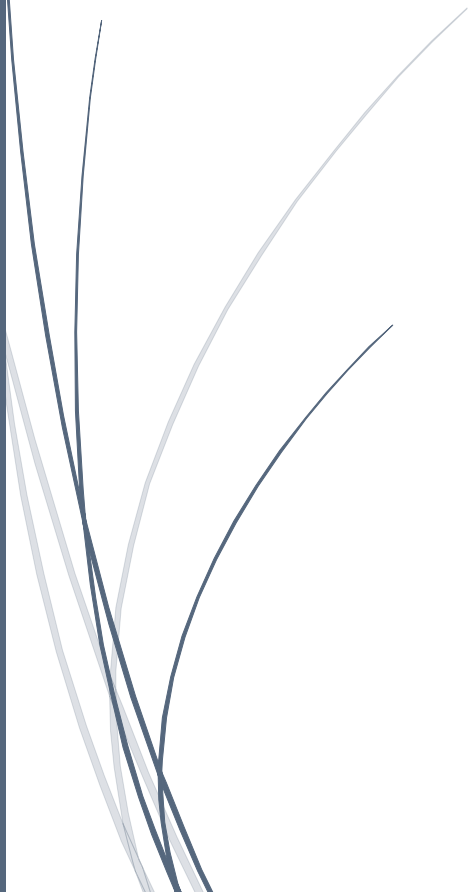
04-06-2015

Virtuel Cardholder

ITSMAP – Theme Project Group 15



Anders Lassen - 201270933, David Buhauer - 201270749, Peter Kragelund - 201270868 & Thomas Neergaard - 11812

Indholdsfortegnelse

Projekt formulering

Det kan tit være et stort irritations element at rende rundt med 10 forskellige medlemskort i sin pung. Derfor har vi valgt at lave en app som kan varetage denne opgave og man slipper for at skulle have de mange medlemskort med på farten. Et medlemskort er sådan et plastic kort ligesom et dankort, men har den funktion at med dette kort viser man at man er medlem af en given butik, det kan f.eks. være et Club Matas Kort. I Matas får man dermed rabat på udvalgte varer. Kortet er gratis, men er med til at sikre butikken at de vælger deres butik frem for en anden, når de nu er medlem allerede.

Projektet er delt om i 3 iterationer,

Iteration 1:

- Layout / GUI delen af app'en
- Det skal være muligt at tilføje et kort

Iteration 2:

- Persistere data lokalt (SQLite DB)
- Det skal være muligt at slette et kort
- Det skal være muligt at ændre et kort

Iteration 3:

- Kan finde den nærmeste butik via google maps.
- Persistering af Billeder i SQLite DB
- Tillad at app'en hjælper dig med at huske benyttelsen af medlemskort (Notifikation)

Iteration 4:

- Benytte NFC til at gemme data om et kort

Vi har nået at udvikle Iteration 1 til og med 3 og har valgt ikke at lave iteration 4. Ideen med at kunne gemme et kort's chip eller magnetstribe synes vi er en brugbar ting at have med, men vi har ingen måde hvorpå vi kan indlæse disse informationer ind i app'en og har derfor valgt ikke at udvikle iteration 4.

Design Valg

Vores app består af fire Activities: MainActivity, AddCard, EditActivity og SettingsActivity. Opdelingen er foretaget ud fra den betragtning, at hver Activity håndterer hver sin kernefunktionalitet i app'en, da det virkede som en logisk og overskuelig måde at dele koden op på, og gjorde det nemmere at dele arbejdet op imellem gruppens medlemmer.

MainActivity er klart den største og væsentligste Activity, da den fungerer som omdrejningspunkt for meget af app'ens funktionalitet, og fungerer ligeledes som app'ens "hjemmeskærm". MainActivity indeholder et overblik over de kort, brugeren har tilføjet til app'en, og derfor er det også MainActivity der er valgt som start Activity for app'en. Som det fremgår af vores kravspecifikation, var der også planlagt en Activity til at logge ind, der ville have været app'ens opstarts-Activity, men pga. mangel på tid og mening med sådan en funktionalitet, blev den opgivet.

