2013

Kjørehjelperen

Prosessdokumentasjon

Høgskolen i Oslo og Akershus

Henrik Hermansen og Lars Smeby
Gruppe 8
26.05.2013

Forord

Det forutsettes at leseren har lest presentasjonen av dette prosjektet før denne rapporten leses. Presentasjonen gir innblikk i viktig informasjon angående gruppen og oppdragsgiveren, samt bakgrunnen for og mål med prosjektet. Når presentasjonen er lest kan denne rapporten leses som et selvstendig dokument.

Videre forutsettes det at leseren har kompetanse innen programmering og systemutvikling, og er kjent med datatekniske begreper. Vi har valgt å legge det tekniske innholdet på et slikt nivå da dette dokumentet i hovedsak er beregnet på sensor og veileder. For ordens skyld vil vi allikevel kort forklare en god del av de ord og uttrykk som kan være fremmed for noen, ettersom man ikke kan forvente at en datakyndig person kjenner til alle datatekniske begreper. Disse forklares i fotnoter, og finnes også i "Ordforklaringer og kilder" bakerst i rapporten.

Prosessdokumentasjonen inneholder en detaljert gjennomgang av arbeidsprosessen vår gjennom hele prosjektet. Vi har her valgt å dele opp denne dokumentasjonen i fire hoveddeler:

- **Planlegging og metode** tar for seg forhold rundt planlegging og samarbeid, kommunikasjon, kompetansebygging i forhold til prosjektet, metodikk og verktøy og teknologier vi har benyttet underveis.
- **Utviklingsprosessen** beskriver prosjektets faser i detalj, og tar også for seg hver enkelt sprint under produksjonsfasen. Denne delen vil også ta for seg større utfordringer vi har hatt underveis, og hvordan vi gikk frem for å løse disse.
  I denne delen vil vi gjerne trekke frem kapittel 2.3.6, "Utfordringer underveis", som et spesielt interessant kapittel for den eller de som skal vurdere dette prosjektet.
- **Kravspesifikasjonen og dens rolle** vil omtale hvordan vi har benyttet kravspesifikasjonen og hvilken rolle den har spilt underveis i prosjektet. Her vil vi også se nærmere på oppfylte krav, avvik fra kravspesifikasjonen, samt endringer av kravspesifikasjonen underveis.
- **Avsluttende del** tar for seg refleksjoner og konklusjoner vi har rundt prosjektet. Vi vil her gå inn på nytteverdien ved produktet, muligheter for videre utvikling, samt hvilket læringsutbytte vi har hatt av dette prosjektet.

Innholdsfortegnelse

1 Planlegging og metode

I denne delen av dokumentasjonen vil vi ta for oss forhold rundt planlegging og samarbeid, kommunikasjon, kompetansebygging i forhold til prosjektet, metodikk og verktøy og teknologier vi har benyttet underveis.

1.1 Arbeidsforhold og samarbeid

Da vi skulle danne grupper til hovedprosjektet var det ikke vanskelig å vite at vi skulle jobbe sammen på dette prosjektet. Vi har under hele studiet jobbet sammen på flertallet av prosjektene vi har hatt, og vi kjenner derfor godt til hverandre fra før. Vi har gjennom de tidligere prosjektene erfart at vi oppfører oss jevnbyrdige i en gruppesammenheng, og at vi er flinke til å drøfte utfordringer og finne de beste løsningene. Ved utfordringer og uenigheter kan vi alltid argumentere vår sak ovenfor hverandre, og vi er da flinke til å kombinere alle disse argumentene slik at begge bidrar til den endelige avgjørelsen. I tillegg kjenner vi hverandres styrker og svakheter, og vet at vi utfyller hverandre godt. Basert på dette, samt at vi begge har det samme ambisjonsnivået, var valget om at vi skulle samarbeide om hovedprosjektet veldig enkelt.

Vi bor begge i Stor-Oslo-området, og det har derfor ikke vært noe problem å samarbeide på skolen eller hos hverandre, noe som har gjort det enklere å tilpasse arbeidssituasjonen og hvor vi har arbeidet fra dag til dag.

Vår oppgave var i utgangspunktet en utfordring for oss begge siden ingen av oss hadde jobbet med utvikling til iPhone[1] før. Utvikling til iPhone innebar at vi måtte lære oss et nytt programmeringsspråk, Objective-C[2], og et nytt rammeverk, Cocoa Touch[3]. Vi måtte også lære oss et nytt IDE[4], Xcode[5], og til og med et nytt operativsystem, OS X[6]. Man må nemlig bruke Mac om man skal utvikle programvare til OS X eller iOS[7]. Dette var mye å sette seg inn i, men det var også et bevisst valg av oss da vi valgte oppgave. Vi ønsket begge å utvide kunnskapsplattformen vår og å utfordre oss selv i hovedprosjektet.

Figur 1: iOS 6.1 er det nyeste operativsystemet til iPhone, og det vi måtte utvikle til.

[1] Smarttelefon fra Apple Inc.
[2] Objective-C er Apples programmeringsspråk for OS X og iOS. For detaljer se kapittel 1.5.5.
[3] Cocoa Touch er et rammeverk for å utvikle programvare for iOS. For detaljer se kapittel 1.5.6.
[4] Integrated Development Environment er programvare som tilbyr omfattende verktøy for programmering og programvareutvikling.
[5] Xcode er Apples IDE for å utvikle programvare til OS X og iOS. For detaljer se kapittel 1.5.4.
[6] OS X er det gjeldende operativsystemet til Mac.
[7] iOS er Apples operativsystem for iPhone, iPad og iPod.

1.2 Kommunikasjon

Vi vil her beskrive nærmere hvordan vi kommuniserte med interessenter og med hverandre underveis i prosjektet. Generelt har ikke kommunikasjon vært noe problem, og vi har som oftest fått tak i de personer vi har hatt behov for innen rimelig tid, slik at ikke arbeidet har lidd unødig på grunn av dette.

1.2.1 I gruppen

Innad i gruppen har kommunikasjon fungert godt under hele prosjektet. Under planleggingen av prosjektet arbeidet vi mye sammen, og vi avtalte da ofte fra gang til gang når vi skulle jobbe videre.

I produksjonsfasen var vi flinke til å fordele oppgaver, og begge hadde derfor alltid noe å gjøre, slik at vi ikke hadde behov for å kommunisere mye hver eneste dag. Under denne fasen holdt vi jevnlig kontakt via Skype[8], eller telefon dersom noe hastet og den ene ikke var pålogget Skype. Utover dette møttes vi ved behov under produksjonsfasen, men det var det ikke mye behov for, ettersom mange avgjørelser kunne tas via Skype.

Vi har også hatt en internettside til prosjektarbeidet vårt, og denne har også fungert som en form for kommunikasjon oss imellom, men har her ikke spilt noen stor rolle. Nettsiden har blitt benyttet til å føre logg for prosjektet, samt å tilgjengeliggjøre diverse dokumenter i forbindelse med planlegging av prosjektet.

1.2.2 Med oppdragsgiver og kunde

Med oppdragsgiver og kunde har all fjernkommunikasjon foregått via e-post. Vi har hatt jevnlig kommunikasjon med vår veileder hos oppdragsgiver, *Christoffer Marcussen*, samt *Frode Reinertsen* som har vært vår kontakt hos BEKK dersom vi har hatt spørsmål eller forespørsler i forhold til det tekniske i NVDB[9] API[10]. Utover dette har vi hatt noe kommunikasjon med *Jan Kristian Jensen* hos Statens vegvesen når vi har hatt spørsmål angående datasettet i NVDB. Samtlige av disse kontaktpersonene har vi hatt god kommunikasjon med per e-post, og nesten uten unntak fått raske og gode svar.

Figur 2: Vår oppdragsgiver var BEKK Consulting AS.

I tillegg har vi jevnlig hatt møter med Christoffer underveis i både planlegging og under produksjon. Møtene har som oftest funnet sted i BEKKs lokaler på Vippetangen. På disse møtene har vi fått faglig og teknisk veiledning, samt veiledning i arbeidsprosessen og fremdriften. Vi har på disse møtene

[8] Programvare for IP-telefoni. For detaljer se kapittel 1.5.1.
[9] Nasjonal vegdatabank, database med vegobjekter forvaltet av Statens vegvesen.
[10] Application Programming Interface (API) er et grensesnitt for kommunikasjon mellom programvare. APIet beskriver de metoder som en gitt programvare eller et bibliotek kan kommunisere med.

også hatt demo for Christoffer, og han har veiledet oss i å presentere produktet, samt gitt oss konkrete tilbakemeldinger på selve produktet vi da har presentert. Dette har vært til god hjelp for oss og representert en trygghet for oss for å sikre at vi gjør et solid arbeid.

1.2.3 Med veileder

Kommunikasjonen vår med veileder *Eva Hadler Vihovde* har for det meste bestått av faste, avtalte møter. I planleggingsfasen var dette veldig nyttig da Eva veiledet oss med å gjøre valg og prioriteringer underveis i planleggingen, og hun ga nyttige råd under arbeidet med kravspesifikasjonen. Ettersom Eva ikke er kjent med utvikling for iPhone og iOS kunne hun naturlig nok ikke gi oss teknisk veiledning under produksjonsfasen, men hun ga oss nyttig veiledning med tanke på brukervennlighet og funksjonalitet i produktet. I tillegg til dette fungerte hun som en pådriver for fremgang i prosjektet, ettersom vi viste frem produktet vårt ved hver sprint til henne også.

HØGSKOLEN I OSLO OG AKERSHUS

Figur 3: Vår veileder ved Høgskolen i Oslo og Akershus var Eva Hadler Vihovde.

Utover dette har det vært sporadisk kommunikasjon med Eva per e-post, sms eller telefon, når det har vært behov for dette. Totalt sett har vi hatt god kommunikasjon med Eva og fått god veiledning.

1.3 Tilegning av kunnskap

Det var mye ny kunnskap vi måtte tilegne oss til dette prosjektet. Før vi kunne starte med noe utvikling for iOS, måtte vi først lære oss programmeringsspråket Objective-C. Dette var noe vanskeligere enn andre språk, da syntaksen er veldig annerledes fra andre språk vi har lært. Da vi etter hvert behersket Objective-C kunne vi begynne å sette oss inn i bibliotekene og rammeverkene for iOS. Vi jobbet hele tiden med forskjellige tutorials[11] på Internett, som tok for seg alt fra hvordan man lager sin første applikasjon, til mer avanserte og konkrete temaer som Core Data[12] og RestKit[13].

Statens vegvesen sier selv at NVDB API er et kraftig verktøy, men krever at man bruker tid på å sette seg inn i det for å kunne utnytte det på en god måte. Vi brukte derfor mye tid på å sette oss inn i APIet, forstå dataene, hvordan man benyttet dem og hvordan de hang sammen. En kort forklaring på hvordan NVDB API fungerer og hvordan det brukes finnes i kapittel 1.2 i produktdokumentasjonen.

[11] En tutorial er en stegvis gjennomgang på hvordan man lager eller utfører noe.
[12] Core Data er et rammeverk for databaselagring på iOS.
[13] RestKit er et Objective-C-rammeverk for iOS som forenkler kommunikasjonen med REST-baserte webtjenester og mappingen av objekter.

Ettersom BEKK var vår oppdragsgiver ga de oss noen retningslinjer for hvordan vi skulle bedrive prosjektutvikling. BEKK er opptatt av smidig[14] utvikling, og benytter Scrum[15] i sine prosjekter. Vi fikk derfor en innføring av dem i grunnleggende Scrum og bruk av brukerhistorier. Les mer om Scrum i "1.4.3 Utviklingsmodell".

1.4 Planleggingsverktøy og metodikk

I dette kapittelet vil vi ta for oss de verktøy og hjelpemidler vi har benyttet til planlegging og prosjektstyring, samt hvilken metodikk vi har fulgt i denne prosessen.

1.4.1 Dokumentasjon

I alle prosjekter er det viktig å ha en viss plan på hva om skal gjøres og hva som skal produseres. I starten er det behov for overordnede planer, og nærmere produksjonsstart er det behov for mer spesifikke planer. Hvor detaljerte og langsiktige disse planene er avhenger av hvilken utviklingsmodell man ønsker å følge, men det vil uansett være behov for en form for planlegging. Denne planleggingen ble dokumentert i diverse styringsdokumenter (se kapittel 1.4.1.2, "Styringsdokumenter", for detaljer).

I tillegg til styringsdokumenter er det viktig å dokumentere arbeidet underveis slik at man kan gå tilbake og se hvem som gjorde hva, og når. Dette valgte vi å gjøre i en egenprodusert, nettbasert løsning (se "1.4.2.1 Prosjektlogg").

1.4.1.1 Dokumentasjonsstandard

Dokumentet "Dokumentasjonsstandard for bachelor-prosjekter (hovedprosjekt) for Institutt for informasjonsteknologi Høgskolen i Oslo og Akershus" av Ann-Mari Torvatn har fungert som en veiledning underveis i arbeidet med dokumentasjonen. Både når det gjaldt styringsdokumentene, men også når det gjaldt sluttrapporten. Vi valgte bevisst å ikke følge dokumentasjonsstandarden slavisk, men heller bruke den som en inspirasjon og veiledning til hva som burde omtales og hvordan det burde struktureres. Dette valgte vi fordi det var elementer i dokumentasjonsstandarden vi ikke følte at passet helt til vår dokumentasjon. Dette valget ble også tatt i samråd med veileder Eva Vihovde.

