

Avstand

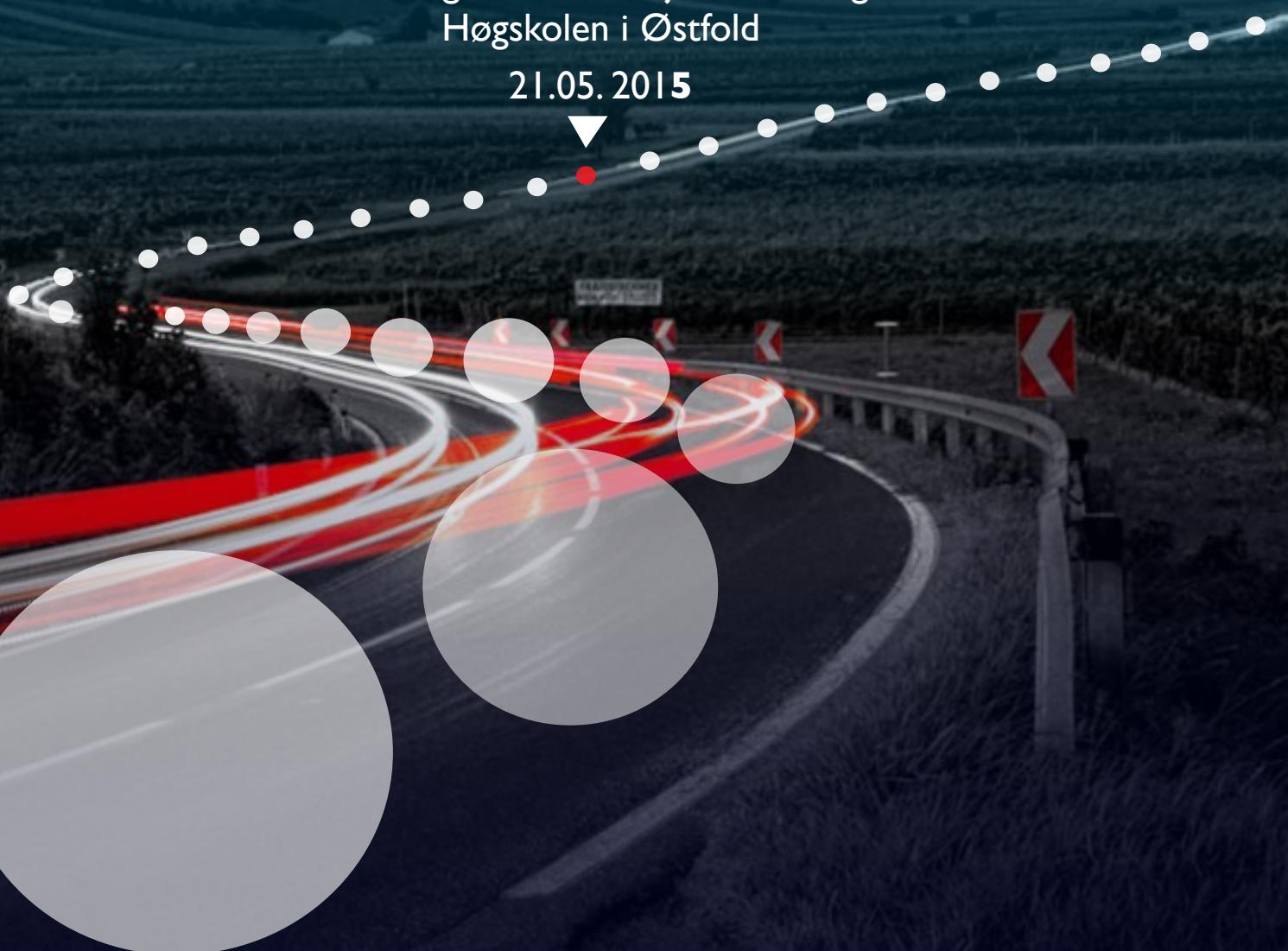
og bompengeberegningsmodul til HRessurs

Bacheloroppgave:

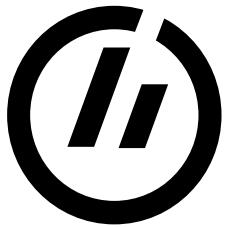
Avdeling for informasjonsteknologi

Høgskolen i Østfold

21.05.2015



• Gruppe BO-G02 • Robin Furu • Ingvild Karlsen Bjørlo • Christian Jacobsen • Glenn Bjørlo



HØGSKOLEN I ØSTFOLD

Avdeling for Informasjonsteknologi
Remmen
1757 Halden
Telefon: 69 21 50 00
URL: www.hiof.no

BACHELOROPPGAVE

Prosjektkategori: Bachelorprosjekt	<input checked="" type="checkbox"/> Fritt tilgjengelig
Omgang i studiepoeng: 20	<input type="checkbox"/> Fritt tilgjengelig etter
Fagområde: Informasjonsteknologi	<input type="checkbox"/> Tilgjengelig etter avtale med oppdragsgiver

Tittel: Avstand og bompengeberegningsmodul til HRessurs	Dato: 21.mai 2015
Forfattere: Robin Furu, Ingvild Karlsen Bjørlo, Christian Jacobsen, Glenn Bjørlo	Veileder: Terje Samuelsen
Avdeling / Program: Avdeling for Informasjonsteknologi	Gruppenummer: BO15-G02
Oppdragsgiver: Infotjenester AS	Kontaktperson hos oppdragsgiver: Petter Ekrann

Ekstrakt:

Infotjenester AS ønsker å utvide deres personalsystem HRessurs med en mulighet for enklere og raskere utfylling av reiseregninger. I dag er man avhengig av å huske hvor mange bomstasjoner man passerer, prisen på hver av disse og avstanden på reisen. Vi skal utvikle en webapplikasjon som gjør utfylling av reiseregninger mer brukervennlig ved at det skal være mulig å beregne både avstand og bompengekostnader direkte i HRessurs. For å utføre disse beregningene skal brukeren av HRessurs kun behøve å angi start, stopp og eventuelle viapunkter på reisen. For at brukeren av HRessurs skal kunne kontrollere at riktig rute er beregnet, skal vi fremstille den beregnede kjøreruta både tekstlig og i et kart. Applikasjonen er utviklet i et .NET rammeverk.

3 emneord:

HRessurs
Reiseregning
Webutvikling

Sammendrag

Infotjenester er et IT/konsulent firma som holder til på Grålum i Sarpsborg. De har utviklet personalsystemet HRessurs som består av HRessurs Sykefraværsoppfølging, HRessurs Reise og Utlegg og HRessurs Personal og Ledersystem. HRessurs Reise og Utlegg benyttes i forbindelse med føring av reiseregninger, og vi har fått i oppdrag å utvikle en tilleggsfunksjonalitet som gjør utfylling av reiseregninger både enklere og raskere.

Infotjenester har fått tilbakemeldinger på at dagens system er lite brukervennlig når det gjelder utfylling av reiseregninger. I dag må man selv beregne avstand og bompengekostnader i forbindelse med reiser, og disse må fylles inn i reiseregningene manuelt. Istedenfor at brukere av HRessurs skal måtte ta i bruk eksterne programmer for å beregne avstand og bompengekostnader, har Infotjenester et ønske om å kunne foreta alle disse beregningene direkte i HRessurs. De ønsker et enkelt system der bruker taster inn start og stopp for reisen, i tillegg til eventuelle viapunkter. Systemet skal så beregne både avstand og bompengekostnader i tillegg til å generere en kort beskrivelse av kjøreruta. Når brukeren bekrefter at kjøreruta som er beregnet er korrekt, skal de beregnede verdiene automatisk fylles inn i riktige tekstfelt i reiseregningen.

For å gjennomføre prosjektet har vi studert andre tjenester som beregner kjøreruter, for å se hva de forskjellige tjenestene tilbyr, og for å finne ut hvilken funksjonalitet vi bør ha med i vår applikasjon. Statens Vegvesen har et gratis API, Ruteplantjenesten, og det er dette vi har utviklet applikasjonen vår rundt. API-et inneholder all informasjon vi trenger i forbindelse med beregning av avstander og bompengekostnader.

Vi har utviklet en frittstående applikasjon som er under implementering i HRessurs. Vi har bygd opp applikasjonen etter Infotjenesters ønske om en lagdelt struktur, slik at både vedlikehold og videreutvikling av applikasjon blir enklere. Vi har testet mange ulike metoder for å komme fram til den beste måten å få til en enkel og brukervennlig applikasjon.

Da vi hadde utviklet en applikasjon vi var fornøyd med, fortok vi en brukertest av den, slik at vi kunne få objektive tilbakemeldinger på eventuelle feil eller mangler. Applikasjonen ble testet av 15 personer med varierende datakunnskaper og ulik erfaring med føring av reiseregninger. Testpersonene oppdaget flere småfeil og mangler vi selv ikke hadde sett fra et utviklikerperspektiv. Vi har utbedret applikasjonen på bakgrunn av tilbakemeldingene, og dette har gjort applikasjonen mer brukervennlig.

I sluttfasen av prosjektet var vi med på første del av implementeringen av applikasjonen i HRessurs. Implementasjonen er godt i gang, og vil bli slutført i løpet av kort tid.

Takk til

Vi ønsker å takke vår oppdragsgiver Infotjenester AS for et spennende og lærerikt bachelorprosjekt. Spesielt ønsker vi å rette en takk til vår kontaktperson Petter Ekrann for god veiledning og oppfølging. Vi ønsker også å takke Per Bissegberg ved Høgskolen i Østfold for gode innspill til hvordan vi kunne bygge opp vår lagdelte applikasjon, i tillegg til vår veileder Terje Samuelsen for god oppfølging og konstruktive tilbakemeldinger på de tidligere versjonene av hovedrapporten.

Vi ønsker å takke alle andre som har bidratt underveis med utviklingen av prosjektet vårt, blant annet alle som tok seg tid til å teste applikasjonen og kom med gode tilbakemeldinger slik at vi kunne levere et best mulig resultat.

Innhold

Sammendrag	1
Takk Til	3
Figurliste	9
1 Introduksjon	1
1.1 Prosjektgruppen	1
1.2 Oppdragsgiver	2
1.3 Oppdraget	4
1.4 Formål, arbeidsmetode og Leveranser	6
1.5 Rapportstruktur	10
1.6 Terminologi	11
2 Bakgrunnsanalyse	13
2.1 Eksisterende løsninger samt regler og hensyn.	13
2.2 Tilgjengelig teknologi for å løse oppgaven	22
3 Veien mot det endelige resultatet	24
3.1 Brukergrensesnitt	24
3.2 Teknisk	34
3.3 Publisering på skolens server	42
3.4 SCRUM - utførelse	43
3.5 Programmer, verktøy og hjelpe midler	44
4 Brukertest av applikasjonen og implementering i HRessurs	46
4.1 Planlegging av brukertesting	46
4.2 Analyse av resultater fra brukerevalueringen	47
4.3 Refleksjon rundt unøyaktigheter i Ruteplantjenesten	51
4.4 Utbedringer og endringer etter brukerevalueringen	58
4.5 Implementering	61
5 Produktet	65
5.1 Produktdokumentasjon	65
5.2 Hvordan kan applikasjonen videreutvikles?	72
6 Diskusjon	74
7 Konklusjon	78

Figurer

1.1	Oversikt over Infotjenesters produkter	3
1.2	Slik fungerer HRessurs i dag	4
1.3	Slik ser vi for oss at HRessurs kan forbedres	5
1.4	Arbeidsflyt ved bruk av Scrum	7
1.5	Grafisk fremvisning av tenkte metoder.	8
1.6	Gantdiagram - Planlagt fremdrift	9
2.1	Meny i form av en pil	13
2.2	Navigering fram og tilbake	13
2.3	Skjermdump fra HRessurs - føring av kjørerute	14
2.4	Registrering av utlegg	15
2.5	Bomstasjoner i Sør-Norge	16
2.6	Sammenligning av eksisterende ruteplanleggingstjenester	18
2.7	Kartutsnitt: Google t.h, 1881 i midten, Ruteplantjenesten t.v	19
3.1	Tidlig skisse på brukergrensesnitt	24
3.2	Inndata fra bruker	25
3.3	Visning av resultater med faner	26
3.4	Første fane - kart	26
3.5	Andre fane - resultat	27
3.6	Tredje fane - kjørerute	27
3.7	Fjerde fane - Bomstasjoner	28
3.8	Det helhetlige brukergrensesnittet	28
3.9	Tekslig beskrivelse av kjøreruta.	29
3.10	Gruppert, tekslig beskrivelse av kjøreruta.	30
3.11	Halden - Kirkenes, to ulike veivalg	31
3.12	Koordinatfeil. Google viapunkter t.v, Polyline t.h	32
3.13	Sarpsborg - Oslo, fremstilt med Google viapunkter	33
3.14	Halden - Oslo, fremstilt som Polyline	33
3.15	Vedlikeholdbar struktur	34
3.16	Fordeler ved lagdeling	35
3.17	Kommunikasjon klient - server	36
3.18	Visning av viapunkter	37
3.19	Viapunkter med et inputfelt og sorterbar liste	39
3.20	Omfattende konvertering	40
4.1	Resultat fra brukertest. Grafisk fremstilling av testpersonenes bakgrunn	47
4.2	Overlapp i viapunktliste	48
4.3	Grå felter i kartvisning	49
4.4	Fremstilling av kjørerute med standard zoomnivå	50
4.5	Lite forklarende knapper i forbindelse med viapunkter	50
4.6	Raskeste vei Halden - Oslo	51

4.7	Korteste vei Halden - Oslo	52
4.8	Turistvei Halden - Oslo	52
4.9	Snarvei gjennom boligstrøk i Halden	53
4.10	Kjørerute følger hovedvei utenom boligstrøk	53
4.11	Ruteplantjenesten viser kjørerute over gangbro	54
4.12	Slik er korteste, lovlige kjørerute	54
4.13	Tresfjord - Stordal over vinterstengt vei	56
4.14	Øverst: viapunkter før testing. Nederst: viapunkter etter utbedring	58
4.15	Listeelement som endrer farge	59
4.16	Automatisk zoomnivå på strekningen Halden - Sarpsborg	59
4.17	Applikasjonen etter utbedring	60
4.18	Skjermbilde av HRessurs med knapp for å åpne vår applikasjon	61
4.19	Skisse av hvordan applikasjonen vil se ut i HRessurs	62
4.20	Interface basert på misforståelse	63
4.21	Skjermbilde av HRessurs med knapp for å åpne vår applikasjon	64
5.1	Klassen TravelRoute der beregninger mot Ruteplantjenesten foregår	67
5.2	Rekkefølgen funksjonene i TravelRoute kjøres	67
5.3	Dataflyten fra klientside, til serverside og tilbake	68
5.4	Klassen Route	69
5.5	TravelRoute implementerer ITravelRoute	69
5.6	CalculatedRoute, objektet som returneres til klientside	70

1. Introduksjon

I dette kapittelet vil vi gi en kort introduksjon om hvem vi er, oppdragsgiver, vårt oppdrag og planlagt arbeidsmetode. Rapportstruktur finnes i kapittel 1.5, og i kapittel 1.6 finnes en beskrivelse av viktige ord og uttrykk vi benytter i rapporten.

1.1 Prosjektgruppen

Glenn og Christian jobbet for Infotjenester høsten 2014 i forbindelse med faget Bedriftspraksis. De fikk en forespørsel om å fortsette ved bedriften i forbindelse med bacheloroppgaven. Dette takket de ja til, og tok så med seg Ingvild og Robin på gruppa. Christian, Ingvild og Robin har tidligere jobbet sammen i labgrupper i faget Industriell IT, og på et eksamensprosjekt i faget Integrerte IT-Systemer, høsten 2014. Her jobbet de med oppkoblingen av passivhuset «Villa Mjølnerød», i regi av Bolig Enøk. Dette gikk ut på å programmere smarthuskomponenter som skulle brukes i huset

Ingvild Karlsen Bjørlo

ingvildkb@hotmail.com - tlf: 90204584. Ingvild bor i Halden, hvor hun også er oppvokst. Hun gikk studiespesialisering på Halden VGS, med et utvekslingsår i Tyskland. Hun jobbet i et år før hun søkte seg til dataingeniørstudiet. Hun har tidligere jobbet 2,5 år på Svinesund Infosenter, som i tillegg til turistinformasjon, fungerer som servicestasjon for bompengeselskapet Svinesunds- forbindelsen. De faglige interessene retter seg mot programmering og dokumentasjon

Robin Furu

robin.furu@hotmail.com - tlf: 41524376. Robin er oppvokst i Fredrikstad. Han gikk idrettslinjen på videregående, før han var ett år i Hans Majestet Kongens Garde. Rett etter førstegangstjenesten begynte han på dataingeniørstudiet. Han har gjennom studiene blitt veldig interessert i HMI/- automatisering og Industriell IT, en mer praktisk tilnærming til IT.

Christian Jacobson

christian.jacobson@hotmail.no - tlf: 41758575. Christian er født og oppvokst i Fredrikstad og bor på Gressvik. Han har tidligere erfaring som teknikker i telekomfaget hvor han jobbet i YIT Building systems i 2 år. Han har fagbrev i telekommunikasjon. Ved siden av studiene jobber han på Expert, og har foto som en stor interesse.

Glenn Bjørlo

g-amb@hotmail.com - 47384852. Glenn er oppvokst i Askim og har bodd i Halden siden høsten 2014. Han ble uteksaminert fra Askim VGS i 2005 og flyttet deretter til Lillehammer hvor han studerte film- og fjernsynsvitenskap. Senere flyttet han til Sarpsborg og begynte studiet DMpro i Halden. Han har nylig kommet hjem fra ett utvekslingsår i Australia.

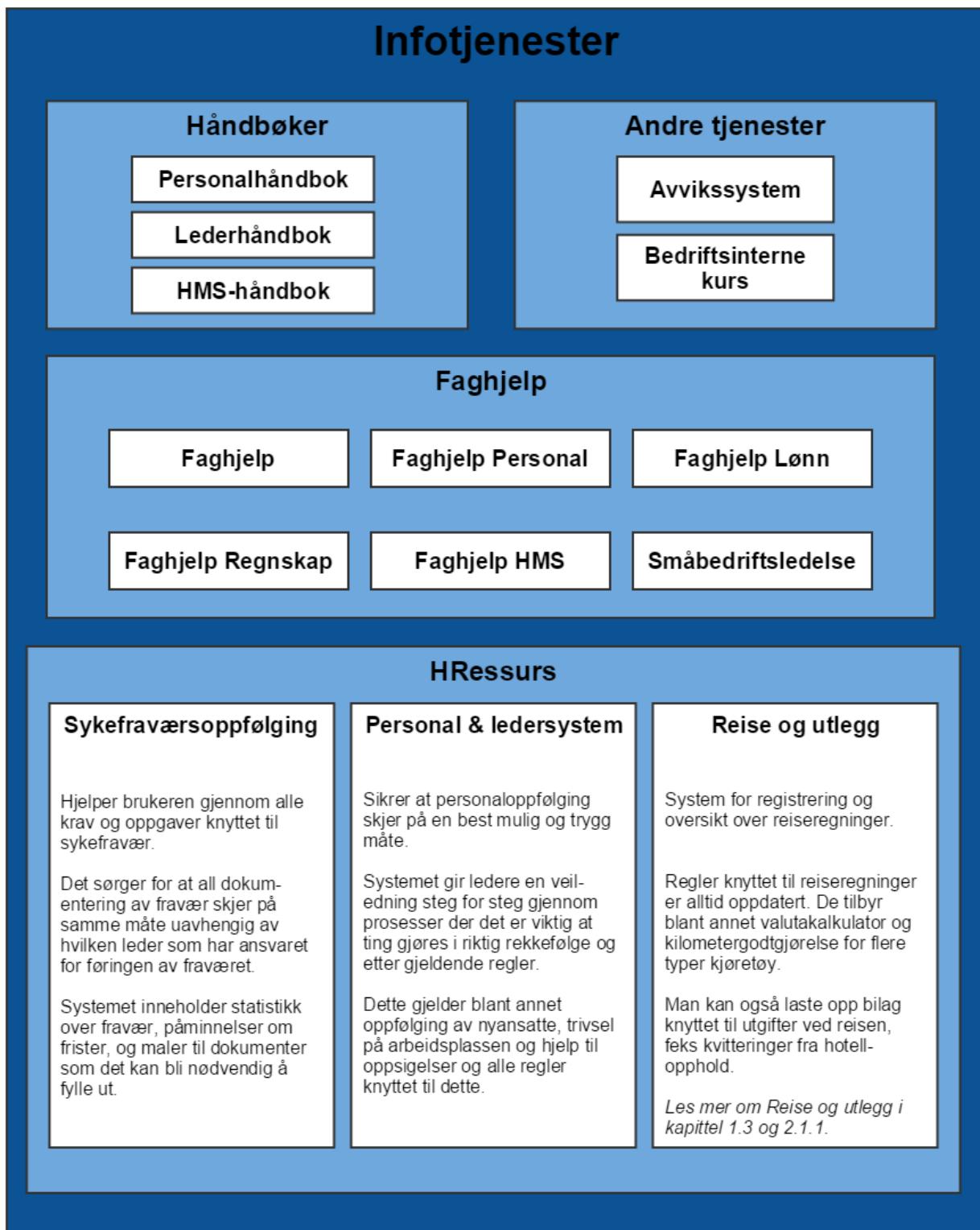
1.2 Oppdragsgiver

Infotjenester er et IT-/konsulentfirma som holder til på Grålum i Sarpsborg. De har siden 1985 levert faglig og juridisk kompetanse til norsk arbeidsliv. Deres hovedfagområder er innen personal, ledelse, HMS, lønn og regnskap. De har i dag ca 150 ansatte og et datterselskap i Sverige, Stage Competence.

De har en ledende posisjon når det gjelder kurs, faghjelp og håndbøker. Håndbøkene er nettbaserte og tar for seg blant annet spørsmål relatert til helse, miljø og sikkerhet. I følge Infotjenester er håndbøkene ment som informasjonsportaler for bedriftene hvor de ansatte kan lete opp informasjon på egenhånd. De tilbyr også faghjelp innenfor mange områder, som for eksempel lønn og regnskap. Faghjelp er først og fremst rettet mot ansatte med lederansvar. Faghjelp inneholder blant annet oppslagsverk, diskusjonsforum og support. Oppslagsverket har mer enn 40 000 brukere. Infotjenesters rådgivere er alle enten jurister eller fagspesialister med lang erfaring innen både offentlig og privat sektor. De svarer på mer enn 50 000 spørsmål hvert år. Sammensenningen av kompetanse hos de ansatte innen juss, HR og systemutvikling gjør at de kan utvikle gode og praktiske tjenester i henhold til regelverk og med god brukervennlighet. Figur 1.1 på neste side illustrerer alle tjenester Infotjenester tilbyr.

Et av deres mest populære produkter er det egenutviklede personalsystemet HRessurs, der man kan behandle alt fra personaladministrering, sykefravær, ferie og reiseregninger. Dette er et nettbasert system, slik at kundene abонnerer på tilgang til nettløsningen og ikke trenger å kjøpe og installere programvare. Dette fører til at systemet alltid er oppdatert, uten at kundene må tenke på å laste ned oppdateringer.

Kundene kan velge mellom tre ulike pakker alt etter hva bedriften trenger: HRessurs Sykefraværsoppfølging, HRessurs personal- og ledersystem og HRessurs Reise og Utlegg. I denne oppgaven er det HRessurs Reise og Utlegg, som behandler reiser og alle utgifter knyttet til dette, vi jobber med. Figur 1.1 inneholder en mer detaljert beskrivelse av alle delene av HRessurs.



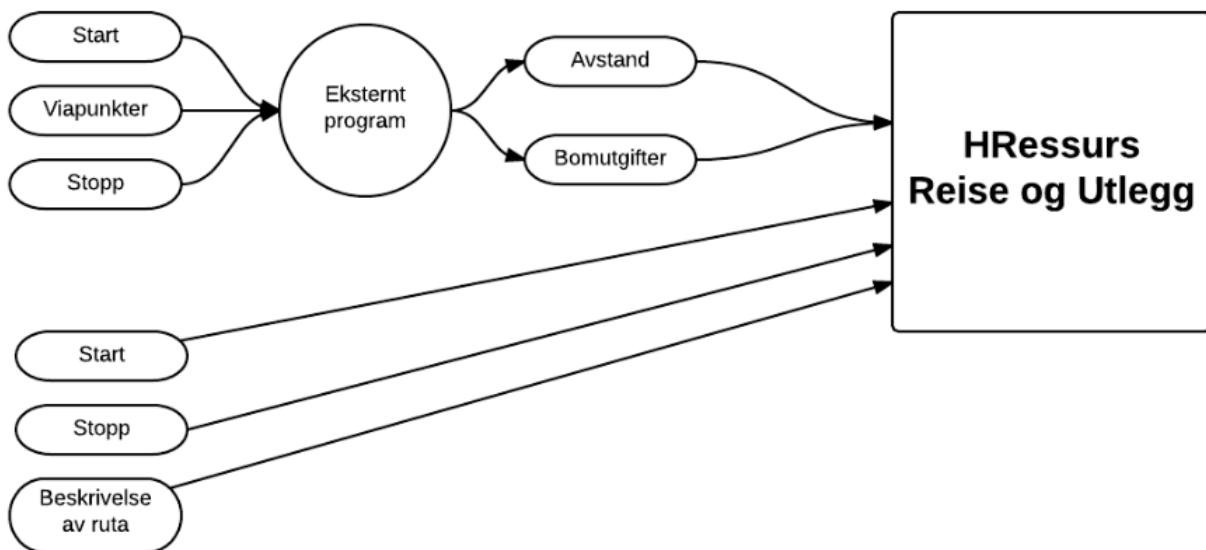
Figur 1.1: Oversikt over Infotjenesters produkter

1.3 Oppdraget

Infotjenester lover på sine nettsider et «*brukervennlig og intuitivt grensesnitt som gjør den ansatte i stand til å registrere/behandle reiser og utlegg uten opplæring. Brukeren trenger ingen kunnskaper om regler og satser*». Infotjenester har fått tilbakemeldinger på at HRessurs ikke er brukervennlig nok, blant annet med hensyn til avstand og bompenger. Vår utfordring er å utbedre denne mangelen som er årsak til noe av misnøyen blant enkelt kunder.

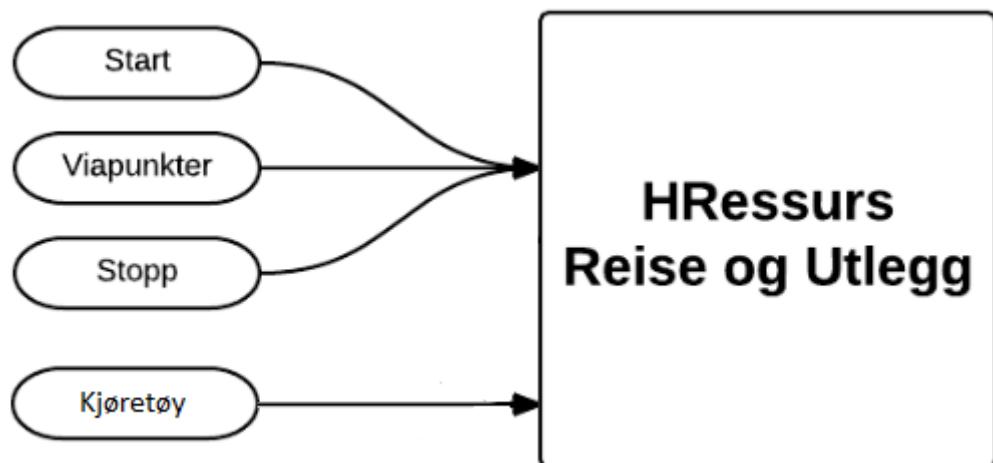
HRessurs skal utvides med en mulighet for å beregne avstand og bompengekostnader under føring av reiseregninger, slik at brukeren ikke skal måtte ta i bruk eksterne verktøy for å innhente denne informasjonen. Denne tilleggsfunksjonen må ta hensyn til statens krav for føring av reiseruter. Disse kravene innebærer at kjørerutene skal beskrives detaljert, med tanke på senere kontroll/etterdokumentasjon av reiseregningene. Brukeren skal kunne angi type fremkomstmiddel, start, stopp og eventuelle viapunkter. Basert på angitt informasjon, skal programmet beregne avstand og bompengekostnader for ruta. For at brukeren skal kunne kontrollere at programmet har beregnet riktig kjørerute, skal programmet skrive ut en kort veibeskrivelse av ruta.

Figur 1.2 viser dagens løsning der man må benytte seg av ekstern teknologi for å innhente avstand og bompengekostnader:



Figur 1.2: Slik fungerer HRessurs i dag

Et annet alternativ til skissen i figur 1.2 er at man gjør slik det ble gjort før denne type beregninger ble tilgjengelig på internett. Man kan selv notere bomstasjoner på ruta og lese av avstanden fra kilometerstanden både før og etter endt kjøretur. Man må så beregne differansen mellom disse avlesningene for å komme fram til hvor langt man har kjørt. Dette er ingen god løsning da det er fort å glemme å lese av kilometerstanden og fordi det i dag ikke er enkelt å oppdage alle bomstasjoner man kjører gjennom. De karakteristiske bomstasjonene der man fysisk stoppet og betalte ved passering blir gradvis byttet ut med automatiske bomstasjoner man knapt registrerer at man kjører gjennom. Vi ønsker at beregningen av reiseregninger skal foregå som vist i figur 1.3.



Figur 1.3: Slik ser vi for oss at HRessurs kan forbedres

Løsningen i figur 1.3 vil være en bedre løsning både for de ansatte som kjører og firmaet, da beregningen foretas automatisk når de registrerer reiseregningen. Løsningen er bedre for de ansatte som kjører, da de slipper mange manuelle beregninger, og de får tilbake det beløpet de faktisk legger ut for. Firmaet betaler kun det beløpet de skal, fremfor et beløp beregnet på hukommelsen til den ansatte. Det viktigste målet er at registreringen av bompengerutgifter på reiser i forbindelse med arbeidet skal bli enklere, og mer korrekt.

Ettersom dette er en tilleggsfunksjonalitet som skal legges til eksisterende HRessurs, er det en stor fordel at løsningen er bygd opp med samme struktur som HRessurs. HRessurs er et program firmaene betaler for tilgang til, og det setter også en del ekstra krav til standarden vi må følge. HRessurs forutsetter ikke at brukere har oversikt over de til en hver tid gjeldene regler for reiseregninger, så det må også sørges for at alle regler følges. Hvordan dette skal løses, og hvilke krav løsningen må følge vil vi komme nærmere til i analysedelen i kapittel 2.

1.4 Formål, arbeidsmetode og Leveranser

1.4.1 Formål

Som vår bacheloroppgave ønsker vi å utvikle en tillegsapplikasjon til HRessurs som skal gjøre det enklere for brukere å føre reiseregninger. De skal slippe å notere hver eneste bomstasjon de passerer og de skal slippe å ta i bruk eksterne verktøy for å beregne avstanden de har kjørt. Vi skal derfor utvikle en applikasjon som følger prinsippet vi illustrerte i figur 1.3 i forrige kapittel. Alt brukeren skal gjøre er å angi start, stopp, eventuelle viapunkter og kjøretøy. Ut fra disse parameterne skal både avstand og totale bompengekostnader beregnes, enkeltpasseringene gjennom bomstasjoner skal listes opp og en spesifisert kjørebekrivelse skal genereres. Denne beskrivelsen skal spesifisere kjøreruta slik at den oppfyller kravene til Statens Reiseregulativ om dokumentering av kjørerute.

Hovedmål

Gjøre utfylling av reisregning enklere og raskere for brukere av personalsystemet HRessurs.

Delmål 1

Foreta en nøyaktig beregning av ruta med fokus på bompengekostander og avstand.