AddCard bliver startet af MainActivity og er der, brugeren tilføjer nye kort ved at tage et billede af forsiden og bagsiden af sit kort og tilføjer et navn, og evt. et kortnummer og noter til dette kort. Brugeren skal tage billeder af begge sider af kortet, så vi er sikre på at få al information med, uanset hvilket kort, der er tale om; nogle kort indeholder ingenting på forsiden, andre ingenting på bagsiden. Navn og nummer er så app'en i sin oversigt over kort, effektivt kan liste, hvilket kort, der er tale om, da den (trods alt) ikke selv kan læse ud fra billedet, hvilket kort, der er tale om. Noter kan optionelt tilføjes, hvis der er omstændigheder omkring kortet, brugeren mener er vigtigt at få noteret (fx Blockbuster: Er lukket alle stedet, bør slette kort).

EditActivity gør langt hen ad vejen det samme som AddCard, men her har brugeren mulighed for at ændre de informationer, der er blevet indtastet i AddCard. Til begge dele gøres der brug af en singleton til at gemme data'en ved lifecycle shifts, da Intents bliver fyldt op og derfor ikke bevarer vores data.

SettingsActivity giver brugeren mulighed for at starte en notifikation, der afspiller en lyd. Dette er ikke den vigtigste funktionalitet for at gøre oplevesen af app'en god, men det giver os mulighed for at vise brugen af Preferences, da den gemmer indstillingerne for denne notifikation i en Preference, så den bliver persisteret og brugbar i hele app'en. Det ønskelige formål med denne Activity, var at implementere geofencing, således app'en ville notificere brugeren om, når vedkommende var inden for en hvis afstand af en forretning tilhørende et eller flere specifikke kort.