1.4.1.2 Styringsdokumenter

Under utrednings- og forprosjektfasen ble det opprettet noen dokumenter som skulle si noe om fremgangen i prosjektet, samt fungere som retningslinjer for den senere utviklingen. Under disse fasene ble følgende styringsdokumenter produsert:

[14] Agile, refererer til en iterativ og inkrementell systemutviklingsmodell.
[15] Scrum er en iterativ og inkrementell systemutviklingsmodell. Les mer i kapittel 1.4.3.

- **Statusrapport** var et dokument som ble levert i oktober 2012. Dette var en kort rapport der vi beskrev hvor langt vi var i prosessen med å finne en oppgave. På dette tidspunktet hadde vi sendt en søknad til BEKK, og avventet svar.
- **Prosjektskisse** var det første dokumentet som beskrev hvilken retning prosjektet faktisk skulle ta. Det var avtalt at BEKK skulle være oppdragsgiver og vi hadde skrevet første utkast av omfanget til prosjektet.
- **Forprosjektrapport** ble skrevet i januar 2013, noe tid etter prosjektskissen. I dette dokumentet var flere detaljer på plass, og vi hadde utformet noen mål og rammebetingelser for prosjektet.
- **Kravspesifikasjon** ble skrevet i starten av februar. Av styringsdokumentene har dette vært det viktigste dokumentet under produksjonsfasen, da det var dette dokumentet som konkret spesifiserte hvilke krav som gjaldt til funksjonalitet, brukervennlighet og design. Les mer om kravspesifikasjonen og dens rolle i del 3 av dette dokumentet.

1.4.2 Prosjektside på nett

Under hele prosjektet har vi hatt en prosjektside på nett. Her har vi hatt generell informasjon om prosjektet, publisert styringsdokumentene underveis og loggført arbeidet vårt i en prosjektlogg. I tillegg har vi hatt lenker til oppdragsgiver og veileder, samt GitHub[16]-repositoriet[17] vårt, på denne siden. Mot slutten av produksjonsfasen la vi også ut en demonstrasjonsvideo av applikasjonen vår på denne siden.

1.4.2.1 Prosjektlogg

Som nevnt over førte vi en prosjektlogg på prosjektsiden vår. Her skrev vi individuelle innlegg hver gang vi hadde jobbet hver for oss, samt innlegg for gruppen om vi hadde vært i møter eller jobbet sammen. Kort sagt har alt arbeid vi har gjort blitt loggført i denne loggen, sammen med tidsbruk for det aktuelle arbeidet. Dette har vært til nytte ved at vi har kunnet gå tilbake og se når vi gjorde hva og hvem som gjorde hva.

[16] GitHub er en nettbasert tilbyder av tjenester for prosjekter som bruker Git versjonskontrollsystem. Les mer om Git under kapittel 1.5.3.
[17] Repository er et begrep fra versjonskontroll og omhandler en datastruktur som bl.a. inneholder filer, mapper, historikk og referanser til tidligere versjoner.

2013-03-18	
	La til et vilttrekk-objekt, lagde et superobjekt som alle vegobjektene arver fra, flyttet kode som gikk igjen inn i passende klasser og metoder, og omstrukturerte tolkingen av returnerte objekter i VegObjektKontroller til å utnytte fordelene ved arv og kunne dermed fjerne mye unødvendig kode.
210 minutter	
2013-03-15	
Selvstudie om brukerpreferanser/innstillinger i iOS.	
120 minutter	
2013-03-15	
Skype-møte ang prosessen videre, drøfting rundt veiskilt og app-innstillinger.	
60 minutter	
2013-03-14	
	Fikset problemene rundt landskapsmodus og HUD ved å opprette 3 forskjellige viewcontrollere hvor den ene tar seg av all logikk, og de to andre arver fra den første og tar seg av henholdsvis portrett- og landskapsvisning.
240 minutter	

Figur 4: Et utdrag fra prosjektloggen på nettsiden.

1.4.3 Utviklingsmodell

Vi har ikke hatt en konkret utviklingsmodell som vi har fulgt helt etter boka, men vi har jobbet etter prinsipper fra smidige utviklingsmodeller og i hovedsak tatt elementer fra Scrum.

Kjennetegnet ved agile utviklingsmodeller er at de er iterative og inkrementelle, og at krav og løsninger utvikles og velges underveis i prosjektet, i samarbeid med de involverte parter. Dette innebærer at det blir mindre planlegging i forkant av prosjektet, mindre byråkrati og enklere å gjøre endringer i rammer og mål underveis, etter hvert som man ser endringer i behov og muligheter underveis i prosjektet. Ved slutten av hver iterasjon skal det alltid leveres en kjørbar versjon av programvaren, slik at denne kan demonstreres.

Som allerede nevnt jobbet vi ikke konkret etter Scrum, men brukte følgende elementer fra modellen:

- **Iterasjonsbasert arbeid**, i Scrum kalt sprinter. Ved starten av hver sprint plukkes det arbeidsoppgaver og funksjoner fra produktkøen, og disse oppgavene skal så sorteres i prioritert rekkefølge. Antallet oppgaver velges basert på grovestimering.
- **Grovestimering** av tidsbruk av arbeidsoppgaver og funksjonalitet som skal implementeres. I praksis estimeres det ikke antall timer per oppgave, men kompleksitet og omfang evalueres, og så estimeres det hvor mye man rekker i løpet av en sprint.
- **Stand-up** er daglige møter der team-medlemmene koordinerer arbeidet seg imellom. Vi hadde ikke alltid daglige møter, men hadde kommunikasjon oss i mellom tilnærmet daglig. Dersom man følger Scrum skal disse Scrum-møtene gjennomføres stående, være relativt korte og team-medlemmene skal omtale følgende tre spørsmål:
    - Hva er gjort siden forrige Scrum-møte?

- Hva skal gjøres før neste møte?
- Hva har (eventuelt) vært til hinder for at team-medlemmet var effektiv i implementeringen av funksjonalitet?

Vi gjennomførte ikke denne delen av Scrum-møtene, men for erfaringens skyld bakte vi den inn i sprint-møtet ved slutten av hver sprint, sammen med sprint retrospective.

- **Sprint review og sprint retrospective** utføres ved slutten av hver sprint. I sprint review demonstreres funksjonaliteten som ble lagt til i det foregående inkrementet, slik at man kan få tilbakemeldinger fra de partene som er interessenter i prosjektet. Deretter gjennomføres sprint retrospective der man samler erfaringer fra siste sprint og finner måter å jobbe litt mer effektivt på neste sprint.

Utover det ovennevnte kan vi legge til at vi arbeidet etter sprinter som varte i to uker, og delte produksjonsfasen vår inn i fem sprinter. Les mer om dette i kapittel 2.3, "Produksjonsfasen".

1.4.4 Versjonskontroll og backup

Til styringsdokumenter og dokumentasjon arbeidet vi begge på hver våre lokale filer i lokale mapper. Disse mappene hadde vi begge synkronisert med hver vår sky-tjeneste, slik at det automatisk alltid var en backup av dokumentene i nettskyen. For ordens skyld kan det nevnes at Lars brukt Microsoft SkyDrive, mens Henrik brukte norske Jottacloud. Ingen av oss opplevde noen problemer med disse tjenestene, og vi var godt fornøyde med denne måten å ha backup på.

Figur 5: Prosjektet har et eget åpent repository på GitHub, her representert av maskoten Octocat (Foto: GitHub).

Under produksjonsfasen brukte vi Git for versjonskontroll, og hadde et åpent repository på GitHub. Vi valgte å bruke dette i hovedsak fordi det er et kjent og solid system, samt at vi hadde benyttet det ved tidligere anledninger, og var derfor noe kjent med det. Vi var fornøyd med dette valget og hadde svært få problemer underveis. De problemene vi hadde skyldtes utelukkende manglende kunnskap fra vår side, men dette fant vi løsninger til på Internett, og tok således lærdom av dette.

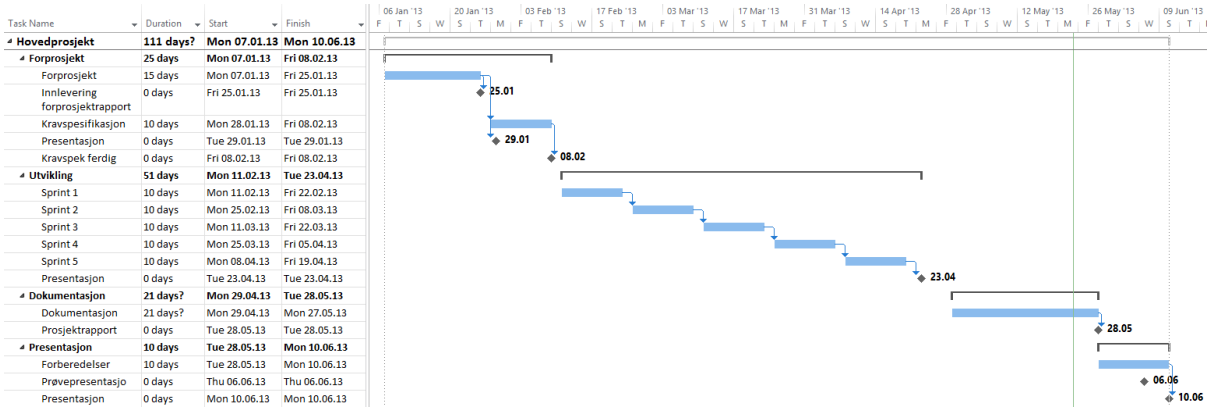
1.4.5 Diagrammer

Som en del av planleggingen tegnet vi også diverse diagrammer underveis. Dette ble ikke gjort på bakgrunn av den utviklingsmodellen vi jobbet etter, men vi ønsket å ha noen overordnede planer for

diverse aspekter av prosjektet. Alle planene er noe revidert underveis, basert på erfaringer og lærdom vi fikk etter hvert som vi jobbet, men ideene er fortsatt de samme. Vi vil her presentere de diagrammene vi har benyttet, slik de så ut etter siste revisjon.

1.4.5.1 Gantt-diagram

Et Gantt-diagram er ikke noe man egentlig skal lage når man arbeider smidig og etter Scrum. Vi utarbeidet heller ikke et Gantt-diagram slik det skal brukes, der man legger inn alle oppgaver og alt som skal utvikles, og så estimerer tidsbruk for hele prosjektet. Istedenfor brukte vi Gantt-diagrammet som en overordnet plan for å holde styr på diverse frister, samt å kunne fordele sprintene våre jevnt utover den tiden vi hadde til produksjonsfasen.



