Rapport Fadderukeappen

INF219 VÅR 2013

av

Johan Rusvik og Marianne Grov

Innholdsfortegnelse

Innledning

Fadderuken involverer flere og flere personer. I 2012 var mellom 800 og 1000 personer engasjert i fadderuken ved MatNat. Med så mange personer involvert kreves mye organisering og logistikk. Til nå har dette skjedd muntlig, via sms og skriftlig på papir, men det er behov for å gjøre denne informasjonen lettere tilgjengelig. I vårt INF219-prosjekt ønsker vi å løse dette problemet ved å gjøre informasjonen tilgjengelig elektronisk.

Idé

Idéen er å utvikle et system som har som hensikt å gjøre informasjon lett tilgjengelig for alle involverte i fadderuken ved MatNat. Måten vi tenker å gjøre dette på, er ved å utvikle en mobilapplikasjon som inneholder relevant informasjon om fadderuken og dens begivenheter, og presenterer dataene på en brukervennlig måte.

Mål

Formålet med applikasjonen er å gjøre organiseringen av fadderuken enklere. Ved bruk av Fadderukeappen vil fadderne på en enklere måte kunne formidle informasjon til studentene. Studentene vil ha tilgang til oppdatert informasjon i applikasjonen, noe som vil gjøre det lettere å holde oversikten over hva man skal delta på, samt lettere å oppdatere seg på eventuelle endringer som kan ha skjedd. Dagens løsning er avhengig av at studentene holder orden på alt av informasjon de får utdelt på papir, og erfaring og observasjoner under tidligere fadderuker tilsier at ikke alle studentene klarer dette. I tillegg oppstår det lett problemer dersom det skulle skje endringer i programmet, da det er vanskelig å gjøre alle studentene oppmerksomme på disse i løpet av kort tid. Målet er å innføre Fadderukeappen slik at disse problemene forsvinner. Fadderukeappen vil gjøre at flere får med seg det som foregår, og dette vil resultere i høyere oppmøte på både obligatoriske arrangementer og arrangementer i regi av Fadderstyret.

Om prosjektet

Vi ønsker først og fremst å utvikle appen for Android OS fordi det er enkelt å komme i gang med Android SDK, og fordi dette er noe vi begge har sterke ønsker om å lære. I andre omgang ønsker vi også å se på iOS, da dette er et operativsystem som også er veldig utbredt hos målgruppen for appen.

Gjennom utviklingen av appen ønsker vi å lære oss følgende:

- **utvikling i Android SDK**
- utvikling i iOS SDK
- nettverksprogrammering - utvikling av protokoller for sending og mottak av data på OS-baserte mobiltelefoner, på tvers av plattformene
- web-basert app-utvikling
- **bli flinkere til å bygge opp en kodebase fra starten**
- **se nærmere på parprogrammering og bruke det i utviklingen av appen**
- **få mer kunnskap om teknologier som git, maven, m.fl.**
- **utvikling av databaseskjemaer og -teknologier som passer til mobilapplikasjoner**
- se nærmere på testdrevet utvikling og bruke det i praksis
- få erfaring i å kommunisere med eksterne parter for å innhente nødvendig informasjon til applikasjonen

Omfang

Ettersom INF219 kun dekker 10 studiepoeng, er det begrenset hvor mye tid vi får brukt på dette prosjektet. Ingen av oss har kjennskap til utvikling i Android SDK fra før av, og en del tid vil dermed gå med til å sette seg inn det mest grunnleggende som man må vite for å komme i gang. Av den grunn vil vi ikke rekke å ferdigutvikle applikasjonen, men vil vektlegge å få implementert de grunnleggende funksjonene som applikasjonen bør inneholde. Av samme grunn vil vi ikke rekke å lære alt på listen ovenfor, selv om dette er hva vi i utgangspunktet ønsker å komme gjennom. De punktene vi har fått gjennomført gjennom prosjektet er markert med fet skrift.

Hvorfor valgte vi dette prosjektet?

Ettersom vi begge har deltatt i fadderuken, og Johan i tillegg har vært med i Fadderstyret, vet vi endel om hvordan organiseringen skjer. Vi mener dagens løsning er tungvint, og vi valgte dette prosjektet fordi vi ønsker å bidra til at de neste fadderukene skal være mindre arbeidskrevende å gjennomføre for de involverte partene.

Arbeidsfordeling

Gjennom utviklingen av appen har vi variert mellom å jobbe individuelt og å jobbe sammen. Stort sett har vi jobbet på forskjellig funksjonalitet, ut fra hva vi selv ønsket å jobbe med. Marianne har jobbet mest med brukergrensesnitt og innhenting av ekstern data, mens Johan har stått for mesteparten av mellomlager-funksjonaliteten og testingen.

Refaktorering, logiske klasser, dokumentasjon av kode og rapportskriving har vi begge jobbet mye med.

Hoveddel

Informasjonskilder

Dataene og informasjonen som publiseres i applikasjonen kan grovt deles inn i to kategorier. Det er data fra Fakultetet, og det er data fra Fadderstyret. Data fra Fakultetet innebærer alt som har med mottaket av studentene å gjøre, og oversikt over fagene som har oppstart i løpet av fadderuken. Det vil si timeplan for denne uken. Data fra Fadderstyret omfatter det sosiale som skjer på ettermiddags- og kveldstid, etter det obligatoriske som har med Fakultetet og gjøre er ferdig for dagen.

Åpne data er et konsept der et antall datasett blir gjort offentlig for alles øyne. IT-avdelingen ved Universitetet i Bergen holder for øyeblikket på med dette for en del datasett som har med Universitetet å gjøre [1]. Et av datasettene som allerede ligger ute inneholder oversikt over alle emnene ved Universitetet. Dette brukes til å fylle ut timeplanen til hver student på UiB sin egen portal MiSide [2]. Av denne enkle grunn vet vi at dette er gode og troverdige data som er relevante for applikasjonen vi ønsker å utvikle. I tillegg er det en stor fordel at dette datasettet kan lastes ned i både XML og JSON format, og det er derfor svært lettvinte å ta i bruk. Ulempen med å bruke dette datasettet er at den første uken i høstsemesteret (fadderuken) ikke følger oppsettet til resten av ukene i semesteret. Denne uken er anderledes på den måten at ikke alle emnene og gruppeøvelsene har oppstart denne uken. I tillegg er det aktiviteter fra Fakultetets side som er unike for fadderuken.