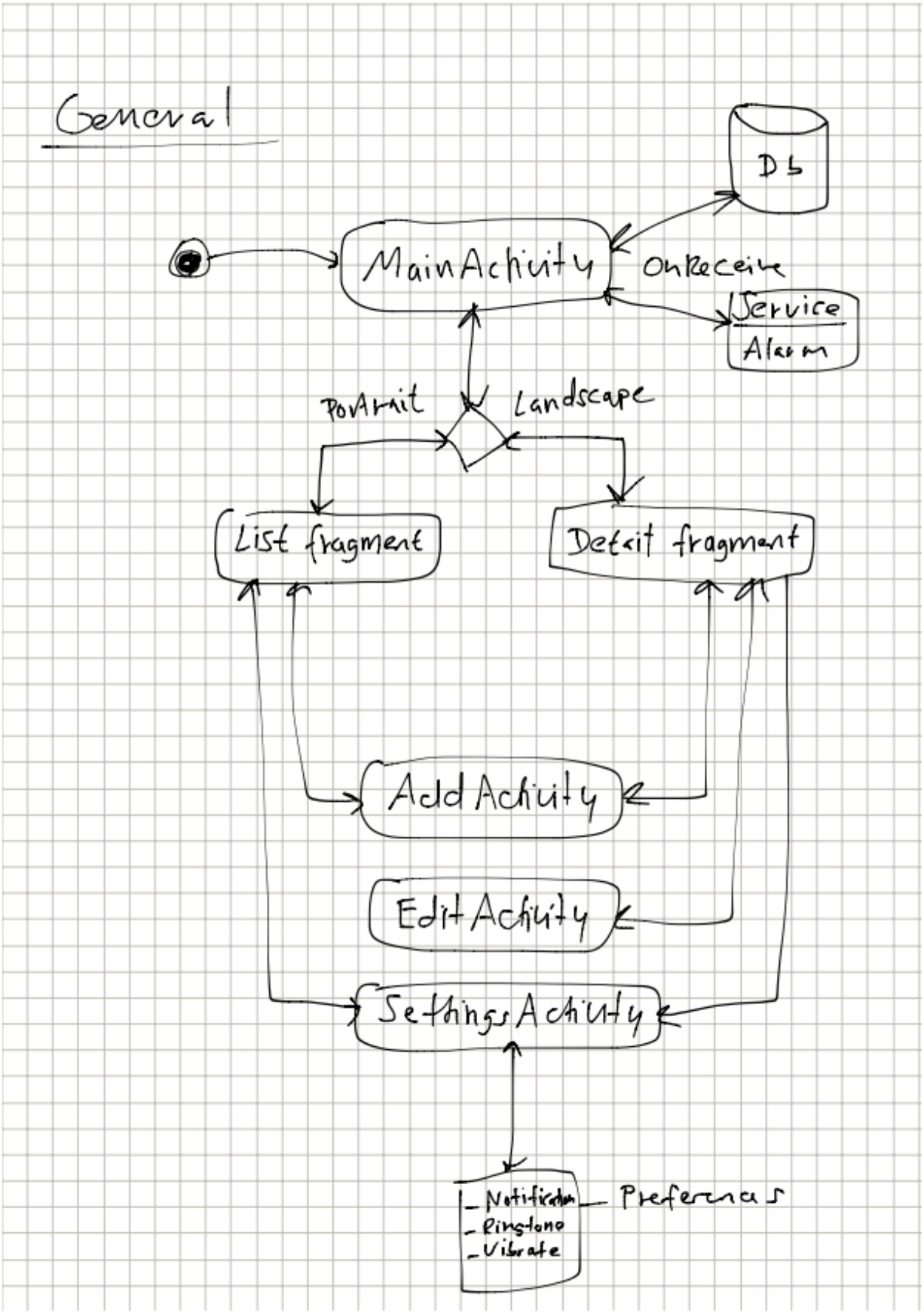
UML Diagrammer

I figur 1 ses det generelle flowdiagram. Essensen er når app'en starter op, starter vi i MainActivity. MainActivity har tilknyttet en SQL database og service til håndtering af notifikationer. Når vi står i MainActivity kan vi vælge at få vist et list view eller detail view over vores virtuelle kort liste. Hvert view består af en fragment som MainActivity står for at initialisere via Fragment Manageren. Essensen er så, at når vi står i portrait mode får vi vist list fragment og dermed listen over kort uden detaljeret info. Her har man mulighed for at tilføje en nyt kort fra actionbaren, hvor AddActivity bliver startet. Udover det, kan man se sine preferencer, hvilket også gøres via actionbaren. Når vi står i landscape mode får vi vist list detail fragment, og dermed listen over kort med detaljeret info. Her har man de samme muligheder som i portrait, blot kan man nu ændre på kortene. Dette kan man dog også gøre i portrait, hvis man trykker/tapper på et kort i listen og derefter trykker på edit ikonet i actionbaren.