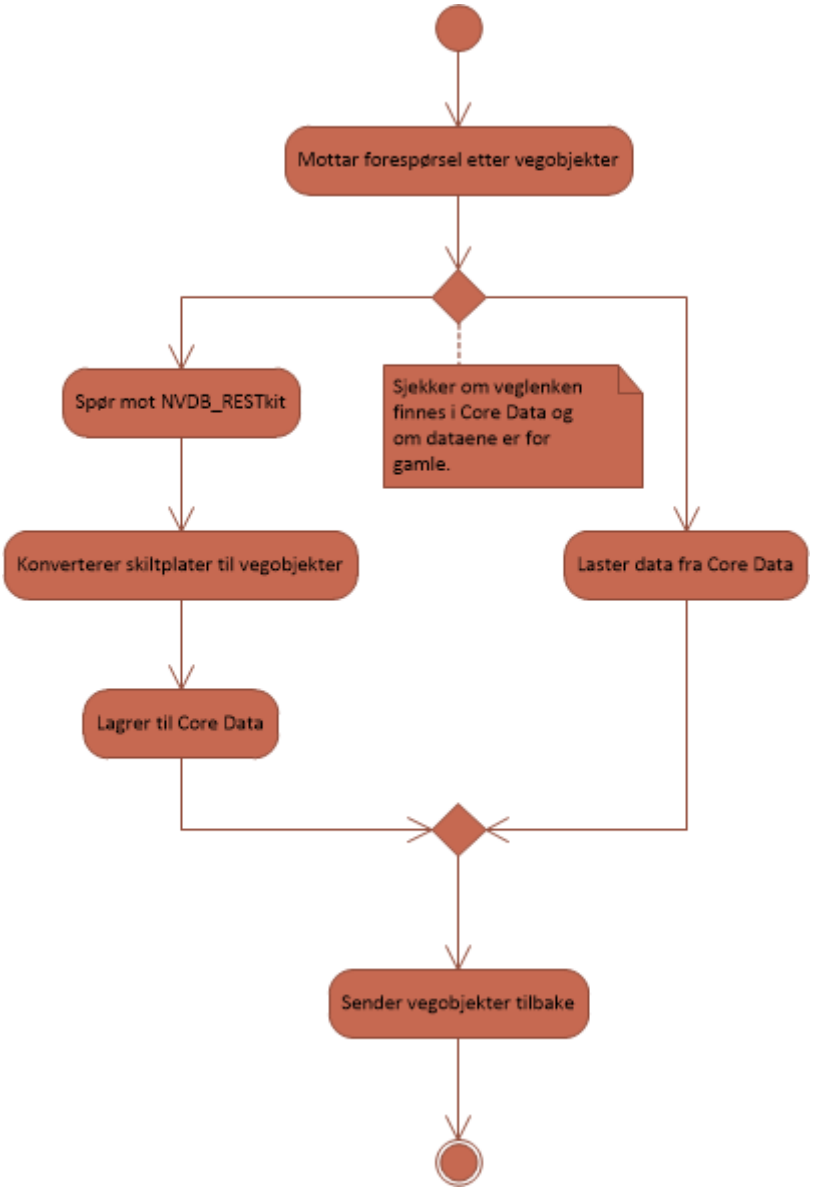
Task Name	Duration	Start	Finish
Hovedprosjekt	**111 days?**	**Mon 07.01.13**	**Mon 10.06.13**
Forprosjekt	**25 days**	**Mon 07.01.13**	**Fri 08.02.13**
Forprosjekt	15 days	Mon 07.01.13	Fri 25.01.13
Innlevering forprosjektrapport	0 days	Fri 25.01.13	Fri 25.01.13
Kravspesifikasjon	10 days	Mon 28.01.13	Fri 08.02.13
Presentasjon	0 days	Tue 29.01.13	Tue 29.01.13
Kravspek ferdig	0 days	Fri 08.02.13	Fri 08.02.13
Utvikling	**51 days**	**Mon 11.02.13**	**Tue 23.04.13**
Sprint 1	10 days	Mon 11.02.13	Fri 22.02.13
Sprint 2	10 days	Mon 25.02.13	Fri 08.03.13
Sprint 3	10 days	Mon 11.03.13	Fri 22.03.13
Sprint 4	10 days	Mon 25.03.13	Fri 05.04.13
Sprint 5	10 days	Mon 08.04.13	Fri 19.04.13
Presentasjon	0 days	Tue 23.04.13	Tue 23.04.13
Dokumentasjon	**21 days?**	**Mon 29.04.13**	**Tue 28.05.13**
Dokumentasjon	21 days?	Mon 29.04.13	Mon 27.05.13
Prosjektrapport	0 days	Tue 28.05.13	Tue 28.05.13
Presentasjon	**10 days**	**Tue 28.05.13**	**Mon 10.06.13**
Forberedelser	10 days	Tue 28.05.13	Mon 10.06.13
Prøvepresentasjo	0 days	Thu 06.06.13	Thu 06.06.13
Presentasjon	0 days	Mon 10.06.13	Mon 10.06.13

Figur 6: Dette Gantt-diagrammet ble laget i begynnelsen av prosjektet.

Underveis i prosjektet endret vi den siste sprinten da vi så at vi hadde behov for mer tid. I utgangspunktet var planen vår å ta fri i påskeferien, men vi så at vi fikk knapt med tid, og valgte derfor å inkludere påskeuka i sprint 5, slik at også denne fikk en varighet på to uker.

1.4.5.2 Aktivitetsdiagram

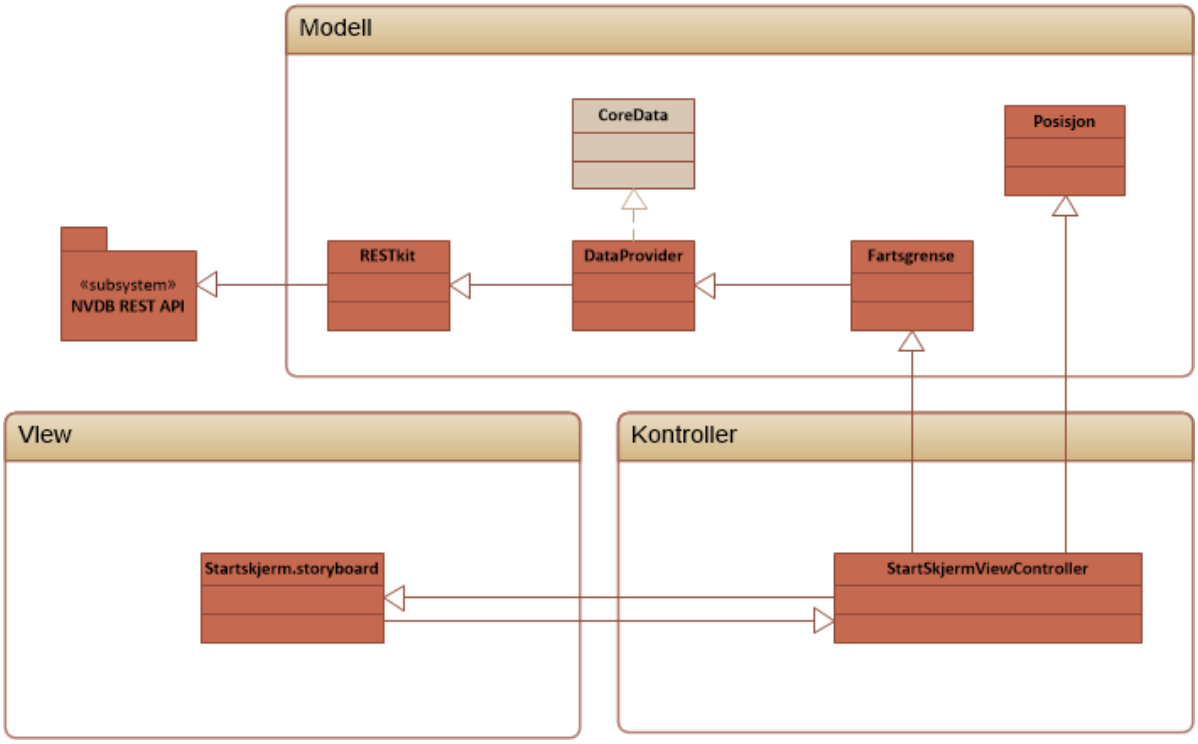
For å hjelpe oss med å se gangen i databehandlingen internt i applikasjonen tegnet vi opp et aktivitetsdiagram. Dette handler altså ikke om aktiviteter som brukeren av applikasjonen vil foreta seg. Dette er heller ikke en naturlig del av smidig utvikling, men vi ønsket å ha dette på papir, slik at vi hadde en felles forståelse av hva målet skulle være da vi utviklet dette.



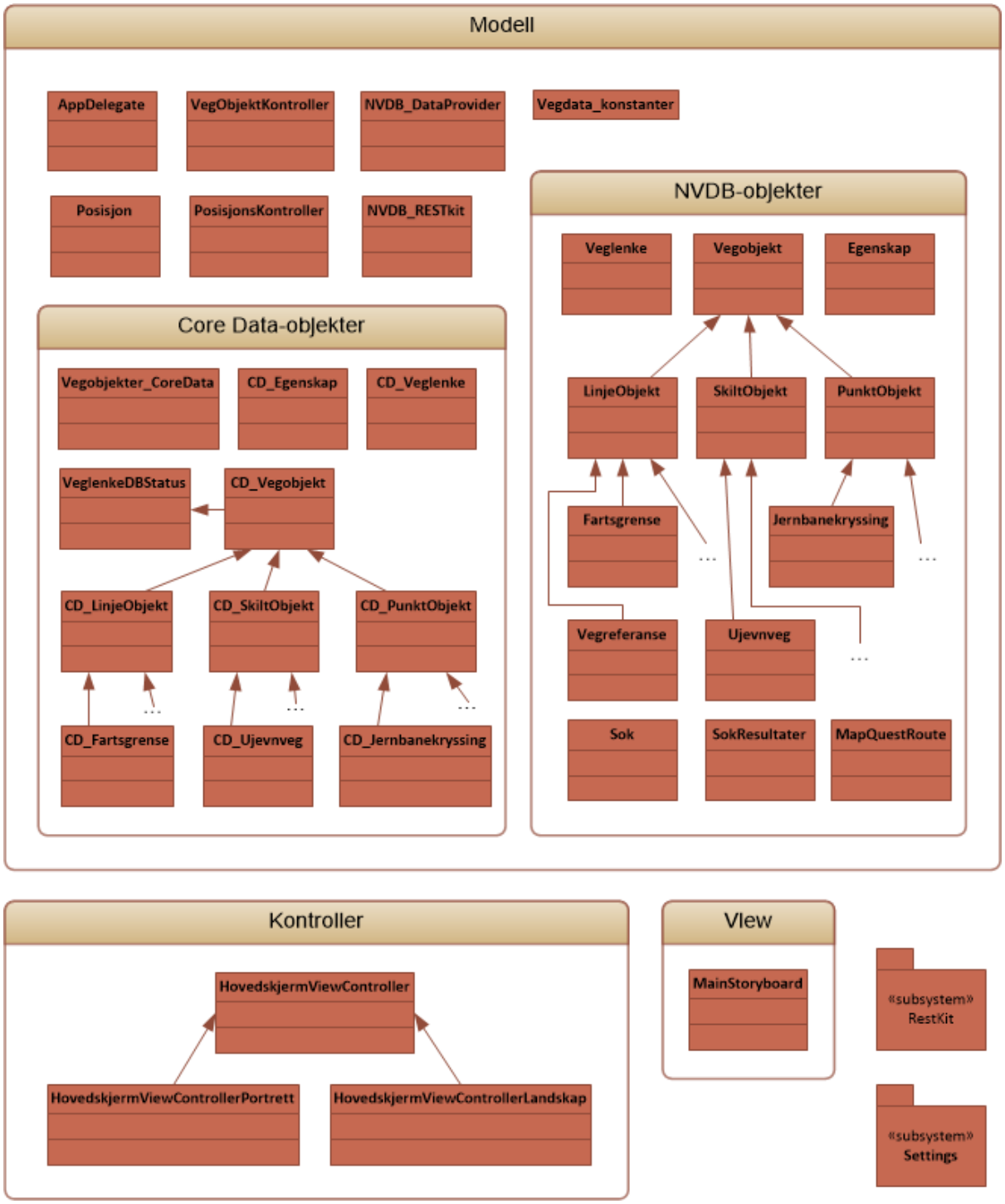
Figur 7: Aktivitetsdiagrammet viser hva som skjer i NVDB_DataProvider-klassen når den mottar en forespørsel etter vegobjekter. Les mer om dette i produktdokumentasjonen, kapittel 4.3.

1.4.5.3 Klassediagram

Ettersom vi ganske raskt så at det ville bli en del klasser å holde styr på, opprettet vi et klassediagram for å få oversikt. Dette diagrammet ble i første omgang konstruert med de klasser vi så behov for i starten. Etter hvert som vi gjorde oss erfaringer med tekniske utfordringer og hvilke behov vi faktisk hadde, ble diagrammet revidert. Revisjonen skjedde da alltid i forkant av endringene, slik at vi hadde en klar plan for vegen videre.



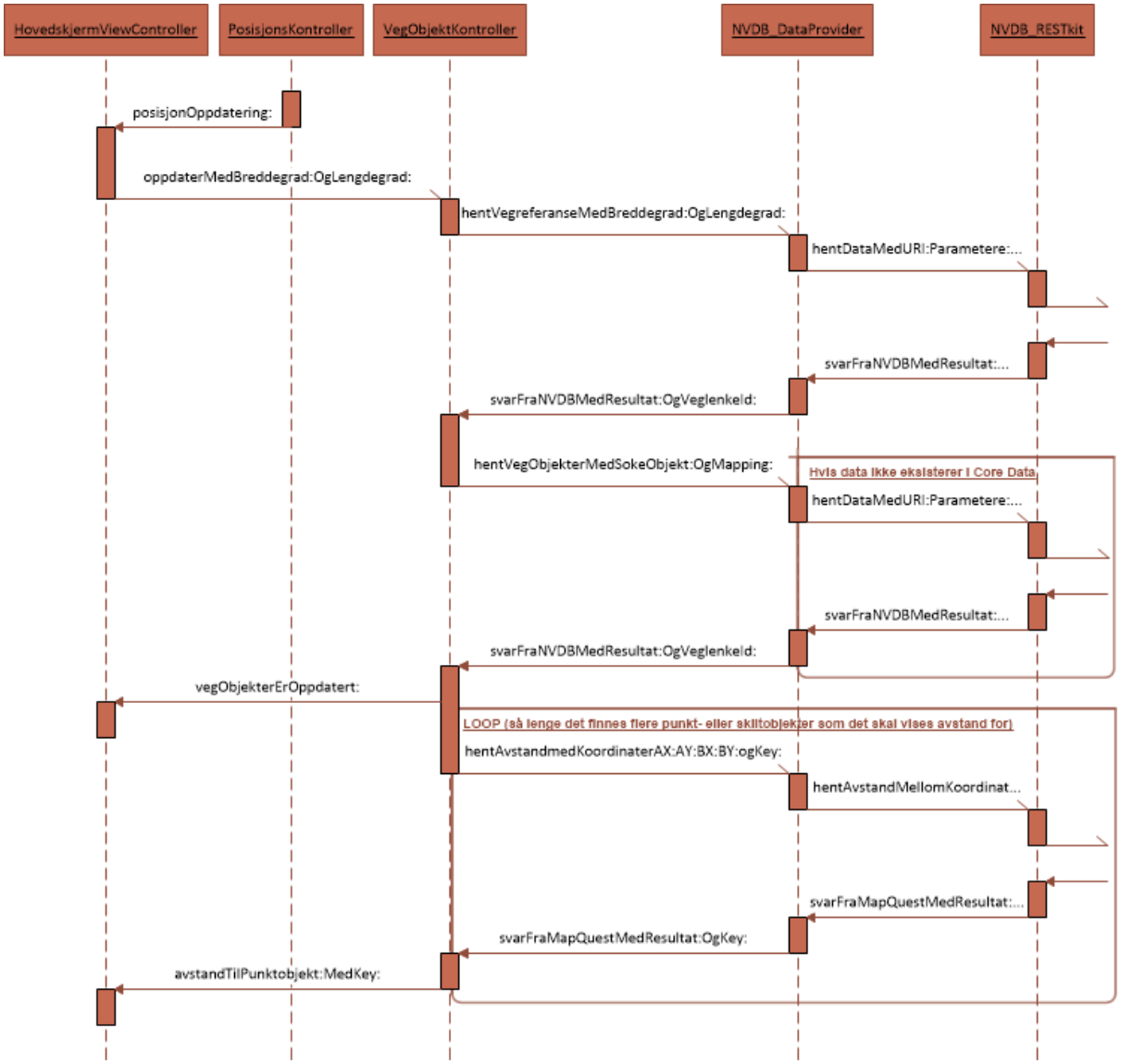
Figur 8: Slik så klassediagrammet ut til første sprint.