Delmål 2

Kunne beskrive den beregnede ruta detaljert nok til å etterkomme alle regler for føring av reiseregninger.

Delmål 3

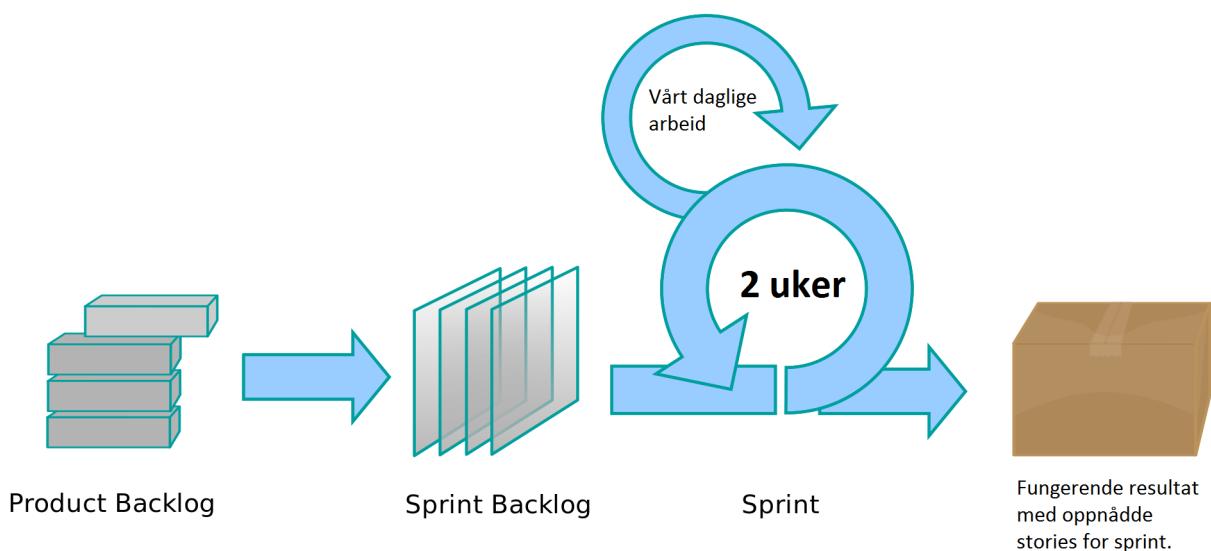
Implementere vår ferdige applikasjon inn i HRessurs

1.4.2 Arbeidsmetode

Arbeidsmetodene vi planlegger å benytte er en kombinasjon av individuelt arbeid, og samarbeid i gruppe. Vi planlegger å jobbe sammen to til tre dager i uka fra skolen, og resten av tiden individuelt. Vi vil kommunisere i en gruppechat på facebook, via mail og skype. Prosjektfiler, både dokumenter og kode, vil vi laste opp i en fellesmappe på dropbox, slik at alle hele tiden har tilgang på de nyeste versjonene. Vi planlegger også å bruke github for ekstra backup av data, og for at oppdragsgiver kan ha tilgang til filene. Da dette er et prosjekt som både innebærer en del forhåndssundersøkelser og programmering, vil vi prøve å få til en oppdeling av prosjektet slik at alle best mulig kan arbeide parallelt. Vi vil dele opp slik at vi jobber på forskjellige deler av applikasjonen, og når de ulike delene er ferdige, vil vi samarbeide om å sette disse sammen.

I samarbeid med arbeidsgiver har vi kommet frem til at bruk av arbeidsmetoden Scrum er det som passer oss best når det kommer til utviklingen av applikasjonen. Vi vil bruke en forenklet versjon av Scrum, da denne metoden er mye brukt i forbindelse med programmering og systemutvikling. I hovedsak går Scrum ut på at det ikke settes konkrete kravspesifikasjoner fra begynnelsen, men at prosjektideen hele tiden videreføres gjennom prosjektperioden. Det utvikles hele tiden ny teknologi, og det som ved prosjektoppstart kan ha vært en revolusjonerende ide, kan underveis i prosjektperioden bli erstattet av ny og bedre teknologi. Vi har en kortere utviklingsfase og et klart

mål for hvordan produktet skal bli, men vi vil allikevel ta i bruk hovedelementene i Scrum da det ikke er fastsatt hvordan applikasjonen skal utvikles. Vi vil definere «user stories», små delmål som beskriver hva som skal kunne gjøres fra brukerens perspektiv, samt definere hvilken funksjonalitet vi bør fokusere på. Vi vil sette opp mål for hva som skal gjøres for to og to uker av gangen, kalt sprinter. Målene som skal utføres i påfølgende sprint lagres i en sprint backlog. Ved hver sprintstart/-slutt vil vi ha et møte med produkteier(oppdragsgiver) hvor vi i fellesskap blir enig om hvilke user stories vi skal fokusere på i neste sprint. Alle definerte user stories som ikke er tildelt sprinter er lagret i en product backlog. Elementer fra Scrum er integrert i Ganttdiagrammet vårt i kapittel 1.4.3, og i figur 1.4 illustrerer vi arbeidsflyten ved bruk av Scrum.

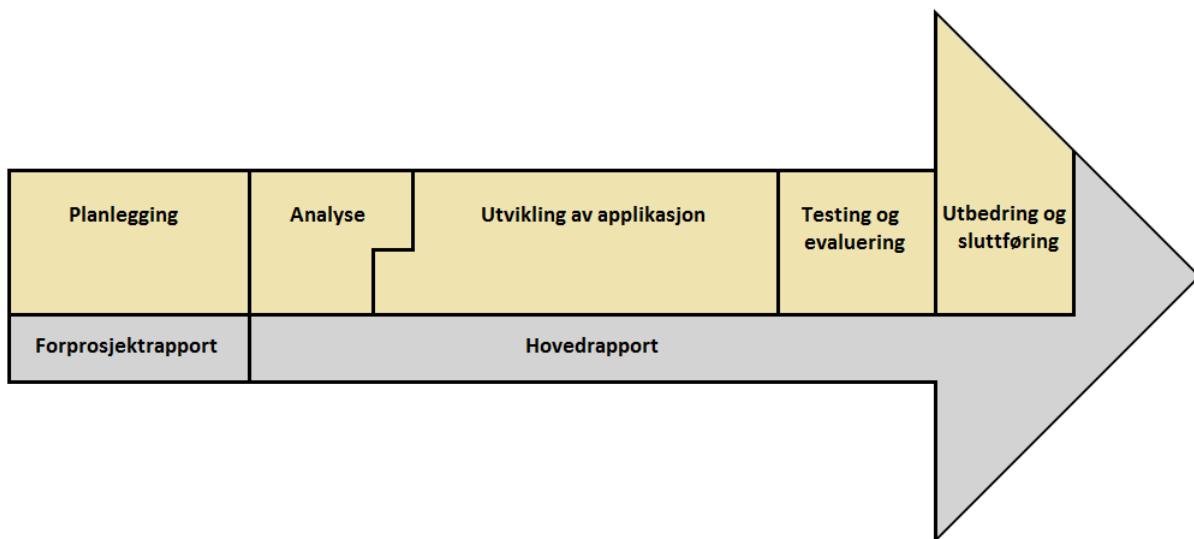


Figur 1.4: Arbeidsflyt ved bruk av Scrum

Vi bruker metoden Scrum for å ha et mer dynamisk arbeidsperspektiv på løsningen, da det hele tiden kan komme endringer eller nye synspunkter på hvordan vi kan løse oppgaven ettersom kunnskapen vår øker. Det kan også hende oppdragsgiver ønsker ekstra funksjoner i applikasjonen. Til å holde styr på back log, sprinter og user stories benytter vi en felles Github, hvor vi lagrer alt under Issues. Ved å annenhver uke bestemme hva som skal gjøres videre, vil vi ha større forutsetninger til å få en løsning både vi og oppdragsgiver er fornøyd med. Etter å ha utviklet applikasjonen skal denne testes, slik at vi får utbedret eventuelle feil og mangler vi selv ikke oppdager. Vi vil la bekjente med varierende dataerfaring og -kompetanse teste systemet, og på bakgrunn av tilbakemeldingene vil vi utbedre og ferdigstille applikasjonen.

I hovedsak kan prosjektperioden deles inn i fem klare faser som illustrert i form av pila i figur 1.5 på neste side. Den første fasen er oppstarten med planlegging og undersøkelse av gjennomførbarhet. Den andre fasen er analysefasen der vi vil foreta analyser av tilsvarende, eksisterende løsninger og teknologi for å finne ut hvordan vi best mulig bør gå fram for å løse oppgaven. Parallelt med at noen av oss fullfører siste del av analysedelen som er beskrevet ovenfor, vil de resterende på gruppa starte

med utviklingen av applikasjonen. Når både vi og oppdragsgiver er fornøyde med resultatet, skal vi gjennomføre testing av applikasjonen, før vi ferdigstiller den ved å utbedre feil og mangler som kommer frem under testingen. For å slippe tekniske problemer mot slutten og å risikere at vi ikke kommer i mål med applikasjonen, planlegger vi å ferdigstille applikasjonen i løpet av april. Da kan vi ha fullt fokus på rapporten frem til innlevering.



Figur 1.5: Grafisk fremvisning av tenkte metoder.

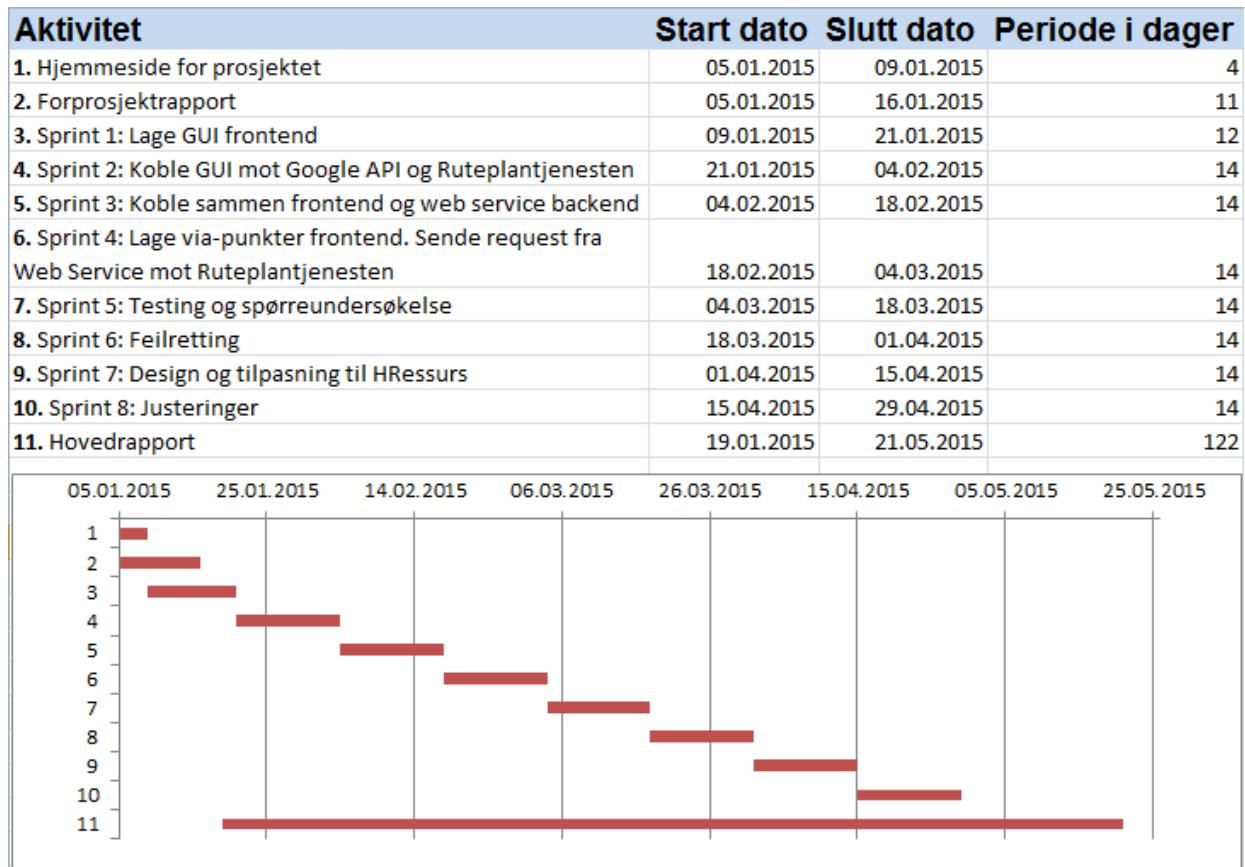
Rapporten vil vi skrive jevnt gjennom hele prosjektet, fra forprosjektrapporten er levert og til rapporten skal leveres. Siden rapporten spiller en veldig stor del av prosjektet, vil vi fordele slik at vi hele tiden har minst en på gruppa som har hovedansvar for å skrive rapporten, mens resten programmerer. Alle har hele prosjektperioden ansvaret for å skrive bidrag til kapittel 3 som omhandler gjennomføringen av prosjektet.

De første ukene vil være en typisk oppstartsfase, der vi vil bruke en del tid på planlegging og forprosjektrapporten, i tillegg til at alle må sette seg inn i programmeringsspråkene oppgaven baserer seg på. I denne perioden skal vi også finne ut om prosjektet er gjennomførbart, blant annet ved å finne ut hvordan vegvesenets API fungerer, om vi finner all informasjon vi trenger der, eller om det finnes andre bedre alternativer. Dersom prosjektet viser seg å bare delvis la seg gjennomføre, eller ikke lar seg gjennomføre i det hele tatt, må vi bruke denne perioden til å finne andre alternativer for å innhente og beregne data. For å se at prosjektet lar seg gjennomføre, vil vi lage en veldig enkelt testversjon. Vi skal så begynne med analyse for å se på hvilke hensyn vi må ta i forhold til den eksisterende løsningen i HRessurs, hvilke regler som gjelder for reiseregninger, hvilke krav til universell utforming som må dekkes og hvordan vi best mulig kan løse oppgaven programmeringsmessig.

Når vi har funnet ut at prosjektet lar seg gjennomføre, og bestemt hvordan vi skal løse det, skal vi begynne å utvikle applikasjonen.

1.4.3 Fremdriftsplan

I diagrammet i figur 1.6 finnes vår oversikt over hvordan vi fra begynnelsen ønsket å strukturere hoveddelene av prosjektet. Vi har delt inn etter de viktigste oppgavene og lagt de under aktivitet. Figuren dekker både innleveringer som er relatert til Høgskolen, slik som innlevering av forprosjektrapport og hovedrapport, og tidsfrister i forhold til oppdragsgiver. De punktene som har med utvikling av applikasjonen er delt inn i sprinter etter Scrummetoden. Sprintene går over to uker, noe som er vanlig praksis innen Scrum.



Figur 1.6: Gantdiagram - Planlagt fremdrift

1.5 Rapportstruktur

Kapittel 2 - Bakgrunnsanalyse

I dette kapittelet tar vi for oss eksisterende teknologi. Vi har delt kapittelet inn i to, eksisterende ruteplanleggingstjenester inkludert regler og hensyn som må følges og tilgjengelig teknologi for å løse oppgaven. Vi går gjennom andre ruteplanleggingstjenester for å se hvilke som egner seg best og eventuelt hvilke deler av ruteplantjenestene som kan knyttes sammen slik at vi oppnår best mulig funksjonalitet i vår applikasjon. Vi tar også for oss fordeler og ulemper rundt valg av teknologi i forhold til programmering.

Kapittel 3 - Veien mot det endelige resultat

I dette kapittelet tar vi for oss veien fram til den ferdige applikasjonen vi gjennomfører brukertesting av. Vi planlegger struktur og brukergrensesnitt, drøfter valg vi har tatt underveis i utviklingsfasen og forklarer prosessen fram mot den ferdig applikasjonen. Vi har delt kapittelet i to, brukergrensesnitt og teknisk. Her skiller vi på de tekniske valgene vi har tatt og valg tatt i forhold til best mulig brukervennlighet.

Kapittel 4 - Brukertest av applikasjonen

I dette kapittelet tar vi for oss prosessen rundt det å brukertestet applikasjonen. Vi tar for oss planleggingsfasen av denne testen, hvilke spørsmål vi har stilt og hvordan vi planlegger å vektlegge tilbakemeldingene. Vi går nærmere inn på tilbakemeldingene vi fikk samt hvilke endringer vi har foretatt som følge av disse.

Kapittel 5 - Det endelige resultatet

I dette kapittelet beskriver vi den ferdig applikasjonen etter utbedringene etter brukertesten. Her har vi utviklet en brukerveiledning for applikasjonen fra et brukerperspektiv. Vi har også foretatt en oppsummering av det tekniske, og beskriver alle hensyn som må tas ved videreutvikling/vedlikehold av applikasjonen.

Kapittel 6 - Diskusjon

I dette kapittelet diskuterer vi hva vi har fått ut av oppgaven, hva vi har lært og utfordringer vi har støtt på. Vi tar for oss hva som burde vært gjort annerledes og om vi har oppnådd målene med prosjektet.

Kapittel 7 - Konklusjon

I dette kapittelet foretar vi en totalvurdering og trekker en konklusjon av hele prosjektet og prosjektperioden.

1.6 Terminologi

.NET (dot NET)

Microsoft sitt rammeverk for programvare. Utviklingsverktøyet kalles for Visual Studio og brukes til å lage alt fra applikasjoner for Windows, lage websider eller jobbe med databaser. .NET rammeverket kommer i to klassebibliotek: Framework Class Library og Base Class Library. I sistnevnte inngår blant annet ASP.NET for utvikling av dynamiske websider.

AJAX

Asynchronous Javascript and XML. Brukes innen webutvikling til å sende eller hente data asynkront, det vil si et skript som kjører i bakgrunnen. Med AJAX kan man for eksempel oppdatere en del av en webside uten å laste inn hele siden på nytt.

API – Application Programming Interface

Et grensesnitt i en softwarekomponent som kan tilpasses til og kjøres fra andre applikasjoner. Et API kan aktiveres fra en applikasjon og kalle på en annen applikasjon som henter data. API-er inneholder et sett med regler som benyttes ved kall til et bibliotek eller en applikasjon. Et populært eksempel er Google Maps. Mange benytter seg av kart levert av Google på sin egen webside. For eksempel kan en bedrift vise kunden nærmeste butikker i forhold til hvor kunden befinner seg. Vi bruker Google Maps til å vise en kjørerute mellom angitte steder.

Brukere

I denne rapporten vil vi referere til de ansatte som bruker HRessurs som «brukere».

C#(sharp)

Dette programmeringsspråket er utviklet av Microsoft innenfor .NET rammeverket og baserer seg på resten av C-familien (C, C++ og Java). C# er et klassebasert objektorientert språk, og er såkalt «sterkt typet» som vil si at man blant annet må definere alle variabeltyper, for eksempel om de inneholder ren tekst (string) eller tall (int). Språket brukes blant annet til å lage applikasjoner i Microsoft Windows eller websider. I webprogrammering kjører C# på serveren som sender relevant informasjon tilbake til nettleseren. En nettleser kan ikke kjøre C# direkte da den ikke kan interpretere språket.

CSS – Cascading Style Sheets

Helt siden sent på 90-tallet har det vært vanlig å legge alt som har med ren design å gjøre i CSS, enten i en egen fil som lastes inn på websiden, eller i egne «tagger» øverst på websiden. Poenget med CSS er at utseende på websiden kan forandres uten å forandre strukturen i HTML. Det vil si at man kan benytte CSS til å endre blant annet farge, marger og skriftype.

HR – Human Resources

en betegnelse som omhandler arbeidskraften i bedrifter, og at man ser på den menneskelige arbeidskraften som en ressurs hver bedrift har.

HRessurs

I resten av rapporten vil vi kun referere til HRessurs Reise og Utlegg som HRessurs, dersom ikke annet er spesifisert. HRessurs er en programvare som forenkler HRM for bedrifter, hvor de gjør det enkelt for både personal og ledelse/økonomi å styre arbeidsstyrken.

HRM - Human Resources Management

omhandler måten man organiserer arbeidsstyrken, HR, til å få en mest mulig effektiv produksjon og hvordan man skal organisere arbeidsoppgaver ut ifra personalets kvaliteter og erfaringer. Under dette ligger også videre utvikling og trening av arbeidsstyrken.

HTML – Hypertext Markup Language

Dette er den vanligste måten å strukturere websider på. HTML regnes som byggeklossene på en webside. Man referer ikke til HTML som et programmeringsspråk, men et markeringsspråk. Dette fordi det brukes til selve struktureringen av elementer på nettsiden og går inn under internasjonal standard for tekstformatering.

Javascript

Javascript regnes i motsetning til C# som et svakt typet programmeringsspråk. Det vil si at man ikke trenger å deklarere verdier på forhånd, som «int» eller «string». Med Javascript kan man la en bruker manipulere websiden. For eksempel kan man sette noen elementer til å være synlig når det er behov for det, eller fjerne og legge til tekst. Javascript er lagret frontend, det vil si at koden kan leses av en bruker og at språket interpreteres og kjøres av nettleseren.

jQuery

jQuery er det mest populære biblioteket for Javascript. Biblioteket er ment å gjøre det enklere å lage funksjoner i Javascript. Eksempler på hva man kan benytte jQuery til er animasjoner og spesielle effekter ved lasting av en nettside. jQuery har også blitt veldig vanlig ved utvikling av AJAX-applikasjoner.

JSON – Javascript Object Notation

Utviklet fra Javascript og brukes primært til å sende tekstbasert data mellom en server og en web-applikasjon. JSON har blitt en populær erstatning for tredjeparts programvare i nettleseren for å muliggjøre kommunikasjon i nåtid mellom server og web-applikasjoner. JSON fungerer godt i AJAX-applikasjoner.

Klientside/Frontend

Regnes som alt brukeren av applikasjonen har tilgang til. Det vil si selve websiden, design og Javascript. All kode er tilgjengelig for utenforstående i klartekst.

Ruteplantjenesten

Vegvesenet sitt API for veibeskrivelser, beregning av kjøreruter og bompenger. Deres implementasjon av API-et kan du finne på <http://visveg.vegvesen.no/visveg/>

Serverside/Backend

All programkode som er lagret utenfor brukerens rekkevidde og som kjører på en server.

XML

Extensible Markup Language. På samme måte som HTML er XML et markeringsspråk. XML brukes blant annet som kommunikasjon mellom forskjellige systemer, og til deling av strukturerte data i Web Servicer. Men fordi XML er mer plasskrevende har JSON tatt mer og mer over for disse oppgavene. Vi benytter oss av JSON og ikke XML.

2. Bakgrunnsanalyse

I dette kapittelet tar vi for oss hvilke hensyn som må gjøres og hvilke eksisterende løsninger som allerede finnes, vi undersøker også hvilke teknologier vi kan benytte for et best mulig resultat på vår applikasjon.

2.1 Eksisterende løsninger samt regler og hensyn.

2.1.1 HRessurs i dag, brukergrensesnitt og sideoppsett

Vi vil nå gå nærmere inn på hvordan HRessurs Reise og Utlegg er designet i dag, slik at vi vet mer om hvilke hensyn vi må ta når vi utvikler vår applikasjon. Brukeren får tilgang til reiseregistrering ved å logge seg inn på sin HRessurskonto og deretter velge «mine reiser». Her får brukeren valget mellom å se på registrerte reiser, eller registrere en ny. Registreringen foregår trinn for trinn, og det er enkelt for brukeren å holde oversikten over hvor i reistreringsprosessen han befinner seg. Det er både en meny i form av en pil (se figur 2.1) som forteller hvor brukeren befinner seg i registreringsprosessen, samt en fram- og tilbakeknapp nederst på skjermen (se figur 2.2 som viser både antall trinn, og hvilket trinn man befinner seg på).



Figur 2.1: Meny i form av en pil

Hver side i applikasjonen er minimalistisk med tanke på design, og følger samme standard gjennom hele registreringsprosessen. Det er hvit bakgrunn og samme teksttype både i hoveddelen, menyen og knappene. Bakgrunnsfargen i menyen er grå, og knappene er blå.



Figur 2.2: Navigering fram og tilbake

Det er ikke mulig å navigere seg videre i registreringen før man har fylt ut alle nødvendige felter. Antall trinn avhenger av om man har benyttet eget kjøretøy, om man skal ha diettgodtgjørelse eller om man har vært i utlandet. Hukes det av for noen av disse punktene, dukker det opp et nytt trinn hvor man må fylle inn informasjon for punktene som er huket av. På første side må brukeren navngi reisen, fylle inn dato og klokkeslett for avreise og hjemkomst og angi reisested og formålet med reisen. Ettersom bruker selv navngir reisene blir det også enklere for bruker å holde oversikten over reisene. Har brukeren huket av for diettgodtgjørelse dukker det opp to nye trinn hvor man på første trinn skal legge inn overnatningssted, ankomst og avreise. På neste trinn kan brukeren legge til måltidsfradrag. Vi vil ikke gå nærmere inn på andre utlegg i forbindelse med overnatting og mat på reiser, slik som diettgodtgjørelse og måltidsfradrag, da disse ikke er relevant i forhold til vår applikasjon.

Figur 2.3: Skjermdump fra HRessurs - føring av kjørerute

Hvis det har vært huket av for eget kjøretøy er neste trinn registrering av reiserute, slik figur 2.3 viser. Ruten må spesifiseres så nøyaktig som mulig i et eget tekstfelt. For eksempel: «Sarpsborg – Oslo via E6 til Tusenfryd, så E18 til Oslo». I neste tekstfelt må brukeren oppgi nøyaktig antall kilometer. Alle felt må fylles ut manuelt, og ingenting beregnes automatisk. Sett fra et brukerperspektiv vil dette kanskje virke gammeldags og tungvint, samt gi inntrykk av lite automatisering. Kjøretøy må velges, bruker kan fylle ut informasjon om kjøring utland hvis dette er relevant, årsak til eventuelle omkjøringer på ruta, antall medpassasjerer og andre spesielle tillegg.

På neste trinn etter ruteregistrering, vist i figur 2.4 på neste side, kan brukeren fylle ut utlegg. Bruker må beskrive type utlegg, dato for utlegget og beløpet. Brukeren kan også laste opp eventuelle bilag. Bompenger krever ikke bilag. På nest siste trinn kan brukeren fylle inn ekstra informasjon, hvis han har mottatt forskudd og kostnadsbærere osv.

På siste side kommer det opp en detaljert oversikt over all informasjon som har vært fylt ut og hvor mye som eventuelt skal utbetales til bruker. Brukeren kan også se nåværende status på registreringen, om den venter på innsendelse eller om den er godkjent/ikke godkjent. Brukeren kan nå velge å signere og sende inn, eller gå tilbake og gjøre endringer.

Type utlegg

Dato for utlegg

Beløp i NOK \$/€

Betalt av arbeidsgiver
(Skal ikke refunderes)

Beskrivelse i

Bilag - Ingen tilgjengelig Upload

Lagre utlegg eller avbryt

Forrige Trinn 5 av 7 Neste

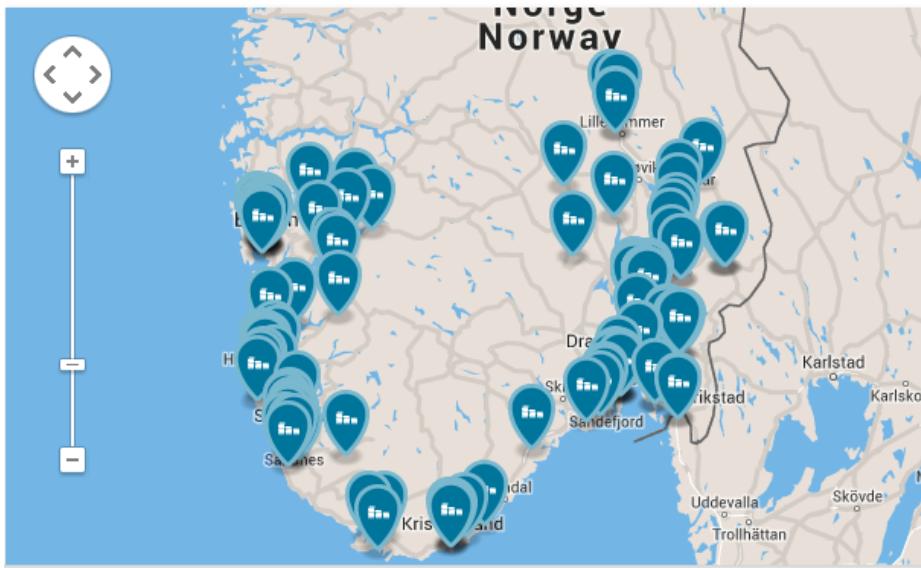
Figur 2.4: Registrering av utlegg

2.1.2 Bomstasjoner - rabattordninger og regler

I Norge øker antall bomstasjoner stadig. Blant annet langs E6 mellom Gardermoen og Hamar har det på få år kommet fem bomstasjoner, og en sjette åpnes i juni 2015[1]. Figur 2.5 viser et kart med alle bomstasjoner i Sør-Norge[2]. Kartet er utviklet av Infotjenester. Senest 3.februar 2015 i Halden Arbeiderblad [3] nevnt en mulighetene for bompengefinansiering i Halden i forbindelse med bygging av to tunneler i fjellet under Fredriksten Festning, sitat: «– En bompengefinansiering gjør at man kan komme forttere i gang med den type prosjekter. Halden bør ta en runde rundt bompengefinansiering, mener han. Ordfører Thor Edquist tror ikke vi kommer utenom bompenger».

Fra og med 1.januar er det obligatorisk for biler med tillat totalvekt over 3,5 tonn å ha autopass når de kjører i Norge. Regelverket gjelder alle biler som tilhører «foretak, stat, fylkeskommune eller kommune, eller som på annen måte hovedsakelig benyttes i næringsvirksomhet» [4] Straffen for å ikke følge disse reglene er en bot på 8000kr, og denne satsen dobles dersom pålegget ikke etterkommes og kjøretøyet stoppes uten brikke igjen innen 2 år fra første straff. Til tross for påbudet om autopassbrikke

i kjøretøy over 3,5 tonn, vil vi når bompengekostnadene beregnes, ikke ta hensyn til eventuelle autopassavtaler de ansatte måtte ha. Dette er med tanke på at priser, rabatter og betalingspraksis varierer veldig over hele Norge [5], slik vi vil gå nærmere inn på i det påfølgende avsnitt.



Figur 2.5: Bomstasjoner i Sør-Norge

Det er i Norge slik at alle bomstasjoner er tilknyttet et bompengeselskap. Det er ikke ett selskap som drifter alle bomstasjoner i Norge, men 44 forskjellige, lokale selskaper som drifter hver sitt/sine bomanlegg [6]. Det finnes ingen standardtakster og rabattavtaler som er gjeldene for alle. Hvert selskap bestemmer selv om de skal ha rabatt til alle med autopass eller rabatt kun til egne abonnenter. Om bompengeselskapene tilbyr forskuddsbetaling, etterskuddsbetaling eller begge deler er også opp til hvert enkelt selskap, og rabattene varierer gjerne ut ifra betalingsalternativene de tilbyr.

Østfold Bompengeselskap [7][8] som drifter seks bomstasjoner i Østfold, tilbyr kun rabatt til sine egne abonnenter. Alle andre med autopass fra andre selskaper, vil måtte betale full pris ved passering. Et annet selskap her i Østfold, som har en annen praksis, er Svinesundsforbindelsen [9]. De tilbyr kun etterskuddsbetaling, og gir 13% rabatt ved passering til alle med autopass uavhengig av bompengeselskap. Tilsvarende rabatter finnes blant annet ved Bomringen i Oslo, Kråkerøyforbindelsen og Bomringen i Kristiansand. Nord-Jæren Bompengeselskap [10] er et selskap som tilbyr både forskudd og etterskuddsbetaling for sine abonnenter, men med 10% rabatt for privatkunder med etterskuddsbetaling, og 20% rabatt for alle med forskuddsbetaling, samt firmakunder med etterskuddsbetaling.