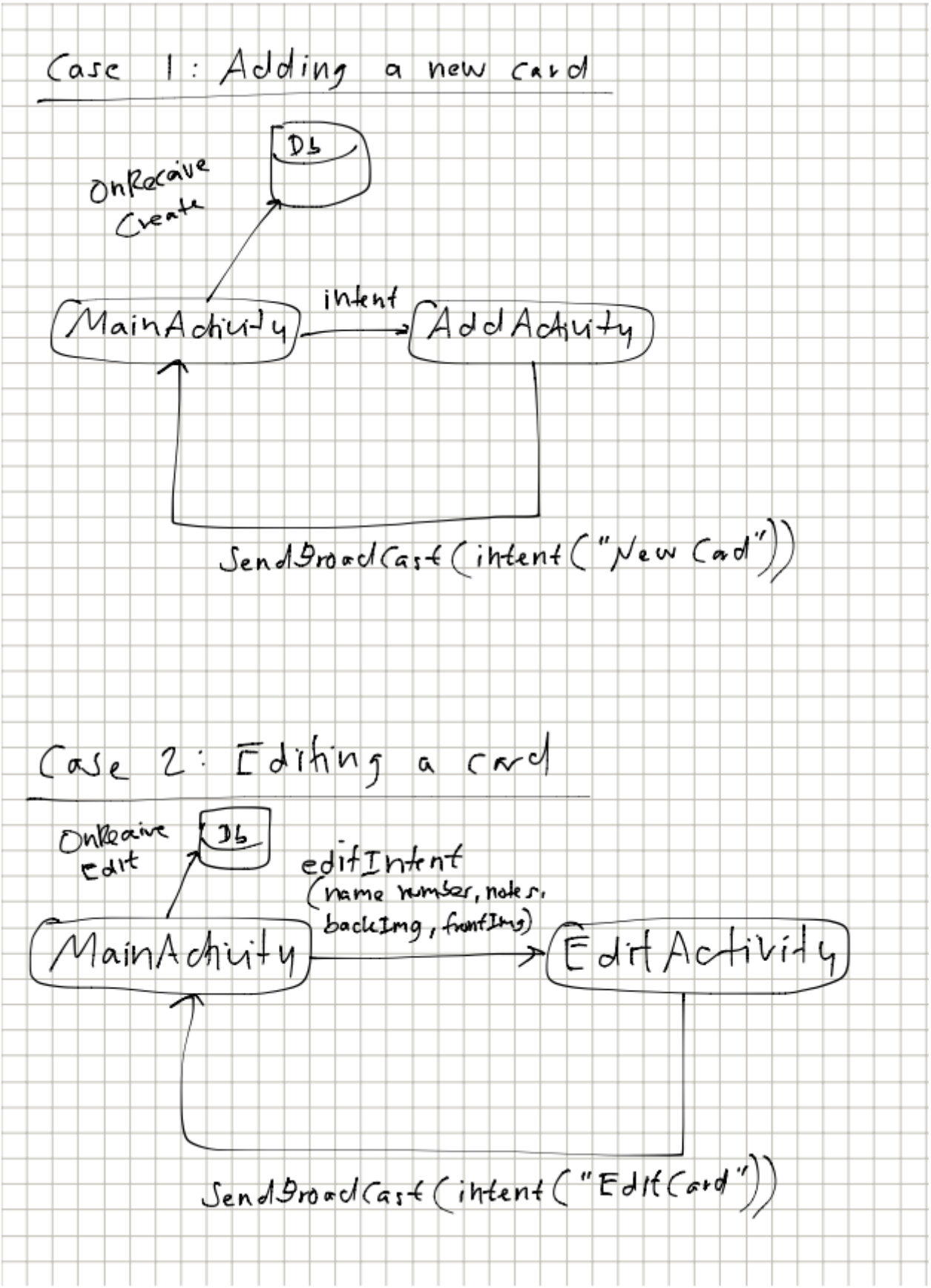
I figur 2 ses de to cases (case 1 og 2) hvor man har mulighed for at tilføje et nyt kort eller ændre på et eksisterende kort. Kigger vi på case 1 – tilføjelse af kort – står vi i MainActivity og afhængig af om vi står i portrait eller landscape, bliver list view fragment vist eller detail view fragment. Når vi så i actionbaren trykker på '+' ikonet åbnes AddActivity hvor vi sender en intent med uden extras. Når man så er færdig med at indtaste info om det nye kort og trykker på 'accept' ikonet, sendes et broadcast fra AddActivity, som MainActivity vil registrer via intentfilteret "NewCard", hvor kortet vil blive tilføjet til databasen og kortlisten vil blive opdateret. Samme fremgangsmåde er blevet anvendt når vi kigger på case 2 – ændring af kort. Her sendes en intent med fra MainActivity til EditActivity med extras der indeholder kortets info. Når nye ændringer er blevet foretaget, sendes ligeledes en broadcast tilbage til MainActivity med et Intentfilter "EditCard".

I figur 3 ses case 3 hvor man har mulighed for at slette et eksisterende kort. Afhængig af om man står i portrait eller landscape er det muligt at slette kortet fra listen og dermed også fra SQL databasen.

