.
    4. Det kommer an på hvad man kan se fra folks profil, da nogen har bestemt at ikke alle skal have adgang til at se alt på Facebook, og dette skal så overholdes. Ellers skal man have nogle vilkår hvor

der står hvilke informationer som i henter og hvordan i bruger dem, så folk har mulighed for at acceptere dette.
5. Næh. Man har jo selv tilmeldt sig det her, når man bruger det. Så længe man selv kan vælge, hvornår andre må finde en.

- I sin nuværende form, ville du være interesseret i at bruge applikationen selv?
  - Ja: Hvornår? Til hvad?
    1. Hvis jeg vidste det var blevet mere udbredt, så kunne det måske være ret smart til store arrangementer så som fodboldkampe osv.
    2. Jeg kunne forstille mig selv bruge applikationen, eksempelvis er den smart hvis man skal finde en anden person i et område, hvor det kan være svært at finde rundt.
    3. Det kunne jeg da sikkert, da jeg er sådan en som tit prøver noget nyt.
    4. Det tror jeg sagtens jeg kunne finde på.
    5. Hvis jeg havde en smartphone, ja. Så ville jeg måske bruge den i toget når jeg keder mig, eller på café med mine veninder og gerne vil vide mere om en eller anden ovre i hjørnet.
- Ville du hellere/oftere bruge applikationen, hvis den var en del af en større applikation?
    1. Måske hvis den fik nogle udvidelser, men det er jo smart nok det den kan.
    2. Applikationen i sin nuværende form ville jeg nok oftere bruge, hvis den var en del at noget, jeg i forvejen bruger – eksempelvis Facebook.
    3. Ja. Det er lige det der mangler for at jeg kan se mig selv bruge den til andet end bare for sjov. Det ville også give flere muligheder til hvad man kunne gøre med applikationen.
    4. Ja. Kan godt se en fidus i det. Gerne sætte det sammen med at kunne finde folk i nærheden ud for interesser.
    5. Måske… Jeg er ikke rigtig vant til at bruge applikationer.

- Har du den officielle Facebook applikation?
    1. Ja
    2. Ja
    3. Ja
    4. Ja
    5. Nej
    - Kunne du forestille dig vores applikation integreret heri?
        1. Tjae, muligvis.
        2. Ja, det kunne jeg godt. I hvert fald dele af den, da man i forvejen kan kontakte andre via Facebook, vil dette måske være overflødigt også at kunne i denne applikation.
        3. Sagtens.
        4. Uden problemer.
        5. Ja det virker da oplagt.
- Har du nogen ideer til, hvad applikationen mangler? Eller hvad der er overflødigt?
    1. Integreret chat, så man ikke behøver gå ind på Facebook.
    2. Jeg synes det kunne være smart, hvis eksempelvis virksomheder eller uddannelsesinstitutioner kunne være med, og at man via den her applikation kunne finde disse, så den kunne bruges som et værktøj til navigation. F.eks. når man er i et storcenter, så kan man få tilbud fra de butikker man går forbi.
    3. Nej ikke lige umiddelbart.
    4. Som sagt tidligere, så det at finde folk i nærheden ud fra fælles interesser, eller en inddeling af folk i grupper så man kunne finde folk som havde interesser inde for en sport eller politiske overbevisninger eller lignende. Og så måske hvis man ikke har nogle fælles venner med den person man kigger på, så vise hvor mange, og hvilke, led man skal ud for at man hænger sammen. Man skal også kunne gå ind på de personer som man finde i fælles venner. Og have en måde at vise hvilke af mine venner, venners venner osv. som er interesseret i det samme som mig.
    5. Der er ikke for meget i hvert fald. Hvis der skal mere på, ville jeg gerne kunne skrive både på folks væg og sende private beskeder.