Det er ikke bare ulik praksis med autopassavtale og passeringsrabatter som skiller bompengeselskapene. I Trondheim praktiseres rushtidspriser, som vil si at i tidsrommet 07-09 og 15-17dobles taksten for å passere bomstasjonene i Trondheim sentrum. Disse tilhører prosjektet «Miljøpakke Trondheim» [11]. Resterende bomstasjoner i Trondheimområdet følger ikke rushtidpraksisen. Miljøpakken er et prosjekt som ble startet i 2009 for å begrense miljøproblemene i forbindelse med at Trondheimsområdet er et av områdene i Norge med størst befolkningsvekst [12]. I Trondheim finnes det også tidsgrenser for om man skal betale for passeringene eller ikke.

Bomstasjonene er grupperte i seks grupper, det Miljøpakken omtaler som «snitt». Hvert av snittene inneholder mellom en og ni bomstasjoner, og innenfor et tidsrom på en time, betaler man kun for en passering i hvert snitt [13] Denne praksisen finner man blant annet igjen i Haugalandspakken [14] som dekker bomstasjoner i Haugesund og på Karmøy, der man også kun betaler en passering innenfor en time.

Med så mange forskjellige rabattordninger og varierende priser innenfor et tidsrom, vil det være tilnærmet umulig å ta hensyn til hver enkelt bruker av HRessurs og en eventuell autopassavtale han eller hun måtte ha. Etterskuddsfakturering av passeringer og utsendelse av oversikt over passeringer for de med forskuddsbetaling, skjer gjerne en tid etter passeringen fant sted. Hyppigheten for utsendelse varierer fra bompengeselskap til bompengeselskap, samt hvilke avtaler man har. Noen selskaper praktiserer månedlig etterskuddsfakturering, og andre sender en oversikt over passeringer når forhåndsbetalt beløp er brukt opp.

Passeringene kan også være vanskelig å holde oversikt over, da disse ofte har lokale navn, som ikke sier den reisende så mye uten å være lokalkjent. Bedrifter har også ofte en månedlig frist for å registrere reiseregninger for siste måned. Dette tilsier at man i de fleste tilfeller ikke har tid til å vente på faktura/oversikt over passeringer, og et annet viktig argument er statens regler i forhold til refusjon av bompenger. Dette er utgifter det ikke kreves kvittering på, og på altinn.no står følgende: «For enkelte småutgifter som bompenger, parkometeravgift og lignende er det ikke nødvendig med originalbilag. Det holder at utgiften spesifiseres på reiseregningen eller et eget oppsett.» [15] Dette er bakgrunn for at det istedenfor å dokumentere hver enkelt passering, med hver enkelte brukes rabatter, kan foretas en generell prisberegning ut ifra vanlige takster.

2.1.3 Eksisterende ruteplanleggingstjenester

I første møtet med oppdragsgiver fikk vi presentert et API fra Vegvesenet, Ruteplantjenesten, som gjør det mulig å beregne bompengekostnader og kjørelengde. Den webbaserte tjenesten deres [16] virker ikke helt optimal, slik at vi bestemte oss for å undersøke om vi hadde andre muligheter enn Vegvesenet ved å gjøre research rundt tilsvarende, eksisterende løsninger. Vi tok for oss andre nettbaserte tjenester for å beregne kjøreruter, for å se hvilken tilleggsinformasjon de kan tilby utover å beregne kjøreruter. Vi testet NAF Ruteplanlegger [17] og veibeskrivelsestjenesten hos 1881 [18], Google Maps [19], Gule sider [20] og Kvasir [21].

Felles for alle er at de oppgir kjørelengde for strekningen mellom to steder, samt at de har en mulighet for å legge til via-punkter. Google Maps og 1881 tilbyr i tillegg en funksjon slik at man grafisk kan endre kjøreruta, kun ved å flytte kjøreruta. Man kan også legge til både start, stopp og viapunkter ved å trykke i kartet. Bompengekostnader er det derimot færre som viser. Kvasir og Gule Sider benytter seg av samme tjeneste der kun logoen og fargevalg skiller de. De inkluderer ingen informasjon om bomstasjoner langs ruta. NAF og Google oppgir at det er bomvei på deler av strekningen, uten å spesifisere nøyaktig hvor på strekningen bomstasjonene befinner seg og ei heller prisen. 1881 oppgir prisene for vanlig personbil og lastebil, i tillegg navn på bomstasjonene og hvor de ligger (grafisk på kart og med koordinater). Siden hovedtanken bak applikasjonen vi skal utvikle, er å beregne både avstand og bompengekostnader, er det kun Ruteplantjenesten og 1881 som tilbyr informasjonen vi trenger.

Figur 2.6 viser en grafisk oversikt over tjenestene vi har testet og hvilken funksjonalitet hver av de tilbyr.

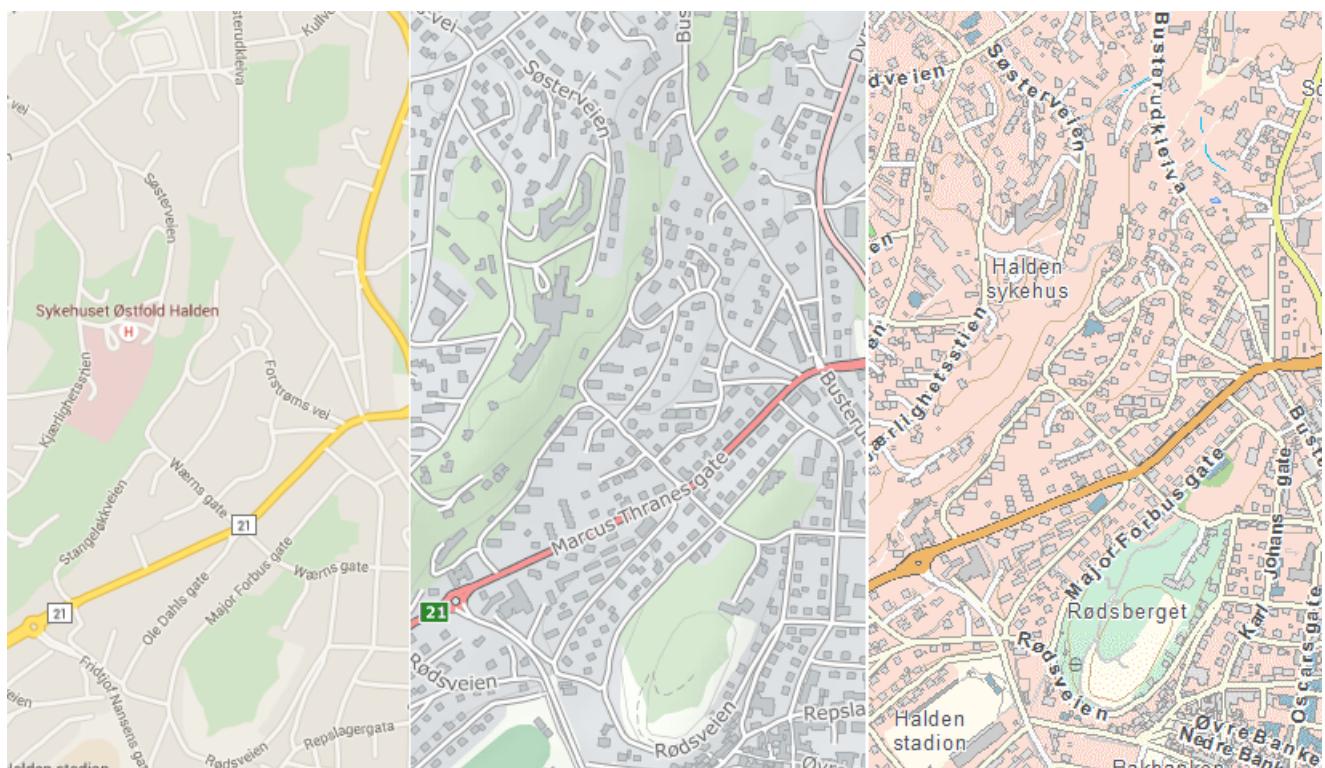
	Kvasir	Gule Sider	Naf	Ruteplantjenesten	1881.no	Google Maps
Avstand	✓	✓	✓	✓	✓	✓
Kjørebekravelse	✓	✓	✓	✓	✓	✓
Bompengekostnader totalt på reisen	✗	✗	✗	✓	✓	✗
Bomstasjoner på strekningen	✗	✗	✓	✓	✓	✓
Bomstasjoner med pris	✗	✗	✗	✓	✓	✗
Spesialpriser, rushtid osv	✗	✗	✗	✗	✓	✗
Autofullført søk	✓	✓	✓	✗	✓	✓
Kart	✓	✓	✓	✓	✓	✓

Figur 2.6: Sammenligning av eksisterende ruteplanleggingstjenester

1881 eller Ruteplantjenesten?

For å finne ut om 1881 eller Ruteplantjenesten eigner seg best, tok vi en nærmere titt på hvilken informasjon hver av tjenestene inneholder. Vi vet som beskrevet i kapittel 2.1.2, at det finnes både rabattordninger og tilleggspriser rundt omkring i Norge. For å finne ut om Ruteplantjenesten eller 1881 inneholder informasjon om disse spesialprisene, har vi testet ruteberegning gjennom bomstasjonene vi på forhånd vet har slike ordninger. Mengderabatter med tidsbegrensning, slik som i Trondheim og Haugesund er ikke tatt hensyn til på hverken 1881 eller Ruteplantjenesten, men 1881 viser informasjon om rushtidsprisene i Trondheim. Ved beregning av kostnader for reiser gjennom bomanlegg med rushtidstillegg, benyttes vanlig takst uten tillegg. Det spesifiseres så i veibeskrivelsen at det mellom 07:00 og 09:00, samt 15:00 og 17:00 er dobbel takst.

1881 tilbyr en bedre kartløsning som er mer brukervennlig. Kartløsningen hos Ruteplantjenesten er uoversiktig og lite brukervennlig og kan, sammenlignet med tjenester som 1881 og Google Maps, virke gammeldags. Se figur 2.7 for en sammenligning av kartutsnitt fra Google, 1881 og Ruteplantjenesten. I Ruteplantjenesten, som er eneste tjenesten uten en funksjon for autofullføring av adresser, må adresser angis korrekt. En tastefeil midt i adressen vil ikke gi resultater. Ved å utelate de siste bokstavene i et stedsnavn, vil den i noen tilfeller gjette riktig adresse, men dersom flere steder begynner med samme navn, vil den velge tilfeldig. Et eksempel på dette er et søker på «Storgata». Dette blir «Storgata, Moss». Hvilken «Storgata» man får opp er helt tilfeldig, og er hverken basert på avstand fra hvor vi befinner oss (da skulle Halden ha blitt valgt) eller alfabetisk sortering. Den eneste muligheten bruker har er å angi fullstendig adresse med by med en gang eller rette opp i ettertid når han eller hun ser at byen er feil. I 1881 som tilbyr autofullførfunksjon, vil man raskt oppdage en eventuell skrivefeil i adressen, siden adressen man ønsker å søker etter, ikke lenger vil dukke opp blant forslagene hvis man har en skrivefeil i adressen.



Figur 2.7: Kartutsnitt: Google t.h, 1881 i midten, Ruteplantjenesten t.v

1881 sitt system for veibeskrivelser innehar alle nødvendige data vi trenger. Vi konkluderte derfor innad i gruppa med at 1881 egnet seg best, siden de tilbyr litt mer detaljinformasjon om noen av bomstasjonene, søkefunksjonen er mer brukervennlig med tanke på autofullføringen, kartløsningen er bedre, og det er mulighet for å grafisk kunne endre kjøreruta, dersom man ser at ruta som beregnes ikke er nøyaktig den man har kjørt. Det eneste som skiller 1881 negativt fra Ruteplantjenesten er at kjøreruter kan bli beregnet gjennom Sverige dersom dette er korteste vei. Dette er uhensiktsmessig siden det vil komplisere beregning av priser, da det etter Statens Reiseregulativ er andre takster for utenlandskjøring. Det er mulig det finnes en måte å tvinge kjørerutene til å holde seg innenfor Norges grenser. 1881 opplyser på nettsidene sine at de har et API som man må betale for etter en gratis testperiode. Oppdragsgiver var enig med oss i at 1881 høres ut som en god løsning, selv om det på sikt ville koste penger. Dersom tjenesten var stabil og ga all nødvendig informasjon, ville ikke de relativt lave kostnadene som ville komme etterhvert være et hinder for fremtidig bruk.

Ved nærmere undersøkelse av API-et viste det seg derimot at dette kun er et API for person- og bedriftssøk, og ikke for veibeskrivelser. Det eneste geografiske innhold i API-et er privatpersoners og bedrifters adresser. Dette til tross for at vi forklarte hva vi skulle bruke API-et til da vi tok kontakt med 1881 for å starte en testperiode. 1881 ville derfor ikke dekke de kravene til informasjon prosjektet vårt har, slik at vi måtte gå tilbake til å benytte oss av Ruteplantjenesten. Ved nærmere undersøkelser av Ruteplantjenesten, viste det seg at JSON-objektet som returneres, har tilleggsinformasjon om navn og priser på alle bomstasjonene på ruta, selv om dette ikke vises i webtjenesten.

Andre muligheter

Ruteplantjenesten er, som vist i figur 1.3, eneste reiseplanleggingsverktøy vi har testet, uten autofullført søk. Da autofullføring bidrar til å gjøre de andre resterende ruteberegningsjenestene mer brukervennlig enn Ruteplantjenesten med tanke på adressessøk, er dette en tilleggsfunksjon vi gjerne ønsket å inkludere i vår applikasjon. Vi var da nødt til å finne en autofullføringsfunksjon for adressessøk som var kompatibel med vår planlagte applikasjon som baserer seg på Ruteplantjenesten. Google sin funksjon for å autofullføre søker både på Google og Google Maps, er en løsning vi alle er kjent med og ser på som god. Med Google Autocomplete får vi også tilgang til koordinatene til adressen, og dette er positivt da beregningene i Ruteplantjenesten er basert på koordinater. Google og Ruteplantjenesten benytter ulike koordinatsystemer som gjør at disse ikke kan benyttes samtidig uten å konvertere koordinatene først. Mer om koordinatsystemene og utfordringene forbundet med dette, finnes i kapittel 3.2.4.

Vi hadde også et ønske om å kunne vise ruta på kart, slik at brukeren kan kontrollere at den beregnede ruta stemmer. Vi forsøkte å sammenligne kjøreruter som beregnes hos Ruteplantjenesten med tilvarende kjørerute på Google Maps, for å se om vi kan benytte Google Maps direkte. På korte strekninger så kjørerutene tilsynelatende like ut, men et problem som dukker opp er at på lengre kjøreruter, og spesielt kjøreruter der korteste vei går via Sverige, vil Ruteplantjenesten og Google vise forskjellige ruter. Google beregner korteste vei uavhengig av landegrenser, mens Ruteplantjenesten beregner korteste vei i Norge. For eksempel kan en bruker ha kjørt fra Oslo til Kirkenes gjennom Norge, mens Google vil vise korteste distanse, altså gjennom Sverige og Finland. Google tilbyr ingen mulighet for restriksjoner i forhold til landegrenser, slik at dette må håndteres dersom Google Maps skulle benyttes.

Det store spørsmålet er om vi virkelig har behov for en kartjeneste i vår applikasjon, siden systemet skal behandle reiser etter at de har funnet sted. Kartet er kun ment som et hjelpemiddel for å kontrollere at beregnet reiserute stemmer med kjørt rute. Kan denne informasjonen vises bedre på andre måter? Istedetfor å presentere ruta på kart, kan vi heller vise den i tekstformat. For eksempel: Bruker A kjørte fra Oslo til Trondheim, E6 til Hamar, om Østerdalen, og videre på E6 frem til destinasjonen.

2.1.4 Universell utforming

1.juli 2014 ble det innført nye regler for universell utforming av IKT-løsninger, «Retningslinjer for tilgjengelig webinhalt (WCAG) 2.0» [22]. Regelen innebærer at alle nye IKT-løsninger i Norge, både innen privat og offentlig sektor, organisasjoner og lag, skal være universelt utformet [23]. Kort sagt er dette en lovpålagt standard for utforming av IKT-løsninger. Eksisterende IKT-løsninger har frist til 21.januar 2021 på å rette seg etter kravene.

Kravene til universell utforming innebærer at IKT-løsninger skal være tilgjengelige for alle. Det skal tas hensyn med tanke på brukervennlighet slik at alle skal ha forutsetninger for å kunne bruke løsningen, også de som har nedsatt oppfattelsesevne i form av dårlig syn og hørsel og fargeblindhet. Brukere av vår applikasjon skal registrere reiseregninger på reiser de selv har kjørt, og det at de kan kjøre bil forusetter at de oppfyller en rekke krav med tanke på for eksempel syn.

Kravene gjelder alle nettsider uavhengig av tenkt brukergruppe, men fordi dette ikke er en side beregnet for svaksynte/blinde, holder det å oppfylle minimumskravene. Kriteriene er delt inn etter 4 prinsipper.

Prinsipp 1 - Mulig å oppfatte

Setter krav til bruken av kontraster, fargevalg, forstørrelsesmuligheter og bruk av tekst som alternativ til bilder, lyd og video.

Prinsipp 2 - Mulig å betjene

Setter krav til brukervennlighet i form av at brukeren fullt ut skal kunne bruke løsningen uavhengig av teknisk utstyr, blant annet slik at en løsning tenkt brukt med mus og tastatur, også skal være mulig å betjene kun via tastatur.

Prinsipp 3 - Forståelig

Innebærer at løsningene skal være forutsigbare, det skal være lett å forstå hvordan de skal brukes og informasjonen man finner skal være enkel å forstå. Krav til bruk av både enkelt språk og hjelpeTekster bidrar til dette.

Prinsipp 4 - Robusthet

Er rettet mot det tekniske. Her settes det noen retningslinjer i form av koding og tilgjengelighet. Innholdet må kunne tolkes på en god måte av forskjellig teknologi, nettsider må valideres, og koden må være riktig. I HTML blir som regel dette ivaretatt, men det er spesielt viktig å sikre god tilgjengelighet dersom man utvikler egne elementer (widgets).

Vi har studert retningslinjene, for å vite hvilke hensyn vi må ta under utformingen av applikasjonen. Vi har utelukket kriteriene som går på lyd og video, da disse ikke vil være aktuelle for vår del. Det er i tillegg en del av kriteriene som ikke vil treffe vår applikasjon direkte da vår applikasjon vil tilpasses eksisterende HRessurs. Vi tar utgangspunkt i at HRessurs allerede oppfyller disse kravene, eventuelt at disse oppfylles når HRessurs relanseres i ny versjon.

Ettersom dette er en applikasjon der brukeren skal angi informasjon, er det viktig for oss å dekke alle krav i forbindelse med inntasting av data. Brukere skal få kontrutive tilbakemeldinger dersom data mangler eller er fylt inn feil. Det er også krav til ledetekster, som innebærer at alle inndatafelt skal ha ledetekst eller instruksjoner som forteller hvilke data som skal angis hvor. Vi vil derfor ha en kort brukerveileddning i applikasjonen, slik at det ikke skal være tvil om hva som skal gjøres og hvilken funksjonalitet som er tilgjengelig.

Som nevnt ovenfor, forutsetter vi at en del retningslinjer er oppfylt overordnet i HRessurs. Dette er retningslinjene som dekker blant annet navigering på nettsidene, fargevalg og kontraster og at brukerne skal varsles på forhånd ved automatisk utlogging. Vi vil tilpasse farger og skriftstørrelser etter slik HRessurs er i dag, slik at applikasjonen skal fremstå som en del av HRessurs, og ikke som en separat «kalkulator». Om kravene til universell utforming er oppfylt i HRessurs eller ikke, kommer ikke vi til å studere nærmere. Nåværende HRessurs er ikke en ny webløsning etter 1.juli 2014, noe som innebærer at den bare omfattes at regelverket for eksisterende løsninger, slik at den først må følge reglene innen 2021. HRessurs skal, som vi vil gå nærmere inn på i kapittel 2.2.2, etterhvert lanseres i ny versjon. Dersom kravene til universell utforming ikke er oppfylt i dag, må de oppfylles ved relansering. Vi har retningslinjene i bakhodet slik at det ikke skal være andre elementer som må tilpasses i vår applikasjon enn de tilpassningene som også må gjøres i resten av HRessurs.

2.2 Tilgjengelig teknologi for å løse oppgaven

2.2.1 .NET

Vi har valgt å utvikle vår applikasjon innenfor Microsoft sitt .NET rammeverk. Vi har undersøkt mulighetene for å benytte oss av andre teknologier, språk og rammeverk som kanskje ville ha ført til at vi satt igjen med en mer åpen og frittstående applikasjon, men denne ville ikke kunne implementeres direkte inn i HRessurs uten tilpassninger. Vårt ønske er at oppdragsgiver skal kunne fase vår applikasjon inn i personalsystemet uten for mange tilpasninger. Infotjenester benytter i dag .NET i hele sin virksomhet. Personalsystemet HRessurs er som nevnt i kapittel 2.1.1 delt inn i flere kategorier, men felles for hele systemet er teknologien som ligger bak. HRessurs som webapplikasjon er laget i ASP .NET. Siden vårt system på sikt skal erstatte deler av dagens ordning for registrering av reise og utlegg er det naturlig at vi velger samme plattform som HRessurs allerede er basert på. Til hvilken grad applikasjonen vil erstatte eksisterende løsning, eller komme i tillegg til eksisterende løsning, avhenger av hvordan vår endelige applikasjon er utformet.

Innenfor .NET rammeverket er det ulike måter å strukturere en webapplikasjon slik som den vi skal lage. Innenfor .NET finnes det flere ulike typer klassebibliotek som er spesialisert mot ulike typer programvare. Den vanligste for web, er ASP.NET biblioteket. Innenfor ASP.NET finnes det igjen ulike plattformer for webutvikling, slik som MVC, web-api osv. Microsoft har planer om å knytte alle disse sammen i et nytt rammeverk som kalles ASP.NET MVC6.

2.2.2 ASP.NET vs ren HTML

I begynnelsen av prosjektet lagde vi en helt enkel testversjon med all logikk på klientsiden. Dette er ingen holdbar løsning, slik at mesteparten av logikken skal flyttes over på server for at systemet skal være mer vedlikeholdbart. I forhold til gjenbruk slipper man for eksempel å lage samme komponent flere ganger. Andre fordeler er at man løser «interoperability problems», altså at vi muliggjør at forskjellige systemer på forskjellige plattformer kan kommunisere med hverandre. En mulighet for å løse dette er å sette opp en Web Service som oppretter kontakt med Vegvesenets server. Det vil si at nettleseren sender en forespørsel til vår Web Service som så sender en forespørsel til Ruteplantjenesten. Når denne svarer får Web Servicen et svar tilbake, som så sender denne tilbake til klientside Javascript som presenterer dataene for brukeren.

Vi har både sett på hvordan man kan gjøre dette i Microsoft sitt ASP.NET rammeverk som gjør jobben med å snakke med en Web Service mindre krevende, men også hvordan dette kan la seg gjøre fra en HTML-side med Javascript. Microsoft sitt ASP.NET er ment å gjøre jobben med å lage en dynamisk nettside enklere. Det vil si at det å vedlikeholde innholdet på nettsiden skal være mer overkommelig. For eksempel hvis man vil inkludere en ny side (web forms/aspx) eller redigere innholdet i en meny. I teorien vil arbeidet med å «snakke» med en Web Service være enklere via web forms enn i HTML og Javascript. Microsoft introduserte muligheten for at serverkoden ligger i «filnavn.aspx.cs»-filer, mens statisk kode ligger i «filnavn.aspx»-filene. Dette vil si at all kode som skal behandle funksjoner basert på brukervalg behandles av filene med «aspx.cs»-endelse. Alle behandlete og kompilerte filer

i ASP.NET lagres i ulike kataloger som er tilgjengelig for alle sider på nettstedet. Det inkluderer også Web Services. Alle filer (inkludert WSDL-filer som bestemmer hvordan kommunikasjonen skal foregå) som refererer til en Web Service ligger i en egen katalog, web forms, slik at denne kan nås enkelt i koden fra alle sider.

Infotjenester er i ferd med å lansere HRessurs i ny drakt, og det innebærer ikke bare rent utseendemessig endringer, men også endringer i det tekniske. Tilbakemeldinger fra flere av utviklerne ved bedriften er at aspx-systemet som ligger bak dagens HRessurs begynner å bli uoversiktig og vanskelig å vedlikeholde. Mest sannsynlig vil neste generasjon HRessurs bestå av rene HTML-filer på klientsiden, mens selve funksjonaliteten fortsatt befinne seg i .NET baserte Web Services. Vårt dilemma blir da om vi skal fortsette å utvikle vår reiseregnsapplikasjon i det dynamiske aspx-biblioteket som teknisk sett skal gjøre jobben med denne typen webapplikasjon mer overkommelig, basert på at dagens ordning er laget i aspx, eller om vi skal utvikle for fremtidens HRessurs og dermed tilpasse oss Infotjenesters behov i større grad. Vi ønsker å utvikle en applikasjon som faktisk vil bli tatt i bruk, og som vil være i bruk en god stund framover. Dette er hovedgrunnen for at vi velger å rette oss inn mot fremtidens HRessurs ved å bruke ren HTML og asmx-Web Service istedenfor ASP.NET.

3. Veien mot det endelige resultatet

I kapittel 3.1 tar vi for oss prosessen fram mot endelig brukergrensesnitt. I kapittel 3.2 tar vi for oss hvordan vi har kommet fram til, og hvordan vi har løst den tekniske delen av applikasjonen.

3.1 Brukergrensesnitt

3.1.1 Planlegging av brukergrensesnitt

Hovedtanken bak applikasjonen vi skal utvikle er å gjøre utfylling av reiseregninger både raskere og enklere. Vi vet derfor at vi må ha minst mulig knapper og at funksjonaliteten bør være selvforklarende. Det viktigste som må være med, er at bruker må ha mulighet til å angi start, stopp, viapunkter og valgt kjøretøy. Vi må ha en knapp for å starte beregningen og en knapp som legger beregnet informasjon inn i riktige felter i HRessurs. Ved behov for eventuelle andre elementer må vi veie funksjonaliteten disse elementene tilfører opp mot hvordan disse vil påvirke brukervennligheten og hvor nødvendige de er i vår applikasjon.

Figur 3.1: Tidlig skisse på brukergrensesnitt

Da vi fikk tilgang til HRessurs lagde vi en skisse over hvordan vi så for oss at vår applikasjon ville passe inn i dagens HRessurs. Se kapittel 2.1.1 for nærmere beskrivelse av HRessurs. I HRessurs er det vanlig at all funksjonalitet ligger på venstre side, med forklarende tekster på høyre side. Vi så for oss at vår applikasjon skulle være en slags tilleggskalkulator man kunne velge å åpne ved siden av dagens utfylling. Vi lagde derfor også en skisse over hvordan det kunne se ut. Dette er illustrert i skissen i figur 3.1. Her er tanken at bruker trykker på «Kalkulator»-knappen for å åpne applikasjonen til høyre for utfyllingen.

3.1.2 Inndata fra bruker

Vi ønsker at brukerne skal fylle inn minst mulig informasjon når de registrerer reiseregninger, samtidig som feilmarginen i forhold til avstand/bompengekostnader i registrerte reiser skal gå ned. Ruteplantjenesten beregner ruten automatisk basert på den raskeste veien, men det er ikke sikkert denne ruten alltid stemmer overens med virkeligheten. For eksempel kan man måtte kjøre omveier som følge av stengte veier, eller plukke opp kollegaer underveis på reisen. Disse omkjøringene må det være mulige for brukeren å legge til, slik at den beregnede ruta blir riktig. Vi har vært gjennom en lang prosess for å få viapunktene mest mulig brukervennlig og funksjonelle. Hele prosessen rundt viapunkter er beskrevet i vedlegg B - Notater fra utviklingsprosessen, kapittel 1.5. Et sammendrag av den tekniske biten finnes i kapittel 3.2.3.

Brukeren skal også kunne sortere viapunktene, dersom det legges inn to eller flere felter. For eksempel hvis brukeren oppdager at han har lagt inn steder han har kjørt via i feil rekkefølge. Da skal han slippe å fjerne teksten i feltene for å fylle de inn på nytt i riktig rekkefølge, og heller kunne endre rekkefølgen på stedene som er angitt. For å løse dette har vi testet løsninger der bruker kan flytte tekstfeltene direkte. Her har det dukket opp ulike problemer med de ulike mulighetene vi har testet. I det endelige resultatet lagrer vi viapunktene i en sorterbar liste. Alle endringer i brukergrensesnittet i forhold til viapunktene henger nøye sammen med de kodemessige utfordringene vi har støtt på. Utfordringene er nærmere beskrevet i kapittel 3.2.3.

Når bruker angir adresser for start, stopp og viapunkter, bruker vi Google sitt autofullførings-API, kalt Google Autocomplete, til å gi brukeren forslag på ulike stedsnavn. Ved bruk av Google Autocomplete får vi også tilgang til stedets koordinater. Disse trenger vi da Ruteplantjenesten beregner ruter basert på koordinater fremfor stedsnavn. Google og Ruteplantjenesten benytter ulike koordinatsystem og vi må derfor ha en kalkulator som konverterer fra det ene til det andre koordinatsystemet før koordinatene kan benyttes. Mer om dette i kapittel 3.2.4.

Figur 3.2 viser inndatafeltene ved oppstart av applikasjonen til høyre. Til venstre viser vi hvor viapunktfeltet plasserer seg i forhold til start/stopp-feltene.

The figure displays two side-by-side user interface screens for route planning, separated by a vertical line.

Left Screen (Initial State):

- Kjøretøy:** A section with four radio buttons representing different vehicle types: bicycle, car, bus, and truck.
- Start:** An input field labeled "Angi startsted".
- Stopp:** An input field labeled "Angi stopsted".
- Eventuell kommentar:** An empty text area for comments.
- Buttons:** Two blue buttons at the bottom: "Legg til viapunkt" (Add waypoint) and "Beregn" (Calculate).

Right Screen (Advanced State):

- Start:** An input field labeled "Angi startsted".
- Viapunkt:** An input field labeled "Angi viapunkt ved behov". To its right are two small blue buttons with "+" and "-".
- Stopp:** An input field labeled "Angi stopsted".

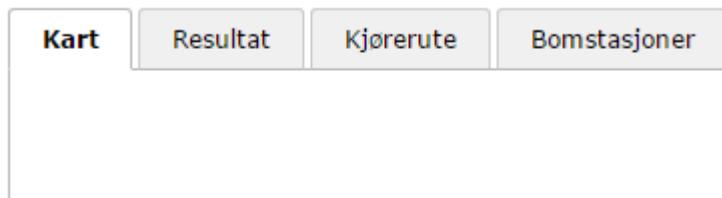
Figur 3.2: Inndata fra bruker

3.1.3 Fremstilling av beregnet rute

I dette kapittelet vil vi gå nærmere inn på hvordan vi kan fremstille den beregnede ruten for bruker slik at bruker best mulig kan kontrollere at opplysningene stemmer.