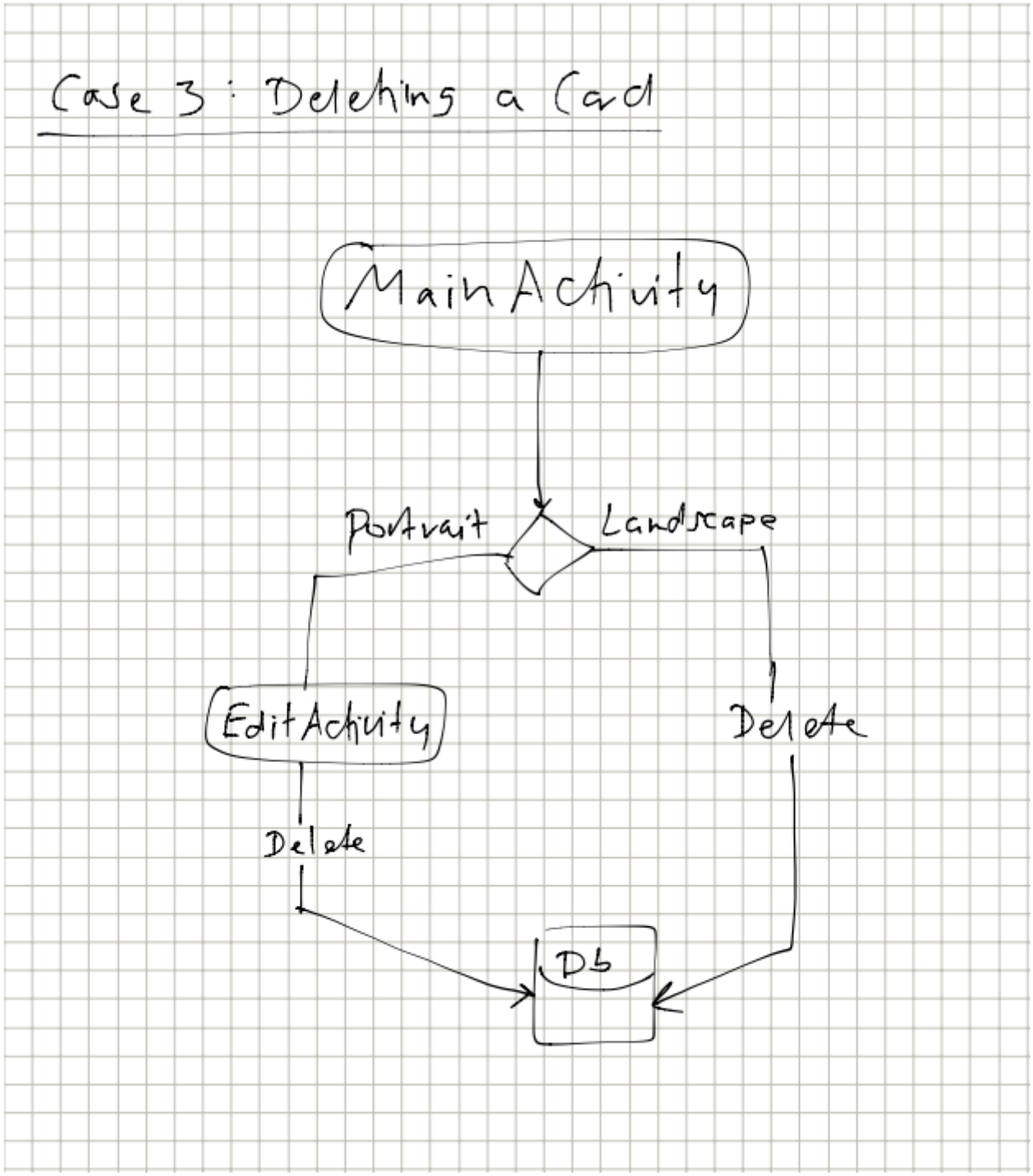
I figur 4 ses case 4 hvor man har mulighed for at åbne Google Maps og finde nærmeste butik fra kortet, man har valgt fra listen. Her vises detail fragment frem samt en knap vil blive vist. Når der trykkes på knappen sendes en mapIntent med, hvorved Google Maps åbnes.