På grunn av disse uregelmessighetene som fadderuken omfatter har vi konkludert med at vi trenger en egenkomponert timeplan fra Fakultetet som dekker denne første uken. Denne timeplanen behøver ikke nødvendigvis ta for seg hver enkelt student sitt opplegg ettersom første semester er såpass standard, men må ta for seg hvert enkelt studieprogram og eventuelt klasseoppsett. For eksempel vil hvert enkelt studieprogram eller hver enkelt klasse ha ulike oppmøtesteder i løpet av uken.

Fadderstyret styrer også en stor del av tiden i fadderuken. Arrangementene i regi av dette organet går på ettermiddags- og kveldstid, og tar for seg det sosiale som fadderuken skal by på. Disse arrangementene er ikke så dynamiske som de Fakultetet står for, det vil si at tid og sted ikke varierer i forhold til hvilken klasse eller studieprogram studenten er en del av. Det finnes ikke noe elektronisk system der disse blir lagret ettersom det ikke er nødvendig da dette kun er arrangementer som går over en uke. Til nå har praksis vært at

planen over disse arrangementene blir satt opp som en ukeskalender, skrevet ut i mange eksemplarer, og delt ut til alle deltagerne i fadderuken. Det har også vært vanlig at en del av fellesarrangementene i regi av Fakultetet blir inkludert i denne kalenderen.

Funksjonalitet

Prioriteringer

Før vi startet kodingen, satte vi opp en prioritert liste med funksjonelle mål for Fadderukeappen. Måten vi valgte å utvikle appen på, var å velge ut noen av målene med høyest prioritet og implementere og teste disse først, for deretter å plukke ut nye mål å implementere. På denne måten får vi på plass den viktigste og mest grunnleggende funksjonaliteten først, for deretter å legge til mer funksjonalitet etterhvert [3].

Formålet med applikasjonen er å gi informasjon om begivenhetene i fadderuken, og på bakgrunn av dette valgte vi å først fokusere på å utvikle en kalenderfunksjonalitet som inneholder oppdatert informasjon om begivenhetene.

Implementert funksjonalitet

Ukesvisningen er hovedmenyen i applikasjonen. Når brukeren starter applikasjonen, kommer den til ukesvisningen. Denne gir en oversikt over dagene som fadderuken består av. Her har brukeren muligheten til å velge en av dagene og bli sendt til dagsvisningen for denne dagen.

Dagsvisningen inneholder en oversikt over arrangementene for dagen den representerer. For hvert arrangement er det oppgitt navn, tid, varighet og sted. Ved å “swipe” over skjermen vil man få frem dagsvisning for dagen før (swipe fra venstre til høyre) og dagen etter (swipe fra venstre til høyre). Det er også mulig å gå tilbake til ukesvisningen og velge en annen dag derfra.

Nedlasting/Oppdatering av data

For at brukeren av applikasjonen skal kunne ha tilgang til oppdatert informasjon, valgte vi å implementere funksjonalitet for å laste inn data fra en ekstern kilde. Arrangementene tilknyttet fadderuken blir hentet fra en eller flere XML-filer som inneholder oppdatert informasjon om arrangementene. Denne nedlastingen vil kun skje dersom brukeren har nettverkstilgang, i andre tilfeller vil applikasjonen vise informasjonen fra forrige nedlasting.

Mellomlager på mobil

For at brukeren skal ha muligheten til å se informasjon om arrangementene uten internettilgang, valgte vi å ha et mellomlager på mobiltelefonen. Når brukeren har

internettilgang vil oppdatert informasjon bli lastet ned og vist, og samtidig vil denne informasjonen bli lagret i mellomlageret på telefonen. Ved en senere anledning, når brukeren starter applikasjonen uten internettilgang, eller dersom det skulle oppstå problemer i nedlastingen, vil informasjonen lagret i mellomlageret på telefonen vises i stedet.

Fremtidig funksjonalitet

Overordnet meny
For å legge til flere funksjonaliteter, kan det være lurt å lage en overordnet meny som igjen leder til de forskjellige funksjonene i applikasjonen. Denne menyen bør da være det første man får opp når applikasjonen starter.

Oppdateringsmuligheter
Enkelte brukere ønsker ikke unødvendig datatrafikk eller har deaktivert internettilkoblingen av ulike grunner. For at brukeren selv skal kunne bestemme når applikasjonen skal sende og motta datatrafikk hadde det vært ønskelig at brukeren selv fikk velge når informasjonen skal lastes ned, og når den skal hentes fra mellomlageret. Dette kan gjøres ved at brukeren får velge om han ønsker nedlasting av informasjon ved oppstart av appen, eller om han kun ønsker å oppdatere informasjonen når han selv vil, gjennom å trykke på en knapp i applikasjonen eller i en meny.

Kontaktinformasjon (UiB)
Nye studenter har ofte ting de lurer på, og de har dermed behov for å kontakte administrative personer ved UiB. Som ny student er det ikke så lett å vite hvilke personer man skal kontakte, eller hvor man skal henvende seg med ulike spørsmål. En side med kontaktinformasjon i applikasjonen ville dermed gjort det lettere for studentene å finne ut hvor de kan få svar på det de lurer på.

Registreringsinformasjon
Første uken er det en del registeringer som må utføres, og dette kan bli mye å holde styr på for en ny student. Ved å legge til registeringsinformasjon som omfatter hvilke registeringer som må utføres og hvordan man går fram steg for steg for å gjøre dette, i applikasjonen, vil gjøre det lettere for studentene å få oversikt også på dette området.

Registering av faddergrupper (Fadderstyret)
For at fadderbarna skal kunne få informasjon om sin egen faddergruppe i applikasjonen, og for å gjøre det enklere for Fadderstyret å sende ut informasjon, burde det legges til

funksjonalitet for at Fadderstyret skal kunne registrere faddergrupper i applikasjonen. Informasjonen som bør registreres er informasjon om fadderne og fadderbarna i faddergruppen.

Poengregistering (Fadderstyret)
Fadderstyret har som oppgave å gi poeng til de forskjellige faddergruppene og i slutten av uken kåre en vinner blant faddergruppene. Ved å legge til et poengsystem i applikasjonen vil det bli lettere å holde styr på poengsummene til de forskjellige faddergruppene. Det bør tilknyttes en poengsum til hver faddergruppe, og medlemmene av Fadderstyret bør ha tilgang til å legge til poeng til hver av gruppene.

Opprettelse av arrangementer (Fadderstyret)
For å forenkle oppretting av arrangementer ytterligere, kan man legge til funksjonalitet for at medlemmene av Fadderstyret skal kunne opprette og endre arrangementer i applikasjonen. Arrangementene som blir lagt til blir overført til og lagret der applikasjonen laster ned informasjon fra. Dette vil da medføre at fadderbarna får tilgang til de nye opplysningene umiddelbart.