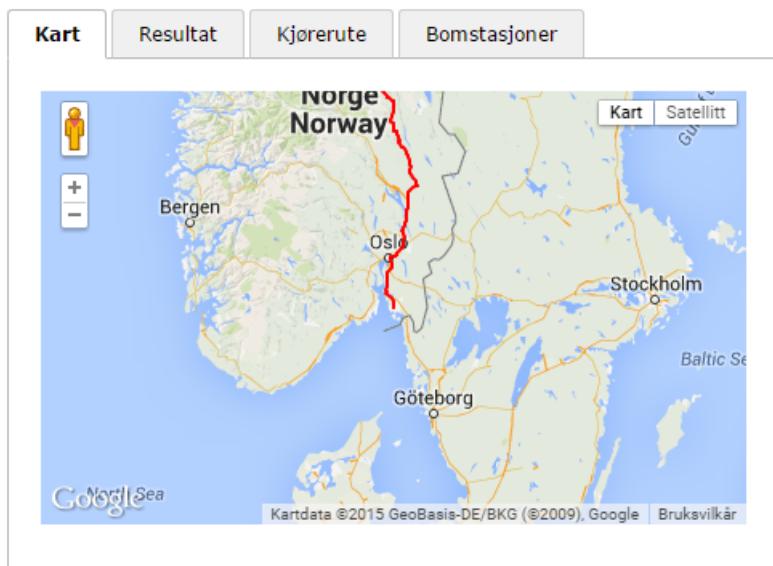
Beregnet resultat

Etterhvert som vi har jobbet med prosjektet, har vi innsett at vi må ha større plass til visning av beregnede ruter enn vi hadde sett for oss i figur 3.1. Vi har valgt å fremstille resultatet av ruteberegningen i en fanebasert visning, se figur 3.3, da dette er en god måte å fremstille mye data på en ryddig måte. Den fanebaserte visningen vår er delt i fire kategorier: kart, resultater, kjørerute og bomstasjoner.



Figur 3.3: Visning av resultater med faner

I den første fansen, Kart (se figur 3.4), som er den fanen man starter i, vises den beregnede kjøreruten grafisk. Her kan man panorere kartet og zoome ut og inn for å få en god oversikt over ruten som er beregnet.



Figur 3.4: Første fane - kart

I fane nummer to, Resultat (se figur 3.5) viser vi resultatet av beregningen. Her har vi lagt inn beregnet avstand, total tidsbruk, de totale bompengekostnadene, og eventuelle kommentarer som er angitt i kommentarfeltet. Vi oppsummerer også start, stopp og viapunkter, slik at bruker raskt og enkelt kan kontrollere at alle angitte steder er med i beregningen.

The screenshot shows a user interface with four tabs at the top: Kart, Resultat (highlighted in red), Kjørerute, and Bomstasjoner. The main content area displays the following information:

- Avstand:** 582km
- Total tid:** 504min
- Bompenger:** 199kr
- Du har kjørt strekningen:**
 - Start: gressvik
 - Stopp: Trondheim S, Trondheim, Norge
- Kommentar:** (empty text area)

Figur 3.5: Andre fane - resultat

I fane nummer tre, kjørerute (se figur 3.6) viser vi den komplette kjørebeskrivelsen. Denne beskrivelsen egner seg ikke uten å kunne følge ruten på kart, men vi har valgt å ha den med slik at de som ønsker, også har valget om å kontrollere ruten ved hjelp av kjørebeskrivelsen.

The screenshot shows a user interface with four tabs at the top: Kart, Resultat, Kjørerute (highlighted in blue), and Bomstasjoner. The main content area displays the following driving instructions:

- Kjørebeskrivelse**
P98833
Start på privat veg.
- K71560**
Ta til høyre på Stangebyeveien,
- F117**
Ta skarpt til venstre på Storveien,
- R110**
I rundkjøring, ta andre avkjørsel og fortsett på Rv110, I rundkjøring, ta andre avkjørsel og fortsett på Rv110, I rundkjøring, ta andre avkjørsel og fortsett på Rv110,
- E6**
I rundkjøring, ta første avkjørsel og fortsett på E6, I rundkjøring,

Figur 3.6: Tredje fane - kjørerute

I fane nummer fire, Bomstasjoner (se figur3.7), lister vi opp alle bomstasjoner på ruta, i tillegg til prisen for valgt kjøretøy. Dersom valgt kjøretøy er motorsykkel, opplyser vi om at bompasseringer er gratis.

Bomstasjon	Pris
Raukerud	23
E6 Europavegen	31
HOVINMOEN	19
Andelva	19
Tømte	15
Ørbekk	20
Espa	21
Kolomoen	23
Klett-E6	10
E6 Tonstad	8
Bjørndal	10

Figur 3.7: Fjerde fane - Bomstasjoner

Brukergrensesnitt i sin helhet

Figur 3.8 viser brukergrensesnittet i sin helhet med inndata til venstre og fremstilling av beregnet kjørerute til høyre.

Dobbelsjekk at all informasjon stemmer i forhold til reiseruten
Hvis reiseruten er korrekt, trykk bekreft. Hvis ikke, prøv å legge til et eller flere viapunkter og beregn rute på nytt

Figur 3.8: Det helhetlige brukergrensesnittet

Tekstlig fremstilling

Ruteplantjenesten er i hovedsak en ruteplanleggingstjeneste. Dette innebærer at innholdet tjenesten returnerer, bærer preg av å være tiltenkt reiseplanlegging og ikke dokumentering av reiser i ettertid. Figur 3.9 viser et utklipp av en veibeskrivelse fra Halden til Drammen.

K3560, Ta til venstre på Jørgen Bjelkes Gate
K2880, Ta til høyre på Hannibal Sehesteds Gate
F22, Ta til høyre på Fv22
R21, Ta til venstre på Rv21
R21, I rundkjøring, ta første avkjørsel og fortsett på Rv21
R21, I rundkjøring, ta andre avkjørsel og fortsett på Rv21
R21, I rundkjøring, ta andre avkjørsel og fortsett på Rv21
E6, Hold til høyre på E6
R23, I rundkjøring, ta tredje avkjørsel og fortsett på Rv23
R23, I rundkjøring, ta andre avkjørsel og fortsett på Rv23
R23, I rundkjøring, ta første avkjørsel og fortsett på Rv23
R23, I rundkjøring, ta andre avkjørsel og fortsett på Rv23
R23, I rundkjøring, ta første avkjørsel og fortsett på Rv23
R23, I rundkjøring, ta andre avkjørsel og fortsett på Rv23
R23, I rundkjøring, ta andre avkjørsel og fortsett på Rv23
R23, I rundkjøring, ta andre avkjørsel og fortsett på Rv23

Figur 3.9: Tekstlig beskrivelse av kjøreruta.

Med mindre man har et kart man kan følge med på er ikke dette en god nok presentasjon av ruta når man i ettertid skal kontrollere at ruta som er beregnet stemmer overens med hvor man har kjørt. Det brukes veldig få stedsnavn i beskrivelsen, og der det gjøres er det kun gatenavn på småveier. Dette innebærer at gatenavnene kun er til hjelp om man er lokalkjent, eller har mulighet til å slå opp gatenavnene i et kart. I det store og hele kan man få en oversikt over hvor den beregnede ruta går, dersom man kjenner veinummeret på de store veiene man har kjørt. Antall tekstlinjer per veinummer kan skape forvirring, da disse ikke har sammenheng med hvor lenge man har kjørt på veien. På kjøreturen fra Halden til Drammen kjører man på E6 i ca 1 time, og E6 har kun en tekstlinje i beskrivelsen. På E6 holder man kun rett fram fra man kjører inn på veien og til man tar av veien igjen. Riksvei 21, veien ut av Halden, har hele fire tekstlinjer ettersom veien inneholder flere rundkjøringer. Der kjører man til sammenligning bare i ca 10 minutter.

For å kunne oppfylle kravet til reiseregninger om at kjøreruta skal være spesifisert tekstlig, er vi nødt til å komprimere veibeskrivelsen, slik at vi kan presentere kun de viktigste punktene på ruta. Det er ikke relevant hverken for brukeren selv eller de som skal kontrollere reiseregningene å vite alle småveier brukeren har kjørt på. Det er heller ikke relevant å vite om man har kjørt gjennom 15 rundkjøringer på R23. Det er viktigst å vite at man har fulgt R23. Det er også mer relevant å vite de store veiene fremfor småveier, da de store veiene gir en overordnet og god nok oversikt over hvor brukeren har kjørt.

For å komme frem til en bedre tekstlig beskrivelse av ruta er det første som må gjøres å gruppere veibeskrivelsen kronologisk etter veinummer. Utskriften av veibeskrivelsen blir da seende ut slik som i figur 3.10.

K2880
Ta til høyre på Hannibal Sehesteds Gate,
F22
Ta til høyre på Fv22,
R21
Ta til venstre på Rv21, I rundkjøring, ta første avkjørsel og fortsett på Rv21, I rundkjøring, ta andre avkjørsel og fortsett på Rv21, I rundkjøring, ta andre avkjørsel og fortsett på Rv21,
E6
Hold til høyre på E6, I rundkjøring, ta andre avkjørsel og fortsett på E6,

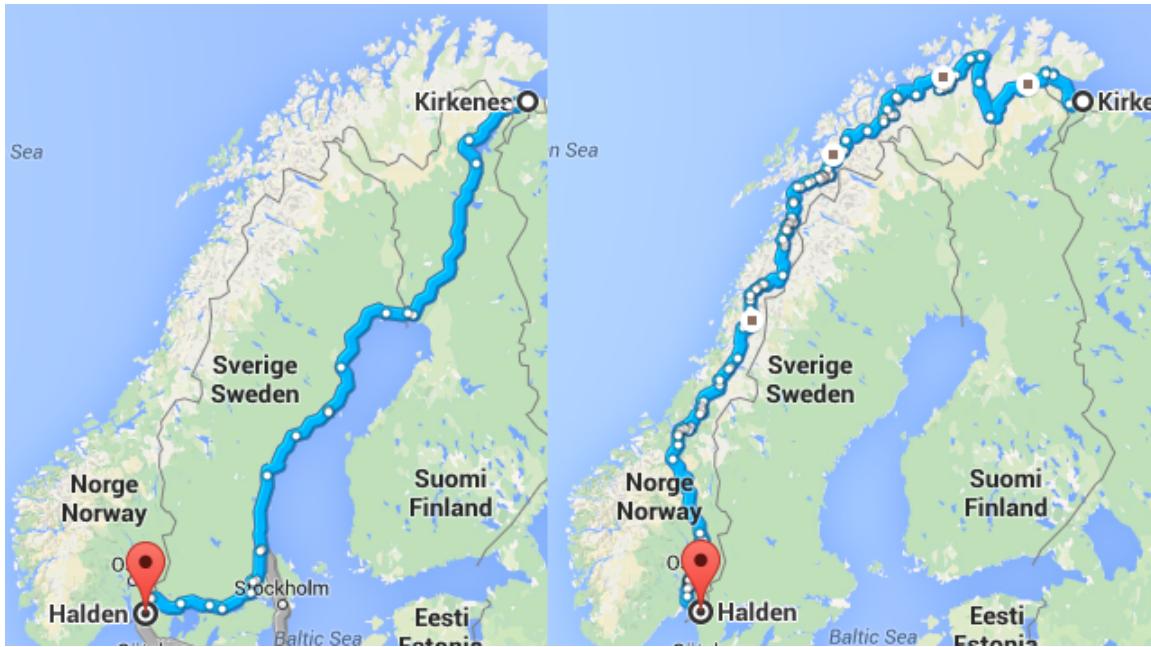
Figur 3.10: Gruppert, tekstlig beskrivelse av kjøreruta.

Å gruppere etter veinummer gir med ett en bedre oversikt over hvor man har kjørt. For brukeren som skal kontrollere reiseruta vil dette vært et mye bedre alternativ enn beskrivelsen i figur 3.9, men det er fremdeles ikke en god nok beskrivelse til at vi ikke trenger å ha med en kartvisning i applikasjonen. For den kortfattede rutebeskrivelsen som skal lagres i reiseregningene beholder vi kun veinumrene på riksveier, fylkesveier og europaveier, og utelater private og kommunale veier (veier med et veinummer som starter med K eller P). Den kortfattetde veibeskrivelsen for kjøreruta i figur 3.10 blir da: «Kjørerute via F22, R21 og E6». Denne må så settes i sammenheng med angitt start og stopp, slik at den endelige beskrivelsen som lagres i reiseregningene blir: «Kjørt fra Halden til Drammen, kjørerute via F22, R21 og E6».

Grafisk fremstilling

Som nevnt i kapittel 2.1.3 - Andre muligheter, foretar Ruteplantjenesten kun beregninger av kjøreruter innenfor Norges grenser, mens Google Maps ikke tar hensyn til landegrenser når kjøreruter beregnes. Dette skaper problemer med tanke på å plotte kjøreruta direkte på et kart fra Google, kun basert på angitt start, stopp og eventuelle viapunkter. Både Ruteplantjenesten og Google Maps beregner raskeste vei, men grunnet landegrenseproblematikken vil vi ikke ha noen garanti for at kjørerutene beregnes likt. Ved å plotte direkte vil de beregnede tallverdiene som presenteres bruker, være beregnet av Ruteplantjenesten, mens ruta som vises på kartet er beregnet av Google.

Det vil være store avvik i disse rutene på for eksempel en kjøretur mellom Halden og Kirkenes, siden Google tar oss via Sverige. I figur 3.11 er ruten mellom Halden og Kirkenes illustrert. Google angir ikke veien gjennom Norge som et alternativ en gang da denne er vesentlig lengre enn alternativene gjennom både Sverige og Finland. Vi må manuelt dra kjøreruten gjennom Norge for å kunne illustrere rutevalget gjennom Norge.

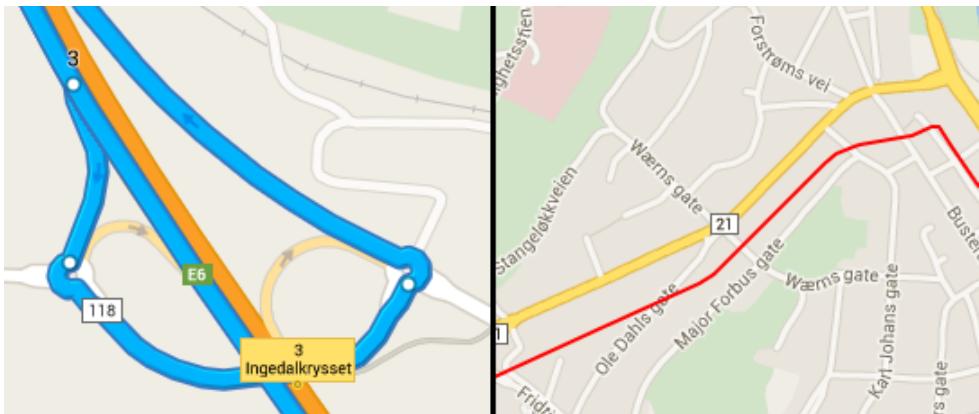


Figur 3.11: Halden - Kirkenes, to ulike veivalg

Det er derimot enda et problem, som tilsier at vi ikke kan ta sjansen på at begge beregner samme rute. Ruteplantjenesten inneholder informasjon fra Vegvesenets trafikkmeldinger om stengte veier, omkjøringer osv. Det kan av den grunn føre til at Google Maps viser raskeste veien uten å ta hensyn til omkjøringer, mens avstanden og bompengekostnadene som beregnes, skjer på grunnlag av en omkjøring Google ikke tar hensyn til. Vi er derfor nødt til å tvinge Google Maps til å legge kjøreruta innom punkter underveis på ruta, slik at ruta som fremstilles av Google er så lik den beregnede ruta som mulig. I kart fra Google har vi to muligheter for å gjøre dette, Polyline og viapunkter. Etter mye testing har vi landet på å benytte oss av Google Polyline for å fremstille kjøreruta på kart. Vi vil nå gå nærmere inn på hvilke argumenter som ligger til grunn for at vi velger Polyline fremfor viapunkter.

Polyline er en funksjon som trekker linjer mellom koordinater, slik at vi er avhengig av nokså mange koordinater for å få en best mulig fremstilling av ruta. Googles viapunktfunksjon kan plotte inntil 8 viapunkter (noen flere ved å kjøpe tilgang). Viapunkter kan angis både ved hjelp av stedsnavn og koordinater. Da Ruteplantjenesten, som beskrevet i kapittel 3.1.3, kun gir tekstlig veibeskrivelse med få stedsnavn, er vi nødt til å plotte disse basert på koordinater. Til viapunkter plukker vi ut 8 viapunkter jevnt fordelt over kjøreruta, mens vi til Polyline benytter oss av alle koordinater. Ruteplantjensten returnerer koordinatene på formen compressed geometry. Dette innebærer at vi både må dekomprimere koordinatene og konvertere de til riktig format før vi kan plotte resultatet på kartet. Se kapittel 3.2.4 for en nærmere beskrivelse av koordinathånderingen.

Vi får en del uventede problemer den første tiden vi prøver å plotte både Polyline og viapunkter. Det begynte med at vi hadde en ustabil løsning, der kjøreruta noen ganger ble plottet ved hjelp av viapunkter, andre ganger ikke. Dersom koordinater havnet i motsatt kjørefelt, for eksempel på E6, opplevde vi at kjøreruta tok en rundtur fram og tilbake mellom avkjørslene. Tilsvarende fikk vi en konsekvent forskyvning gjennom hele ruta da vi testet Polyline. Figur 3.12 illustrerer hvordan rutene ble fremstilt. Feilen viste seg å ligge i koordinatkonverteringen, og da vi fikk løst denne feilen (se kapittel 3.2.4), fikk vi et bedre perspektiv på hvordan Polyline og viapunkter takler koordinatfeil. Dette var avgjørende da vi tok det endelige valget med hvordan vi skal fremstille kjøreruta.

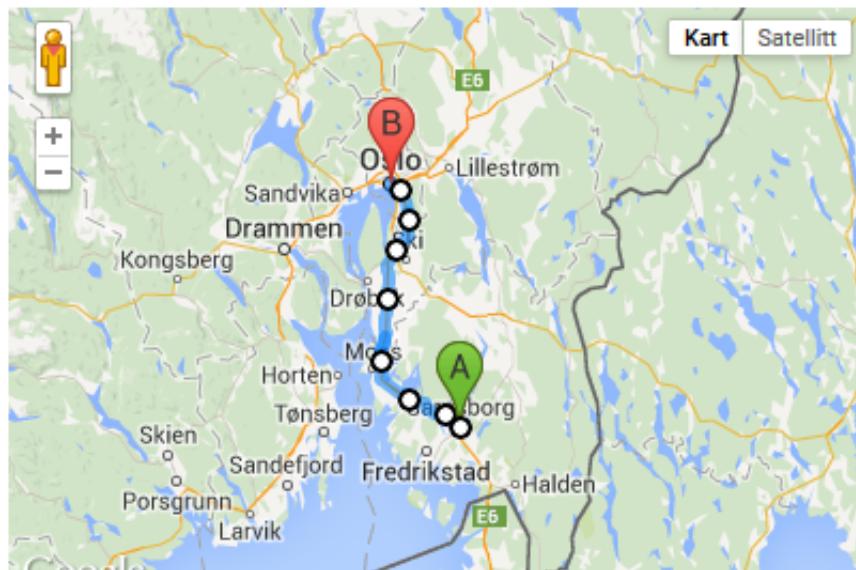


Figur 3.12: Koordinatfeil. Google viapunkter t.v., Polyline t.h

Som vi merket da vi testet å bruke Google viapunkter, hender det at Google ikke klarer å tegne inn kjøreruta, dersom for eksempel et koordinat ligger litt for langt utenfor veien. Etter å ha løst problemet med at koordinatene forskyves, er ikke dette lenger like kritisk. Til tross for riktige koordinater, kan det fremdeles skje at et koordinat havner for langt utenfor veien til at Google klarer å plotte det på kartet. Et par eksempler er hvis Google og Ruteplantjenesten har ulike oppfatninger av hva som defineres som en kjørbar vei, eller ved avsidesliggende gårder og lignende. En annen ulempe med viapunktene, siden man bare kan angi åtte viapunkter, er at vi ikke har noen kontroll på hva som skjer mellom disse åtte punktene. På en strekning fra Halden til Oslo vil ikke dette være noe problem da avstanden mellom punktene er liten og veivalgene mellom punktene er gitt av raskeste vei. Dersom man skal fra Halden til Kirkenes, blir det adskillig lengre mellom viapunktene, og ruta kan bli annerledes. Vi har heller ingen garanti for at Google og Ruteplantjenesten har samme definisjon for veistandarden de beregner raskeste rute via. Vi har ingen garanti for at begge beregner en ferjeforbindelse som det beste alternativet, dersom et alternativ uten ferje vil kunne være nesten like godt.

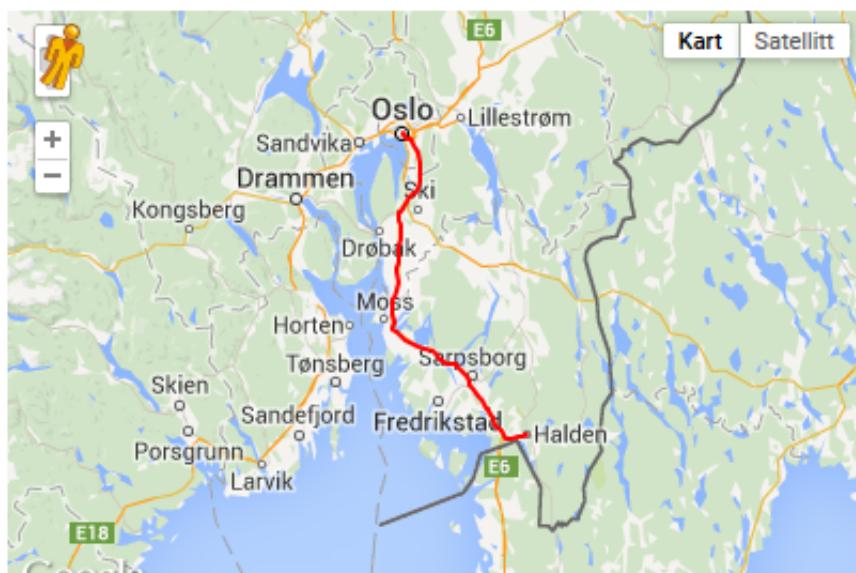
Vi ender til slutt opp med å bruke Polyline for å fremstille den beregnede ruta grafisk. Å bruke Polyline gir flere fordeler med tanke på at vi kan være sikre på at ruta vises slik den skal, og at den alltid vises. Dersom vi bruker Polyline, spiller det ingen rolle at linja havner utenfor veien, den vises uansett. Ved å bruke alle punktene fra Ruteplantjenesten blir det tett nok mellom punktene til at selv svingen gjennom en rundkjøring kommer tydelig frem på kartet. Ulempen er at jo lengre ruta er, jo lengre tid vil det ta å beregne koordinatene. Det er kun snakk om noen få sekunder, slik at dette ikke blir noe stort problem. Ved bruk av Polyline trengs alle koordinatene for å få en nøyaktig linje, og vi anser viktigheten av en korrekt rutevisning som alltid fungerer for å være større enn at det må gå veldig raskt.

Utseende på den plottede kjøreruta har også betydning for at vi velger Polyline framfor viapunkter. Figur 3.13 viser hvordan kjøreruta ser ut etter å bli plottet med Google viapunkter. På en kort strekning slik som Sarpsborg – Oslo, der de åtte punktene har en kort avstand mellom hverandre, fremstår viapunktene som forstyrrende elementer. På en lengre strekning vil punktene fordeles utover et større område, slik at de ikke fremstår like forstyrrende.



Figur 3.13: Sarpsborg - Oslo, fremstilt med Google viapunkter

Figur 3.14 viser strekningen Halden – Oslo, plottet ved hjelp av Polyline. Her ser kjøreruta mer oversiktlig ut, og den er på denne strekningen på 119km plottet gjennom ca. 1300 koordinater. Dette tilsvarer ca. et koordinat pr 10m.

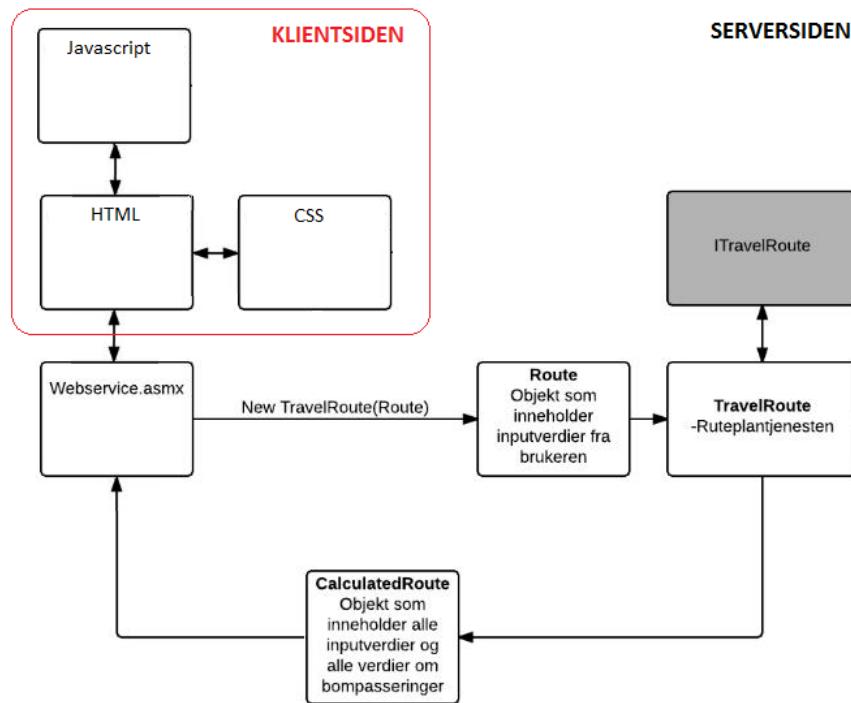


Figur 3.14: Halden - Oslo, fremstilt som Polyline

3.2 Teknisk

3.2.1 Planlegging av struktur

Til vår applikasjon er det en rekke ønsker fra oppdragsgiver. Det er et ønske om at applikasjonen skal være bygd etter en lignende struktur som HRessurs har i dag, med brukergrensesnitt på klientsiden (også kalt frontend), og en serverside (backend) der alle beregninger foregår. Figur 3.15 viser den planlagte strukturen. Dette er både med tanke på vedlikeholdbarhet og for at Infotjenester skal ha en bedre mulighet til å overvåke trafikken på serveren. Da vil de ha mulighet til å logge eventuelle feil som oppstår og dermed enklere kunne finne feilene og rette dem. Vi står relativt fritt til å bestemme hvordan vi ønsker å gjøre dette, men for at vi ikke ender opp med å utvikle noe som er helt forskjellig fra HRessurs, tar vi imot forslag og veiledning før vi går i gang med denne delen av prosjektet. Vi får ett krav, og det er at alle beregninger som kan foregå på en server, skal foregå på en server.



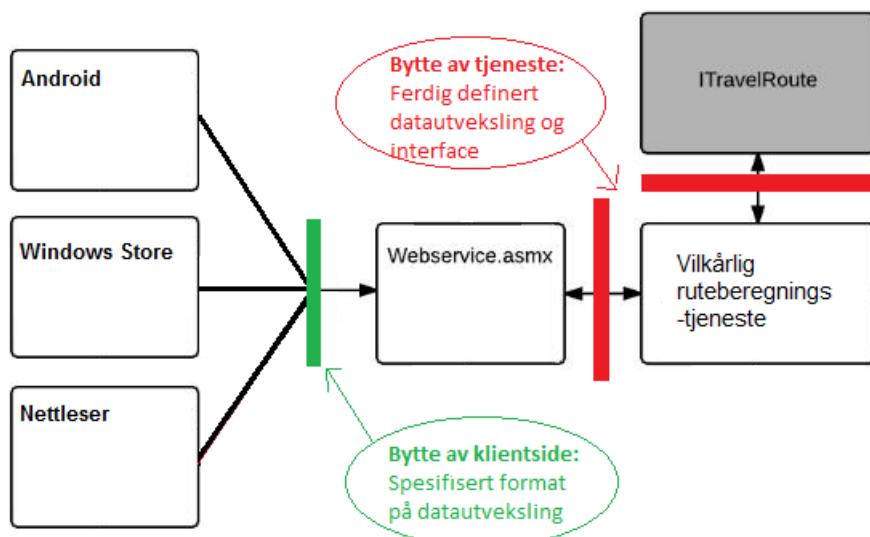
Figur 3.15: Vedlikeholdbar struktur

På serveren skal alle beregningsrutiner være skrevet i C#. På klientsiden skal brukeren angi all informasjon som er relevant for beregningene. Klientsiden vil inneholde en HTML-side og en CSS-fil der vi tilpasser utseendet på HTML-siden. I tillegg har vi flere Javascriptfiler. Javascript brukes for å hente og kontrollere alle data som fylles inn av brukeren, og sende disse dataene til serveren. Når responsen fra serveren kommer, vil vi også behandle responsdata i Javascript, slik at vi kan presentere de beregnede dataene for brukeren. På serveren vil vi ha en Web Service som håndterer kommunikasjonen mellom klient og server, denne er nærmere beskrevet i kap 3.2.2.

Fordeler med valgt struktur

Noe av hovedtanken bak en slik lagdeling av applikasjonen er at alle deler skal være utskiftbare. Dette innebærer at alle delene applikasjonen består av, skal kunne byttes ut og erstattes av en tilsvarende del, uten at dette innebærer endring i de resterende delene i prosjektet. For at dette skal være mulig, må det ligge en grunnleggende struktur i bunn. Dette vil i praksis si at det skal være mulig å bytte ut Ruteplantjenesten med en annen ruteberegningsleverandør eller nettsiden med en Windows Store- eller Androidapp.

Det kan hende at 1881 etterhvert vil tilby et API for ruteberegningsjenester, mens Ruteplantjenesten kutter ut sin tjeneste. Byttet av tjenestetilbyder kan da gjennomføres uten at andre deler av prosjektet må endres. Ved å fastsette hvilke parametere som mottas fra klientsiden, og hvilke parametere som returneres tilbake til klientsiden, spiller det ingen rolle for de resterende deler av applikasjonen om beregningene foregår ved hjelp av Ruteplantjenesten eller 1881. Web Service kontrollerer at inputdata fra klientsiden er på riktig format, og ved at klassen TravelRoute (der beregninger foregår) implementerer interfacet ITravelRoute, vil vi være sikret at denne strukturen opprettholdes. Dette er illustrert med rødt i figur 3.16.

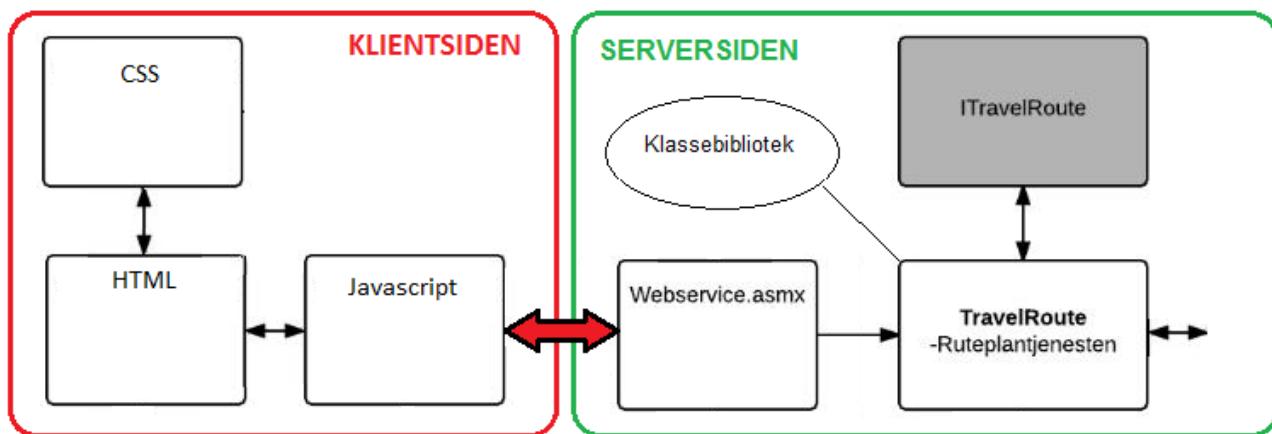


Figur 3.16: Fordeler ved lagdeling

Dersom man ønsker å lage en annen type applikasjon på klientsiden kan man også bytte ut denne uten at resten av koden på serversiden må endres. Hvis det er fastsatt at Web Service skal ta i mot stedsnavn og koordinater på et bestemt koordinatformat i tillegg til kjøretøy, kan nettsiden byttes ut med en Windows Store App eller Androidapp uten at det må gjøres endringer på serversiden. Dette forutsetter at parameterne sendes inn på formatet som er spesifisert. Dette er illustrert med grønt i figur 3.16. Her viser den grønne streken at data må være på et spesifisert format når data passerer dette punktet.