Appendiks 3 – Cd-rom indhold

Cd-rommen der medfølger rapporten indeholder en digital version af denne rapport samt kildekoden. I det følgende beskrives mappestrukturen. Det samme er afleveret elektronisk på CampusNet:

- **/apk/NeabyPeople.apk**
  - .apk filen med vores applikation
  - Er også tilgængelig på www.olivergroth.dk/app (tilgå fra Androidenhed for nem installation)
- **/NearbyPeople**
  - Eclipse projektet
  - **/res**
    - Ressourcerne i applikationen. Billeder, layouts osv.
  - **/src**
    - Kildekoden med kommentarer
  - **/com_facebook_android_src**
    - Facebook SDK til Android
  - **/lib**
    - ListUtils biblioteket brugt til at udregne fælles venner
  - **/AndroidManifest.xml**
    - Konfigurationsfil til vores applikation
- **/rapport.pdf**
  - Digital version af denne rapport

Figurer og tabeller

Kilder

1 eBizMBA. Top 15 Most Popular Social Networking Websites, juni 2011. http://www.ebizmba.com/articles/social-networking-websites

2 Gartner. Gartner Identifies 10 Consumer Mobile Applications To Watch In 2012, marts 2011. http://www.gartner.com/it/page.jsp?id=1544815

3 International Communication Union. The World In 2010, marts 2011. http://www.itu.int/ITU-D/ict/material/FactsFigures2010.pdf

4 Danmarks Statistik. Smartphone i hver tredje familie, maj 2011. http://www.dst.dk/pukora/epub/Nyt/2011/NR206.pdf

5 Facebook. Facebook Statistics, marts 2011. http://www.facebook.com/press/info.php?statistics

6 Michael Eyal Sharon. Who, What, When, and Now…Where, april 2011. http://blog.facebook.com/blog.php?post=418175202130

7 Google, Google Blog. See where your friends are with Google Latitude, april 2011. http://googleblog.blogspot.com/2009/02/see-where-your-friends-are-with-google.html

8 Wikipedia. foursquare (social network), april 2011. http://en.wikipedia.org/wiki/Foursquare_%28social_network%29

9 PR Newswire. Report March 2011 U.S. Mobile Subscriber Market Share, marts 2011. http://www.prnewswire.com/news-releases/comscore-reports-march-2011-us-mobile-subscriber-market-share-121396514.html

10 Android. Bluetooth, marts 2011. http://developer.android.com/guide/topics/wireless/bluetooth.html

11 Android. JSON Package Summary, org.json, marts 2011. http://developer.android.com/reference/org/json/package-summary.html

12 Peter Svensson. Apple publishes guidelines for app approval, marts 2011.
http://www.msnbc.msn.com/id/39076144/ns/technology_and_science-wireless/t/apple-publishes-guidelines-app-approval/

13 Facebook. Facebook Developers – Graph API, marts 2011.
http://developers.facebook.com/docs/reference/api/

14 OAuth. OAuth 2.0, marts 2011.
http://oauth.net/2/

15 howstuffworks. How Bluetooth Works, marts 2011.
http://electronics.howstuffworks.com/bluetooth1.htm

16 Facebook. Facebook Developers, Mobile Apps – Android, marts 2011.
http://developers.facebook.com/docs/guides/mobile/#android

17 SP Technolab Blog. Android: Connecting the MySQL using PHP, marts 2011.
http://blog.sptechnolab.com/2011/02/10/android/android-connecting-to-mysql-using-php/

18 Android. Painless Threading, marts 2011.
http://developer.android.com/resources/articles/painless-threading.html

19 Android. ArrayAdapter, marts 2011.
http://developer.android.com/reference/android/widget/ArrayAdapter.html

20 Android Forums. Bluetooth kills WIFI? (help), april 2011.
http://androidforums.com/droid-x-support-troubleshooting/129738-bluetooth-kills-wifi-help.html

21 Rolf Molich. *Usable Web Design*, kapitel 11, side 128-158. Nyt Teknisk Forlag, første udgave, 2007.

Institut for Informatik og Matematisk Modellering
Danmarks Tekniske Universitet
Bygning 321
DK-2800 Kgs. Lyngby
Danmark

www.milab.imm.dtu.dk
Tel: +45 45 25 33 51 / Fax: +45 45 88 26 73
E-mail: milab@imm.dtu.dk