Kontaktinformasjon (medlemmer i faddergruppene)
Tilgang til kontaktinformasjon om de andre medlemmene av faddergruppen kan være en nyttig funksjonalitet. Dersom noen skulle komme bort fra gruppen vil de da ha tilstrekkelig informasjon for å kunne kontakte andre i faddergruppen og bli gjenforent.

Push-notifikasjoner
For at brukerne av appen skal få med seg at det har skjedd endringer uten å måtte sjekke selv i applikasjonen, kan det være lurt å benytte push-notifikasjoner. Push-notifikasjoner gjør det mulig for applikasjonen å varsle brukeren om nye meldinger eller arrangementer når brukeren ikke har applikasjonen åpen. Når mobiltelefonen mottar push-notifikasjon, vil et ikon og en melding vises på statuslinjen. Ved å trykke på notifikasjonen blir man sendt til applikasjonen. [4]

I forbindelse med dette bør det legges til mulighet for at brukeren kan skru av push-notifikasjon funksjonaliteten, og mulighet for å selv velge intervall for hvor ofte det skal bli sendt push-notifikasjoner. Denne funksjonaliteten kan implementeres ved bruk av Google Cloud Messaging. Dersom meldingen som har blitt sendt er liten (mindre enn 4kb), kan den være inneholdt i push-notifikasjonen, ellers er push-notifikasjonen et varsel om oppdatert informasjon som man finner i applikasjonen.[5]

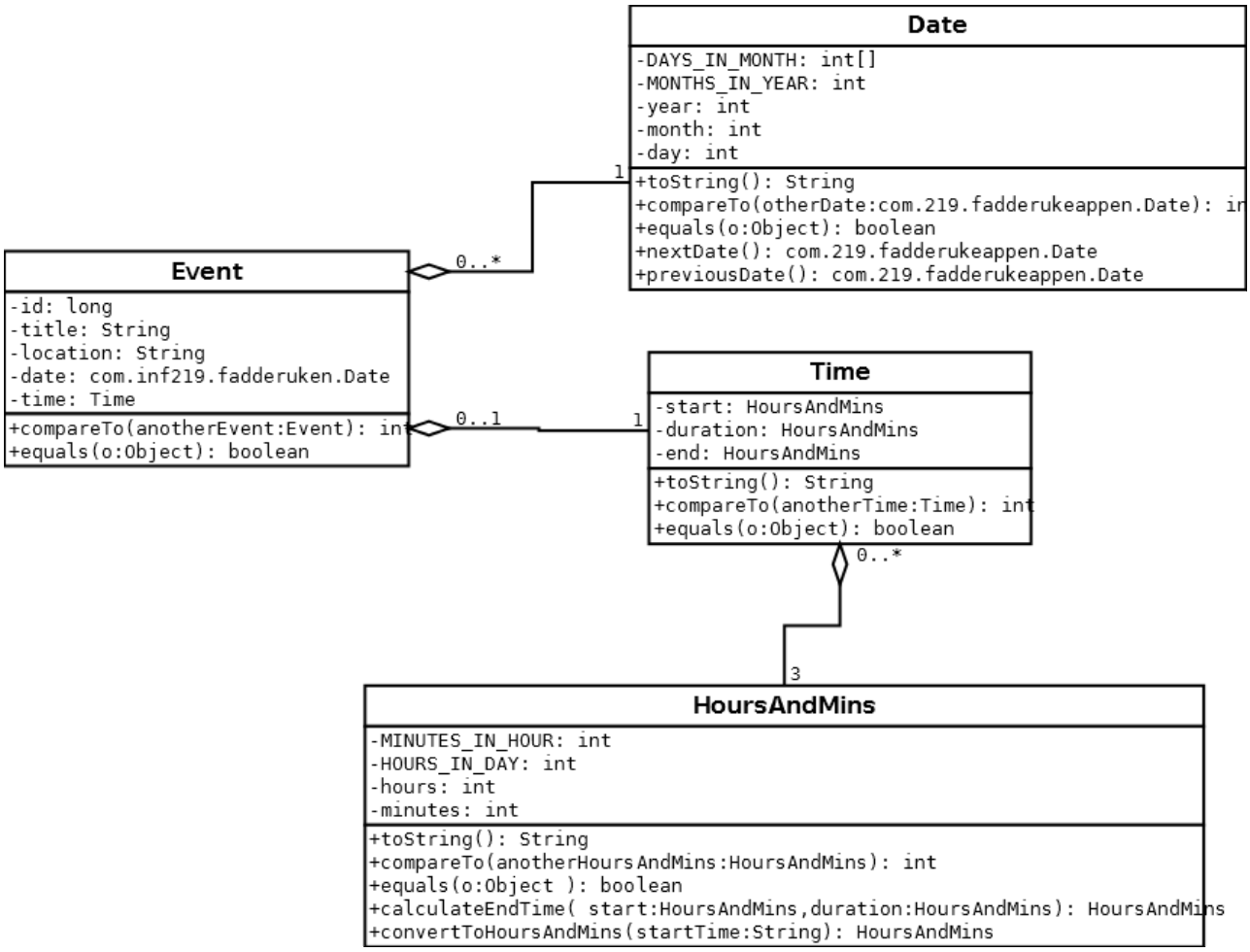
Meldingssystem

For å forenkle organiseringen av fadderuken ytterligere, ville et meldingssystem for kommunikasjon mellom de involverte i fadderuken vært nyttig. Medlemmer av Fadderstyret burde ha mulighet til å sende ut fellesmeldinger til alle, eller et utvalg av faddergruppene. I tillegg ville muligheten for kommunikasjon mellom faddere og fadderbarn innenfor en faddergruppe vært ønskelig. På denne måten vil fadderbarna på en lettere måte kunne ta kontakt med fadderne dersom de skulle ha spørsmål eller dersom noe skulle være uklart. I tillegg til at et meldingssystem integrert i applikasjonen ville forenklet kommunikasjonen og organiseringen, ville det også redusert kommunikasjonskostnadene som påløper i fadderuken. Frem til nå har mye av kommunikasjonen foregått over SMS. Ved å erstatte en stor del av SMSene med meldinger sendt over internett, ville disse kostnadene bli redusert betraktelig.

Man kan selvfølgelig ikke gå ut fra at alle studenter er smarttelefonbrukere, eller at alle har datatrafikk aktivert. På grunnlag av dette bør det sendes ut SMS til de som ikke har smarttelefon, og til de som ikke leser meldingen innen et gitt tidsrom.