3.2.2 Kommunikasjon mellom klient og server

Som beskrevet i kapittel 3.2.1, har vi bestemt oss for å dele applikasjonen inn i flere lag. Hovedskillet mellom disse lagene ligger mellom brukergrensesnittet på klientsiden der brukerdata angis, og serveren der alle beregninger foregår. For å kommunisere mellom klient og server benytter vi oss av Web Service. Fra Javascriptrutinen på klientsiden vil vi foreta en request mot Web Service på Serversiden. Denne requesten vil være selve bindeleddet i applikasjonen, slik som figur 3.17 viser. Det viser seg å være tidkrevende å få dette til å fungere optimalt da vi aldri har jobbet med Web Service før, men vi lander til slutt på å bruke en AJAX-request. Hele prosessen med å få dette til å fungere, er utdypet i vedlegg B - Notater fra utviklingsprosessen, kapittel 1.3 og 1.4.



Figur 3.17: Kommunikasjon klient - server

For å finne ut hvordan vi best mulig skulle løse denne kommunikasjonen mellom klient og server testet vi blant annet Microsoft useService, som viste seg å være foreldet. Vi testet også XMLHttpRequest, men da fikk vi problemer med at denne kun returnerte XML. Navnet indikerer at en XMLHttpRequest har noe med XML å gjøre, men det skulle være mulig å tvinge den til å returnere JSON istedenfor. Dette fikk vi ikke til å fungere, slik at den endelige løsningen ble å benytte oss av en AJAX-request. Her var mulighetene bedre både med tanke på å sende data til serversiden, og vi fikk data returnert som JSON.

Til å begynne med benyttet vi oss kun av testdata for å få kommunikasjonen mellom klient og server opp å gå. Hovedfokuset var at data skal kunne sendes begge veier og returneres som JSON. Data skal sendes fra klientsiden, som parameter i AJAX-requesten og til serversiden via Web Service. Vi startet med å sende over testparametere og returnere testresultater. Gradvis utvidet vi både AJAX-requesten til å sende inndata fra bruker til Web Service, og Web Service til å ta i mot denne informasjonen og foreta beregninger av kjørerute. Strukturen på objektene som sendes til og fra Web Service og hvordan beregninger foregår på serveren vil vi komme nærmere tilbake til i kapittel 5.1.2.

3.2.3 Viapunkter

Vi har gjennom hele perioden forsøkt oss på flere måter å lage en god løsning for viapunkter. Hele prosessen er utdypet i vedlegg B - Notater fra utviklingsprosessen, kapittel 1.5. Den første løsningen vi så på innebar at vi bruker designprinsipper, CSS, kombinert med Javascript. Vi hadde da HTML-tekstfelt som var synlig i koden, men ikke for brukeren. Dersom brukeren ønsket å legge til viapunkter måtte han trykke på en «legg til» knapp som gjorde feltene synlige og klare til å fylles inn. En ulempe her var at vi da ville ha mange tekstfelt som kanskje ikke brukeren har behov for, og de skjulte tekstfeltene ville fremdeles oppta plass på siden. Vi konkluderte med at dette ville bli en uoversiktlig og forvirrende løsning for bruker.

Den andre løsningen vi prøvde innebar å kun holde seg til Javascript, og droppe CSS helt. Ved hjelp av Javascript kan man lage nye HTML-elementer. Vi knyttet en funksjon opp mot «legg til»-knappen som lager ett nytt tekstfelt for hver gang knappen trykkes. Dermed ville brukeren kunne legge til kun de feltene han har bruk for, samt at det er lett forstålig hvordan det skal gjøres. Figur 3.18 viser de to første løsningene for viapunkter. Rute 1 og 3 viser bruk av CSS og skjulte tekstfelt, mens rute 2 og 4 viser bruken av Javascript til å lage nye tekstfelt.

<p><input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/></p> <p>Angi startsted</p> <p>Angi stopsted</p> <p>Send rute</p>	<p>Legg til viapunkt</p>
1	
<hr/>	
<p><input checked="" type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/></p> <p>Angi startsted</p> <p>Angi Viapunkt</p> <p>Angi Viapunkt</p> <p>Angi Viapunkt</p> <p>Angi stopsted</p> <p>Send rute</p>	<p>Legg til viapunkt</p>
2	
<hr/>	
<p><input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/></p> <p>Angi startsted</p> <p>Angi stopsted</p> <p>Nytt viapunkt</p> <p>Angi viapunkt</p> <p>Angi viapunkt</p> <p>Angi viapunkt</p> <p>Beregn</p>	
3	
<hr/>	
<p><input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/></p> <p>Angi startsted</p> <p>Angi stopsted</p> <p>Nytt viapunkt</p> <p>Angi viapunkt</p> <p>Angi viapunkt</p> <p>Angi viapunkt</p> <p>Beregn</p>	
4	

Figur 3.18: Visning av viapunkter

Selv om det tilsvynelatende virker mer fornuftig å lage nye tekstfelter når brukeren vil legge til nye viapunkter, oppstår det store problemer på klientsiden i enkelte nettlesere, i tillegg til at det blir vanskelig å få Google Autocomplete til å fungere. Det første problemet som oppstår ved denne løsningen begrenser seg til nettleseren Mozilla Firefox, men er et såpass alvorlig problem at vi ikke kan overse det. Dersom en bruker lager nye tekstfelt ved å trykke «Legg til»-knappen, får han ikke muligheten til å trykke seg direkte inn i selve tekstfeltet med musepekeren. Dette fungerer i start- og

stoppfeltene som alltid eksisterer, men ikke det nye inputfeltet som legges til etter at nettsiden lastes inn første gang. Den eneste måten brukeren har for å kunne skrive noe som helst i viapunktfeltene er ved å bruke tabulatorknappen på tastaturet. Har man skrevet feil i et viapunktfelt må man bruke tabulatorknappen for å navigere seg gjennom alle elementer på siden før man kommer tilbake til det elementet man vil redigere. Funksjonaliteten er også testet i Google Chrome og Internet Explorer uten problemer, men siden Firefox er en såpass utbredt nettleser kan vi ikke overse denne feilen. Dette problemet oppstår som følge av at vi kombinerer Google Autocomplete i nye tekstfelt med en sorteringsfunksjonalitet som vi kommer tilbake til i de påfølgende avsnittene.

Vi støter også på en annen alvorlig feil som får applikasjonen til å fungere dårlig. Denne har ikke direkte å gjøre med hvordan viapunkter legges til, men problemet kan løses ved å endre måten dette gjøres på. La oss si en bruker vil legge til fem viapunkter, slik at det inkludert start og stopp blir sju steder der koordinater må hentes og konverteres. Koordinatene må hentes fra Google Autocomplete, før disse konverteres av vår konverteringsfunksjon som er beskrevet i kapittel 3.2.4. Det finnes få muligheter for å hente ut koordinatene etterhvert som steder angis, slik at vi er nødt til å hente alle koordinater når «Beregn»-knappen trykkes. Det tas i liten grad hensyn til at de ulike tjenestene kan bruke lang tid på å svare, slik at dette kan føre til at applikasjonen vil kunne stoppe. Problemet med at applikasjonen stopper har oppstått flere ganger, rett og slett fordi for mye informasjon behandles på en gang. Her er det i hovedsak i Google Autocomplete og uthenting av mange koordinater samtidig at feilen ligger. Ved å kun ha et tekstfelt og lagre et og et viapunkt underveis mens brukeren fyller inn de nødvendige viapunktene, vil vi kunne unngå denne feilen.

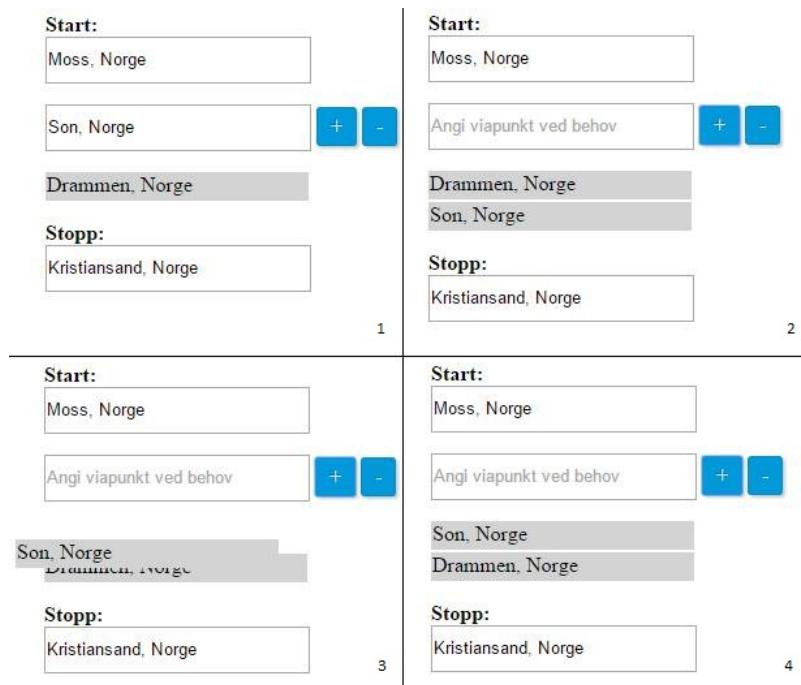
Vi har satt en maksgrense på fem viapunkter, og når det er lagt til fem punkter vil brukeren få beskjed om at han ikke kan legge til flere viapunkter samt at lagrekappen fjernes. Dersom «Slett»-knappen trykkes slik at viapunktlisten tömmes, vil «Legg til»-knappen vises igjen slik at brukeren kan legge til nye viapunkter. Når en beregning har blitt gjennomført vil viapunktlisten slettes.

I vår endelige viapunkthåndtering valgte vi å gå litt tilbake for å kombinere CSS med Javascript. Vi behøver ikke å ha mer enn ett tekstfelt for viapunkter, og vi kan bruke det samme tekstfeltet til å lagre alle viapunktene. Brukeren fyller inn et stedsnavn i tekstfeltet, og trykker lagre. Hver gang brukeren lagrer et viapunkt, hentes koordinatene fra Google, før disse konverteres og lagres i en egen sortørbar liste Samtidig slettes innholdet i tekstfeltet slik at et nytt viapunkt kan legges til. På denne måten unngår vi at programmet blir overbelastet.

Sortering av viapunkter

Som beskrevet i kapittel 3.1.2 har vi valgt å legge til en ekstra funksjonalitet for at brukeren kan sortere listen av viapunkter etter at de er lagt til. Brukeren kan sortere listen over viapunkter ved å dra rundt på hvert listeelement som inneholder et viapunkt. Både stedsnavn og koordinater er lagret i listen, men kun stedsnavn skrives ut. I tillegg har brukeren mulighet til å slette listen over viapunkter og begynne på nytt dersom han har behov for det. Vi har hatt to utfordringer med tanke på sortering. Den ene utfordringen er hvordan sorteringen skulle fremstilles i brukergrensesnittet. Den andre utfordringen var hvordan vi skulle hente ut den nye rekkefølgen etter sortering.

Den beste måten vi har kommet fram til for å sortere en liste er å bruke Javascript biblioteket jQuery, som er et bibliotek som gjør det enklere å manipulere HTML-elementer. Før vi støtte på problemet med for mange koordinater som skal hentes ut samtidig, sorterte vi tekstfeltene brukeren fylte inn viapunktene i. Da måtte vi knytte tekstuflfeltet sammen med et bildelement for å muliggjøre sortering. Ettersom vi gikk over til å lagre viapunktene i en liste før sortering, kan vi nå sortere listeelementene istedenfor tekstuflfeltene. Dette er en mye bedre og mer stabil måte å gjøre dette på. Vi bruker også jQuery til å hente ut den nye rekkefølgen etter sortering. Figur 3.19 viser sorteringen av listeelementer.



Figur 3.19: Viapunkter med et inputfelt og sorterbar liste

3.2.4 Koordinathåndtering

Som beskrevet i kapittel 3.1.2, benytter vi Google til å hente både adresser/stedsnavn og tilhørende koordinater. Google og Ruteplantjenesten benytter ulikt referansesystem (datum). Google benytter det amerikanske refereansesystemet WGS84 med koordinater på det tradisjonelle presentasjonsformatet desimalgrader (latitude og longitude) som er benyttet i mange hundre år. WGS84 (World Geodetic System 1984) er et globalt referansesystem som bl.a. brukes av GPS-systemet. Ruteplantjenesten benytter referansesystemet ETRS89 (European Terrestrial Reference System 1989) og presenterer koordinatene på UTM-format (Universal Transverse Mercator), med kartesiske koordinater (X og Y). ETRS89 er et europeisk refereansesystem som også brukes i Norge. ETRS89 er referert til den eurasiske kontinentalplaten slik at ETRS89-koordinater i Europa er statiske i forhold til kontinentet uavhengig av kontinentaldriften. WGS84-koordinater vil i Europa endre seg over tid selv om endringene er små.

Det sies at avvikene mellom ETRS89 og WGS84 i de fleste tilfeller er under en halv meter [29][30], slik at det i utgangspunktet vil være mulig å bruke begge standarder. Det vil si at dersom man bruker en kartvisning som skal finne et tilfeldig sted på kartet, vil punktet som vises, kun avvike opp mot en halv meter avhengig av om man bruker koordinater beregnet etter WGS84 eller ETRS89.

Ruteplantjenesten støtter begge standarder dersom koordinatene presenteres i UTM-formatet, men de oppgir at de selv benytter koordinater etter ETRS89. Selv om disse referansesystemene er nok så like, er problemet at koordinatene fra Google er presentert i form av desimalgrader, slik at de må konverteres til UTM. Det trengs flere matematiske formler for å foreta konverteringen mellom dem. I tillegg deles ulike land inn i ulike soner. Norsk sone regnes gjerne som 33N for hele landet, selv om Norge egentlig strekker seg fra sone 32 til 36. Avvikene er derimot såpass små at det holder med å definere sone 33N som standard. Vi har benyttet oss av et ferdig konverteringsskript [33] for å konvertere fra desimalgrader til kartesiske koordinater. Vi har tilpasset dette skriptet til å benytte UTM 33N. Vi har sett bort fra den minimale forskjellen mellom WGS84 og ETRS89, siden Ruteplantjenesten aksepterer begge to.

Når kjøreruta er ferdig beregnet, må koordinatene Ruteplantjenesten returnerer, konverteres tilbake til latitude og longitude i desimalgrader slik at koordinatene kan plottes på et kart fra Google. Koordinatene som returneres er compressed geometry (komprimerte koordinater). Et eksempel på compressed geometry er: <<+34+pcuhk+jirn14+ls+34+fk+io+0+fk-68+cg-cg+cg-fk+34-fk-68-1>>. Dette er ikke koordinater som kan overføres direkte til Google Maps. Først må disse koordinatene dekomprimeres, slik at vi får de som vanlige UTM koordinater. Her har vi gått for å benytte oss av en ferdig konverteringsfunksjon i ren C#-kode fra Esri [28, 34], en kjent tilbyder av geografiske tjenester.

```

zcm = 3 + 6 * (utmz - 1) - 180;
e0 = e / Math.sqrt(1 - e * e);
esq = (1 - (b / a) * (b / a));
e0sq = e * e / (1 - e * e);
N = a / Math.sqrt(1 - Math.pow(e * Math.sin(phi), 2));
T = Math.pow(Math.tan(phi), 2);
C = e0sq * Math.pow(Math.cos(phi), 2);
A = (lngd - zcm) * drad * Math.cos(phi);
M = phi * (1 - esq * (1 / 4 + esq * (3 / 64 + 5 * esq / 256)));
M = M - Math.sin(2 * phi) * (esq * (3 / 8 + esq * (3 / 32 + 45 * esq / 1024)));
M = M + Math.sin(4 * phi) * (esq * esq * (15 / 256 + esq * 45 / 1024));
M = M - Math.sin(6 * phi) * (esq * esq * esq * (35 / 3072));
M = M * a;
M0 = 0;

```

Figur 3.20: Omfattende konvertering

De dekomprimerte koordinatene er på UTM-format. Google krever som nevnt ovenfor koordinatene i latitude og longitude, slik at vi igjen er nødt til å foreta en konvertering. Å konvertere koordinatene tilbake er en like omfattende beregning som det var å konvertere motsatt vei, og også her tar vi utgangspunkt i en ferdig konverteringsfunksjon [32] og tilpasser denne. Figur 3.20 viser et utdrag av konverteringsfunksjonen for å vise hvor komplekse beregningene er.

Da vi konverterte fra UTM til latitude og longitude opplevde vi først at konverteringen ble unøyaktig. Det viste seg at konverteringsfunksjonen inneholdt en konstant som ble lagt til de beregnede koordinatene, slik at koordinatene hadde en konsekvent forskyvning. Vi fant til slutt feilen, slik at koordinatene nå konverteres riktig. Vi konverterer alle koordinatene Ruteplantjenesten beregner, slik at det vil ta lenger tid å beregne en lang rute, for eksempel Halden til Kirkenes, enn det vil ta å beregne kjøreruten Halden til Sarpsborg. Tidsbruken er nøyne vurdert opp mot fordelene ved å konvertere alle koordinatene, og bakgrunn for dette er beskrevet i kapittel 3.1.3.

Konverteringen fra latitude og longitude til UTM foregår gjøres i Javascript i nettleser. Dette strider mot oppdragsgivers ønske om å ha alle beregninger på serversiden, se kapittel 3.2.1. Vi har brukt mye tid på å prøve å finne en ferdig konverteringsfunksjon i C# slik at vi kunne flytte konverteringen til serveren. Vi har ikke funnet ferdig C#-kode til dette, og vi har også, uten hell, i flere omganger forsøkt å selv gjøre om Javascript-konverteringen til C#. Et annet alternativ vi har testet er Vegvesenets Vegreferanse-API [31] som tilbyr konvertering mellom ulike presentasjonsformer. Dette må gjøres i form av en forespørsel mot Vegvesenet. Her må man foreta en forespørsel per koordinat, slik tidsforbruket ble for stor til at det var hensiktsmessig. Vi har derfor landet på at det beste alternativet er å beholde konverteringen fra latitude og longitude til UTM i Javascript på klientsiden, inntil man kan få implementert en fungerende rutine i C# på server.

3.2.5 Optimalisering av kode

Optimalisering av kode - Javascript

Etter de siste endringene vedrørende viapunkter, har vi gjennomgått all Javascriptkode på klientsiden og foretatt en opprydning. En del kode lå igjen fra tidligere versjoner i prosjektpérioden, og kunne enten fjernes eller forenkles. Vi har samlet det meste av Javascriptkoden i to filer. Her har vi delt inn koden etter om den har med innhenting/presentasjon av data, eller om koden er i forbindelse med koordinatkonvertering/forespørsel til Web Service. Noe kode er det begrenset hva vi kan forandre på siden vi benytter oss av Google API enkelte steder. Koden i forbindelse med Google API er ikke veldig mottakelig for redigering, og vi var nødt til å beholde både globale variabler og noen gjentagelser for at disse skulle fungere optimalt.

Noe kode har vi kunnet utbedre, slik at vi ikke lenger har gjentagelser. For eksempel har vi hatt én kalkulatorfunksjon for hver av koordinatene vi har måttet konvertere inne i konverterskriptet. Det vil si en for start, en for stopp og en for viapunkter. Den viktigste årsaken til dette var å unngå overbelastning. Siden vi har gått over til å konvertere ett og ett viapunkt som beskrevet i kapittel 3.13 har vi gått over til å benytte samme konverteringsfunksjon for start og stopp. Det lot seg gjøre ved at vi byttet ut globale variabler med lokale variabler som sendes som parametere til selve funksjonen.

Optimalisering av kode - C#

I vår applikasjon har vi benyttet mange lister, blant annet nestede lister (en liste, med flere underlister i) og lister med array. For oss som har utviklet koden, er ikke dette noe problem, da vi har full oversikt over hvordan listene er bygd opp, og hvor vi finner alle verdier vi er ute etter. For de som ikke har skrevet koden selv, kan dette være tidkrevende å finne ut av, og siden applikasjonen vi utvikler skal tas i bruk i HRessurs, er det viktig at den er vedlikeholdbar og lett forståelig. Det første vi begynner med er å redusere antall lister, og gjøre koden mer objektorientert.

Vi tilpasser også koden slik at objektet som returneres til klientsiden, inneholder et eget objekt som kan legges direkte inn i HRessurs. Vi har tidligere kun returnert et objekt med all informasjon, men for at informasjonen enklere skal kunne legges til i reiseregningene, lagrer vi nå informasjonen som skal til HRessurs i et eget objekt.

Alle for-løkker i koden, endres til foreach, da dette optimaliserer koden. Ved for-løkker der man sjekker lengden på en liste/et array, kjøres denne beregningen for hvert trinn i løkka. Ved bruk av foreach beregnes lista kun en gang, før programmet går gjennom hele lista/arrayet. I skolesammenheng har for-løkker blitt brukt oftere enn foreach, og fordelen med foreach har vi først blitt gjort oppmerksomme på av kontaktperson Petter Ekrann da han ble obs på at vi konsekvent brukte for-løkker framfor foreach.

3.3 Publisering på skolens server

For å få testet applikasjonen uten å kjøre den via Visual Studio, utviklingsverktøyet vi har benyttet, var vi nødt til å laste opp applikasjonen på en server hos Høgskolen i Østfold. Det er ikke bare å laste den opp slik som en vanlig nettside, ettersom vi benytter oss av en Web Service som må ligge og kjøre i bakgrunn.

Vi fikk noen problemer i forbindelse med publisering. Ved å publisere applikasjonen gjennom et publisingsverktøy i Visual Studio, skulle appliasjonen være klar til å lastes direkte opp på en server. Dette fungerte ikke, og viste seg å skyldes at serveren vi hadde fått tildelt, var en Linux-server. Web Service-modulen som skal kjøres på serveren er en Windows-Web Service, slik at vi var avhengige av å få tilgang på skolens Windows-server, «donau».

Da vi fikk tilgang til «donau», viste det seg at serveren kjørte et for gammelt .NET-rammeverk (2.0), og vi måtte vente noen dager på at serveren ble oppdatert til 4.5-rammeverk. Når rammerverket var oppdatert, fungerte alt slik det skulle, men vi måtte utesette brukertestinga i ca. en uke på grunn av at oppretting av domene og retting av Framework feil tok tid.

Applikasjonen er å finne på: http://donau.hiof.no/b015_g02/

3.4 SCRUM - utførelse

Som nevnt i kapittel 1.4.2, har utviklingsprosessen vår fulgt metoden Scrum. Vi avtalte tidlig med oppdragsgiver at vi skulle ha Scrummøte annenhver onsdag, og dette har stort sett gått greit, med unntak av noen forskyninger på grunn av møtevirksomhet og ferier. På disse møtene har vi gått gjennom hva som var gjort siden sist, og hva vi skulle gjøre videre. Vi startet de første ukene med å sette oss i teknologien, analysere muligheter og finne ut om prosjektet var gjennomførbart osv, slik at vi først startet med sprinter fra uke 6. På neste side følger en oversikt over datoer for sprinter og hva som skulle gjøres. 18. februar til 12. mars ble sprintmøtet utsatt med en uke, slik at det gikk tre uker mellom møtene. Dette hentet vi inn igjen ved å begynne på det vi visste ville være de neste punktene som skulle gjøres, samt at vi kunne bruke litt mer tid på rapporten siden 1.versjon skulle leveres.

Vi fikk utført de fleste arbeidsoppgavene for alle sprintene. Et unntak er feilhåndteringen i Sprint 2. Generell feilhåndtering ble ikke fullført før det nærmet seg slutten av prosjektperioden.

4.feb - 18. feb Sprint 1.

1. Få opp kommunikasjon mellom klientside og serverside ved hjelp av Web Service
2. Samle inn brukerdata fra klientside som er klare til å sende til Web Service
3. Hente stedsnavn fra klientside. Hente ut avstand og bompenger fra Ruteplantjenesten, i Web Service basert på hardkodede koordinater.

18. feb - 12. mars. Sprint 2.

1. Flytte logikk fra Web Service til TravelRoute, som implementerer ITravelRoute.
2. Hente koordinater fra klientside til start og stopp.
3. Hente ut alle bomstasjoner på ruta.
4. Feilhåndtering på request mot ruteplantjenesten.

12. mars – 24. mars. Sprint 3.

1. Brukergrensesnitt tilpasset HRessurs.
2. Parse JSON til forhåndsdefinert klasse, framfor vanlig parsing
3. Fullt fungerende løsning for innsamling av viapunkter
4. Utbedre feilhåndteringen

24. mars – 8. april. Sprint 4.

1. Få frem Veibeskrivelse
2. Fullt fungerende løsning for sortering av viapunkter
3. Plotte rute på kart, feilhåndtering med forskjellige søk
4. Konvertering av koordinater tilbake til LatLng

8. april – 15. april. Sprint 5.

1. Komprimert veibeskrivelse.
2. Eget objekt som returneres til HRessurs (ett objekt med bomutgifter og ett objekt med kjørerute)
3. Gjøre om Array og Liste struktur til Objekt struktur
4. Gjøre om for-løkker til foreach

15. april – 29. april. Sprint 5.

1. Sluttføring fram mot brukertesting

3.5 Programmer, verktøy og hjelpeemidler

3.5.1 Programmer og nettbaserte verktøy vi har brukt:

Vi har benyttet oss av følgende programmer og verktøy for å utvikle applikasjonen, utføre brukertesting og dokumentere til rapporten:

- Visual Studio 2013
- Visual Paradigm Free UML design tool
- Surveymonkey.com
- Github
- Dropbox
- Google Forms

3.5.2 Innhenting av informasjon

Den informasjonen vi har trengt for å kunne fullføre arbeidet har blitt innhentet hovedsakelig fra nett. Glenn organiserte også et møte med en av hans forelesere Per Bissegberg ved HIOF for å få flere synspunkt og forklaringer på problemstillingen rundt lagdeling, da dette var veldig nytt for oss i begynnelsen.

- Stackoverflow: Dette nettsamfunnet har vi benyttet til alle deler av arbeidet
- Youtube: Her finner man mange gode tutorials. Særlig har vi lett etter informasjon på hvordan man bruker en Web Service i Visual Studio
- Google Developers: Fremgangsmåte for å bruke Google Autocomplete og Google Maps
- W3School: Mye grunnleggende programering, spesielt innen HTML, Javascript og CSS er beskrevet her, med tilleggsfunksjon for å teste/modifisere eksemplene.

4. Brukertest av applikasjonen og implementering i HRessurs

I dette kapittelet tar vi for oss brukertesting av applikasjonen, utbedringer som følge av tilbakemeldinger og helt til slutt implementeringen i HRessurs. Fullversjon av testskjema og testresultater følger finnes i vedlegg A - Brukertest.

4.1 Planlegging av brukertesting

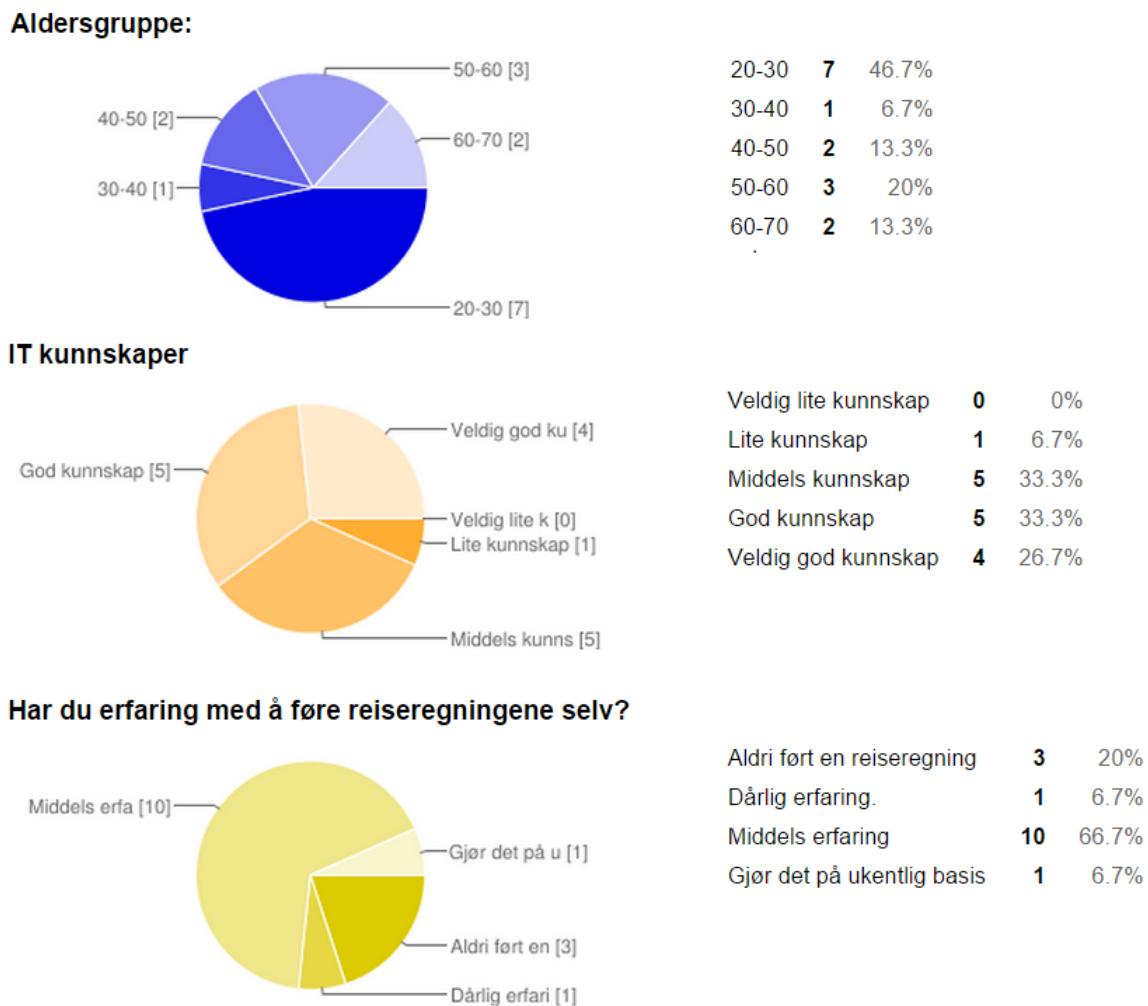
For å kvalitetssikre applikasjonen vår når det nærmet seg implementasjon i HRessurs, bestemte vi oss for å la familie og venner teste applikasjonen når vi så oss fornøyde med resultatet. Ved hjelp av denne brukertestingen ønsket vi å få tilbakemeldinger på applikasjonen vi har utviklet, samt få rede på eventuelle feil og mangler vi har oversett fra vårt utviklerperspektiv. Vi valgte derfor ut et antall personer i forskjellige aldre og med ulik erfaring både med bruk av datamaskin og føring av reiseregninger. Vi ønsket og få et bredt spekter av testpersoner for å kunne sammenligne feil og vanskelighetene en bruker med middels datakunnskaper opplevde i forhold til en person med gode datakunnskaper.