Figur 9: Dette er det endelige klassediagrammet i prosjektet, og viser hvordan Kjørehjelperen ser ut i dag.

1.4.5.4 Sekvensdiagram

I forhold til å holde oversikt over kommunikasjon mellom klasser og den logiske sammenhengen i programmet ønsket vi også å benytte et sekvensdiagram. Dette ble opprettet nokså sent i prosjektet. Diagrammet har ikke hatt spesiell stor nytteverdi for oss, men det er inkludert i produktdokumentasjonen, og vil der kunne ha stor nytteverdi for eventuelle utviklere som skal videreutvikle prosjektet.



Figur 10: Sekvensdiagrammet viser en normalkjøring i applikasjonen.

1.4.6 Brukerhistorier

Som en del av planleggingen benyttet vi oss også av brukerhistorier. Dette er noe BEKK er vant med å gjøre, så de ønsket tidlig at vi skulle sette oss litt inn i dette, ettersom de er kjent med nytteverdien av dette. Brukerhistoriene ble skrevet ved starten av hver sprint, og fungerte således som en rettesnor for målet med den funksjonaliteten vi til en hver tid arbeidet med å implementere. Vi vil i de følgende underkapitler gjengi brukerhistoriene vi skrev, sprint for sprint.

1.4.6.1 Sprint 1

Fartsgrenser

Som sjåfør skal jeg kunne se fartsgrensen på vegen jeg kjører på slik at jeg kan justere farten min.

Akseptansekrav:

- Jeg skal kunne se fartsgrensen på vegen jeg kjører på

- Jeg skal slippe å interagere med telefonen
- Fartsgrensen skal presenteres i samme stil som fartsgrenseskiltet

1.4.6.2 Sprint 2

Hindre telefonen fra å sovne

Som sjåfør skal jeg kunne se applikasjonen hele tiden slik at jeg ikke behøver å interagere med den mens jeg kjører.

Akseptansekrav:

- Mens applikasjonen kjører skal telefonen ikke skrus av automatisk

Forkjørsveg

Som sjåfør skal jeg kunne se om jeg er på en forkjørsveg eller ikke slik at jeg vet om jeg må være ekstra oppmerksom på trafikk fra høyre.

Akseptansekrav:

- Jeg skal kunne se om vegen jeg er på er en forkjørsveg eller ikke
- Informasjonen skal presenteres i samme stil som forkjørsvegskilt

Landscape

Som sjåfør skal jeg kunne velge om jeg vil ha telefonen i portrett eller landskapsmodus slik at jeg kan legge fra meg telefonen slik jeg vil og fremdeles kunne lese informasjonen.

Akseptansekrav:

- Jeg skal kunne vende telefonen og fremdeles få presentert informasjonen riktig veg
- Informasjonen skal tilpasses telefonens visningsmodus best mulig

1.4.6.3 Sprint 3

HUD (Head-up display)

Som sjåfør skal jeg kunne legge telefonen på dashbordet slik at jeg får presentert informasjonen som en refleksjon i frontruten[18].

Akseptansekrav:

- Jeg skal kunne se informasjonen på riktig måte reflektert i frontruten
- Informasjonen skal presenteres i samme stil som ellers, men tilpasses for klarest mulig refleksjon

[18] Head-up display eller heads-up display, også kjent som HUD, er en gjennomsiktig skjerm som viser informasjon uten at brukerne må se bort fra sine vanlige synspunkter. Opprinnelsen til navnet stammer fra at en pilot skal kunne se informasjon med hodet "opp" og se frem, i stedet for å se ned på instrumentene i cockpiten.

Varselskilt

Som sjåfør skal jeg kunne få informasjon om kommende og gjeldende varselskilt* om diverse farer og annet som varsles om slik at jeg kan gjøre nødvendige forberedelser når jeg nærmer meg og være tilstrekkelig oppmerksom når jeg er i et område der et varselskilt er gjeldende.

Akseptansekrav:

- Jeg skal få et varsel om kommende varselskilt* i rimelig tid før det vil være gjeldende.
- Jeg skal se hvilke varselskilt* som er gjeldende der jeg er nå.
- Informasjonen skal presenteres i samme stil som varselskiltet den refererer til.

* Det er kun krav om utvalgte varselskilt. Hvilke varselskilt dette gjelder vil bestemmes nærmere under utvikling.

1.4.6.4 Sprint 4

Brukerpreferanser

Som sjåfør skal jeg kunne velge hvilken informasjon jeg skal få presentert, og hvilke varsler jeg ønsker å motta slik at applikasjonen fungerer etter mine ønsker og behov.

Akseptansekrav:

- Jeg skal kunne endre hvilken informasjon jeg får presentert av applikasjonen.
- Jeg skal kunne endre hvilke varsler jeg får presentert, bl.a. varsler om varselskilt.

Lydvarsling

Som sjåfør skal jeg bli varslet ved hjelp av lyd når det på skjermen bli presentert nye skilt for strekningen jeg er på slik at jeg ikke behøver å konstant følge med på skjermen, men kan bli gjort oppmerksom på dette gjennom lydvarsling.

Akseptansekrav:

- Jeg skal ved hjep av audio bli gjort oppmerksom på at applikasjonen oppdager og presenterer nye skilt underveis på streknigen jeg er på.

Avstandsvisning

Som sjåfør skal jeg få informasjon om avstanden til skilt og objekter på strekningen jeg er på slik at jeg kan ta høyde for dette og gjøre eventuelle forberedelser i god tid.

Akseptansekrav:

- Jeg skal kunne se avstanden til et skilt eller det objektet skiltet representerer.

1.4.6.5 Sprint 5

Ingen brukerhistorier.

1.5 Verktøy og teknologi

1.5.1 Skype

Skype er programvare som brukes til IP-telefoni. Brukere som har installert programvaren og opprettet konto kan ringe til hverandre via datamaskinen. Ettersom samtalen skjer over Internett er dette helt kostnadsfritt med tanke på antall samtaler og lengden på samtalene. Programmet har også funksjonalitet for IM[19], filoverføring og videosamtaler. Med unntak av videosamtaler benyttet vi oss også av disse funksjonene.

Figur 11: Skype ble ofte benyttet til kommunikasjon.

1.5.2 Trello

Trello er en nettbasert applikasjon for prosjektstyring. Funksjonalitet som skal implementeres, eller spesifikke oppgaver som skal gjøres, opprettes som et kort og legges i en liste over oppgaver som skal gjøres, en såkalt "backlog". Hvert kort kan tildeles en eller flere deltakere i prosjektet, som skal løse den spesifikke oppgaven. Når en oppgave er påbegynt flyttes kortet til listen over oppgaver som jobbes med, og når den er ferdig flyttes kortet videre til en liste over ferdige oppgaver. Slik kunne vi enkelt holde oversikt over hva som skulle gjøres, hvem som gjorde hva og når oppgaver var utført.

1.5.3 Git

Git er et distribuert versjonskontrollsystem med mulighet for alle-til-alle synkronisering. Git tilbyr desentralisert arbeidsflyt med en tillitsbasert pyramidestruktur. Ved hjelp av Git var det enkelt å utvikle hver for oss, selv om vi til tider gjorde endringer på de samme filene, ettersom Git for det meste håndterer sammenfletting av disse endringene.

1.5.4 Xcode

Xcode er Apples integrerte utviklingsmiljø for utvikling av programvare til OS X og iOS. Ettersom Xcode kun kan benyttes i OS X er man avhengig av å ha en Mac for å kunne bruke det. Xcode har også støtte for andre språk enn C og Objective-C, men det var kun dette vi hadde behov for i vårt prosjekt.

[19] Instant Messaging, eller lynmelding på norsk, er en type samtale over et nettverk der man sender tekstbeskjeder til hverandre i sanntid.

1.5.5 Objective-C

Objective-C er Apples objektorienterte høynivå-programmeringsspråk[20] for OS X og iOS som tilfører Smalltalk[21]-lignende kommunikasjon til programmeringsspråket C. Objective-C ble opprinnelig utviklet av NeXT på starten av 80-tallet for deres operativsystem NeXTSTEP.

1.5.6 Cocoa Touch

Cocoa Touch er et rammeverk for å utvikle programvare som kan kjøre på iPhone, iPod Touch og iPad. Cocoa Touch tilbyr et abstraksjonslag for iOS og er basert på MAC OS X Cocoa API. I likhet med Cocoa følger også Cocoa Touch designmønsteret MVC[22]. Verktøy for å utvikle applikasjoner basert på Cocoa Touch er inkludert i iOS SDK[23].

[20] Et høynivåspråk har stor abstraksjon fra måten datamaskinen fungerer.
[21] Smalltalk er et dynamisk, objektorientert og reflekterende programmeringsspråk fra 1980.
[22] Model-view-controller er et designmønster for programvareutvikling. For detaljer se produktdokumentasjonen kapittel 2.1, "Model View Controller".
[23] Software development kit (SDK) er verktøy som lar deg utvikle programmer til en spesifikk programvarepakke, et rammeverk, en hardware-plattform, et operativsystem e.l.

2 Utviklingsprosessen

I denne delen av prosessdokumentasjonen ønsker vi å belyse utviklingen av prosjektet. Noen av de følgende kapitler er også dels omtalt i andre kapitler, men der med et annet fokus. I denne delen vil det være et kronologisk fokus der vi vil belyse fremdriften i prosjektet.

Den kronologiske fremdriften kan også ses på figur 6 i kapittel 1.4.5.1, "Gantt-diagram".

2.1 Utredningsfasen

Utredningsfasen var en noe vag fase på slutten av 2012. Den begynte allerede da vi satte sammen grupper, og varte frem til vi kunne begynne å konkretisere oppgaven vår.

I denne fasen skrev vi først en statusrapport, og senere en prosjektskisse. Vi begynte også smått å forberede oss for iOS og Objective-C da prosjektskissen var klar og det var klart at det var dette vi skulle jobbe med.

2.2 Forprosjektsfasen

Denne fasen begynte rett over nyttår, og det var nå vi måtte begynne å sette de første konkrete målene og rammebetingelsene for prosjektet. Dette var en myk start på denne fasen, med en glidende overgang til utformingen av konkrete krav til prosjektet.

I denne fasen skrev vi i januar en forprosjektrapport der overordnede mål og rammebetingelser ble formulert. Da dette var klart, i slutten av januar, begynte vi arbeidet med kravspesifikasjonen. 8. februar var denne ferdig, og vi nærmet oss da å kunne gå over i produksjonsfasen.

Underveis i denne fasen startet vi også arbeidet med å tilegne oss nødvendig kunnskap om de språk og rammeverk vi ville få bruk for i produksjonsfasen.

2.3 Produksjonsfasen

Kort tid etter at kravspesifikasjonen var klar gikk vi over i produksjonsfasen. Vi hadde på det tidspunktet tilegnet oss tilstrekkelig med kunnskap innen Objective-C og iOS SDK til å kunne begynne på basisen til produktet. Som nevnt i "1.4.3 Utviklingsmodell", arbeidet vi iterativt i såkalte sprinter. Vi hadde totalt fem sprinter, og vi vil her presentere grovt hvordan arbeidet ble fordelt over de fem sprintene. Dette var ikke noe vi kunne planlegge på forhånd – hver sprint ble planlagt ved starten av den aktuelle sprinten.