Tekniske løsninger/design

Entiteter



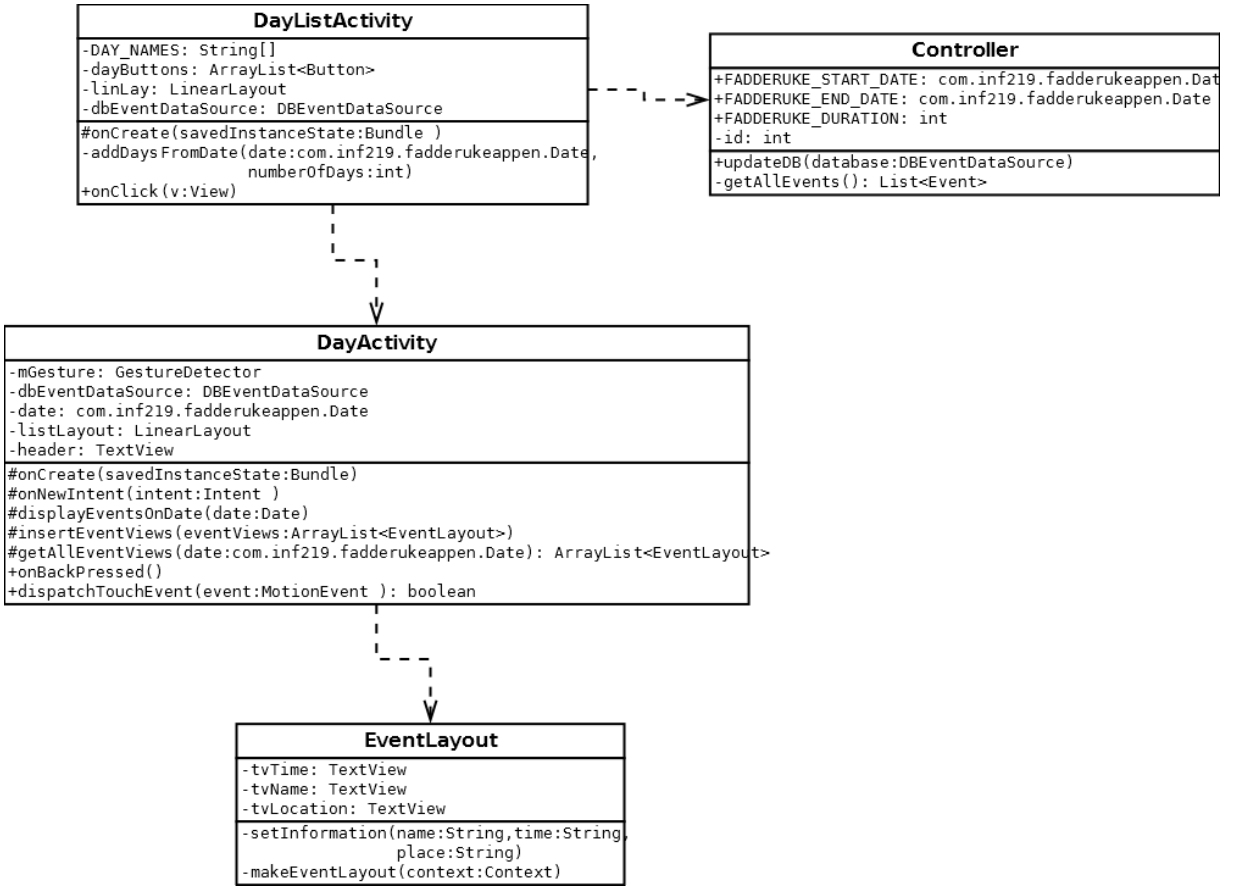
Dette klassediagrammet viser en oversikt over entitethierarkiet vårt. Det er verdt å legge merke til at vi implementerte vår egen versjon for en spesifikk dato, *Date,* som er tilpasset fadderuken. Det vil si at vi valgte å ikke bruke klassen *java.util.Date* [6]. Grunnen til dette er at vi så vi hadde behov for metoder som finner henholdsvis neste og forrige dato i forhold til en gitt dato. Når det i tillegg var såpass fort gjort å implementere klassen er dette et valg vi er fornøyd med.

Klassen *HoursAndMins* holder rede på et tidspunkt på dagen i form av antall timer og minutter. Det var først litt ut i prosjektet at vi fant ut at dette var en nyttig klasse og ha for å kunne utføre operasjoner eller beregninger på et tidspunkt. Klassen *Time* holder rede på starttidspunkt og sluttidspunkt, samt varigheten til et *Event.* Dette gjør den ved hjelp av tre instanser av *HoursAndMins*. Ettersom denne kun holder rede på to heltall (antall timer og minutter) fungerer den godt til å holde rede på varighet i tillegg til tidspunkter.

Data

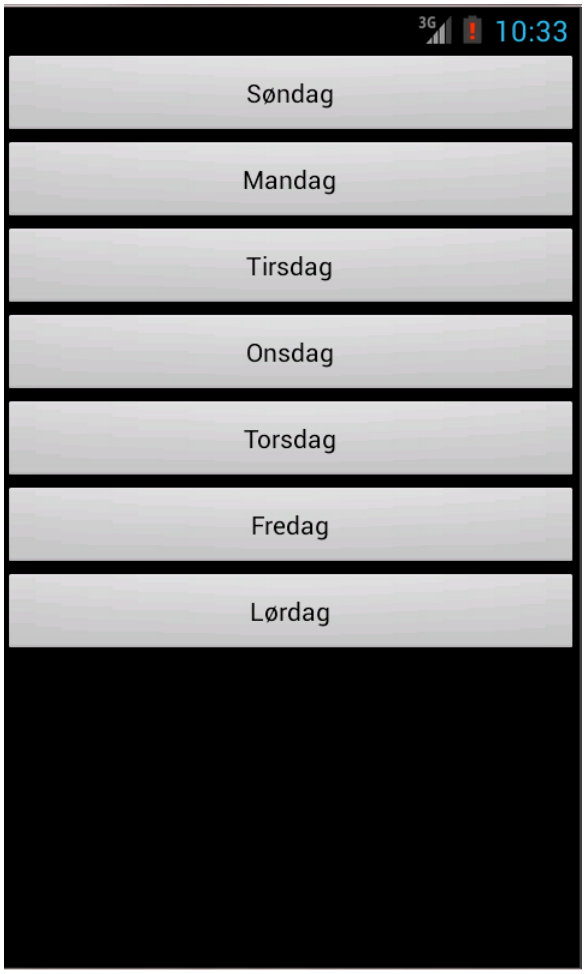
Dette klassediagrammet viser en oversikt over hvordan applikasjonen mottar og lagrer data på enheten. Når metoden *updateDB()* i *Controller* blir kalt, kaller denne igjen på *getAllEvents()* i samme klasse. Denne bruker *DataThread* klassen til å starte en separat tråd som laster ned XML dokumenter fra en gitt URL ved hjelp av *XMLParser*. En liste med *Events* blir returnert tilbake til *Controller* klassen. Deretter blir *eventene* satt inn i databasen ved hjelp av *DBEventDataSource* og *DBEventHelper.*