Vi bestemte oss for noen klare retningslinjer for hvilke tilbakemeldinger vi skulle fokusere på hvis vi fikk motstridende tilbakemeldinger. Ved stort sprik i tilbakemeldingene har vi valgt å prioritere svar fra brukere som registrerer reiseregninger og som har middels eller bedre datakunnskaper. Det vil ikke være hensiktsmessig å tilpasse applikasjonen til de som sjeldent bruker pc og/eller har veldig lave datakunnskaper. Dette ville gått utover viktig funksjonalitet som er nødvendig for applikasjonen. Spørsmålene vi planla var delt i to deler. I første del kartlegger vi testpersonenes bakgrunn og erfaring innen pc og reiseregninger. Del to tar for seg alt det tekniske rundt applikasjonen. Her kartlegger vi om brukeren opplever feil/får spesifikke feilmeldinger, om det er funksjonalitet de savner, samt generelle tilbakemeldinger på selve applikasjonen. Da får vi et godt grunnlag for å finne ut om det er klare likheter/forskjeller mellom erfaring og oppfatninger av applikasjonen.

Da vi skulle lage svarskjema stod valget mellom SurveyMonkey [25] og Google Forms. Dette ble et enkelt valg da SurveyMonkey kun tilbyr 9 spørsmål i deres gratis utgave, mens Google tilbyr et ubegrenset antall spørsmål. Google gir også en mye ryddigere og enklere grafisk visning av spørreundersøkelsen. Google forms tilbys også en bedre løsning med tanke på å analysere resultatene, blant annet ved at man kan få frem svarene i form av grafer og skjemaer. Dette gjør at behandlingen av resultater går raskere og blir mer oversiktlig. For fullstendig spørreundersøkelse og resultater, se vedlegg A - Brukertest.

4.2 Analyse av resultater fra brukerevalueringen

Ved gjennomføring av brukertesten, delte vi ut en link til applikasjonen og en link til spørreskjema [26]. Når brukertesten var gjennomført satt vi igjen med 15 besvarelser. Hvordan testpersonene våre er kategorisert etter erfaring, er illustrert i figur 4.1



Figur 4.1: Resultat fra brukertest. Grafisk fremstilling av testpersonenes bakgrunn

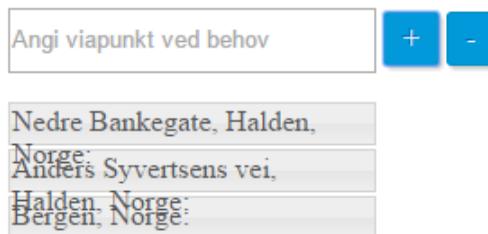
Våre observante testere oppdaget en del feil og svakheter som vi ikke har lagt merke til under utviklingen. Dette gjør at vi er nødt til å utbedre en del av disse før endelig versjon skal implementeres i HRessurs. Vi har også fått veldig mange gode tilbakemeldinger. Brukerne med minst data- og reiseregningserfaring la ikke merke til noen feil og syns at applikasjonen var forståelig og enkel. Våre testere med best erfaring innen data, avdekket en del feil og mangler som bør utbedres for å løfte standarden på applikasjonen. Grunnen til dette er nok at jo mer erfaring tester har, jo større forståelse har han eller hun for hvordan en brukervennlig applikasjon bør være.

4.2.1 Feil og mangler fra brukertest:

Vi har valgt å dele tilbakemeldingene inn i to kategorier, feil som må utbedres og feil/mangler som bør utbedres. Under feil som må utbedres kommer blant annet feil som gjør at applikasjonen ikke fungerer som den skal og at det beregnede resultat blir galt. Feil som bør utbedres er mangler som ikke er kritisk for at applikasjonen skal fungere, men som vil bidra til en mye bedre brukervennlighet.

Feil som må utbedres

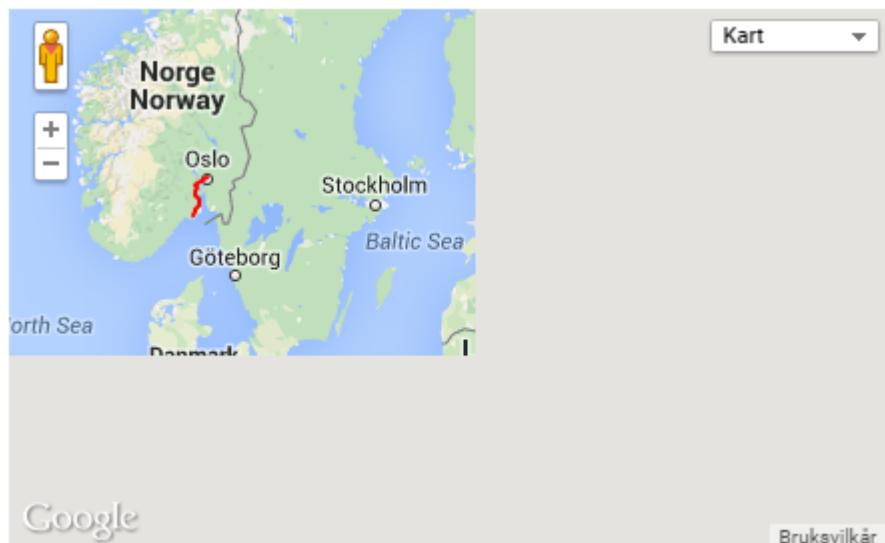
- Et problem som dukket opp var at ved lange stedsnavn var det ikke plass til hele navnet i viapunktlista. Dette førte til at stedsnavnet fortsatte på neste linje, slik at det ble en overlapp slik figur 4.2 viser. Stedsnavnene ble skrevet ut oppå hverandre, og det var knapt mulig å se hvilke viapunkter som var angitt. Her fikk vi forslag om å enten lage scrollfunksjon i lista, eller kutte slutten av ordet slik at det ikke fortsatte utover feltet og laget problemer på neste linje.



Figur 4.2: Overlapp i viapunktliste

- Ved å trykke på «Beregne»-knappen to ganger, siden det ikke gis tilbakemelding på at beregninger pågår, kan det komme en feilmelding med OVER_QUERY_LIMIT. Denne feilmeldingen kommer hvis det går så kort tid mellom hver gang knappen trykkes, at Google ikke rekker å hente koordinatene fra første trykk i mellomtiden. I tillegg til feilmelding, slettes viapunkter når «Beregne»-knappen trykkes, slik at de ikke blir tatt med i beregningen når knappen trykkes for andre gang. Vi bør sperre knappen, slik at det ikke er mulig å trykke en gang til mens beregning pågår. Vi bør også gi bruker en tilbakemelding på at beregning pågår.
- Det har blitt tydeliggjort at beregnede veivalg ikke alltid er ideelle. Det har blant annet blitt beregnet kjøreruter over en gangbro, via småveier gjennom boligstrøk istedenfor hovedvei og over fjellet på veier som er vinterstengt. Dette problemet går vi nærmere inn på i kapittel 4.3
- Det har kommet mange tilbakemeldinger på at viapunktene bør forbedres. Feil som har kommet frem ved via-punkter er:
 - Får ikke slettet via-punkter når maks antall via-punkter er valgt
 - Kan lagre viapunkter uten verdi
 - Uklart hvilken rekkefølge via-punktene beregnes i
 - Ved å trykke beregn flere ganger mens ruten beregnes, slettes viapunktene slik at de ikke kommer med i beregningen

- Enkelte stedsnavn som finnes i Google Maps, klarer ikke vår tjeneste å finne. Tester oppgav ikke hvilke steder som ikke ble funnet, slik at vi selv har måtte teste oss fram for å finne ut hvilke stedstyper det gjaldt. Det kan se ut som dette gjelder bedrifter, gårder, småbruk og lignende. Her bør vi se om det er mulig å legge til flere stedstyper i Google Autocomplete. Vi har kun benyttet byer og adresser til nå.
- En bruker fikk problemer når han forsøkte å beregne ny rute mens han sto i resultatfanen. Dette førte til at kartet ble vist i feil fane helt til tester trykket seg videre til en annen fane.
- Flere personer opplevde problemer med kartvisning dersom de beregnet en ny rute mens de befant seg i en annen fane enn kart. Enkelte ruter av kartet ble grå slik figur 4.3 viser, og det var ikke mulig å zoome/panorere fornuftig etter dette. Denne feilen må utbedres, slik at kartet alltid fungerer som det skal.



Figur 4.3: Grå felter i kartvisning

- Det oppstår et problem med Google Autocomplete hvis man ikke velger et forslag fra Autocompletelista. Her bør det komme en feilmelding når kjøreruta ikke kan beregnes som følge av at bruker ikke har valgt adresse fra Goggle Autocomplete
- Flere personer ønsket en forklarende tekst til alle funksjoner slik at all funksjonalitet ble tydelig fra første bruk.
- Dersom applikasjonen av en eller annen grunn ikke klarer å beregne kjøreruta, får ikke bruker tilbakemelding om dette. Feilmelding vises kun ved å inspisere siden, for eksempel i Firefox med Firebug, eller Google Chrome ved å høyreklikke og velge «Inspiser element». Her må bruker få generell tilbakemelding dersom en feil oppstår i forbindelse med Web Service og innhenting av informasjon fra Ruteplantjenesten.

Feil som bør utbedres for optimal brukeropplevelse

- Hele fire personer har gitt tilbakemelding på at tidsbruk skrives ut i minutter. Denne bør gjøres om til timer og minutter.
- Flere har kommentert at applikasjonen ikke er mobilvennlig. Dette vil antageligvis bli rettet ved implementasjon til HRessurs, da HRessurs er mobilvennlig
- Kartet tilpasses ikke kjøreruten, og har et standard zoomnivå uavhengig om ruten er kort eller lang. Som vist i figur 4.4, blir kjøreruta kun en liten strek på kartet. Ved enda kortere kjøreruter på kun et par kilometer, kan man kun se kjøreruta som en liten prikk i kartet før man selv zoomer inn manuelt. Dette bør endres slik at zoom-nivået tilpasses kjøreruta automatisk.



Figur 4.4: Fremstilling av kjørerute med standard zoomnivå

- Flere har kommentert at knappene i forbindelse med viapunkter er vanskelige å forstå, se figur 4.5. Vi bør endre navn på lagreknappen <+> til «Lagre» og minusknappen <-> til «Slett alle». Vi bør også legge til funksjonalitet for å slette kun ett og ett viapunkt. En bruker syns det var ukjart når <-> slettet alle viapunktene.



Figur 4.5: Lite forklarende knapper i forbindelse med viapunkter

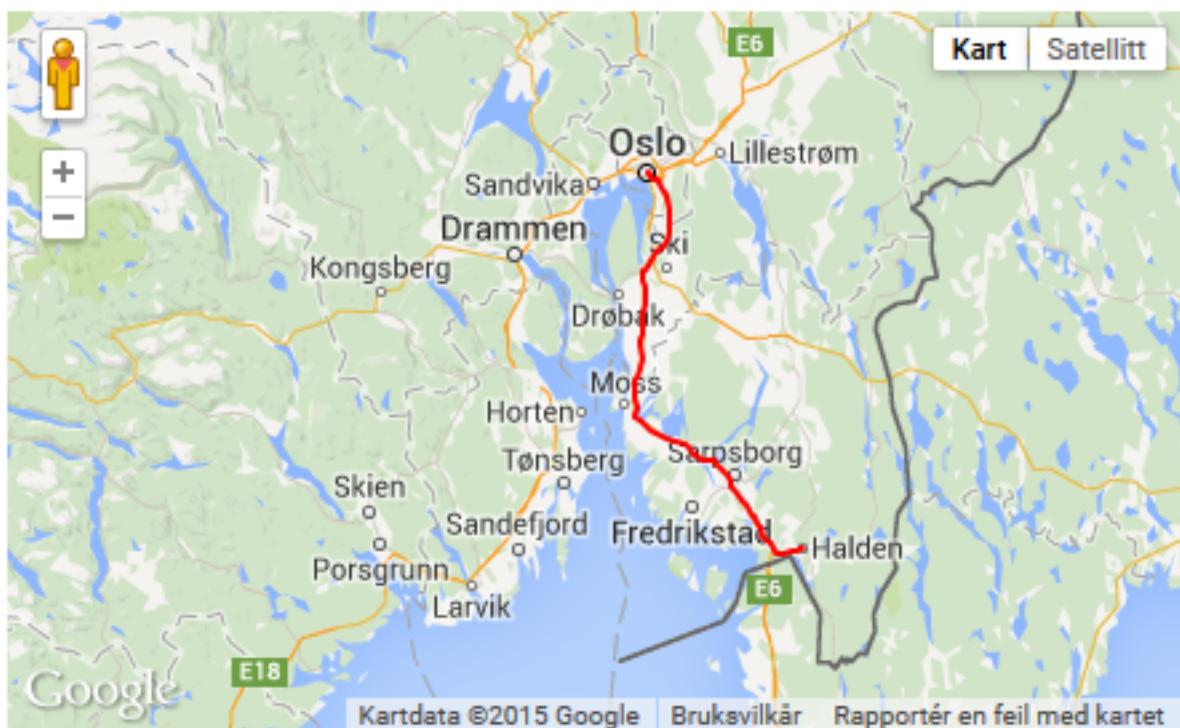
- Flere har ønsket en bedre måte å endre kjørerute på. Det har kommet ønsker om å kunne dra kjøreruta dit man ønsker for å endre kjøreruta i ettermiddag, og kunne sette start, stopp og viapunkter direkte i kartet fremfor å angi stedsnavn.

I kapittel 4.4 tar vi for oss hvilke endringer vi har gjort som følge av brukertestingene.

4.3 Refleksjon rundt unøyaktigheter i Ruteplantjenesten

Etter brukertestingene har vi blitt gjort oppmerksomme på at Ruteplantjenesten har varierende grad på hvor nøyaktig og riktig de beregner rutene sine. Et problem som dukker opp er at de ikke har en konsekvent måte å beregne kjøreruter, og at rutene noen ganger kan bli beregnet gjennom boligstrøk, framfor å følge hovedveien. Dette virker litt motstridene mot vegvesenets skilting, som i all hovedsak prøver å lede trafikken utenfor boligområder. Det er også tilfeller der kjøreruta beregnes via veier der det ikke er lov eller mulig å kjøre. Eksempler på dette er at kjørerute beregnes over en gangbro og at kjøreruta beregnes over et vinterstengt fjell.

I første omgang vil vi studere de ulike veivalgene vi selv kan påvirke kodemessig, for å være sikre på at det ikke er vi som har tatt et valg som fører til at rutene ikke beregnes optimalt. Her tilbyr vegvesenet tre forskjellige valg. Raskeste og korteste vei, samt turistvei. Vi har hele tiden operert med raskeste vei, da dette er mest relevant i forhold til jobbreiser, der det gjerne skal brukes minst mulig tid på kjøring. Vi velger å teste med Halden – Oslo, som er en kjørerute vi er godt kjent med. Det beste valget er å følge E6 hele veien fra Svinesundsparken og inn til Ring3, der man tar av mot Oslo sentrum. Denne ruta beregner vegvesenet når vi velger parameter raskeste vei, se figur 4.6.



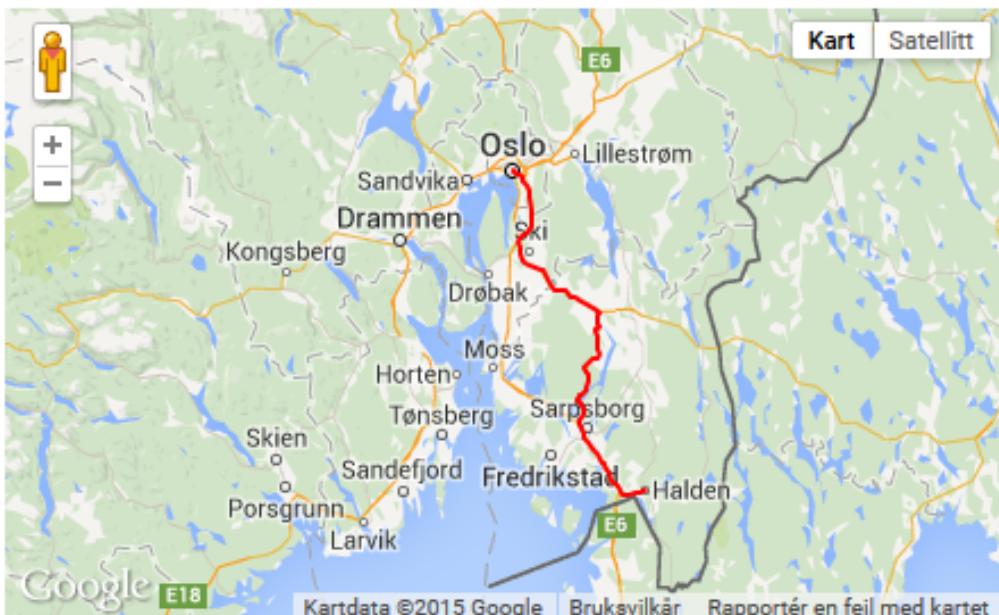
Figur 4.6: Raskeste vei Halden - Oslo

Vi tester også Halden – Oslo med korteste vei, og får beregnet kjøreruten i figur 4.7. Denne ruta følger småveiene gjennom Østfold, fra Halden via Rakkestad, Mysen og Askim før den til slutt svinger inn på E6 ved Ski og fortsetter siste stykket inn til ring 3 på E6.



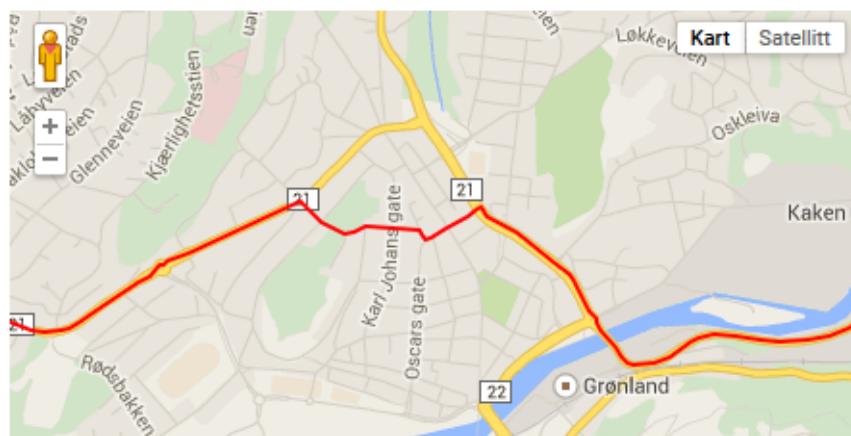
Figur 4.7: Korteste vei Halden - Oslo

Helt til sist tester vi parameter turistvei. Denne tar E6 til Sarpsborg, før den svinger innom Askim og kommer inn på E6 igjen ved Ski.



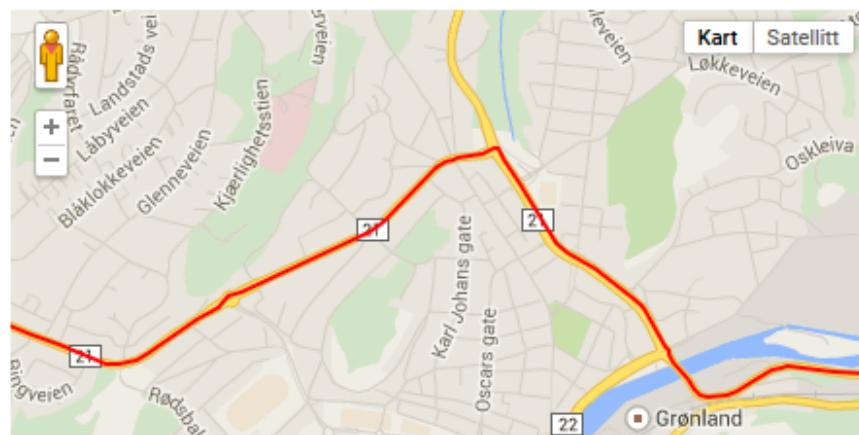
Figur 4.8: Turistvei Halden - Oslo

Etter testingen av alle tre veivalgene, slår vi fast at raskeste vei, som vi har brukt hele veien til nå, er den desidert beste ruta. Hovedtrekkene i de beregnede rutene er riktige, men avstikkerne gjennom boligstrøk vil kunne gjøre beregningene unøyaktige. Et typisk eksempel er Tistedal – Sarpsborg, se utsnitt av ruta i figur 4.9. Her beregnes kjøreruta via små sentrumsgater: Vognmakergata, Arbeidergata og Wærns gate. Sistenevnte benyttes vinterstid som akebakke, og er både smal og uegnet for stor trafikk i tillegg til at den er bortimot ufremkommelig for vogntog og andre store biler. Rett utenom går riksvei 21, og det er via R21 man blir ledet ved å følge trafikkskiltene. Trafikkskiltene i Norge driftes av Vegvesenet, men allikevel dirigerer de brukerne utenom de skiltede rutene i tjenesten de tilbyr. Vi sjekket, og den samme ruta beregnes i webtjenesten deres [38].



Figur 4.9: Snarvei gjennom boligstrøk i Halden

Vi bestemmer oss for å gjøre flere tester rundt dette. Vi tester da Aremark – Sarpsborg, og får ruten som vist i figur 4.10. Der kan man se at kjøreruten følger R21, og ikke småveiene slik kjøreruta mellom Tistedal og Sarpsborg gjør. Det som gjør dette ulogisk, er at kjøreruta er akkurat den samme de siste 3-4km fra Tistedal fram til den ene ruta svinger inn i boligstrøket. Etter at kjøreruta som snegler seg gjennom småveiene tar inn på R21 igjen, er kjørerutene helt like de siste 30km til Sarpsborg.

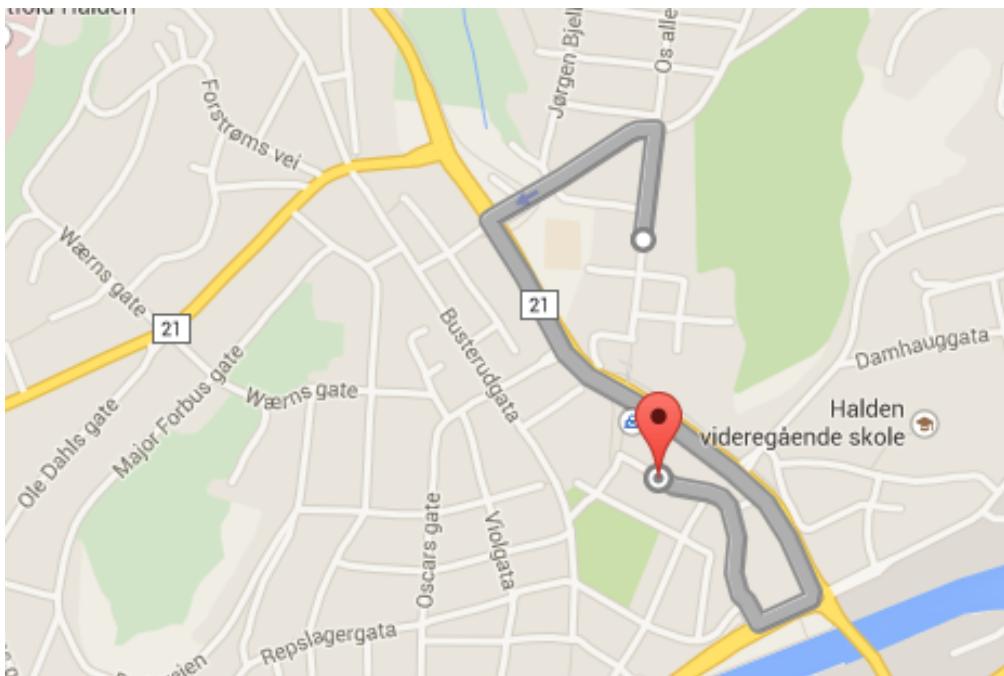


Figur 4.10: Kjørerute følger hovedvei utenom boligstrøk

Et annet problem vi blir gjort oppmerksomme på, er at Vegvesenet beregner kjøreruta via ulovlige ruter. Os Allé, Halden – Urtegata, Halden ligger bare noen hundre meter fra hverandre, men korteste kjørerute med bil er 1,2km. Ruteplantjenesten beregner kjøreruta via gangbrua over R21, slik figur 4.11 viser. Den korteste, lovlige kjøreruta som er beregnet via Google Maps er vist i figur 4.12.



Figur 4.11: Ruteplantjenesten viser kjørerute over gangbro



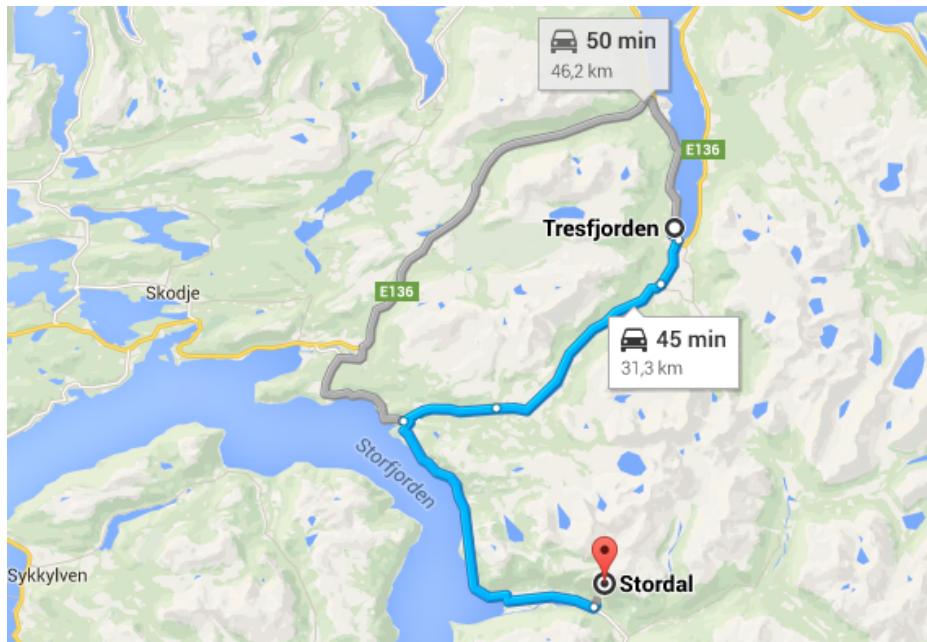
Figur 4.12: Slik er korteste, lovlige kjørerute

Dessverre er dette et problem vi først blir gjort oppmerksomme mot slutten av prosjektperioden, siden problemet kun oppstår ved enkelte adresser. I hovedsak er kjørerutene riktige, og ettersom problemet ikke er konsekvent, er vi avhengig av at en adresse der feilen oppstår angis. Alle adresser vi har testet under utviklingsperioden (der vi kjenner kjøreruta), har stemt overens med kjøreruten vi ser for oss. Vi har hatt et større fokus på at brukergrensesnittet og funksjonalitet skal være optimalt. Vi har fokusert på at kjøreruta skal beregnes via alle punkter angitt av bruker, istedenfor å studere absolutt alle kjøreruter vi beregner i detalj for å lete etter feil. At ruter kan bli beregnet via småveier som ikke eigner seg å kjøre på, der det er smalt og ufremkommelig, eller vinterstengt er ikke kritisk da denne applikasjonen kun skal beregne kjøreruter i ettertid. Vårt største problem i forhold til at rutene ikke alltid følger beste veien, er at tjenesten ikke vil være pålitelig nok. Da applikasjonen skal beregne kostnader og avstander i forbindelse med reiseregninger, som har strenge krav til etterprøvbarhet, er ikke Vegvesenet sin tjeneste optimal slik den er i dag.

Applikasjonen vil ikke bli fullstendig brukervennlig dersom vi skal kreve at brukere zoomer seg inn på detaljnivå for å kontrollere at kjøreruta som er beregnet er korrekt. En bruker av HRessurs skal ikke være tvunget til å benytte seg av applikasjonen vår ved utfylling av reiseregninger, og det er kun ment som en ekstratjeneste til de som ønsker å benytte den. Vi vil ta et forbehold i applikasjonen om at bruker selv må kontrollere at de beregnede opplysningene stemmer, slik man alltid bør gjøre uavhengig av tjeneste som benyttes. Det viktigste vi kan gjøre i denne forbindelse er å gjøre fremstillingen av de beregnede resultatene så brukervennlige som mulig, for å best mulig legge til rette at bruker kan kontrollere kjørerutene.

I eksempelet med akebakken i Halden sentrum i figur 4.9 og figur 4.10, er forskjellene i avstand minimale. Det er 34,6km via R21, og 34,5km via sentrumsgatene. Et slikt avvik på 100m kan aksepteres, ettersom dette er et problem som kun oppstår ved få adresser. Et problem som kan oppstå som følge av disse små avstikkerne fra hovedveien, er at ruten kan styres utenom en bomstasjon, slik at de totale bompengekostnadene som beregnes ikke vil stemme med de faktiske utgiftene. I all hovedsak er bomstasjonene rundt om i landet lagt opp slik at det ikke skal være mulig å ta av på en vei som går parallelt med bomveien for å unngå bompenger. Korte avstikkere som dette vil derfor i hovedsak ikke påvirke bompengekostnadene, med mindre kjøreruten blir beregnet via en gangvei, slik vi viste eksempler på i figur 4.11 og figur 4.12.

Hvis det er et større avvik i avstanden, slik som i eksempelet i figur 4.13 på neste side med kjøreruten mellom Tresfjord og Stordal (Møre&Romsdal) som beregnes over vinterstengt fjell, vil avviket i avstand kunne skape problemer hvis dette ikke oppdages. Vi har valgt å illustrere dette problemet i Google Maps, da de viser både prioritert rute og alternativ rute på samme kart. Her er differansen i avstanden på hele 15km, som er vesentlig mer enn 100m. Dette vil kunne føre til at man får beregnet 50-60kr mindre i kjøregodtgjørelse etter statens satser [39]. Fordelen er at en slik kjørerute vil oppdages raskere enn en liten snarvei gjennom et boligstrøk. I denne sammenheng skal det nevnes at både Google Maps, Finn ruteplanleggingstjeneste [40] og 1881 veibeskrivelse alle har veien over den vinterstengte fjellveien som 1. valg av rute. Vi hadde derfor ikke kommet unna dette problemet ved valg av en annen tjeneste.



Figur 4.13: Tresfjord - Stordal over vinterstengt vei

Vi antar at det er en feil/unøyaktighet i algoritmen Vegvesenets API benytter seg av i beregningen av kjøreruter, ettersom kjøreruter som følger samme vei, enkelte steder beregnes ulikt selv om parameterne for rutevalg er de samme. Vi sender følgende mail til vegvesenet angående denne problemstillingen:

Hei,

Vi er en gruppe fra Høgskolen i Østfold som jobber med bacheloroppgaven. Vi har det siste halvåret utviklet en applikasjon som kan benyttes til reiseregninger (basert på ruteplan-apiet), ved at man taster inn start, stopp og viapunkter og får returnert avstand og bompengekostnader på strekningen. Vi har benyttet oss av de komprimerte koordinatene, og plottet absolutt alle koordinatene på den beregnede strekning inn på et kart, slik at vi grafisk kan kontrollere hvor ruta er beregnet. På strekningen Halden – Oslo er det ca 1300 koordinater, som tilsvarer at koordinater ca hver 10.meter plottes på kartet. Kjøreruta som plottes skal derfor være ganske nøyaktig lik kjøreruta som beregnes via API-et deres. Nå er applikasjonen tilnærmet ferdig, slik at vi har foretatt en brukertest av applikasjonen. Observante testere har lagt merke til at kjørerutene som beregnes, ikke er optimale.