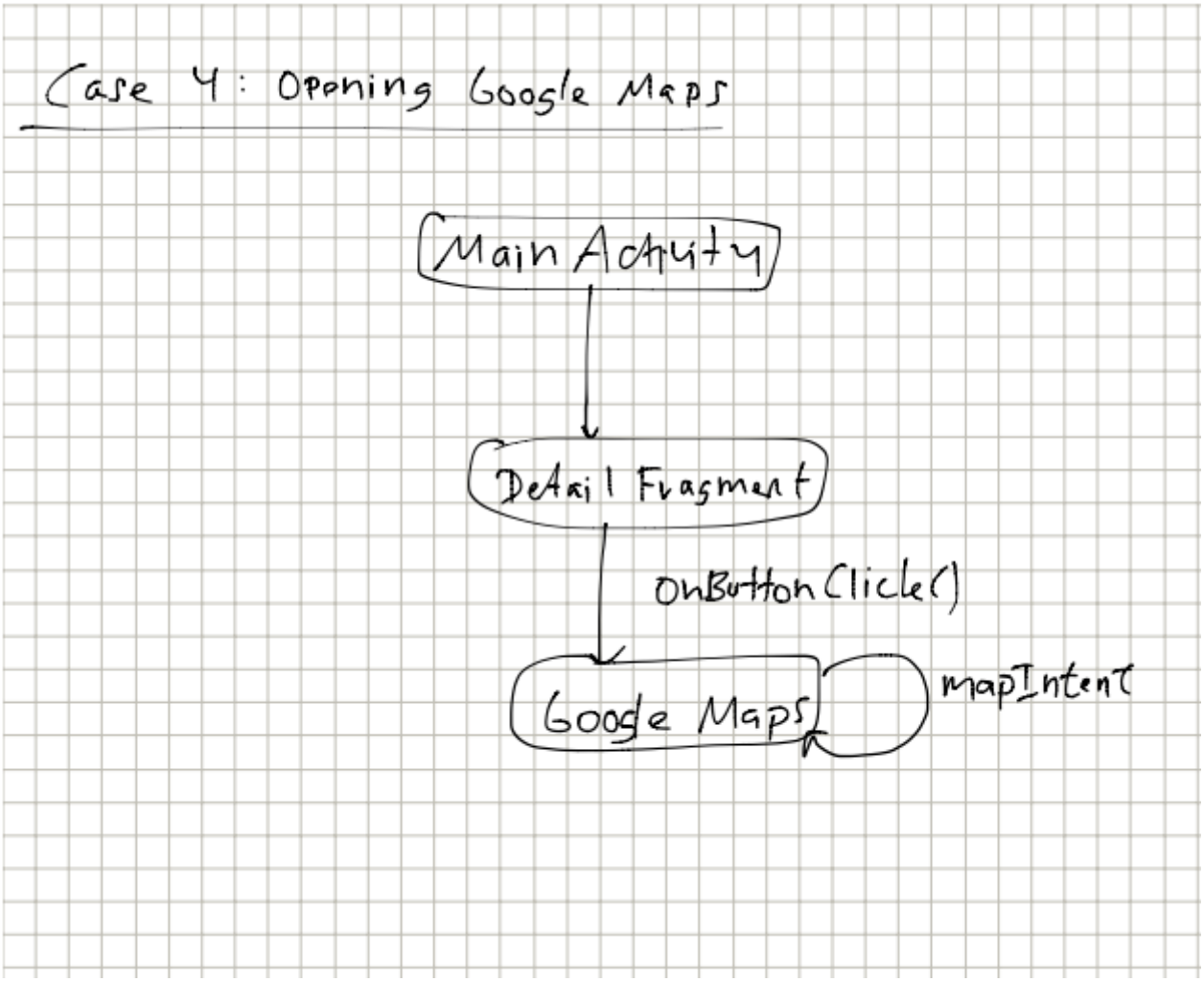
Figur 1 - Generel flow diagram



Figur 2 - Case 1 og 2: Tilføj/Ændre kort



Figur 3 - Case 3: Slette kort



Figur 4 - Case 4: Åbning af Google Maps

Work plan

MainActivity + CardDetailsFragment + CardListFragment	**Deltagere**
Denne del har vi stort set været fælles om. Samtidig er vores MainActivity samlepunkt for mange af vores ting, og vi har derfor alle skulle rette i denne på et tidspunkt. Vi har samtidig taget udgangspunkt i Kaspers udleverede demo – ArnieMoviesApp. Da der derfor også er meget der ligner det som benyttes i hans app. • Vi opfylder her brugen af Activities og fragments. • Intents til at starte andre activites. • Broadcastreceiver til at modtage intents fra Add- og EditActivity.	Alle

AddActivity	**Deltagere**
I denne activity skal der håndteres noget billedetagning, så brugeren har mulighed for at tage et billede af det nye kort. Her startes kameraet. Efter man har taget et billede startes crop, så man har mulighed for at beskære billedet. • Vi opfylder brugen af resources (kamera). • Vi bruger en singleton til at gemme data ved lifecycle skift – Vi kunne have brugt onSavedInstance, men denne blev resat ved at rotere 2 gange, og vores data ville derfor stadig ikke være gemt.	Peter Kragelund

EditActivity	**Deltagere**
Denne activity minder meget om AddActivity. Her har arbejdet dog ligget i at skulle føre data fra MainActivity over og vise dette i activitien. Der er her et problem hvis billederne (front og back) fra mainactivity er for store. De vil derfor fylde vores intent op, og vores intent vil derfor skulle bruge mere plads end den har til rådighed. Vi har her igen benyttet en singleton til at gemme data på tværs af lifecycle skift.	David Buhauer

SQLite database	**Deltagere**
Dette består af to java klasser. Den ene består i at lave databasen vha. SQLiteOpenHelper, hvor den anden stiller en tabel og nogle funktioner til rådighed, som f.eks. tilføj kort, edit kort og delete kort fra databasen. • Vi opfylder brugen af en contentprovider, med adgang til en local SQLite database.	Anders Lassen