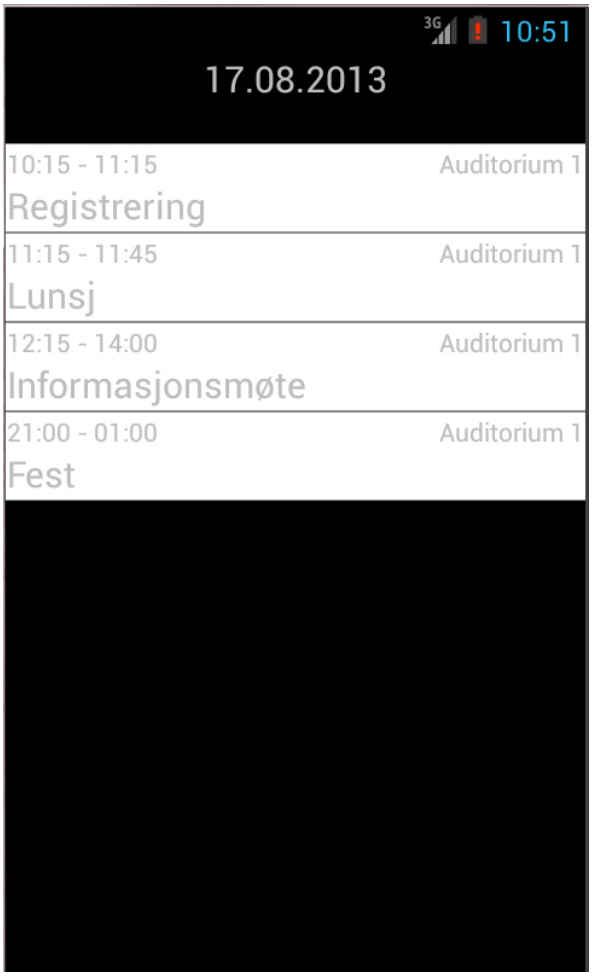
GUI



Dette klassediagrammet viser en oversikt over det grafiske brukergrensesnittet til applikasjonen. Når applikasjonen starter begynner *DayListActivity* å kjøre. Denne viser en oversikt over dagene i fadderuken. I tillegg kaller denne på metoden *updateDB* i *Controller.* Hva denne gjør ble beskrevet på forrige side. Når brukeren velger en spesifikk dag blir *DayActivity* kalt. Denne visualiserer alle *eventer* for den dagen ved hjelp av klassen *EventLayout.*

Skjermskisser





Den første skjermskissen er det første skjermbildet brukeren får se når han starter applikasjonen. Dette er ukesvisningen, hvor brukeren får opp de forskjellige dagene i fadderuken og kan velge å trykke på en av dem for å komme til den valgte dagen. Ved å trykke på tilbake-tasten på enheten vil applikasjonen bli avsluttet.

Den andre skjermskissen viser oversikten over en bestemt dag, nemlig dagsvisningen. Her vises en liste med de aktuelle begivenhetene som forekommer på denne dagen. Dersom det finnes flere eventer enn det er plass til å vise på skjermen, er det mulig å "scrolle" seg gjennom dem. For å gå til en annen dato kan brukeren benytte seg av to muligheter. Det er mulig å gå tilbake til oversikten over alle dagene, ved å benytte tilbake-tasten på enheten, og velge en ny dag der, eller man kan gå direkte fra en dag til den forrige eller neste. Dette blir gjort ved at man drar fingeren fra høyre til venstre for å komme til neste dag, eller venstre til høyre for forrige dag.

Veivalg

API (target/minimum)

En av de første tingene vi måtte gjøre når Android prosjektet ble opprettet var å velge Laveste Påkrevet SDK og Mål SDK [7]. Laveste Påkrevet SDK definerer den laveste mulige Android-versjonen som applikasjonen skal være kompatibel mot. Mål SDK definerer den høyeste Android-versjonen som applikasjonen er testet mot. Laveste påkrevet SDK er desidert viktigst av disse to. Her er det viktig å velge riktig slik at man får tak i all den funksjonalitet man trenger fra Android-bibliotekene, samtidig som man tar hensyn til at svært mange brukere fortsatt bruker eldre versjoner av Android. Vi tok mest hensyn til å gjøre applikasjonen så tilgjengelig som mulig, og valgte Android SDK 9 som Laveste Påkrevet SDK [8]. Mål SDK har vi satt til 17 ettersom dette er den nyeste versjonen. Andelen brukere som har API-level fra og med 9 er 94,5%, og vi vil da gjøre applikasjonen tilgjengelig for de aller fleste aktuelle brukere ved å velge 9 som Laveste Påkrevet SDK [9].

Datalagring

Android SDK tilbyr flere datalagringsalternativer [10]. Strukturen vår er enkel og ikke altfor kompleks, og vi valgte derfor å bruke Android sin støtte for SQLite. Ettersom Android sin støtte for SQLite er komplett var det ikke nødvendig å legge til eksterne biblioteker for dette i prosjektet. Oppsettet for SQLite i Android er også meget intuitivt og enkelt å bruke. Andre alternativer for datalagring innebar rene tekstfiler i enhetens minne eller totalt webbasert lagring. Vi forkastet disse alternativene henholdsvis fordi dataene applikasjonen krever er godt strukturerte og vi ønsker at applikasjonen skal fungere uten tilgang til Internet.