Et eksempel er kjøreruta fra Tistedal til Sarpsborg. Se vedlagt bilde. I Halden sentrum blir denne kjøreruta dirigert gjennom små og trange sentrumsgater og opp gjennom en bakke som vinterstid er stengt og brukt som akebakke. Kjøreruten beregnes via Vognmakergata, Arbeidergata og Wærns gate (akebakken). All skilting av veier prøver konsekvent å lede trafikken utenom boligstrøk, mens ruteplanleggingsstjenesten deres leder folk gjennom boligstrøk der det går en riksvei rett ved siden av. Tilsvarende problem finnes i Møre og Romsdal. Dersom man beregner en kjørerute fra Tresfjord til Stordal, blir denne beregnet via en liten vei over fjellet. Denne veien er vinterstengt pga snø, og den er ikke egnet for stor trafikk, og spesielt ikke tungtrafikk. Kjøreruten fra Os Allé, Halden – Urtegata, Halden blir beregnet over en gangbro, der det ikke er lov å kjøre.

Disse kjørerutene finner vi også igjen i tjenesten deres, visveg.vegvesen.no. Er dette en feil i systemet deres, eller er det bevisst? Hvor går grensa for hva dere definerer som en vei i ruteplanleggingstjenesten, og er det eventuelt andre valg man kan gjøre i spørringene for å tvinge ruta til å holde seg på de største veiene? Vi har testet alle tre rutene, både korteste vei, raskeste vei og turistvei, og det er raskeste som viser riktig veivalg, sett bort fra de små, ulogiske avstikkerne gjennom boligstrøk.

Er dette noe som vil bli utbedret i forbindelse med fullføring av den nye tjenesten dit.no? Vi har forsøkt å teste denne fra mobil for å se om kjørerutene er de samme, men får feilmeldingen «En ukjent feil oppstod. Feilen er logget og vil bli undersøkt». Eller er denne type feil med ufremkomelige småveier, ulovlige veier via gangbroer osv noe som vil vedvare?

Vedleggene er figur 4.10 og 4.9.

Vi ser for oss flere mulige løsninger på problemstillingen angående applikasjonens pålitelighet. Løsningene avhenger av tilbakemelding fra Vegvesenet. Dagens løsning kan benyttes uavhengig av svar, men vi er nødt til å ta forbehold om at brukeren selv må kontrollere at beregnet rute er korrekt. Det beste vil være å få et svar om at dette er ting som vil bli utbedret etter hvert, da Vegvesenet tilbyr all informasjon vi trenger. Det eneste problemet er usikkerheten om tjenesten vil beregne den riktige ruta. Vi håper på en positiv tilbakemelding fra Vegvesenet om at disse feilene er under utbedring, da vi vet at de ruster opp tjenesten sin til bruk i den nye rute- og trafikktjenesten Dit.no [37].

Et annet alternativ er å bytte ut Ruteplantjenesten med en annen tjeneste. Her var problemet at Ruteplantjenesten var den eneste tjeneste av sitt slag som inneholder bomstasjoner, slik at det kan være problematisk å finne andre tjenester. Med tanke på lagdelingen av applikasjonen (se kapittel 3.2.1), vil dette være en relativt rask og enkel prosess dersom det viser seg at en tilsvarende tjeneste er tilgjengelig.

Vi får raskt et oppløftende svar fra Jan Kristian Jensen i seksjon for NVDB og Geodata hos Vegvesenet, og vi konkluderer med at Ruteplantjenesten fremdeles kan benyttes, ettersom de jobber med å utbedre slike feil som vi har oppdaget. Svaret er følgende:

Dette er slikt som vi jobber systematisk for å fjerne. Vi har et hierarki av veger, og ruteplanleggeren skal velge veger høyere opp på rangstigen fremfor kommunalveger. Dagens hierarki er bygd på hvem som eier vegen (Staten? Fylket? Kommunen?). Dette fungerer stort sett greit, men har en del svakheter, og i noen tilfeller blir det slik som dere beskriver. Vi ser virkelig fram til vi kan gå over til å bruke informasjonen «funksjonell vegklasse». Det vil gi langt mer riktig beskrivelse av vegens funksjon enn dagens hierarki. Da vil mange av dagens problemer bli borte.

Dagen etterpå får følgende oppfølgingssvar:

Hei igjen! Jeg diskuterte dette med kolleger + utvikler hos leverandør, og ble minnet på et par ting: Vi legger mer vekt på hierarki på lange ruter enn på korte. Det vil si at tjenesten har større fleksibilitet til å bruke det kommunale vegnettet ved korte ruter. Dette betyr at selv om det rutes på suboptimale kommunalveger på korte strekninger så vil lengre strekninger som regel følge hovedvegene.

På Tistedal-Sarpsborg (evt Unneberg)-eksemplet kan dette vises ved at startpunkt trengs bare flyttes ørlite grann lengre øst for at det IKKE skal via Vognmakergata, Arbeidergata og Wærns gate.

<http://goo.gl/awDJhr>

Tilsvarende gjelder for Tresfjord-Stordal. Flyttes startpunkt ørlite grann unna Tresfjord så rutes det ikke over fjellet: <http://goo.gl/Uo7QlB>

Det må være mulig å navigere lokale veger – samtidig som vi skal unngå lokale veger på lange strekninger. For at begge deler skal bli riktigst mulig må vi inngå en del kompromisser – og INGEN kompromisser klare å gi perfekte løsninger i alle situasjoner. Dette jobber vi hele tiden med å bli bedre på.

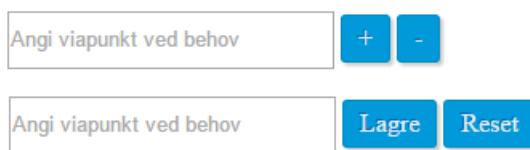
MVH Jan Kr. Jesnen

Svarene fra Vegvesenet er oppløftende, og det virker som tjenesten vil bli bedre med tiden. Ettersom feilen med at kjørerute beregnes over vinterstengte veier også oppstår i de andre ruteplanleggingstjenestene, er det kun problemet med at kjørerutene beregnes utenfor hovedveiene, som er spesielt for Ruteplantjenesten. Vi vet også at dette kun er på lokale ruter og i forbindelse med start og stopp. Vi vil derfor sørge for at det står en tekst om forbehold om at brukeren må kontrollere kjøreruten, spesielt i forbindelse med lokale kjøreruter.

4.4 Utbedringer og endringer etter brukerevalueringen

4.4.1 Forandringer i brukergrensesnittet

Vi har, som beskrevet i kapittel 4.2, fått tilbakemeldinger på at det kan være noe uklart hvordan viapunktene fungerer. Blant annet har det vært forvirrende hva knappene «+» og «-» gjør. Plussknappen skal lagre, mens minusknappen sletter hele listen. Noen har trodd at denne knappen sletter ett og ett viapunkt. Dette ser vi at kan skape forvirring og vi har derfor valgt å bytte ut symbolene med tekstene: «Lagre» og «Reset» som vist i figur 4.14. Vi tror det nå blir mer forståelig hva de to knappene gjør. Vi skulle også gjerne fått til en mulighet for å fjerne ett og ett viapunkt, men vi har så langt ikke fått dette til å fungere. Tiden strekker dessverre ikke til for å finne ut av dette.



Figur 4.14: Øverst: viapunkter før testing. Nederst: viapunkter etter utbedring

Vi fikk også tilbakemelding på at viapunktene ikke alltid tas med i beregningen. Dette skyldes at det er fylt inn tekst i viapunktfeltet som ikke er lagret. Det er kun de stedene som er lagret og som av den grunn er lagt til i den sorterbare listen som tas med. Vi har forståelse for at dette kan være forvirrende, siden start og stopp ikke må lagres for å tas med i beregningen. For å forhindre at brukere tror at viapunkter som ikke er lagret blir med i beregningen, har vi lagt inn en advarsel hvis det er skrevet noe i tekstfeltet for viapunkter som ikke er lagret når «Bereg» trykkes. Brukeren vil da måtte velge å lagre viapunktet eller fjerne innholdet fra tekstfeltet.

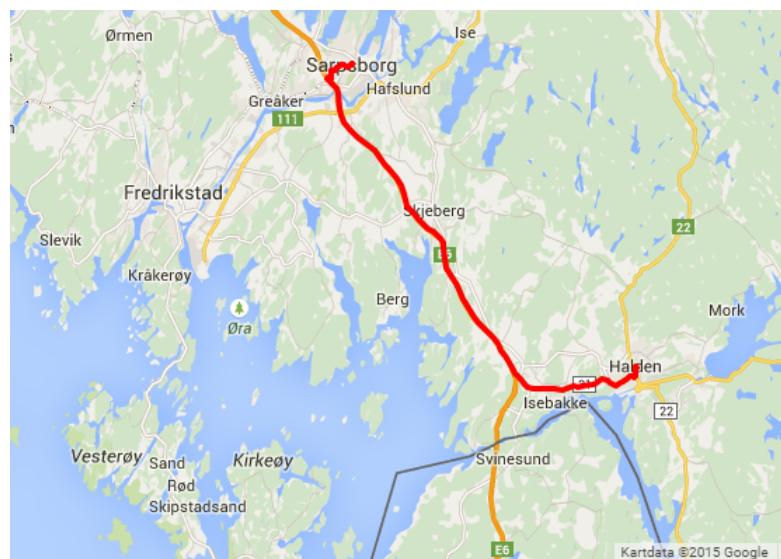
For å gjøre det mer forståelig at man kan sortere viapunktene har vi lagt til funksjonalitet i CSS som gir en annen farge når man holder musepekeren over listen med viapunkter, slik som figur 4.15 viser.



Figur 4.15: Listeelement som endrer farge

For å utbedre kartvisningen, har vi sett på mulighetene for å tilpasse zoomnivå etter beregnet rute. Google har ingen funksjonalitet for å zoome kartet direkte til riktig nivå når man har plottet en rute via Polyline. Vi har derfor vært nødt til å definere zoomnivå og midtpunkt manuelt. Vi har løst dette ved å hente ut koordinatet midt i koordinatlista, og setter dette som midtpunkt i kartet. På kjøreruter som svinger mye innenfor et område, vil det kunne oppstå et avvik i forhold til at midtpunktet i kartet ikke nødvendigvis er midt på reisen. Vi har testet mange ulike ruter, og til tross for disse avvikene, blir kartvisningen betydelig mye bedre enn utgangspunktet.

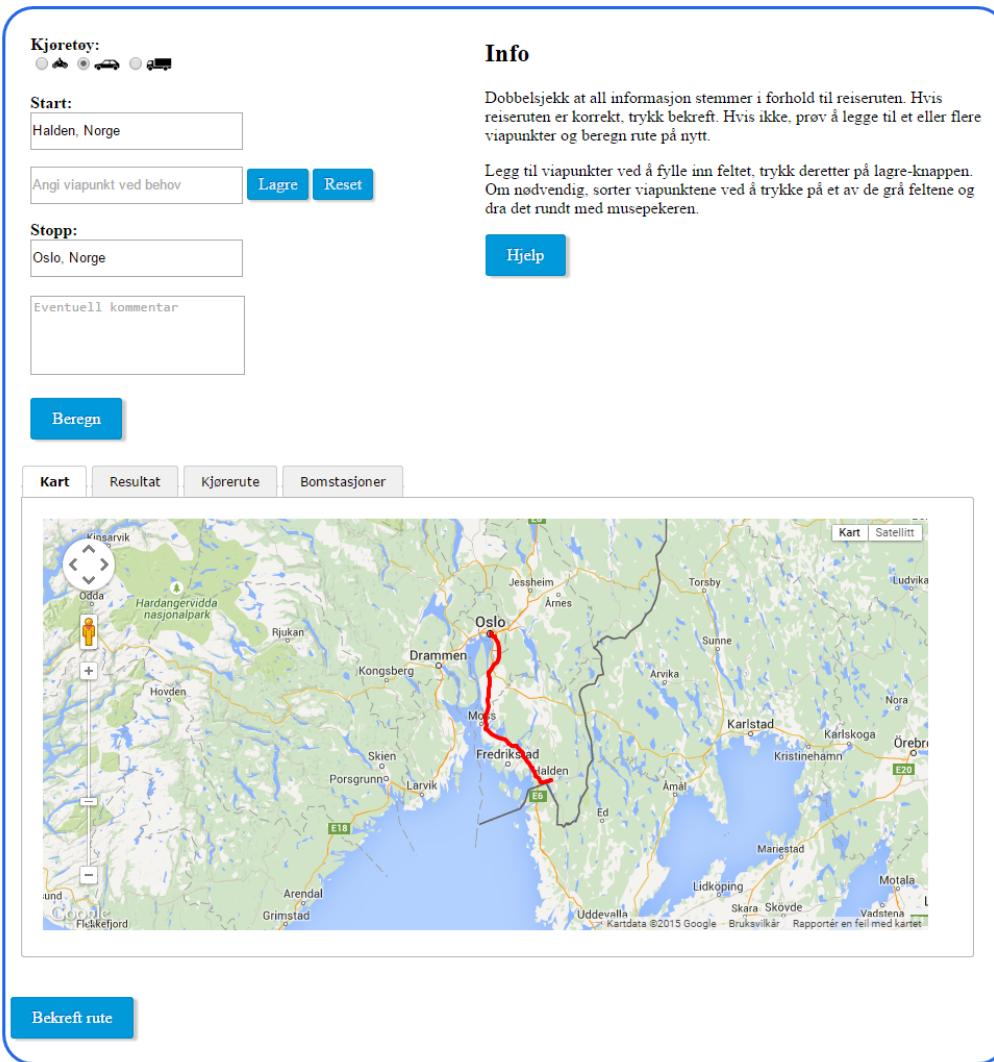
For å få kartet til og zoome til riktig nivå var vi nødt til å teste en del frem og tilbake for å finne ut hvilket zoomnivå som egnet seg best i forhold til lenge på kjøreruten. Vi har valgt å gjøre 6 tester for å bestemme zoomnivået. Vi har definert ulike zoomnivåer avhengig av om ruten er under 10km, 25km, 75km, 125km, 200km eller 500km. Figur 4.16 illustrerer automatisk zoomnivå på kjøreruten mellom Halden og Sarpsborg.



Figur 4.16: Automatisk zoomnivå på strekningen Halden - Sarpsborg

60 KAPITTEL 4. BRUKERTEST AV APPLIKASJONEN OG IMPLEMENTERING I HRESSURS

For å være sikre på at førstegangsbrukere føler seg komfortable med systemet, har vi valgt å legge til et felt med forklaringer til høyre på nettsiden. Det er slik Infotjenester har valgt å gjøre det i HRessurs og det er naturlig at vi gjør det samme for å tilpasse brukergrensesnittet vårt etter HRessurs. Dette betyr at vi må flytte fanevisingen av resultater lenger ned på siden. Vi har fått tilbakemelding på at kartet som vises er for lite, og ved å flytte kartet ned for å vise forklaringstekster, får vi også løst problemet med for lite kart, slik som figur 4.17



Figur 4.17: Applikasjonen etter utbedring

Vi har også fikset en annen feil i forhold til fanevisningen. Etter at man har beregnet en rute og navigert seg rundt i fanevisingen, vil den fanen som sist var åpen, være fanen man starter i når man gjør en ny beregning. Dette skaper problemer ved at kartet ikke lastes inn skikkelig. Vi har løst dette ved hjelp av Jquery ved å tvinge applikasjonen til å alltid starte i fanen nummer en, kart, både når en beregning utføres og når applikasjonen lastes inn på nytt.

4.5 Implementering

4.5.1 Tilpassing av brukergrensesnitt

Vi har sendt skjermbilder av applikasjonen vår til designansvarlig for HRessurs, og fått tilsendt skisser på hvordan det er tenkt at applikasjonen vår skal se ut i HRessurs. Vi kan da se, som vist i figur 4.18 og 4.19 på neste side, at applikasjonen er tenkt som en tilleggstjeneste. Dette er slik vi har forestilt oss at implementeringen vil foregå. Vår applikasjon skal være et tillegg brukere som registrerer reiseregninger kan velge å benytte seg av ved å trykke knappen «Kalkulator» som vist i øverste figur 4.18. Vi mener at dette navnevalget bør endres til et mer forklarende navn, da kalkulator høres mer ut som en kalkulator for å summere tall enn en beregningskalkulator for avstander og kostnader forbundet med bomstasjoner.

Start → Kjøring → Utlegg → Tillegg → Fullfør

Legg til kjørerute ved hjelp av vår kalkulator:

Kalkulator

Rute

Legg til ruten manuelt:

Kjøreruten skal beskrives så nøyaktig som mulig, med adresser for hvert delstopp. Kjøreruten skal oppgis så nøyaktig at den kan etterprøves i antall kilometer.

Total kilometer

Fremkomstmiddel

Privat bil

Kilometer utland

Årsak til eventuell omkjøring

Info

Kilometer med passasjer oppgis ved at man summerer antall kilometer pr passasjer, og legger inn totalsummen.

Eks. Per sitter på 50 km, mens Kari sitter på 30 km. Da legger du inn 80 km med passasjer.

⊕ Passasjerer

⊕ Spesielle tillegg

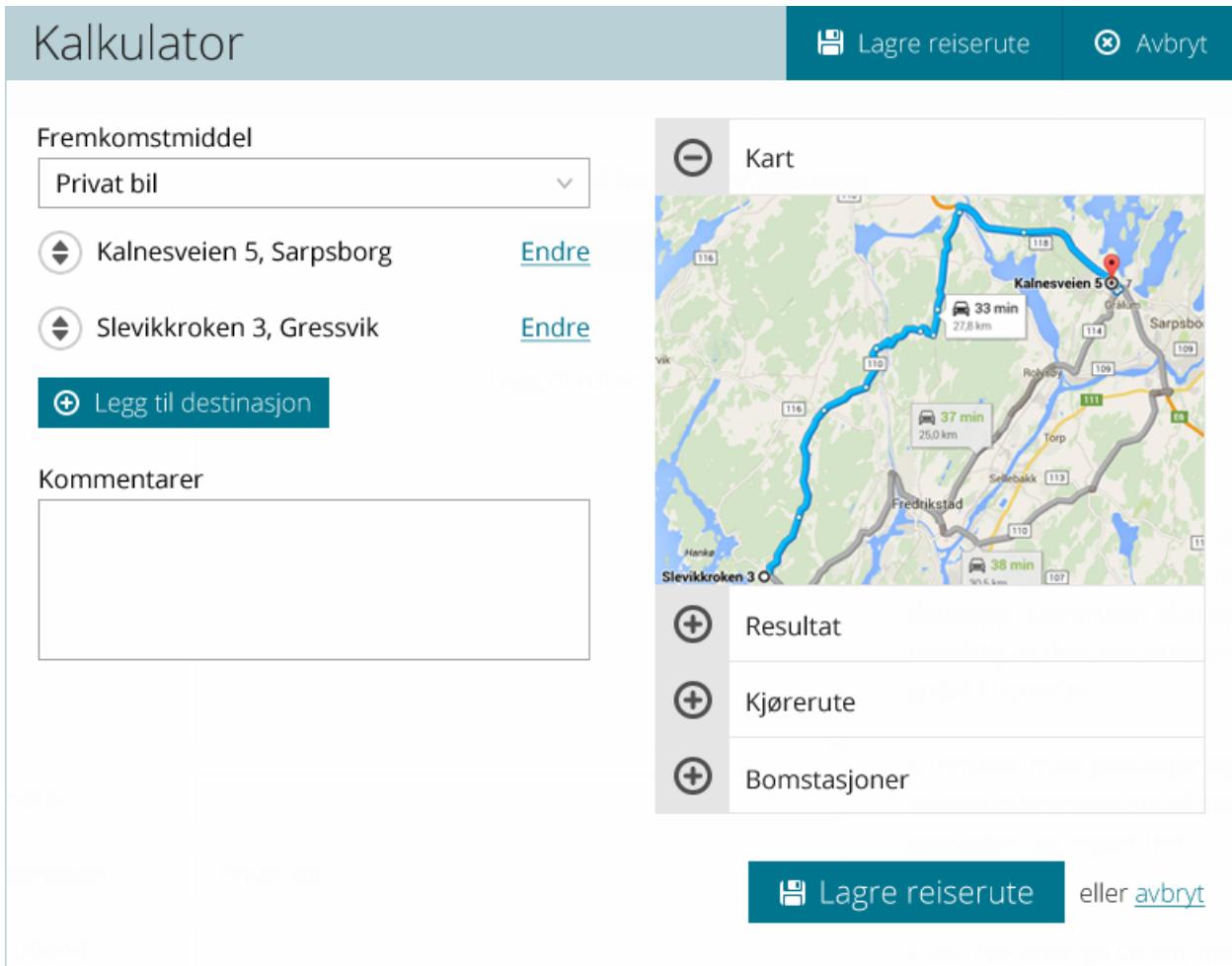
Legg til

Forrige

Trinn 2 av 5

Neste

Figur 4.18: Skjermbilde av HRessurs med knapp for å åpne vår applikasjon



Figur 4.19: Skisse av hvordan applikasjonen vil se ut i HRessurs

Det må også legges til en knapp som beregner verdiene før resultatene vises. Ved å trykke «lagre» etter å ha foretatt beregningene vil applikasjonen returnere de nødvendige data tilbake til HRessurs og fylle ut de beregnede verdiene i skjemaet for utfylling, slik at bruker ikke trenger å overføre data fra beregningskalkulatoren til riktige tekstfelt.

4.5.2 Implementering av kode

For å implementere applikasjonen må det foretas enkelte justeringer. For enkelthets skyld med tanke på å legge applikasjonen inn i HRessurs, gjøres de fire prosjektene applikasjonen er bygd opp av (CoordinateManager, Visveg02, Ruteplantjenesten og TravelRoute) om til ett prosjekt. Vi har lagret credentials (påloggingsinformasjon mot Vegvesenet) i koden, og dette flyttes over i en konfigurasjonsfil. Dette er blant annet for at det skal bli enklere å bytte ut disse senere.

Vi måtte også gjøre små endringer i koden, blant annet i interfacet ITravelRoute. Her viste det seg at vi hadde misforstått litt av hensikten med et interface, siden vi sjeldent har vært borti dette. Klassen TravelRoute benyttes i forbindelse med beregninger mot Ruteplantjenesten, og denne klassen implementerer interfacet ITravelRoute. Vi hadde misforstått og lagt alle funksjoner som er nødvendig for å foreta beregningene mot Ruteplantjenesten inn i interfacet. Interfacet skulle kun inneholde funksjoner som er nødvendig for å opprettholde lagdelingen ved bytte av ruteberegningsjeneste. Interfacet vi benyttet er vist i figur 4.20, og av alle funksjonene vi hadde definert, var det kun «Calculate» (illustrert med rød stjerne) som skulle være i interfacet.

```
public interface ITravelRoute
{
    2 references
    Result Search(Route input);
    2 references
    Result CalculateResultFromJson(String jsonInput);
    2 references
    Directions CalculateRoadDescriptionDirectionsObject(Result result);
    2 references
    CalculatedRoute Calculate(Route input);
    2 references
    ToHRessurs CalculateHRessursObject(Route input, Result result, Directions listOfRoadDescription);
    2 references
    CalculatedRoute CalcReturn(Route input, Result result, Directions listOfRoadDescription, ToHRessurs finalResult);
}
```

Figur 4.20: Interface basert på misforståelse

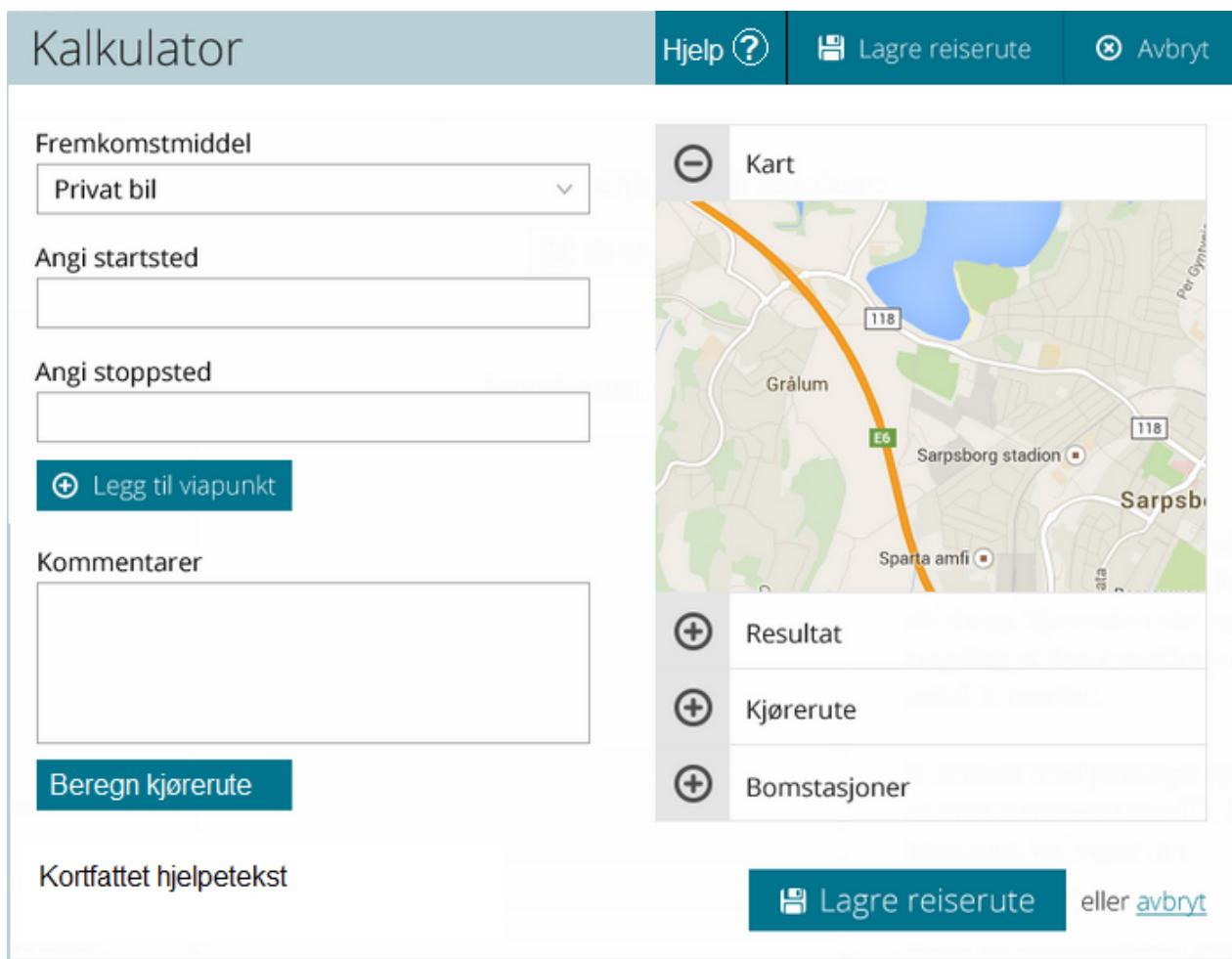
Ettersom vi har benyttet oss av en Web Service til kommunikasjon mellom klient og server, var det en veldig enkel prosess å få serverdelen av prosjektet vårt opp å gå på Infotjenesters server. Det var derimot en litt mer tidkrevende prosess å implementere klientsiden av applikasjonen. HRessurs består av mange forskjellige Javascriptfiler, og for å ha system i disse, strukturerer Infotjenester disse på en måte som virker tungvint for oss som ikke har kjennskap til strukturen fra før. Det vi derimot blir fortalt er at når man lærer seg hvordan strukturen er bygd opp, vil det være mye enklere å håndtere filene enn om de ikke hadde vært strukturert på den måten.

Infotjenester har utviklet sitt eget rammeverk, LazyFramework [36], som de benytter i forbindelse med HRessurs. De benytter også gjennomgående Ractive [35] i forbindelse med Javascript. Under implementering av klientsiden av applikasjonen har vi forklart hvordan Javascriptkoden vår er bygd opp, mens kontaktperson Petter Ekrann har tilpasset koden etter LazyFramework og Ractive. Vi ble ikke fortalt at de benytter Ractive i hele HRessurs før dagen vi startet med implementering. Vi har derfor ikke benyttet oss av Ractive i vår applikasjon. Hvis vi hadde fått vite dette tidligere, hadde vi kunnet benytte Ractive fra begynnelsen av, slik at implementeringen kunne gått raskere. At koden må tilpasses Ractive og LazyFramework innebærer ikke at hele koden må gjøres om, men at den «pakkes» inn i kode i henhold til Ractive og LazyFramework. Dette tar en del tid, men det er bedre å legge litt mer arbeid i å implementere applikasjonen da vedlikeholdet vil bli vesentlig enklere når applikasjonen følger samme standard som resten av HRessurs. Dersom vi skulle ha satt oss inn i både LazyFramework, strukturen på Javascriptfilene og hvordan Infotjenester benytter Ractive, for å selv implementere applikasjonen, ville dette tatt alt for mye tid av bachelorperioden. Det har derfor vært mer effektivt at vi har laget en separat applikasjon som implementeres av de som kan det.

4.5.3 Etter implementering

Vi har ikke fått tilsendt endelig skjermbilde av HRessurs der applikasjonen er ferdig implementert, ettersom applikasjonen fremdeles er under implementering. Det meste av funksjonalitet er allerede lagt inn, men dette er enda ikke knyttet opp mot et funksjonelt brukergrensesnitt. Det vil kun forekomme små justeringer i forhold til skissene i figur 4.18 og 4.19, og vi ser for oss at resultatet vil bli slik noe i figur 4.21. Her har vi gjort små endringer i skissene, og lagt til «Beregn kjørerute» kjørerute-knappen som manglet, og vi har også lagt til en knapp øverst til høyre, «Hjelp» der man kan åpne bruksanvisning av applikasjonen. Denne har vi lagt til som følge av at det er avsatt liten plass til hjelpetekst i skissene.

Med bakgrunn i rettigheter og sikkerhet får ikke vi som studenter tilgang til kildekoden for HRessurs. Vi får først tilgang til det endelige resultatet av implementeringen av applikasjonen vår, når en ny versjon av HRessurs publiseres. Dette vil ikke skje i løpet av de neste ukene, ettersom de må ferdigstille implementeringen og foreta en grundig testing før de kan lansere ny funksjonalitet.



Figur 4.21: Skjermbilde av HRessurs med knapp for å åpne vår applikasjon

5. Produktet

I dette kapittelet beskrives det endelige produktet. Her finnes en bruksanvisning for applikasjonen, og en veiledning i forhold til å videreutvikle/vedlikeholde koden.

5.1 Produktdokumentasjon

5.1.1 Brukerveiledning

Slik fungerer tjenesten.

1. Velg kjøretøy i kjøretøyvelgeren.
2. Fyll inn start og stopp i tekstfeltene. Velg et stedsnavn fra listen over steder som kommer frem under tekstfeltet etterhvert som stedsnavn skrives inn. Dersom stednavnene ikke hentes fra denne listen, vil ikke alltid ruten kunne beregnes.
3. Ved behov for viapunkter, fyll inn et stedsnavn slik som for start og stopp. Trykk «Lagre», og viapunktet legges til i listen over viapunkter. Et nytt viapunkt kan nå legges til hvis nødvendig. For å fjerne alle viapunktene, trykk «Reset». Alle viapunkter må lagres for at de skal tas med i beregningen. Ved feilmeldingen «Du har glemt å lagre et viapunkt», lagre viapunktet som er fylt inn i tekstfeltet eller fjern teksten i feltet. Viapunktene kan sorteres ved å ta tak i et viapunkt og dra dette til riktig plass. Rekkefølgen for viapunkter i listen, er rekkefølgen de beregnes i på kjøreruta. Øverste viapunkt er første viapunkt på reisen.
4. Ved tilleggskommentarer til kjøreruten, slik som forklaring i forbindelse med eventuelle omkjøringer, skrives disse i kommentarfeltet.
5. Trykk nå «Beregn». Kjøreruten beregnes.
6. Kontroller den beregnede kjøreruta. Vegvesenet har per i dag noen unøyaktigheter i forhold til kjørerutene de beregner. Disse unøyaktighetene oppstår i hovedsak i nærhet til start og stopp, slik at det på lokale ruter er viktig å være obs på at kjøreruten er riktig.
7. Når kjøreruten som er beregnet ser riktig ut, trykk knappen «Bekreft rute». Start, stopp, eventuelle viapunkter, kjøretøy, avstand, totale bompengekostnader og den komprimerte kjørebeskrivelsen legges nå til i de aktuelle feltene i reiseregningen, slik at disse ikke må fylles ut manuelt.