2.3.1 Sprint 1

Mandag 11.02.13 – fredag 22.02.13

I den første sprinten var det mye grunnarbeid som måtte gjøres, og produktet vi leverte ved slutten av sprinten hadde derfor lite funksjonalitet. Den vesentlige basisfunksjonaliteten som ble implementert her var:

- Finne lokasjonskoordinater ved hjelp av telefonens GPS[24]
- Et kommunikasjonslag for å kommunisere med NVDB API
- Et skjermbilde å vise data i

Figur 12: Etter første sprint gjennomførte vi en testtur av applikasjonen.

Videre benyttet vi denne basisfunksjonaliteten til å implementere følgende funksjonalitet:

- Benytte kommunikasjonslaget mot NVDB API for å hente ut informasjon om hvilken veg man er på
- Benytte informasjonen om hvilken veg man er på for å hente ut fartsgrensen på den aktuelle vegen
- Vise mottatte data om veg og fartsgrense på skjerm. Fartsgrense ble her vist grafisk som et skilt

Da denne sprinten var over var vi godt fornøyd med å være i gang, og å ha noe konkrete data å vise. Vi ble også raskt bevisste på hvor lang tid utvikling tok med et språk vi var såpass ferske i, spesielt når vi også jobbet mot et API vi ikke var drevne i, men dette var nyttige erfaringer vi tok med oss til de følgende sprintene. Vi avsluttet denne sprinten med en testtur, noe som var veldig nyttig ettersom vi ble oppmerksom på hvordan GPSen oppførte seg, hvordan kommunikasjonen mot NVDB API gikk over tid og hvordan vi kunne forbedre disse aspektene. Les mer om selve testen i testdokumentasjonen, kapittel 2.1.

[24] NAVSTAR Global Positioning System (GPS) er et nettverk av satellitter som er plassert i bane rundt Jorden av det amerikanske forsvaret. Systemet gjør det mulig for en mottaker å fastsette egen posisjon med svært stor nøyaktighet overalt i verden, under nær sagt alle værforhold.

2.3.2 Sprint 2

Mandag 25.02.13 – fredag 08.03.13

Da neste sprint startet nølte vi ikke med å ta fatt på utfordringene. I denne sprinten tok vi fatt på diverse problemer vi oppdaget på testturen i slutten av sprint 1, og mot slutten av sprinten fikk vi implementert noe ny funksjonalitet. De sentrale forandringene i denne sprinten var som følger:

- Hindre telefonen fra å "sovne", slik at applikasjonen alltid er åpen og på når den først er åpnet
- Diverse forbedringer i oppdateringen av GPS-posisjon, som optimalisert oppdateringsfrekvens og økt presisjon
- Endret måten vi hentet informasjon fra NVDB API for å begrense databruk og øke nøyaktigheten i resultatet
- La til landskapsmodus for applikasjonen
- La til forkjørsveg og presenterte denne informasjonen på samme måte som fartsgrense

Etter denne sprinten valgte vi å ikke ta noen testtur, ettersom det var såpass lite ny funksjonalitet. Vi fikk dekket de behov vi hadde for testing gjennom iOS-simulatoren i Xcode.

2.3.3 Sprint 3

Mandag 11.03.13 – fredag 22.03.13

Ved tredje sprint begynte vi å bli varme i trøya. Vi så at grunnmuren i applikasjonen nå begynte å være solid, og vi fokuserte nå mer på å utvide applikasjonen til å kunne presentere mer informasjon. I løpet av denne sprinten fikk vi gjort følgende endringer:

Figur 13: I sprint 3 la vi blant annet til varsling av vilttrekk.

- Fikset bug der telefonen "spammet" forespørsler mot NVDB API om man sto helt stille
- La til funksjonalitet for HUD
- La til noen fareskilt: vilttrekk, høydebegrensning, jernbanekrysning og fartsdemper
- Tilpasset skjermbildet for opptil 5 skilt (7 på iPhone 5)[25]

Av samme grunner som etter sprint 2 valgte vi også her å ikke ta noen testtur. Det var dessuten noen logistikkutfordringer knyttet til fysisk testing (les mer om dette i testdokumentasjonen). Etter denne sprinten begynte vi å se applikasjonen ta form, men vi så at det fortsatt gjensto mye arbeid, og gikk derfor raskt i gang med sprint 4.

[25] iPhone 5 har en 4" skjerm, mens tidligere modeller har 3,5" skjermer.

2.3.4 Sprint 4

Mandag 25.03.13 – fredag 05.04.13

I den fjerde sprinten bestemte vi oss for at det aller meste av funksjonalitet nå måtte implementeres, slik at vi hadde sprint 5 til diverse bugfiksing og finpussing. I løpet av denne sprinten fikk vi gjort følgende arbeid med applikasjonen:

Figur 14: Under testturen etter sprint 4 testet vi blant annet HUD-modus.

- Core Data ble implementert slik at data vi hentet fra NVDB API kunne caches[26] for gjenbruk
- Lydvarsling ble implementert, slik at brukeren varsles når det dukker opp et nytt skilt på strekningen
- Brukerpreferanser ble implementert for de skilt og funksjoner som allerede var implementert
- Avstandsvisning til punktobjekter ble implementert
- Filtersøk mot NVDB API ble implementert for å redusere mengden datatrafikk

Ved slutten av denne sprinten følte vi at det var behov for en ny testtur. Vi følte nå at vi hadde gjort grunnmuren i applikasjonen solid nok, samt at vi hadde nok skilt og funksjonalitet til at det var hensiktsmessig å teste det på en kjøretur. I tillegg var dette viktig med tanke på at neste sprint ville være den siste.

2.3.5 Sprint 5

Mandag 08.04.13 – fredag 19.04.13

Da vi kom til den siste sprinten så vi at vi fremdeles hadde mye arbeid foran oss, og vi måtte derfor arbeide ekstra effektivt her. Vi hadde nå også god kontroll på koden vår, og det var tydelig lettere å utvikle mer funksjonalitet når vi hadde blitt så mye stødigere på språket og miljøet siden første sprint. I denne sprinten fikk vi gjort følgende tillegg og endringer:

- Diverse brukerinnstillinger ble lagt til, som visning av HUD-knappen, lydvarsling og begrensning av lokale data i cache
- Gikk over til MapQuest[27] for å finne avstander. Endret fra Google Maps av hensyn til Googles bruksvilkår[28]

[26] Catching er mellomlagring av data.

- Mange nye skilt ble lagt til (ca. 30 nye skilt) og brukerinnstillinger for disse ble lagt til
- Grafikk for ikon og lasteskjerm ble implementert
- Diverse bugfiksing og små optimaliseringer

Figur 15: Vi laget et eget ikon til applikasjonen som vises på startskjermen på telefonen og i App Store.

Etter denne sprinten tok vi en tredje testtur for å se at det ikke var noen alvorlige feil eller mangler. Det vi fant kan man lese om i testdokumentasjonen, kapittel 2.3. Generelt var vi godt fornøyd med både siste sprinten og produktet etter denne turen.

2.3.6 Utfordringer underveis

Underveis i produksjonsfasen støtte vi på flere utfordringer som gjorde at vi måtte endre eller tilpasse løsningen vår. Noen var vanskelige å løse, noen krevde ikke like mye innsats, mens noen rett og slett ikke lot seg løse. Vi ønsker gjennom dette kapitlet å belyse det faktum at dette ikke har vært en triviell prosess. Selv om applikasjonen vår ikke har flust med funksjonalitet for brukeren å utforske, så er det viktig å være oppmerksom på at det er mye data som skal lagres, behandles og presenteres, og dette krever en god del funksjonalitet og en del mekanismer som vil være skjult for brukeren.

De følgende punktene er omtalt i tilfeldig rekkefølge.

2.3.6.1 Brukerinnstillinger

Da vi begynte arbeidet med brukerinnstillingene oppdaget vi tidlig at denne delen av iOS opererte med en veldig banal løsning.

Når man skal definere innstillinger for en applikasjon i iOS gjøres dette via et enkelt grensesnitt, som så genererer XML. Når man så starter Settings-applikasjonen i iOS, og åpner innstillinger for applikasjonen man har laget, vil Settings-applikasjonen tolke XML'en og presentere innstillingene man har ønsket.

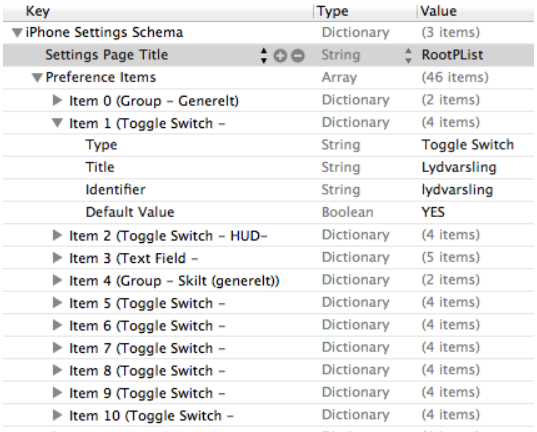
Dessverre legger dette en alvorlig begrensning på innstillings-skjermen, ettersom man ikke kan gjøre den dynamisk på noen måte. Dette la følgende begrensninger på brukerinnstillingene:

- Vi ønsket en knapp som kunne slette lokale data i applikasjonen om man trykket på den. Dette lot seg ikke gjøre. Vi fant ingen løsning på problemet.

[27] MapQuest er en online karttjeneste som tilbyr blant annet vegkart og navigasjonshjelp.
[28] Google krever at deres navigasjonstjeneste benyttes i deres eget kart.

- Vi ønsket en trinnbasert slider for å begrense cache size. Slidere i iOS Settings er trinnløse og viser ikke sin nåværende verdi på noen måte. Vi valgte da å la brukeren definere en cache size limit ved hjelp av et tekstfelt, noe vi selv anser som en stygg løsning.

Vi er klar over at det finnes rammeverk for iOS som kunne løst dette problemet for oss, der man kan lage egendefinerte, dynamiske og iOS-lignende brukerinnstillinger. Grunnet den gitte tidsrammen kunne vi dessverre ikke prioritere å tilegne oss kunnskap om disse rammeverkene, og måtte derfor frastå fra å forsøke å bruke dem.



Key	Type	Value
iPhone Settings Schema	Dictionary	(3 items)
Settings Page Title	String	RootPList
Preference Items	Array	(46 items)
Item 0 (Group – Generelt)	Dictionary	(2 items)
Item 1 (Toggle Switch –	Dictionary	(4 items)
Type	String	Toggle Switch
Title	String	Lydvarsling
Identifier	String	lydvarsling
Default Value	Boolean	YES
Item 2 (Toggle Switch – HUD–	Dictionary	(4 items)
Item 3 (Text Field –	Dictionary	(5 items)
Item 4 (Group – Skilt (generelt))	Dictionary	(2 items)
Item 5 (Toggle Switch –	Dictionary	(4 items)
Item 6 (Toggle Switch –	Dictionary	(4 items)
Item 7 (Toggle Switch –	Dictionary	(4 items)
Item 8 (Toggle Switch –	Dictionary	(4 items)
Item 9 (Toggle Switch –	Dictionary	(4 items)
Item 10 (Toggle Switch –	Dictionary	(4 items)

Figur 16: Det er veldig begrenset hvordan innstillingene til en applikasjon kan se ut. Her fra grensesnittet i Xcode.

2.3.6.2 Avstander

En sentral og veldig viktig del vi ønsket å få med i produktet vårt var avstander til skilt og objekter. Da vi valgte denne oppgaven gikk vi løs på den i den tro at NVDB API inneholdt tilstrekkelig informasjon slik at dette skulle være en enkel oppgave. Dette viste seg raskt å ikke stemme – NVDB API inneholder ingen informasjon om avstander eller lengder på vegstrekninger.

Løsningen ble da å involvere et eksternt API for å håndtere avstander. Da vi først kom til denne løsningen brukte vi Google Maps. Vi fant snart ut at Google krevde at for å bruke APIet deres til å hente ut kjøreavstanden mellom to punkter, så måtte dette brukes sammen med Google Maps. Siden vi ikke gjorde dette, måtte vi finne et annet API for denne jobben Vi endte da med å bruke MapQuest.

Baksiden med denne løsningen er at vi nå må hente informasjon om avstanden asynkront med oppdateringen av skilt og objekter. Dette kan i noen tilfeller føre til at avstandene ikke oppdaterer seg så jevnt og korrekt som vi ønsker.

2.3.6.3 Kobling mellom veglenker

En annen informasjon vi også antok at var en del av NVDB API var referanser mellom veglenker[29], slik at vi alltid kunne sett en viss avstand fremover på vegen. Dessverre fungerer NVDB API slik at når man er på en veglenke, så kan man kun se objekter på den aktuelle veglenken. Dette innebærer at når man befinner seg på de siste hundre meterne av en veglenke, så vil man ikke få informasjon om kommende skilt og objekter, selv om de i noen tilfeller kan ligge noen titalls meter lengre frem.

[29] Et objekt i NVDB som representerer en sammenhengende vegstrekning med en unik ID.