Hvor ofte oppdatere lokal database

Det finnes i hovedsak to måter å flytte oppdatert data til brukerens enhet på. Den ene innebærer å bruke en ekstern server som pusher data til enheten når en administrator velger det, eller applikasjonen gir et signal om at den lokale databasen på enheten skal oppdateres. Den andre er å kun benytte seg av Google sin klient for å oppdatere Android applikasjoner, *Google Play* [11]. Da slipper man å ta i bruk en ekstern server og vedlikehold av den. Med den siste måten legger administrator ut en oppdatering av applikasjonen, med oppdatert database, på Google Play, og brukere av applikasjonen kan da laste ned denne når de ønsker det. Ulempen med Google Play er at brukere kan velge å ikke oppdatere, eller glemme å gjøre det. Ettersom hensikten med vår applikasjon er såpass tidsbestemt, er det viktig at brukeren får oppdateringer i databasen ned på

enheten så fort det lar seg gjøre. Med en ekstern server som så og si tvangspusher oppdatert data til enheten er det sannsynligvis lettere for en bruker å akseptere oppdateringen. Derfor har vi kommet frem til at en ekstern server er den beste løsningen for vår applikasjon.

Målet med applikasjonen er å ha all nødvendig data på telefonen før fadderuken begynner. Dette er hensiktsmessig ettersom deltagerne i fadderuken flytter mye rundt på seg i løpet av uken, og derfor ikke alltid er koblet til Internet. Det er likevel viktig at applikasjonen støtter oppdateringer i databasen i løpet av uken hvis det skjer viktige endringer i ukeprogrammet. Vi kom frem til at det enkleste og mest effektive er å oppdatere databasen når applikasjonen starter opp. Dette krever selvfølgelig at enheten er koblet til Internet. Ettersom datamengden er såpass liten har vi valgt å tømme hele den lokale databasen og laste inn hele på nytt, med endringene inkludert. Ulempen med dette er at oppstart kan ta unaturlig lang tid i brukerens øyne og oppfattes som en krasj. Det er derfor viktig å ha en god løsning for å fortelle brukeren at applikasjonen oppdaterer. Dette kan man gjøre ved å ha en informativ “loading screen”. Vi har ikke rukket å implementere dette, men vi har lagt til rette for at det ikke skal være vanskelig å legge til denne funksjonaliteten.

Hvorfor xml

Vi valgte å bruke XML som standard i første omgang av den enkle grunn at det var det formatet vi begge kjente best til. Dette innebærer at enheten mottar data i XML-format som blir reformatert til strukturert data som blir lagret i databasen. Underveis har vi også vurdert å endre standardformatet vårt til JSON [12] eller CSV [13], men så lenge vi har en sterk og sikker reformateringskomponent for XML, utgjør det ikke noe forskjell.

Integrere med android-kalender?

Vi vurderte å gjøre det mulig for brukerene av applikasjonen å få lagret begivenhetene i android-kalenderen på enheten. For de som benytter android-kalenderen for å holde styr på avtalene sine ville dette gjort det enklere for dem å holde oversikten, og å få samlet alle avtalene sine på samme sted.

Å implementere denne funksjonaliteten ble vanskeligere enn vi først hadde tenkt, og vi valgte derfor å ikke prioritere denne funksjonaliteten. Grunnen til dette er at Androids kalender-API ikke eksisterte før API-level 14. [14] Ettersom applikasjonen vår har minimum-SDK satt til 9, vil dette by på problemer da man trenger en annen måte å få tilgang til kalenderen på for enheter med lavere API-level enn 14. Et annet alternativ ville vært å droppet funksjonaliteten for API-level lavere enn 14.

Testing

Unit testing

Komplett testdekning av logikken i prosjektet er meget viktig. Dette er en svært enkel måte for å finne ut om alt fungerer som det skal, spesielt for “edge-cases”. Det vil si tilstander som oppstår svært sjeldent, men som likevel må kunne håndteres. Målet med testing av koden er å oppnå komplett testdekning. Testing av enklere komponenter som ikke har så mye logikk i seg er ikke like viktig å teste, men vi ønsker likevel å ha testdekning for så mye av prosjektet som mulig.

For å klare å teste mye av logikken i koden, spesielt det som har med lesing og skriving til lokal database på enheten å gjøre, samt mottak av data fra eksternt hold, krever at strukturen og komponentene til prosjektet er satt opp på en spesiell måte. Logikken må skilles fra resten i egne metoder, på best mulig måte. Dette gjør logikken enkel å få tak i, og dermed enkel å teste. Klarer man å oppnå dette, slipper man å ta i bruk verktøy for å simulere eksterne komponenter, såkalte mock-rammeverk [15]. For prosjektet vårt sin del måtte det blitt gjort en god del refaktorering for gjøre det mulig å teste uten mock-rammeverk, blant annet ved å dele opp store metoder i mindre komponenter, samt endre deler av strukturen, og teste hver av delene for seg.

Kommunikasjon mellom lokal database og server

Når det gjelder tester for å sjekke hvordan applikasjonen håndterer mottak av data fra eksternt hold, samt lesing fra og skriving til lokal database, har vi valgt å ikke legge særlig vekt på dette. Denne typen testing krever at man simulerer disse eksterne enhetene ved hjelp av et mock-rammeverk, noe vi vi ikke hadde særlig kjennskap til fra før, og heller ikke hadde tid til å sette oss tilstrekkelig inn i. Dette får man i tillegg testet meget grundig når man kjører applikasjonen med reelle data på en enhet, og forsøker å oppdatere den lokale databasen. Hvis applikasjonen fungerer som den skal og den viser alt av data som er forventet, og ikke mer enn det, kan man til en viss grad gå ut i fra at kommunikasjonen mellom de forskjellige komponentene fungerer som den skal.

Tekniske utfordringer

Hvis enhet går tom for strøm

At enheten applikasjonen kjører på går tom for strøm, vil i de fleste tilfeller ikke være et problem. Applikasjonen bruker data fra en lokal database som kjører på enheten, og denne blir ikke borte selv om enheten plutselig skrur seg av. Et problem oppstår hvis

enheten går tom for strøm samtidig som applikasjonen oppdaterer databasen. Da har applikasjonen allerede lastet ned en del oppdatert data og tømt den lokale databasen. Databasen blir tømt i forkant for å ikke ende opp med duplikater. Det er to måter man kan håndtere dette problemet på.

Applikasjonen kan opprette en midlertidig kopi av den lokale databasen før den blir tømt. Går oppdateringen knirkefritt slettes denne kopien når oppdateringen er ferdig. Hvis oppdateringen ikke går som den skal, blir den ufullstendige databasen tømt og kopien lastet inn igjen.

En annen mulighet er at applikasjonen oppretter et mellomlager på enheten før dataene blir lagret i databasen. Nedlastingen av oppdatert data blir først lastet ned til dette mellomlageret, og applikasjonen gir et signal hvis oppdateringen går uten problemer. Først når dette signalet kommer tømmer applikasjonen databasen, og legger inn de nye dataene.