Connecting to maps	**Deltagere**
Det er muligt at connecte til google maps via vores applikation. Hvis man f.eks. er inde på et matas kort kan man connecte til google maps og finde den nærmeste matas butik. • Vi gør her brug af external services.	Thomas Neergaard

SettingsActivity	**Deltagere**
Her har vi en klasse som gør brug af preferences, som gør det muligt at gemme og tilgå settings på tværs af activites. • Vi gør her brug af preferences.	David Buhauer

NotificationService	**Deltagere**
Vi bruger en service til at starte en alarm. Når alarmen lyder vil den starte en notifikation. Vi har ikke fået implementeret det som der skal køres en notifikation på. • Vi gør her brug af services samt notifikations.	Peter Kragelund

Konklusion

Vi har udviklet en applikation, der langt hen ad vejen opfylder de krav, vi stillede op i vores problemformulering og kravspecifikation. App'en er i stand til at gemme, vise, tilføje og ændre kort, hvilket er de væsentligste funktionaliteter, vi satte os for at implementere. I vores projektformulering har vi delt arbejdet op i fire iterationer. Alle iterationer er blevet opfyldt, undtagen den sidste omhandlende udnyttelse af telefonens NFC, da vi har valgt ikke at kigge på indlæsning af data pga. mangel på tid.

Det største problem under udviklingen lå i persisterings-delen, helt specifikt når vi forsøgte at gemme billeder i databasen. Her stødte vi ind i problemer me at billederne kunne være for store (tog for meget plads) og at lifecycle shifts nulstillede billeddataen, så de ikke blev gemt. Dette løste vi ved at bruge en singleton.

Reference liste

- http://www.vogella.com/tutorials/AndroidSQLite/article.html
- http://code.tutsplus.com/tutorials/capture-and-crop-an-image-with-the-device-camera--mobile-11458
- http://developer.android.com/guide/topics/ui/notifiers/notifications.html
- http://stackoverflow.com/questions/4989182/converting-java-bitmap-to-byte-array
- http://developer.android.com/index.html