Vi så en mulig løsning i å se på koordinatene til de tre siste punktene på veglenken, for deretter å beregne hvor vegen vil gå videre, slik at vi kunne hentet ut informasjon om veglenken på den beregnede posisjonen. Grunnet den gitte tidsrammen var dette dessverre ikke noe vi kunne prioritere å gjøre.

2.3.6.4 Manglende data i datasettet

Selv om NVDB API er et kraftig verktøy med mye informasjon, viste det seg etter hvert at det var sentrale data som rett og slett manglet for en del områder. For eksempel fant vi ingen forkjørsveger da vi lette i hele Oslo og Akershus. Vi kontaktet Statens vegvesen angående dette, og fikk til svar at mye av informasjonen i NVDB er det lokale avdelinger av Statens vegvesen som har ansvaret for å bidra med. Dette vil altså si at det ikke er NVDB API som har en feil i seg selv, men at Statens vegvesen fremdeles har en stor jobb med å ferdigstille datasettet i NVDB.

Statens vegvesen

Figur 17: Lokale avdelinger i Statens vegvesen er ansvarlige for å bidra med informasjon til NVDB.

Dette var selvsagt ikke et problem vi kunne gjøre noe med, og vi må derfor dessverre godta at applikasjonen ikke kan være bedre enn datasettet den er bygget på.

2.3.6.5 APIet er i demoversjon

Da vi tok på oss dette prosjektet i november/desember 2012 ble vi informert om at APIet foreløpig kun eksisterte som en lukket demoversjon, men at det ikke var noe problem å ta utgangspunkt i denne. Vi ble videre informert om at APIet skulle komme ut for offentligheten hos Statens vegvesen i løpet av desember.

Det viste seg dog at APIet ikke ble frigitt for offentligheten før i mars 2013. I mellomtiden hadde demoversjonen fått ny funksjonalitet for søk, funksjonalitet som vi valgte å benytte oss av fordi den gjorde det mye enklere å foreta søk i databasen.

Da APIet ble frigitt, og vi ble oppmerksomme på at dette var den første demoversjonen vi hadde sett, og ikke den vi nå jobbet mot, tok vi kontakt med BEKK og Statens vegvesen for å forhøre oss om når den versjonen vi jobbet mot ville settes ut i produksjon. Vi fikk da til svar at det burde skje innen en måned, men selv om vi nå er i slutten av mai, jobber vi fremdeles mot en demoversjon som ikke er offentlig tilgjengelig.

Dette har for det meste ikke vært til noe stort hinder for arbeidsprosessen, men satt en stopper for vårt ønske om å få publisert produktet i Apples App Store[30] innen prosjektets frist.

2.3.6.6 Automatisk parsing av JSON for søk

Da vi arbeidet med å automatisere søkeprosessen opprettet vi klasser i Objective-C som skulle settes sammen til en struktur lik den som benyttes for JSON[31]-baserte søk i NVDB API. Vi brukte innebygde funksjoner fra iOS SDK, men fikk ikke dette til å fungere. Vi forsøkt også å traversere søke-objektet og dets objektorienterte egenskaper manuelt, men det fungerte fremdeles ikke.

Etter grundig feilsøking viste det seg at vi hadde truffet på en ukjent bug i Objective-C. Da vi satte sammen klassene slik vi gjorde ble den ene klassen ubrukelig. Vi forsøkte mye frem og tilbake, men det lot seg ikke gjøre å opprette objekter av den aktuelle klassen. Vi fant ingen hjelp for dette på Internett, og vår veileder Christoffer Marcussen fra BEKK hadde aldri sett noe lignende.

Resultatet ble at vi laget en "workaround" hvor vi opprettet JSON-strengen manuelt. Dette ble noe mer tungvint med tanke på å utvide søkeobjektet, men når koden først var skrevet ble resultatet tilfredsstillende, ettersom JSON-strengen for søk ble generert korrekt slik den skulle.

2.3.6.7 Mange objekter langs vegen er ikke objekter i NVDB

Mange av objektene langs vegen vi ønsket å varsle om i applikasjonen eksisterer ikke som objekter i NVDB. Implementasjon av objekter som også er objekter i NVDB var en relativt triviell prosedyre, men vi måtte finne andre løsninger for å varsle om de øvrige objektene.

Figur 18: I applikasjonen er bl.a. "Farlig sving" tolket fra et skiltobjekt, og ikke et eget objekt i NVDB.

Løsningen her ble at vi måtte se på skiltplater i NVDB. Selv om mye ikke er lagt inn som objekter i NVDB er i hvert fall alle skiltplater registrert som skiltplate-objekter. Vi kunne således søke opp de skiltplatene vi hadde behov for og benytte denne informasjonen for å varsle brukere om de aktuelle objektene. Det kan for øvrig nevnes at for å få til dette var vi avhengige av den noe mer avanserte søkefunksjonaliteten som foreløpig ikke er offentlig tilgjengelig, som omtalt i kapittel 2.3.6.5, "APIet er i demoversjon".

2.3.6.8 Nedetid på APIet

Ettersom vi hele tiden jobbet mot en demoversjon av APIet hadde vi ingen garanti på at ting var oppe og kjørte hele tiden. Dette innebar at vi ved enkelte anledninger ble forsinket i produksjonsarbeidet da APIet kunne være utilgjengelig i opptil flere timer. I kortere perioder kunne

[30] Apple App Store er Apples distribusjonsplattform for applikasjoner til iOS.
[31] JavaScript Object Notation, en enkel tekstbasert standard for datautveksling.

vi også oppleve at APIet svarte ekstremt tregt, noe som gjorde at simulering av applikasjonen ble en langtrukken prosess.

Ettersom dette var utenfor vår kontroll var det ikke annet vi kunne gjøre enn å kontakte Frode Reinertsen hos BEKK for å høre om han kunne forbedre situasjonen.

2.3.6.9 Upresis bounding-box

Under testturen etter første sprint oppdaget vi at applikasjonen hadde problemer med å "holde seg på vegen". Da det var sideveger eller paralleltgående veger virket det nesten tilfeldig hvilken veg APIet trodde vi var på.

Dette var en problemstilling vi tok opp med Frode Reinertsen hos BEKK, og han utførte så forbedringer rundt dette i APIet. I tillegg kom det ny funksjonalitet i demoversjonen av APIet. Da vi utforsket og tok i bruk denne funksjonaliteten, samtidig som vi forbedret våre egne metoder for posisjon- og veglenkesøk, fikk vi økt presisjonen betraktelig.

Selv etter de ovennevnte forbedringene ser vi fremdeles at det er vanskelig for APIet å vite nøyaktig hvilken veg vi er på dersom det er flere veger innenfor en radius på noen få meter, eller dersom man kjører over/under en bro. Selv på vår siste testtur, som varte i ca. 2 timer, var dette et relativt sjeldent problem.

2.3.7 Apps4Norge

Apps4Norge var en konkurranse i regi av Difi og IKT-Norge. Konkurransen gikk ut på å lage en applikasjon som benytter eller er basert på åpne, offentlige data. Konkurransen ble arrangert for å få fokus på de mange nye offentlige datasettene, fra offentlige etater, som har blitt lansert i 2012/2013.

I starten av april ble vi tilfeldigvis tipset om konkurransen av Christoffer Marcussen i BEKK, og vi kikket litt på vilkår og regler for å delta. Da vi oppdaget at applikasjonen vår faktisk var midt i blinken for denne konkurransen, ettersom vi benytter åpne, offentlige data, bestemte vi oss for å melde oss på. Etter dette arbeidet vi videre som normalt, og beholdt fokuset på at prosjektet skulle gjennomføres etter de betingelser og vilkår vi hadde tatt sikte på fra starten av. Den eneste forandringen vi gjorde i prosjektet på

Figur 19: Kjørehjelperen vant 3. plass i lagkonkurransen i Apps4Norge (Foto: Cecilie Vaagen Smestad, Statens vegvesen).

grunn av konkurransen, var å fremskynde den siste testturen med noen få dager, slik at vi kunne få laget en god demovideo.

Da vi deltok på prisutdelingen 8. mai var overraskelsen derfor stor da vi gledelig nok ble tildelt 3. plass i kategorien for applikasjoner laget av grupper. I ettertid har applikasjonen vår og konkurransen blitt omtalt på nettsidene til bl.a. Statens vegvesen[32], Høgskolen i Oslo og Akershus[33], IKT-Norge[34] og Teknisk Ukeblad[35].

2.4 Dokumentasjonsfasen

Etter at produksjonsfasen var endt, og produktet var ferdig utviklet, var det klart for å dokumentere arbeidet. Vi satte av ca. en måned til denne fasen, og begynte på den mandag 29. april, med innleveringsfrist for prosjektet tirsdag 28. mai.

I likhet med arbeidet med styringsdokumentene, arbeidet vi også under dokumentasjonsfasen med dokumentasjonsstandarden[36] til Ann-Mari Torvatn som en veiledning, uten at den ble fulgt slavisk. Også her var dette med overlegg, ettersom det var aspekter vi følte burde omstruktureres, kuttes eller legges til. Også disse avvikene ble gjort i samråd med veileder Eva Vihovde.

[32] http://www.vegvesen.no/Om+Statens+vegvesen/Media/Siste+nyheter/Vis?key=472208
[33] http://www.hioa.no/Aktuelle-saker/Fikk-premie-for-kjoerehjelper
[34] http://ikt-norge.no/2013/05/vinnere_apps4norge/
[35] http://www.tu.no/it/2013/05/08/norges-mest-lovende-teknologier
[36] For detaljer om dokumentasjonsstandarden se kapittel 1.4.1.1, "Dokumentasjonsstandard".

3 Kravspesifikasjonen og dens rolle

I denne delen vil vi beskrive rollen kravspesifikasjonen har hatt under utviklingsarbeidet med produktet vårt. Vi vil også se nærmere på hvilke krav og rammer vi har klart å oppfylle, hvilke områder vi har avvik fra kravspesifikasjonen, samt hvilke utvidelser eller endringer vi har gjort i kravspesifikasjonen underveis.

3.1 Kravspesifikasjonens rolle

Kravspesifikasjonen skal fungere som et styringsdokument for hovedprosjektet. Den skal definere rammer og retningslinjer for prosjektet, utarbeides i samarbeid med oppdragsgiver, og sikre at både vi og oppdragsgiver får klarlagt hva betingelsene for prosjektet skal være. Dette gjelder både funksjonelle og ikke-funksjonelle krav, samt krav til arbeidsprosess og kvalitetssikring.

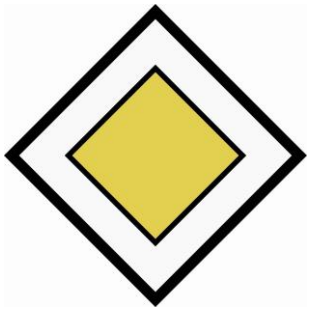
Kravspesifikasjonen skal opptre som en bindende avtale mellom oss og oppdragsgiver, men det skal også tas hensyn til krav fra Høgskolen i Oslo og Akershus. Kravspesifikasjonen, og alle revisjoner, skal derfor være godkjent av alle parter før det kan anses som gyldig.

3.2 Samsvar med kravspesifikasjonen

I dette kapittelet tar vi for oss alle kravene i kravspesifikasjonen, og diskuterer i hvilken grad disse har blitt oppfylt. Kravspesifikasjonen det refereres til her er den endelige kravspesifikasjonen, som er vedlagt som et eget dokument i denne oppgaven. Endringer som er gjort i kravspesifikasjonen underveis kan man lese om i kapittel 3.4.

3.2.1 Funksjonelle krav

Vi delte opp de funksjonelle kravene[37] i 3 deler: Prioritert funksjonalitet, ønsket funksjonalitet og eventuell tilleggsfunksjonalitet. Dette var for å gi en pekepinn på hvordan vi prioriterte de forskjellige forslagene til funksjonalitet. Underveis har funksjonalitet blitt omprioritert, flyttet og lagt til.



Figur 20: Dette skiltet vises på skjermen hvis brukeren kjører på en forkjørsveg.

Brukeren skal kunne se gjeldende fartsgrense for vegstrekningen

Så lenge det er registrert en fartsgrense på vegen brukeren befinner seg på i NVDB vises denne på skjermen. Fartsgrensen vises som et fartsgrenseskilt.

[37] Funksjonelle krav er krav til funksjonalitet som skal implementeres i systemet.

Brukeren skal kunne se hvorvidt han/hun er på en forkjørsveg eller ikke

Hvis brukeren er på en veg som er registrert som en forkjørsveg i NVDB vil det vises et forkjørsvegskilt på skjermen. Hvis vegen ikke er en forkjørsveg vil det derfor heller ikke vises noe forkjørsvegskilt. Manglende forkjørsvegskilt kan dog også skyldes manglende data i NVDB. Hvis brukeren er på en motorveg eller motortrafikkveg[38] vil disse skiltene vises i stedet for forkjørsvegskiltet.