Hvis ikke internettilgang

Det er ikke gitt at enheten applikasjonen kjører på alltid er koblet til Internet. Ettersom vår applikasjon har en lokal database er den ikke direkte avhengig av Internet for å kjøre. Det er selvfølgelig anbefalt å oppdatere applikasjonen så ofte som mulig for å få med seg viktige endringer.

For å sikre at brukeren får med seg endringer i fadderukeprogrammet, kan man se på mulighetene for å sende melding om endringer via SMS. Eventuelt kan SMSen inneholde en beskjed om at brukeren bør oppdatere applikasjonen så fort som mulig. Ulempen med å benytte SMS for varsling om endringer i applikasjonen er at det er kostbart, men ettersom fadderuken går over så kort tid, og disse endringene kan være av stor betydning, kan man se på mulighetene for å få finansiert en slik tjeneste av MatNat-Fakultetet.

Bruk av tråder

Innhenting av data fra internett er en tidkrevende prosess, og dette kan gjøre applikasjonen treig. For applikasjoner med API-level høyere enn 11 (Honeycomb SDK), blir det kastet et unntak ved navn *NetworkOnMainThreadException* for å forhindre nettverksoperasjoner på hovedtråden. For applikasjoner med lavere SDK, er det tillatt å ha nettverksoperasjoner på hovedtråden, men det er sterkt frarådet.[16]

Nedlastingen av ekstern data må altså foregå på en annen tråd enn hovedtråden. Måten vi valgte å løse dette problemet på, var ved bruk av AsyncTask. AsyncTask gir mulighet for å

kjøre operasjoner i bakgrunnen, og vi lot alle nettverksoperasjonene foregå i AsyncTask. [17]

Testing av enkelte Android biblioteker

I starten av prosjektet testet vi logikk på samme måte som vi har testet vanlige javaprosjekter før. Det vil si unit testing og mocking. Unit testing gikk greit, men å mocke server og database viste seg å være vanvittig vanskelig. Etterhvert fant vi ut at Android SDK tilbyr et fullstendig rammeverk for JUnit testing [19], som ville løst problemet med testing som involverer enkelte Android biblioteker. Dette er selvfølgelig noe vi burde ha sjekket opp helt i starten av prosjektet. På grunn av problemene vi støtte på i forbindelse med mockingen har vi dessverre ikke fått tid til å teste så mye som vi skulle ønske. Hvis dette prosjektet blir bygget videre på senere anbefaler vi sterkt å sette seg inn i dette rammeverket og ta det i bruk fra dag én.

Brukertesting

I planleggingsfasen av applikasjonen var brukertesting en vital del av prosjektet. Vi ønsket å få applikasjonen såpass komplett slik at vi kunne teste den hos en liten del av deltagerne i fadderuken høsten 2013. Dette målet klarte vi dessverre ikke å nå. Det vil si at applikasjonen verken er god nok funksjonelt sett eller har god nok testdekning. Derfor har vi konkludert med at det å teste applikasjonen denne fadderuken ikke har noen nytteverdi. Vi har likevel valgt å skrive litt om strategien vår for å få testet applikasjonen i forbindelse med selve fadderuken, og med vilkårlige personer.

Ettersom Johan er med i Fadderstyret 2013 og var med i samme styre 2012, har han god kommunikasjon med de som har ansvaret for oppstartsuken på MatNat-Fakultetet. Vi ville derfor hatt bedre muligheter enn de fleste til å få testet applikasjonen i fadderuken. Hvis vi skulle gjennomført en slik test ville vi kun tatt for oss en liten del av deltagerne av fadderuken, for eksempel en eller to faddergrupper. Vi måtte ha samlet de utvalgte til et informasjonsmøte der vi informerer om applikasjonen og forteller hvor man kan laste den ned. Etter fadderuken hadde vi måttet utføre en undersøkelse der vi ba om tilbakemeldinger på forskjellige aspekter ved applikasjonen. For eksempel hvor enkelt det var å ta i bruk applikasjonen, om den krasjet ofte, hva testbrukerne savnet ved applikasjone, om informasjonen applikasjonen viste var rett/nyttig osv. Etter å ha mottatt alle tilbakemeldingene hadde vi lest gjennom og tatt til oss hva som bør endres på, hva som var bra, og hva som var dårlig.

En annen brukertestingstrategi vi så for oss var å spørre tilfeldige personer om de kunne

tenke seg å teste applikasjonen. Disse personene ville mest sannsynlig være mer eller mindre kjente av enten Johan eller Marianne, men ville likevel gjort nytten. Vi ville mest sannsynlig prøvd å spørre noen som ikke studerer informatikk ettersom informatikere ikke er representative for “den vanlige mannen på gata” når det gjelder IT-teknologi. Strategien for disse testene ville for eksempel være å be testpersonene navigere seg fra et sted til et annet eller utføre en enkel operasjon inne i applikasjonen. Hver test ville ikke vart lenger enn 5-10 minutter. Av disse testene kunne vi veldig enkelt funnet ut hvordan applikasjonen ville blitt håndtert av vanlige personer; blant annet hvor intuitiv og brukervennlig den er.

Konklusjon

Læringsutbytte

Tidkrevende

I forkant av prosjektet satte vi opp en liste over hva vi ønsket å lære, og hvor langt vi ønsket å komme i utviklingen av applikasjonen. Vi ønsket blant annet å utvikle applikasjonen for Android OS, og i tillegg starte implementasjon for iOS. Etter å ha jobbet med prosjektet en stund, innså vi at vi ikke ville rekke å komme så langt som vi ønsket. Ingen av oss hadde erfaring med utvikling i Android SDK, og vi måtte derfor bruke en del tid på å sette oss inn i dette før vi fikk startet utviklingen av applikasjonen.

Lært mye

Selv om vi ikke fikk gjennomført alt vi ønsket, har vi likevel hatt et stort læringsutbytte av dette prosjektet. Gjennom planleggingen av prosjektet lærte vi mer om hvordan man bør strukturere prosjektet for å holde ulike komponenter adskilt. Hovedmålet vårt med prosjektet var å lære å utvikle applikasjoner i Android SDK, og dette er noe vi har jobbet masse med gjennom utviklingen av applikasjonen. Som nybegynnere har vi hatt endel utfordringer og problemer underveis, og selv om dette har gjort at vi ikke kom så langt som vi ønsket, har vi lært utrolig masse av dette. Vi har funnet ut av hva som fungerer og ikke fungerer ved å prøve og feile, og vi har blitt flinkere til å tenke i gjennom fordeler og ulemper ved forskjellige løsninger.