Brukeren skal kunne se diverse gjeldende varselskilt

Brukeren blir presentert følgende fareskilt:

- Vilttrekk
- Farlig sving
- Smalere veg
- Steinsprut
- Farlig vegskulder
- Tunnel
- Trafikklyssignal
- Syklende
- Møtende trafikk
- Sidevind
- Annen fare
- Fartsdemper
- Farlige svinger
- Ujevn veg
- Rasfare
- Bevegelig bru
- Farlig vegkryss
- Avstand til gangfelt
- Ku
- Kø
- Skiløpere
- Jernbanekryssing
- Bratt bakke
- Vegarbeid
- Glatt kjørebane
- Kai, strand eller ferjeleie
- Rundkjøring
- Barn
- Sau
- Fly
- Ridende

Figur 21: Dette skiltet varsler om glatt kjørebane.

I tillegg blir brukeren presentert for:

- Høydebegrensning
- Videokontroll/-overvåkning
- Automatisk trafikkontroll
- Særlig ulykkesfare

Figur 22: Dette skiltet brukes til å varsle om nødtelefoner langs vegen. Det benyttes ikke i applikasjonen, men var inspirasjonen til skiltet som varsler om SOS-lommer.

Brukeren skal kunne endre hvilken informasjon som skal vises

Alle vegobjektene som støttes i applikasjonen kan deaktiveres i applikasjonens innstillingsmeny.

Brukeren skal kunne endre hvorvidt det brukes lyd for å varsle brukeren

Lydvarsling kan deaktiveres i applikasjonens innstillingsmeny.

Figur 23: Uoffisielt skilt som brukes til å varsle om SOS-lommer i applikasjonen.

Informasjonen på skjermen skal kunne speilvendes med det formål å lage et HUD i frontruten

Head-up display er implementert. En av/på-knapp speilvender informasjonen på skjermen og låser orienteringen til landskapsmodus.

Brukeren skal kunne se avstand til førstkommende rasteplass

Førstkommende rasteplass varsles med et rasteplasskilt og avstanden dit.

[38] Motorveg og motortrafikkveg er de nye navnene på henholdsvis motorveg klasse A og motorveg klasse B.

Brukeren skal kunne se avstand til førstkommende SOS-lomme

Førstkommende SOS-lomme varsles med et serviceskilt med bokstavene SOS. Dette er ikke et offisielt skilt. Avstanden dit står under eller ved siden av.

Brukeren skal kunne se avstand til nærmeste toalett (rasteplass eller bensinstasjon)

Førstkommende toalett varsles med et serviceskilt med bokstavene WC. Avstanden dit står under eller ved siden av. Det opplyses kun om toaletter forvaltet av Statens vegvesen, da NVDB ikke inneholder data om bensinstasjoner.

3.2.2 Ikke-funksjonelle krav

De ikke-funksjonelle kravene[39] er delt opp i produktkrav, prosesskrav og eksterne krav.

3.2.2.1 Produktkrav

Produktkrav omhandler krav til det ferdige produktet som ikke er en direkte funksjon. Dette omhandler krav til brukervennlighet, effektivitet og pålitelighet.

Applikasjonen skal være på norsk

Dette kravet er oppfylt.

Applikasjonen skal i så stor grad som mulig følge de 5 E'ene

Vi vil påstå at den gjør det[40]. Grunnet at applikasjonen kun består av ett skjermbilde, begrenses mulighetene til å bryte disse prinsippene betraktelig. Les mer om dette i produktdokumentasjonen, kapittel 5.3: "De fem E'ene".

Applikasjonen skal ikke kreve interaksjon under bruk (kjøring)

Første gang man starter applikasjonen må man godkjenne at den får bruke telefonens posisjonstjeneste, samt at man må godta en ansvarsfraskrivelse for bruk. Hvis telefonen ikke klarer å finne posisjonen under bruk kommer det dessuten opp en dialogboks[41] som informerer om dette. Utenom dette krever applikasjonen ingen interaksjon.

Applikasjonen skal begrense egen datatrafikk i så stor grad som mulig

Applikasjonen mellomlagrer data på enheten. Hvis applikasjonen ser at den er på en veg den har vært på før, og dataene ikke er eldre enn 30 dager, bruker den de mellomlagrede dataene. Dette sparer mye datatrafikk, siden enkelte vegstrekninger inneholder mye data.

[39] Ikke-funksjonelle krav er krav til systemets egenskaper og rammeverk.
[40] Prinsipper for brukervennlighet: effective, efficient, engaging, error tolerant og easy to learn (Stone, Jarret, Woodroffe, Minocha, "User Interface Design and Evaluation", 2005, Elsevier Inc.).
[41] Liten boks med informasjon som legger seg over skjermbildet. Man må trykke på ”OK” for at den skal forsvinne.

Figur 24: Applikasjonen skal begrense datatrafikk i så stor grad som mulig (Foto: Jollygoo.com).

Hvilken veg man befinner seg på er det NVDB som avgjør ved hjelp av telefonens GPS-koordinater. Det må derfor gjøres en spørring mot serveren for hver oppdatering. I tillegg må det spørres mot MapQuest for hver oppdatering for å beregne avstand til punktene lenger fremme. Det vil derfor brukes noe datatrafikk hele tiden, men mellomlagringen av data reduserer datatrafikken med 95%.

Applikasjonen skal være så energigjerrig som mulig

Det er ikke gjort noen egne tiltak for å begrense strømbruken. Telefonen sørger selv for å skru av tjenester når applikasjonen minimeres. Når den er i bruk benyttes den høyeste graden av nøyaktighet ved posisjonering. Dette krever en del strøm, men er nødvendig for at posisjonen skal være nøyaktig nok. Apple anbefaler at telefonen er koblet til en strømkilde når denne tjenesten benyttes. Vi anslår allikevel at applikasjonen kan kjøre i omtrent 5 timer på et fulladet batteri[42].

Applikasjonen skal begrense bruk av telefonens minne

Det er under implementasjonen forsøkt å designe applikasjonen på en måte som gjør at det ikke befinner seg store mengder data i minnet til enhver tid. Det brukes i stor grad arrays[43] og dictionaries[44] til å holde på informasjon istedenfor store objekter.

Med introduksjonen av ARC[45] i iOS 5[46] overføres dessuten minnehåndteringen til kompilatoren[47]. Vi trengte derfor ikke bekymre oss for minnelekkasjer[48] i noen stor grad.

Posisjonering av bruker skal være innenfor en nøyaktighet på 10 meter

Ved bruk av telefonens lokasjonstjeneste med høyeste nøyaktighet leverer den posisjoner med 5 meters nøyaktighet.

Posisjonering av objektdata skal være innenfor en nøyaktighet på 1 meter

Vi har ikke utført noen konkrete tester på hvor nøyaktig posisjonene i NVDB er, men ut i fra eget inntrykk er posisjonene langt mer nøyaktige enn 1 meter.

---

[42] Basert på egen testing på en iPhone 4S.
[43] Et array er en endimensjonal tabell.
[44] En dictionary er en datastruktur bestående av nøkkel- og verdipar.
[45] Automatic Reference Counting (ARC) er at ansvaret for minnehåndtering flyttes fra programmereren til kompilatoren.
[46] iOS er Apples operativsystem for iPhone, iPad og iPod.
[47] En kompilator er et dataprogram som oversetter et dataprogram fra kildekode til maskinkode.
[48] Feil ved minnehåndteringen i programmet.

Dataene som presenteres skal være oppdaterte

Hvis de mellomlagrede dataene på telefonen er eldre enn 30 dager, oppdateres dataene. Hvorvidt dataene i NVDB er oppdatert eller ikke har vi ingen kontroll over.

3.2.2.2 Prosesskrav

Prosesskrav omhandler krav til prosessen rundt utviklingen av produktet, deriblant metodikk, leveranser, implementasjon og standarder.

Applikasjonen skal være kjørbar på iPhone 3GS, 4, 4S og 5 (oppdatert til iOS 6.1)

Ved å utvikle applikasjonen til iOS 6.1 vil den være kjørbar på alle enheter med dette operativsystemet. iPhone 3GS og nyere støtter iOS 6.x.

Figur 25: Applikasjonen skal kunne kjøre på en iPhone 3GS eller nyere.

Kildekoden for leveransen skal lisensieres under GNU GPL[49] og være tilgjengelig via GitHub

Kildekoden er lisensiert under GLP 3 og er tilgjengelig på https://github.com/lsmeby/Vegdata.

Applikasjonen skal utvikles for iOS 6.1

Applikasjonen skal utvikles med Xcode 4.6 på OS X

Objective-C benyttes som programmeringsspråk

Applikasjonen skal benytte Cocoa Touch og de andre rammeverkene i iOS 6.1 SDK

Disse kravene er selvskrevne og kan ikke omgås. Ved utvikling av applikasjoner til iOS tvinges du til å bruke Xcode på OS X, der du igjen må bruke overnevnte rammeverk og programmeringsspråk. Vi brukte nyeste versjon av Xcode og iOS SDK, som på det tidspunktet var henholdsvis 4.6 og 6.1.

Applikasjonen skal testes på så mange fysiske iOS-enheter som mulig

Per i dag er applikasjonen kun testet på en iPhone 4S i tillegg til simulatoren i Xcode. Det planlegges testing på flere enheter før endelig lansering i App Store.

Applikasjonen skal benytte det åpne biblioteket RestKit for kommunikasjon og parsing av data med NVDB APIet

Applikasjonen benytter RestKit. Se kapittel 3.2 i produktdokumentasjonen for mer informasjon om RestKit.

[49] GNU General Public License (GNU GPL eller GPL) er en lisens som tillater fri bruk, kopiering og endring av koden.

Brukerpreferanser skal lagres som "User preferences"

Vegdata skal lagres som Core Data

Disse kravene er innfridd. Se "3.1 Core Data" og "4.2.12 Settings" i produktdokumentasjonen for detaljer rundt lagring av data.

Det skal benyttes JSON for kommunikasjon med NVDB API

Datautveksling med både NVDB og MapQuest kan gjøres med JSON og XML. Kjørehjelperen benytter JSON. Les mer om dette i første kapittel i produktdokumentasjonen.

Applikasjonen skal benytte MVC-patternet (som forventes i iOS)

Strukturen i et iOS-prosjekt er i utgangspunktet basert på MVC-patternet. Vi har utvidet dette ytterligere ved å gjøre en enda tydeligere lagdeling. Les mer om dette i produktdokumentasjonen, kapittel 2.

3.2.2.3 Eksterne krav

Eksterne krav omhandler krav som ikke passer inn under produkt- eller prosesskrav, deriblant estetiske og lovmessige krav.

Applikasjonen skal være i samsvar med Apples retningslinjer for design på iOS

Retningslinjene[50] er til dels subjektive, men det er forsøkt å følge disse så godt som mulig. Les mer om dette i produktdokumentasjonen under kapittel 5.2.

Dataene som presenteres skal være store nok til å kunne sees på 1 meters avstand

Dataene er godt synlige for en person med normalt syn på 1 meters avstand.

Det skal i så stor grad som mulig benyttes ikoner og grafikk fremfor tekst

Det benyttes skilt til å varsle om ulike farer og hendelser. Avstander må naturlig nok oppgis som tekst.

Designet skal følge de 7 designprinsippene

Vi mener designet følger prinsippene[51]. Les en vurdering av dette i kapittel 5.1 i produktdokumentasjonen.

[50] https://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html

[51] Structure, simplicity, consistency, tolerance, visibility, affordance og feedback (Stone, Jarret, Woodroffe, Minocha, "User Interface Design and Evaluation", 2005, Elsevier Inc.).

Applikasjonen skal ikke kreve interaksjon under kjøring som strider mot norsk vegtrafikklov

Applikasjonen krever i utgangspunktet ingen interaksjon under kjøring. Les mer om dette i testdokumentasjonen, kapittel 1.

Applikasjonen skal inneholde teksten: "Inneholder data under norsk lisens for offentlige data (NLOD) tilgjengeliggjort av Statens vegvesen."

Applikasjonen skal opplyse om, og avskrive seg selv og Statens vegvesen ansvar for, eventuelle feil og mangler som måtte forekomme i NVDB APIet

Ved første gangs oppstart vises overnevnte tekst, en link til MapQuest og en kort ansvarsfraskrivelse. Ansvarsfraskrivelsen gjelder både for Kjørehjelperen selv, samt data fra NVDB og MapQuest.

Figur 26: Denne meldingen vises første gang Kjørehjelperen starter.

3.3 Avvik fra kravspesifikasjonen

Vi vil her ta for oss de krav som vi ønsket å oppfylle, men som av diverse grunner ikke har blitt oppfylt. Vi vil også forklare kort hvorfor det aktuelle kravet ikke ble oppfylt.

Brukeren skal kunne endre hvor tidlig varsler skal vises

Dette kravet ble vanskelig å innfri. Grunnene for dette handler dels om det omtalt i kapittel 2.3.6.1, "Brukerinnstillinger", men i hovedsak handler det om utfordringen omtalt i kapittel 2.3.6.3, "Kobling mellom veglenker".