Viktigheten av testing

En viktig erfaring vi har gjort oss, er at det er lurt å tenke på testing fra starten av prosjektet. Ettersom vi ikke tenkte så veldig mye på testing under implementeringen, var vi nødt til å gjennomføre en del refaktorering for å gjøre det enklere å teste deler av kildekoden. Vi ser nå i ettertid at vi burde skrevet tester underveis, både for å være sikre på at

implementasjonen fungerer som tenkt, og for å slippe alt arbeidet med refaktorering og skriving av tester i etterkant. Hadde vi testet underveis ville vi trolig hatt en mer gjennomtestet applikasjon enn det vi har i dag, ettersom vi nå får knapt med tid til å gjøre dette grundig.

Konkludering av appens nyttighet

Ettersom vi ikke har fått brukertestet applikasjonen til den grad vi hadde som mål, blir det vanskelig å si noe om applikasjonens nytteverdi fra en vanlig bruker sin synsvinkel. Vi kan likevel argumentere for at applikasjonen vil ha en generell nytteverdi fra vår egen synsvinkel.

I dagens digitale samfunn er det nesten pinlig at informasjonen som alle de nye studentene ved Fakultetet får ikke er bedre organisert. Til nå har alt kommet på papirformat og muntlig. Det er rett og slett umulig for alle å holde orden på alt sammen. Det er ikke vanskelig å tenke seg at mange blir stresset og redde for at de ikke kommer til å få med seg eller huske på alt. Det er akkurat derfor vi ønsket å utvikle en mobil applikasjon som vil spare alle de nye studentene for dette.

I tillegg vil en slik applikasjon gjøre arbeidet enklere for de som organiserer fadderuken. Dette er primært MatNat-Fakultetet og Fadderstyret. For å sikre at alle deltagere i fadderuken mottar viktig informasjon, har det før vært nødvendig å henvende seg til mange individer. Eventuelt kan Fadderstyret ta i bruk SMS-løsningen nevnt tidligere i rapporten, men dette er kostbart. En tilnærmet gratis og lettvint løsning vil være å sende ut en melding eller notis via vår applikasjon.

Veien videre, forslag til videreutvikling (INF219)

Tidligere i rapporten ble mulige fremtidig funksjonalitet presentert. Dette er funksjonalitet som vi mener ville økt applikasjonens nytteverdi. Å legge til dette i applikasjonen krever en del tid og jobbing, samtidig som det er veldig lærerikt. Vi mener derfor at en videreutvikling av applikasjonen hadde passet veldig godt som INF219-prosjekt for de som er interessert i å gjøre gjennomføringen av fadderuken enklere.

Vi har også presentert hvordan vi ville gått frem for å brukerteste applikasjonen om vi hadde hatt tilstrekkelig med tid, og her kan det være en del tips å plukke opp om noen skulle ønske å videreutvikle applikasjonen.

Et forslag til videreutvikling av applikasjonen kan være å først og fremst gjøre ferdig den

mest grunnleggende funksjonaliteten, samt å sørge for at applikasjonen har høy testdekning. Deretter kan det være lurt å gjennomføre brukertesting som tidligere beskrevet. Etter at dette er gjort, vil man sitte med litt mer informasjon om hva en bruker ønsker av funksjonalitet, og om det er noe i brukergrensesnittet som bør være annerledes. Ved å gjennomføre brukertestingen før man legger til mer funksjonalitet, vil det være enklere å rette opp feil, og å endre på deler som ikke fungerer optimalt, uten at man blir nødt til å endre store deler av implementasjonen som eksisterer nå.

Tips og triks til videreutvikling

Dersom noen ønsker å videreutvikle applikasjonen, er det veldig viktig at vedkommende setter seg inn i den allerede eksisterende koden og får oversikt over hvilken funksjonalitet som allerede eksisterer, og hvordan den fungerer. Vi har lagt til endel kommentarer i koden for å gjøre dette lettere. Utenom dette er det viktig å tenke på strukturering av komponentene før man starter med implementering. Dette vil forhindre at man ender opp med dårlig kode, eller må starte på nytt gang på gang. Når det gjelder strukturen, er det viktig å hele tiden tenke på at koden skal testes, og at dette bør være lett å gjennomføre. Ved å teste underveis vil man slippe refaktorering og en vanskelig jobb med å teste, mot slutten av prosjektet.

Vår personlige visjon

Med denne mobil-applikasjonen ønsker vi å tilby en enkel og billig måte å organisere fadderuken på. Applikasjonen er til for både deltagere og de som står for organiseringen. Ettersom vi ønsker at så mange som mulig skal ta i bruk applikasjonen vår, er det viktig at den er gratis å laste ned. For å gjøre den attraktiv og lettvint er det også viktig å ikke fylle den med reklame. Vi har utviklet applikasjonen for samfunnet og profitt har aldri vært et mål. Vi ser derfor ingen grunn til å vurdere fortjenestemuligheter. Vi håper at våre etterfølgere har samme visjon og at applikasjonen videreutvikles på dette grunnlaget. Vi håper at den er klar til å testes i forbindelse med fadderuken 2014, og at den blir tatt godt i mot.

Link til prosjektet på GitHub

https://github.com/livarb/uibmobilapp.git

Kilder/Ressurser

[1] https://it.uib.no/%C3%85pne_data
[2] https://miside.uib.no
[3] Robert C. Martin. (2002) *Agile Software Development: Principles, Patterns, and Practices.* New Jersey, Prentice Hall.
[4] https://www.parse.com/tutorials/android-push-notifications
[5] https://developer.android.com/google/gcm/index.html
[6] http://docs.oracle.com/javase/6/docs/api/?java/util/Date.html
[7] http://developer.android.com/training/basics/firstapp/creating-project.html
[8] http://developer.android.com/about/dashboards/index.html
[9] http://developer.android.com/about/dashboards/index.html
[10] http://developer.android.com/guide/topics/data/data-storage.html#filesInternal
[11] https://play.google.com/store
[12] http://www.w3schools.com/json/
[13] http://en.wikipedia.org/wiki/Comma-separated_values
[14] http://developer.android.com/reference/android/provider/CalendarContract.html
[15] http://en.wikipedia.org/wiki/Mock_object
[16]http://developer.android.com/reference/android/os/NetworkOnMainThreadException
[17] http://developer.android.com/reference/android/os/AsyncTask.html
[18] http://www.oracle.com/technetwork/articles/javase/index-137171.html
.html
[19] http://developer.android.com/tools/testing/testing_eclipse.html