Den ferdige applikasjonen skal være godkjent og tilgjengelig i Apples App Store

Det er to vesentlige grunner til at dette kravet ikke har blitt innfridd. Den mest konkrete grunnen er at NVDB API enda ikke er offentlig tilgjengelig i den versjonen vi har arbeidet mot. Les mer om dette i "2.3.6.5 APIet er i demoversjon".

Den andre grunnen er at selv om vi er godt fornøyde med arbeidet vi har gjort, og synes vi har arbeidet godt gjennom hele prosessen, så er vi begge av den oppfatning at applikasjonen har behov for litt finpussing før vi stolte kan lansere den for verden. Les mer om dette i kapittel 4.3, "Videre utvikling".

3.4 Endringer i kravspesifikasjonen

Selv om noen avvik fra kravspesifikasjonen ikke er til å unngå, har vi til en viss grad forsøkt dette ved å revidere kravspesifikasjonen underveis. Disse endringene ønsker vi også at skal være omtalt i denne delen av dokumentasjonen.

Brukeren skal kunne se avstand til neste mulige forbikjøring

Denne funksjonaliteten ble droppet både på grunn av dårlig datagrunnlag i NVDB, og at det datagrunnlaget som fantes gjaldt noen få skiltede forbikjøringsfelt. Vi følte at dette ikke ville gi noe særlig nytteverdi til brukeren.

Brukeren skal kunne se hva klokka er

Dette var funksjonalitet vi anså som relativt lite viktig, ettersom de fleste har andre klokker tilgjengelig i bilen, og kravet frafalt derfor underveis.

Brukeren skal kunne se et varsel ved overtredelse av fartsgrense

Dette kravet ble fjernet fordi det var funksjonalitet vi anså som relativt lite viktig. Vi ønsket å prioritere fokuset vårt mot den funksjonaliteten vi anså som mest relevant slik at vi kunne spisse inn applikasjonen mot å være en skilt-påminnelses-applikasjon. Vi ville dessuten bruke all tilgjengelig plass på skjermen til skilt.

Brukeren skal kunne se kommende bomstasjoner på vegen

Dette kravet ble i hovedsak frafalt fordi det ville være lite hensiktsmessig for brukere av applikasjonen å bli varslet om dette. Som beskrevet i 2.3.6.3, "Koblinger mellom veglenker", var det vanskelig å kunne se veldig langt fremover på en vegstrekning. Ettersom vi da ikke kunne varslet sjåføren før det var et par kilometer igjen til bomstasjonen, følte vi at dette var noe sent å varsle om en bomstasjon.

Det skal være mulig å få opp utvidet informasjon om fasiliteter ved rasteplasser

Dette kravet ble droppet av samme grunn som vi droppet kravet om varsel ved overtredelse av fartsgrense: denne funksjonaliteten var noe avvikende fra kjernen i applikasjonen, og var derfor påtenkt som en mulig utvidelse dersom vi fikk god tid.

Brukeren skal kunne se avstand til førstkommende bensinstasjon

Da vi formulerte dette kravet var det en ren teoretisk mulighet dersom vi fikk veldig god tid. Ettersom bensinstasjoner ikke er en del av datasettet i NVDB ville dette krevd innhenting av data fra andre hold, og ville derfor vært en større oppgave å utføre. Kravet ble derfor fjernet.

Varsler skal kunne dikteres for brukeren

Dette kravet omhandlet funksjonalitet som vi anså som en spennende mulighet. Allikevel var vi fullt klar over, da vi formulerte det, at det ville vært en stor jobb. Kravet ble derfor skrevet ned for å sikre at vi hadde tenkt ut nok funksjonalitet til å unngå at vi fikk for lite å gjøre i prosjektet. Det ble etter hvert tydelig at vi hadde planlagt veldig mye arbeid for oss selv, og dette kravet ble derfor fjernet.

Det skal leveres en kjørbar applikasjon ved hver leveranse

Det ble gjort en endring i detaljene til dette kravet: Den ene leveransedatoen ble endret fra 12.04 til 05.04, dette valgte vi fordi vi så et behov for å også jobbe i påskeferien, for å få mer tid til utviklingen.

4 Avsluttende del

Etter et langt, utfordrende og spennende prosjekt har vi her kommet til slutten på dette eventyret, så i denne delen vil vi gjøre oss noen tanker og oppsummeringer som vil være naturlige å gjøre ved slutten av et prosjekt.

4.1 Vurderinger av data og teknologi

I prosjektet har vi benyttet nye verktøy og data, som vi hadde som et mål at vi skulle vurdere.

4.1.1 NVDB

NVDB inneholder mye data, noe som gir mange muligheter til implementasjon og bruk av APIet. Noe vi riktig nok savner i APIet er avstander og lengder på veglenker. I Kjørehjelperen har vi måttet ty til en løsning der vi benytter MapQuest for å finne avstanden til kommende objekter. Hvis NVDB hadde oppgitt lengden på veglenkene, hadde vi både fått en smidigere og mer nøyaktig avstandsangivelse i applikasjonen.

En annen ting vi savnet i NDVB API var en sammenheng mellom veglenkene. Slik NVDB fungerer i dag finnes det ingen direkte link mellom veglenker som henger sammen, og vi kan derfor ikke på noen enkel måte forutse hva som dukker opp på kommende veglenker.

Selv om NVDB inneholder over 250 objekttyper, skulle vi gjerne sett at det var noen flere. Flesteparten av objektene langs vegen som vi benytter i Kjørehjelperen eksisterer ikke som egne objekter i NVDB, men som et skiltplate-objekt som vi måtte tolke og konvertere til egne objekter. Dette er en løsning som fungerer for øyeblikket, men er noe svak siden den utelukkende baserer seg på sammenligning av tekststrenger.

4.1.2 iOS

Etter at vi begge tidligere har jobbet med både Android[52] og Windows Phone[53], var det en spennende utfordring å jobbe med iOS. Vi ser flere likheter mellom plattformene i hvordan prosjektet struktureres, men også vesentlige ulikheter, da spesielt syntaksen i Objective-C.

iOS og Cocoa Touch-rammeverket bygger på gamle C-klasser og kan derfor oppleves som noe utdatert og ustrukturerte. For øvrig ser vi at Apple har lagt mye arbeid i å bygge et solid SDK som

[52] Android er et operativsystem til mobile enheter basert på Java. Android utvikles av Google.
[53] Windows Phone er Microsofts operativsystem til smarttelefoner.

enkelt gir tilgang til telefonens hardware og egenskaper. Vi var også godt fornøyd med delegater[54], og hvordan dette benyttes for å kommunisere asynkront innad i applikasjonen.

4.2 Nytteverdi

I dette delkapittelet vil vi se nærmere på hvilken nytteverdi prosjektet og produktet vil ha for sentrale parter, utenom oss som har utført arbeidet. Vi vil se nærmere på vårt eget utbytte av dette i kapittel 4.3.

4.2.1 For brukere av produktet

Brukere av produktet er helt klart den parten i dette prosjektet som vil ha størst nytteverdi av produktet i ettertid. Brukere har ikke vært en direkte interessent i prosjektet, men ettersom de er målgruppen må de også være å anse som en sentrale.

I retrospekt, føler vi at vi har produsert et godt produkt der vi har oppnådd vårt mål om et allment nyttig produkt. Anerkjennelsen vi fikk gjennom vår plassering i konkurransen Apps4Norge utgjør også en ekstra trygghet om at vi har utviklet et solid og nyttig produkt. Vi velger derfor å konkludere med at brukere av produktet vil kunne ha stor nytte av det i ettertid av dette prosjektet.

4.2.2 For oppdragsgiver

Ettersom oppdragsgiver tradisjonelt ikke produserer kommersielle produkter av denne typen vil ikke oppdragsgiver ha noen nytteverdi av produktet i ettertid. Dette har heller ikke vært noe mål for dem, og må derfor ikke anses som noe negativt.

Derimot kan selve prosessen rundt prosjektet ha en viss nytteverdi for oppdragsgiver i ettertid. Vi antar ikke at BEKK som firma vil kunne dra spesiell nytte av dette prosjektet, men det vil være rimelig å anta at erfaringen for de involverte personene hos BEKK kan være nyttig å ta med seg videre. Vi håper også at denne dokumentasjonen kan være til nytte for oppdragsgiver ved en senere anledning.

BEKK har riktignok fått både oppmerksomhet rundt og en nyttig testing av APIet de har utviklet for Statens vegvesen, noe som utvilsomt er noe de kan ta med seg videre.

4.2.3 For kunde

Til forskjell fra mange andre oppgaver har ikke denne handlet om å utvikle et produkt som skal brukes av kunden, altså Statens vegvesen. Derimot har denne oppgaven handlet om å synliggjøre

[54] En delegat i Objective-C er en klasse som implementerer en gitt protokoll. En annen klasse kan kalle denne delegaten uten å vite noe om hvilken klasse den kaller. Les mer om delegater i produktdokumentasjonen, kapittel 4.1.4.

mulighetene i et allerede eksisterende produkt, NVDB API. I lys av disse forutsetningene kan vi så vurdere nytteverdien dette produktet vil ha for Statens vegvesen i ettertid.

For å vurdere nytteverdien av produktet for kunden vil vi se på de samme argumentene som vi så på under vurderingen av nytteverdien for brukere. For det første føler vi selv at vi har utviklet et solid produkt. I tillegg kan vi med sikkerhet si at anerkjennelsen vi fikk gjennom vår plassering i Apps4Norge utgjør en god publisitet for Statens vegvesen og deres API med åpne data, NVDB. De har også fått erfaringer og tilbakemelding på hvordan det er å bruke APIet deres. Vi vil derfor konkludere med at kunden i ettertid vil ha god nytteverdi av produktet vi har utviklet og publisiteten det skaper for NVDB.

4.3 Videre utvikling

På kort sikt er det ikke store endringer vi ser for oss at det er behov for i applikasjonen. Vi er godt fornøyd med funksjonalitet og stabilitet, men ønsker allikevel å utføre noen endringer før vi vil anse produktet nå som en god førsteversjon, med tanke på lansering. I det vi skriver denne dokumentasjonen benytter applikasjonen vår fremdeles en demoversjon av APIet, og det kan derfor ikke publiseres for allmennheten i Apples App Store. Når denne demoversjonen blir satt i produksjon har vi som mål å finpusse på applikasjonen og få den publisert. Les mer om dette i produktdokumentasjonen, kapittel 6: "Gjenstående arbeid".

4.4 Læringsutbytte

I løpet av dette semesteret og dette prosjektet føler vi begge at vi har fått kunnskap og erfaring innen flere områder. Vi ser at prosessen har utviklet oss, og modnet oss med tanke på å være forberedt på arbeidslivet. Ikke bare har dette vært et omfattende prosjekt som har tatt tid å gjennomføre, men vi har også hatt et ansvar med å sette mål og rammer for prosjektet. Det har også vært en viktig prosess med å planlegge og strukturere tiden, der vi har sett hvor utfordrende det kan være å ha frister å overholde og leveranser man skal rekke. Sist, men ikke minst, har det også vært en verdifull erfaring å ha en konkret oppdragsgiver fra arbeidslivet, noe som har gitt oss god innsikt i hvordan et prosjekt gjennomføres i "den virkelige verden".

I tillegg til den viktige erfaringen med prosjektarbeid og prosjektstyring, har vi også fått et stort faglig læringsutbytte av dette prosjektet. Som nevnt i "1.1 Arbeidsforhold og samarbeid" ønsket vi begge "å utvide kunnskapsplattformen vår" samt å "utfordre oss selv i hovedprosjektet". Dette var altså et bevisst valg fordi vi ønsket at hovedprosjektet skulle være noe mer enn å bare gjøre noe vi allerede hadde lært på studiet; vi ønsket å lære mer, og det gjorde vi ved å lære oss et nytt programmeringsspråk, et nytt rammeverk, et nytt utviklingsmiljø og et nytt operativsystem! Vi er

derfor stolte av å se tilbake på dette prosjektet der vi ikke bare utviklet et produkt vi er stolte av, men også tilegnet oss mye ny faglig kunnskap som vil gi oss tyngde og bredde når vi skal ut i arbeidslivet.

4.5 Konklusjon

Vi har nå gjennomført et omfattende prosjekt der vi både har fått erfare hvordan slikt skal gjennomføres profesjonelt i arbeidslivet, men også tilegnet oss mye ny kunnskap og erfaring. Vi har utfordret oss selv, både faglig og med tanke på omfang, og kommet frem til et solid og anerkjent produkt. Derfor er vi veldig fornøyde når vi nå har kommet til vegs ende, og vi kan se tilbake på prosjektet.

Når vi nå kan reflektere over denne prosessen i retrospekt så ser vi at dette ikke bare har vært et semester og et prosjekt, men heller en kulminasjon av en tre år lang prosess med læring. Grunnen til at vi ser på det slik er at vi her har fått bruk for alt vi har lært underveis i studiet, både programmeringsferdigheter, evnen til å lære seg nye språk, prosjektplanlegging, visuelt design, interaksjonsdesign, og mye, mye mer.

På bakgrunn av graden av måloppnåelse, produktets anerkjennelse, samt vår egen tilfredshet med både prosess, produkt og egen utvikling, vil vi si at prosjektet har vært svært vellykket.