Obs: «Bekreft rute»-knappen er spesiell for vår frittstående applikasjon. I brukerveiledningen som legges til i HRessurs, må «Bekreft rute» byttes ut med «Lagre».

5.1.2 Vedlikehold

I dette kapittelet tar vi for oss de tekniske hensyn det er viktig å følge dersom applikasjonen skal videreutvikles/vedlikeholdes. Det er utfyllende kommentarer i koden, men her vil vi gå gjennom de viktigste trekkene og objektflyten som må følges i henhold til lagdelingen.

Klientside - Javascript

frontend.js

I denne filen ligger all kode i forbindelse med Google API-et og viapunkter. Øverst finnes koden i forbindelse med Google Autocomplete og innhenting av koordinater før beregning. Den siste funksjonen Plot, plotter ruten på kart etter alle beregninger er fullført.

convert.js

I denne filen ligger all kode i forbindelse med konvertering av koordinater, lagring og sortering av viapunkter, kommunikasjonen mellom klient-/serverside og presentasjon av de beregnede data (med unntak av plotting på kart, som ligger i frontend.js).

Gjennomgang av funksjoner og rekkefølge

Det er bare et fåtall funksjoner som aktiveres direkte fra brukeren. Når brukeren trykker «Beregn» kjøres funksjonen calcRoute som først sjekker om riktige data er fylt inn. Dersom ingen feil registreres kalles funksjonen på getCoordinateStart. getCoordinateStart henter koordinater for startpunkt og kaller så getCoordinateStop som henter koordinater for stoppunkt.

Når brukeren legger til et viapunkt vil først funksjonen via1 som henter koordinatene fra Google kjøre, før de sendes til konverteren i convert.js. Her legges de inn i et array hvor nøkkel-elementet, stedsnavnet, legges til i viapunktlisten (ul-liste). Funksjonen saveValue henter nøkkel-elementene fra viapunktlisten før rekkefølgen på viapunktene kontrolleres og eventuelt settes i ny rekkefølge.

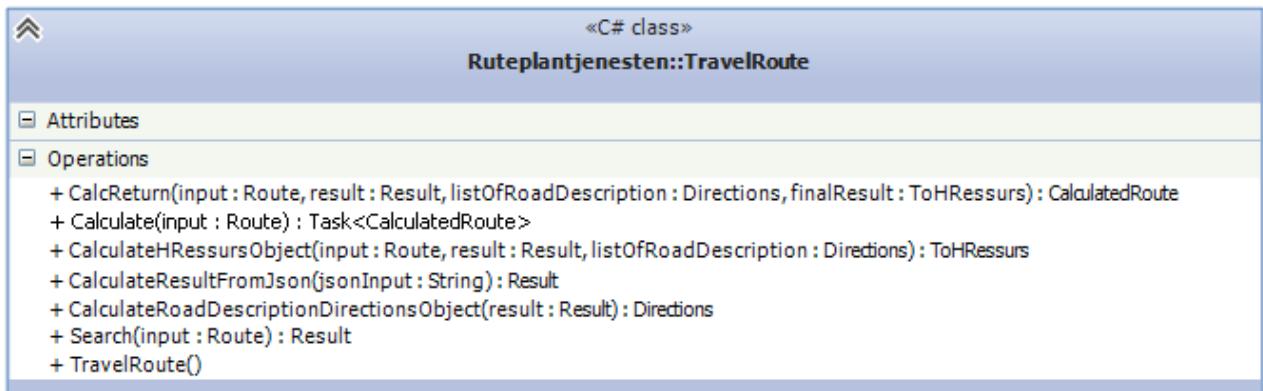
Når all informasjon er samlet inn fra inputfeltene, sendes disse til Web Service i funksjonen openWebservice. Denne foretar en AJAX-request mot Web Service. Når AJAX-requesten mottar responsen fra Web Service, oppdateres disse i DOM slik at de blir synlige for bruker. Ved feilmelding fra Web Service, skrives det ut en popup-melding til bruker, om at angitt rute ikke kunne beregnes.

Viktig informasjon angående Google

Dersom Infotjenester ikke ønsker å benytte seg av Google API-et lenger, er de nødt til å gjøre større endringer i koden. Vår applikasjon er bygget opp rundt Google, og vi benytter oss både av Google Maps og Autocomplete til de viktigste delene av applikasjonen. Uten Google vil vi ikke få tilgang til koordinatene Ruteplantjenesten trenger, og vi vil heller ikke kunne vise kjøreruta grafisk.

Serverside - C#

På serversiden ligger alle funksjonene i TravelRoute, se figur 5.1. Her styrer funksjonen Calculate alle beregninger. Denne kalles fra Web Service, med input-objektet av klassen Route. Rekkefølgen funksjonene kjøres i er illustrert i figur 5.2.



Figur 5.1: Klassen TravelRoute der beregninger mot Ruteplantjenesten foregår

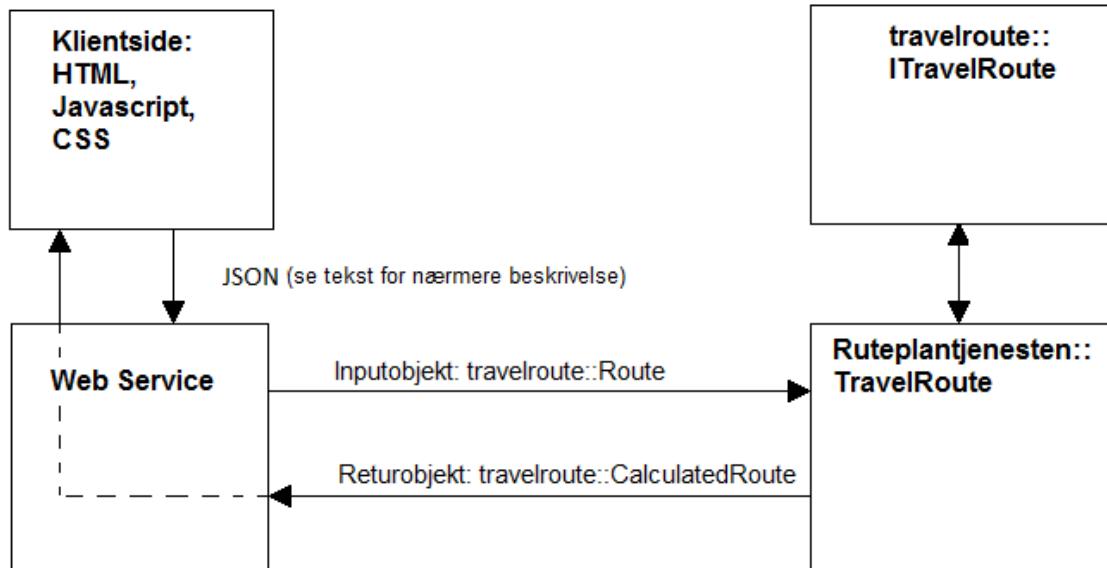


Figur 5.2: Rekkefølgen funksjonene i TravelRoute kjøres

All bruk av koordinatkonvertering er spesifikk for Ruteplantjenesten. Ved et bytte av ruteberegningsjeneste som benytter seg av koordinater på et annet format, kan man trenge en annen koordinatkonvertering. Dersom ruteberegningsjenesten benytter seg av latitude/longitude, slipper man å benytte en konverteringsfunksjon.

Dataflyt

Hovedtrekkene bak strukturen i applikasjonen er nærmere beskrevet i kapittel 3.2.1. I dette kapittelet blir dataflyten mellom de ulike delene applikasjonen bestå av nærmere beskrevet. En fullstendig oversikt over dataflyten er illustrert i figur 5.3.



Figur 5.3: Dataflyten fra klientside, til serverside og tilbake

Fra klientsiden foretas det en AJAX-request mot Web Service. Her skal data være spesifisert på følgende format:

Start

start = [Stedsnavn, Xkoordinat, Ykoordinat]

Stopp

stopp = [Stedsnavn, Xkoordinat, Ykoordinat]

Via

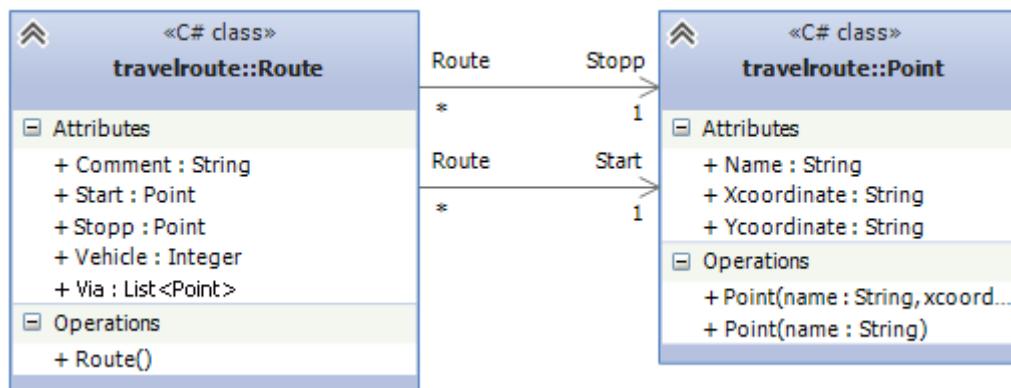
via = [[Stedsnavn, Xkoordinat, Ykoordinat], [Stedsnavn, Xkoordinat, Ykoordinat]]
(opp til fem viapunkter)

Vehicle

Vehicle = intverdi 1-3. 1 = motorsykkel, 2 = personbil, 3 = lastebil

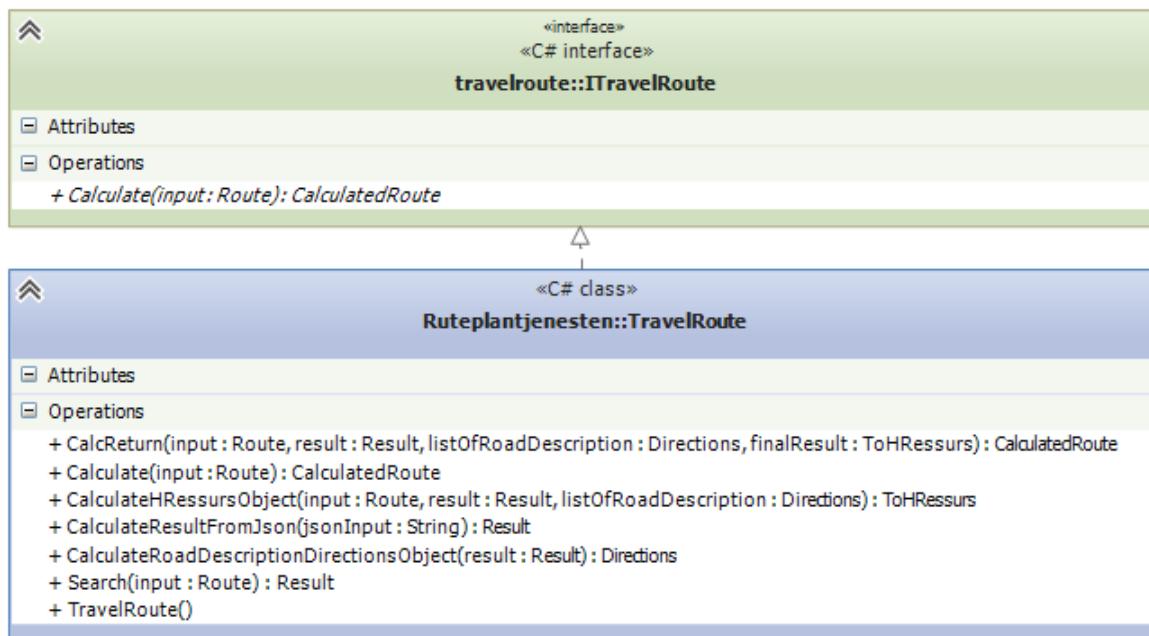
Disse inputverdiene skal sendes over som et JSON-objekt, og variablene må konverteres til JSON slik: JSON.stringify(Start: start, Stopp: stopp, Via: via, Vehicle: vehicleSelected, Comment: comment);

I Web Service gjøres JSON-input fra klientsiden om til et objekt av type Route, se figur 5.4. Start, stopp og viapunkter i Route, er objekter av typen Point (stedsnavn, x- og y-koordinat)



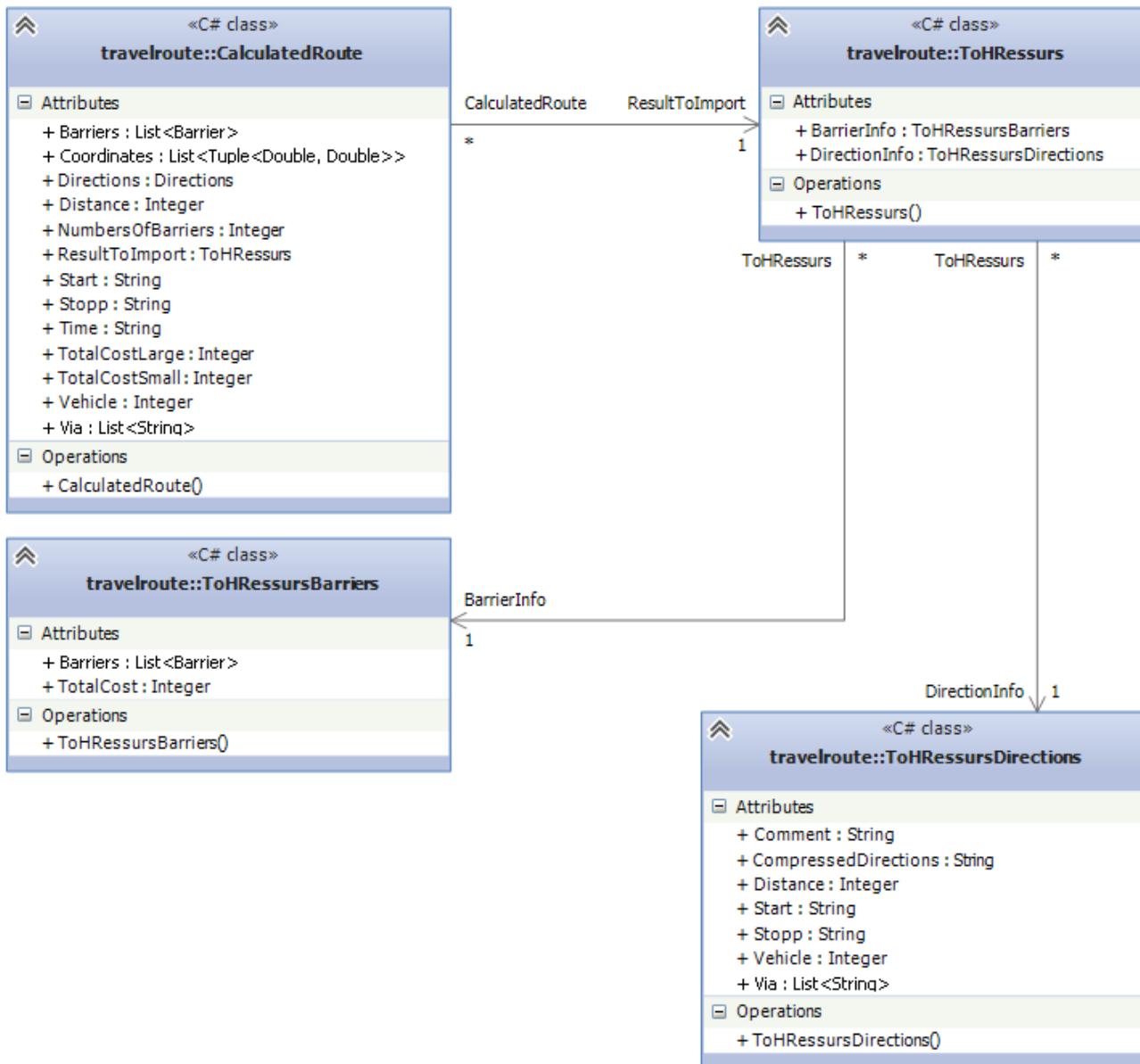
Figur 5.4: Klassen Route

Fra Web Service sendes objektet av type Route over til klassen Ruteplantjenesten::TravelRoute, som foretar beregninger mot Ruteplantjenesten. Denne klassen implementerer interfacet `travelroute::ITravelRoute`, se figur 5.5.



Figur 5.5: TravelRoute implementerer ITravelRoute

I Ruteplantjenesten::TravelRoute beregnes kjøreruta, før resultatet returneres til klientsiden via Web Service, som et object av klassen CalculatedRoute, se figur 5.6. Dette objektet består av et eget ToHRessurs-objekt, slik at det vil bli enklere å legge riktig informasjon over i HRessurs etter implementering. ToHRessurs er igjen delt opp i to objekter, et objekt som inneholder all informasjon om kjøreruta, derav avstand, komprimert kjørerute, kjøretøy og angitte steder. Det andre objektet inneholder all informasjon i forbindelse med bomstasjoner. ToHRessurs er delt i to objekter siden kjørerute og utlegg er to ulike deler av HRessurs, og de beregnede verdiene skal derfor legges til på forskjellige sider i utfyllingen av reiseregningen.



Figur 5.6: CalculatedRoute, objektet som returneres til klientside

Begrensninger i Ruteplantjenesten

Vegvesenet har en maksgrense på 2500 spøringer om dagen. Det betyr at dersom Infotjenesters kunder skulle registrere mer enn 2500 reiseregninger om dagen der de benytter seg av beregningskalkulatoren, vil de kunne miste tilgangen til API-et ut dagen. Vi har tatt dette opp med oppdragsgiver som mener det er lite sannsynlig at de vil komme over 2500 spøringer i løpet av en dag.

Vi har tatt kontakt med Vegvesenet for å høre hva vi kan gjøre dersom man går over denne grensen. De sier denne grensen mest sannsynlig vil øke med tiden, men at de ikke tilbyr noen løsning for flere enn 2500 spøringer, men de sier også: «Grensen vår er ikke hugget i stein, men et greit kompromiss for å dele våre beregningsressurser». Vi har også vært i kontakt med selskapet som står bak selve Ruteplantjenesten, Triona [41]. De forteller også at det ikke er en nøyaktig grense på 2500 spøringer. Det er derfor ikke sikkert det har noe å si om man skulle komme over denne grensen, men for å være helt sikre på at dette ikke skulle skape problemer for vår Infotjenester har vi bedt Vegvesenet om en ekstra bruker slik at vi kan teste grensen for å se hva resultatet blir når man passerer 2500 spøringer i løpet av en dag. Grunnet treghet i forbindelse med å opprette en ny bruker hos Vegvesenet, har vi ikke fått tilgang til en ny bruker tidsnok til at vi rekker å teste dette. Vi kan ikke risikere å miste tilgang til Ruteplantjenesten i denne perioden, selv om det bare er ut dagen, da vi er avhengige av kontinuerlig tilgang for å foreta den siste finjusteringen av applikasjonen før innlevering.

Vegvesenet gjorde oss oppmerksomme på firmaet Norkart [42], som er tilbyder av ruteberegningsjenesten Ferd [43] som blant annet Finn og 1881 benytter seg av. Dersom det skulle være nødvendig med en tjeneste som tilbyr mer enn 2500 spøringer, er dette et alternativ. De har en kostnad på 0,15kr per spørring. Vi tar utgangspunkt i at det foretas minimum 2500 spøringer om dagen, hvis Infotjenester tvinges til å bytte ruteberegningsjeneste som følge av for mange spøringer. Minimumskostnadene for bruk av Ferd blir da:

$$2500 \text{ spøringer} \times 0,15 \text{ kr.} = 375 \text{ kr. per dag.}$$

I utgangspunktet regnes det 230 arbeidsdager i løpet av et år [44] for en heltidsstilling, men siden fordelingen av disse dagene varierer utifra bransje og varierende arbeidstider, er vi nødt til å foreta en omrentlig beregning av totalkostnader ved bruk av Ferd.

$$230 \text{ dager} \times 375 \text{ kr.} = 86\,250 \text{ kr. i året.}$$

$$365 \text{ dager} \times 375 \text{ kr.} = 136\,875 \text{ kr. i året.}$$

Minimumskostnadene i løpet av et år, med 2500 spøringer, vil være et sted mellom 86 000 og 137 000 kr. i året. Dette er nok så dyrt som en gratis tilleggstjeneste, så da er et alternativ at beregningsapplikasjonen blir en tilleggstjeneste i en dyrere abonnementsløsning.

Vi har foreslått for Infotjenester at de skaffer seg en ekstra bruker som backupløsning hvis de mot formodning skulle komme over 2500 spøringer enkelte dager. Hvis de stadig passerer 2500 spøringer om dagen, vil Ruteplantjenesten måtte byttes ut med en annen leverandør, men hvis grensen kun passeres for eksempel en gang i måneden, vil det være en akseptabel løsning med en backupbruker.

5.2 Hvordan kan applikasjonen videreutvikles?

Vi har gjort oss noen tanker om hvordan applikasjonen vår kan utbedres. Vi har sett på enkelte av disse, og gjort oss noen tanker om hvordan de kan gjennomføres.

Ide 1: Ferjekostnader på strekningen

Slik applikasjonen er nå, er det ikke mulig å beregne kostnader i forbindelse med ferjeoverganger. Vi sjekker kun om ruten inneholder ferje. Dersom kjøreruten inneholder ferje, gir vi tilbakemelding om at det er ferjeovergang(er) på strekningen og at ekstra kostnader kan påløpe.

Det finnes i dag ingen samlet oversikt over ferjepriser i Norge, men den dagen dette blir tilgjengelig kan man videreutvikle applikasjonen slik at den også beregner ferjekostnader. Dersom denne funksjonaliteten skal legges til, må man legge til en mulighet for å angi antall personer i bilen, og eventuelt alder, da det er av betydning for prisen på de fleste ferjeoverganger.

Før dette forslaget kan gjennomføres i praksis må det også skje en endring i regelverket for reiseregninger. Det er i dag strengere regler for ferjeutgifter enn for bompengekostnader. I dag må ferjekostnader dokumenteres med kvittering, da det er MVA på disse utgiftene. Så lenge man er avhengig av kvittering for ferjekostnader, vil det ikke være hensiktsmessig å legge til en funksjonlitet for å beregne ferjekostnader.

Ide 2: Kjørerute i utlandet

I dag er applikasjonen forbeholdt kjøreruter i Norge. Dette er en følge av at vi benytter oss av Ruteplantjenesten som kun foretar beregninger i Norge. Dersom hele eller deler av reisen foregår i utlandet, kan man ikke benytte applikasjonen til å beregne avstand og bompenger. Eneste mulighet for å benytte applikasjonen ved delvis kjøring i utlandet, er å benytte applikasjonen til å beregne avstand og bompenger fram til grensen, for så å ta i bruk en annen ruteplanleggingstjeneste for å beregne avstanden i utlandet.

Vi ser for oss at det vil være hensiktsmessig å finne en løsning slik at man også kan beregne ruter i utlandet, men da må man eventuelt bytte ut Ruteplantjenesten med et annet API, eller finne en løsning for å knytte flere API-er sammen.

Ide 3: Mobilvennlig applikasjon

Etter brukertesten fikk vi tilbakemeldinger på at applikasjonen ikke er mobilvennlig. Applikasjonen fungerer på mobil, men brukergrensesnittet er ikke optimalt siden et er tilpasset en dataskjerm som er vesentlig større enn en mobilskjerm.

Vi har valgt å ikke fokusere så mye på dette i denne omgang, siden applikasjonen skal integreres i et system som allerede har en god løsning for bruk på mobil.

Ide 4: Dynamisk valg av steder

Vi har fått tilbakemeldinger fra flere testere med et ønske om en mer dynamisk endring av rute. Vi vet at Google har funksjonalitet for å flytte kjøreruten ved å ta tak i den inntegnede «streken» som viser kjøreruta og flytte denne dit man ønsker.

Vi har undersøkt mulighetene for å implementere dette i vår applikasjon, men ikke funnet noe støtte for denne funksjonaliteten ettersom vi benytter oss av Polyline. Det er mulig å gjøre dette ved bruk av en tilleggsfunksjonalitet hvis man plotter ruta ved hjelp av Google viapunkter, men det finnes derimot ingen funksjonalitet for å stanse Google fra å beregne en rute utenfor landegrensene. Så lenge denne funksjonaliteten ikke er tilgjengelig, vil det heller ikke være nødvendig med en funksjon for å dynamisk flytte kjøreruta på kartet. Denne funksjonaliteten vil ikke hindre Google fra å tegne ruten utenfor Norge, og det vil derfor ikke være mulig å kombinere Ruteplantjenesten med Googles funksjonalitet for dynamisk endring av kjørerute.

Ide 5: Gjenta beregning av samme rute

Ofte vil ansatte kjøre samme strekning flere ganger, for eksempel hvis bedriften de arbeider i ofte er på kurs/møter i en annen bedrift eller at de ansatte pendler mellom ulike kontorer bedriften har.

Dersom bruker kjører samme rute flere ganger, vil avstanden i de fleste tilfeller være den samme hver gang med mindre bruker var tvunget til å kjøre en omvei. Bompengekostnadene vil derimot kunne være endret siden sist denne kjøreruten ble beregnet. I dagens løsning har vi ikke tatt med koordinater i resultatet som lagres i reiseregningene, men dersom URL-en mot Ruteplantjenesten (denne inneholder koordinatene for angitte steder) også lagres i forbindelse med reisen, vil det være enkelt å gjennomføre en ny beregning av den samme strekningen. Da vil man kunne få oppdaterte priser på bomstasjonene uten å måtte fylle inn alle detaljer på nytt. Dette er en funksjonalitet som må legges inn i HRessurs i forbindelse med visning av tidligere reiser.

Ide 6: GPS

Dersom en bruker benytter seg av HRessurs på mobil til å fylle ut reiseregninger, kan det være hensiktsmessig å kunne benytte nåværende GPS-posisjon framfor å måtte angi stedsnavn. Hvis man ønsker å fylle ut reiseregningen mens man befinner seg på enten start- eller endepunktet for kjøreruta, for eksempel på kontoret som var utgangspunktet for reisen, bør bruker ha mulighet til å benytte GPS-posisjon istedenfor å måtte fylle inn stedsnavn. Dette er en funksjonalitet som vil være spesielt rettet mot mobilversjonen av HRessurs.

Ide 7: Tilpasset zoom-område basert på koordinater

En annen måte å tilpasse zoomnivået ved plotting på kart, er å tilpasse området til alle koordinater på kjøreruten. Ved å benytte fire variabler (for eksempel limitEast, limitNorth, limitSouth og limitWest), kan man gå gjennom alle koordinater på kjøreruta, og dersom et koordinat er lengre nord enn koordinatet som er lagret i variabelen north, overskrives dette med det nye koordinatet. Tilsvarende må alle koordinater sjekkes mot koordinatene som er lagret i variablene east, south og west. Slik kan man få lagret alle ytterpunktene på kjøreruta. Siden vi allerede konverterer alle koordinater fra UTM til latitude/longitude, vil det være mest fornuftig å legge denne grenseberegningen inn i sløyfen hvor koordinatene konverteres.

Når alle koordinatene er kontrollert, må zoomnivået tilpasses til nærmeste nivå der alle koordinater havner innenfor kartutsnittet definert av de fire grenseverdiene. Hvordan dette håndteres i Google er ikke sjekket nærmere opp i, annet enn at de har ulike måter å tilpasse kartutsnittet.

6. Diskusjon

I dette kapittelet vil vi oppsummere prosjektperioden, i den hensikt å finne ut hvorvidt vi har oppnådd målene med oppgaven, om gjennomføringen av prosjektet gikk slik vi hadde planlagt, hva vi kunne gjort anderledes og hva vi har lært.

Gjennomføring av arbeidsmetoden

I kapittel 1.4.2 gikk vi gjennom hvordan vi planla å gjennomføre bachelorprosjektet. Vi har i stor grad fulgt planen i forhold til å dele opp prosjektet i ulike faser, som illustert i figur 1.5. Vi har hatt noen avvik i forhold til planen, men dette går for det meste på lengden av de ulike fasene. Planen var å ha liten overlapp mellom analyse og utvikling, men at det allikevel skulle være et klart skille mellom fasene. Ettersom vi var avhengig av å begynne å utvikle deler av applikasjonen for å se om løsningsmetodene vi kom frem til under analysefasen lot seg gjennomføre, ble det til at analysefasen i all hovedsak foregikk parallelt med første del av utviklingsfasen. Testperioden og slutføringen ble kortere i forhold til det pila i figuren viser.

Vi har jobbet 2-3 dager sammen på skolen hver uke slik vi planla, i tillegg til å jobbe en del hjemmefra når det gjaldt oppgaver som godt kunne løses individuelt. Vi har hele tiden kommunisert via Facebook, men i perioder der mange beskjeder har måttet gis, har det blitt vanskelig å holde oversikt over alle meldingene. Under siste innspurten av prosjektet der viktige beslutninger har blitt tatt, har vi derfor også kommunisert på Skype for å enklere ta avgjørelser i fellesskap og for å formidle viktige beskjeder.

Siden vi har benyttet Scrum-metodikk i utviklingen av applikasjonen, har vi fått et innblikk i hvordan lignende prosesser ofte foregår ute i arbeidslivet. Vi har fått oppleve at oppdragsgiver ikke alltid vet hvordan de ønsker at ting skal være og at det kan bli endringer underveis. Dette har vært en god erfaring for oss og vi føler at applikasjonen har blitt mye bedre basert på dette enn den ville blitt dersom vi skulle planlagt alt fra begynnelsen. Store deler av applikasjonen har blitt til som en følge av at vi har tilegnet oss mye ny kunnskap underveis.

Vi har ikke fulgt Scrum helt slik vi hadde planlagt med nøyaktig føring av sprinter, user stories, product backlog og sprint backlog. Vi har hatt en oversikt over ønsket funksjonalitet, og her har nye ideer blitt lagt til etterhvert som de har oppstått. Til hver sprint har vi bestemt hva som skulle gjøres i løpet av de neste to ukene. Vi gjennomførte to-ukers sprinter slik som planlagt, men det ble litt forskyvning i forhold til de planlagte to-ukers sprintene i figur 1.6. Vi startet ikke med Scrum før i begynnelsen av februar, siden det var mye nytt å sette seg inn i før vi kunne komme i gang med effektiv jobbing. De første ukene gikk mye ut på å prøve oss frem ved å utvikle en helt enkel testversjon, både for å se at prosjektet lot seg gjennomføre og for å få et innblikk i hvilke muligheter vi hadde når vi skulle utvikle applikasjonen. Vi begynte først med sprinter da vi begynte å dele opp applikasjonen etter lagdelingen som er beskrevet i kapittel 3.2.1.

Det å dele prosjektet slik at alle kunne jobbe parallelt var veldig enkelt etter at vi begynte med lagdelingen. Vi hadde derimot problemer med å bestemme oss for hvordan vi skulle dele prosjektfilene i forbindelse med applikasjonen. Allerede første uka testet vi GitHub, ettersom de tilbyr en god versjonskontroll og lar flere jobbe parallelt mot samme prosjekt. Dette fungerte ikke for oss, og det var kun den som opprettet prosjektet, som fikk åpnet dette. Vi andre fikk feil i forbindelse med filstier (path) som lå igjen i prosjektet fra den som opprettet prosjektet. Vi bestemte oss for å ikke bruke mer tid på GitHub og dette er grunnen til at vi som nevnt i kapittel 1.4.2 valgte å benytte Dropbox. Dropbox har ingen mulighet for å slå sammen endringer fra flere filer til en fil («merge» filer). Det vil si at vi ikke kan gjøre endringer i samme fil samtidig, slik som man kan i GitHub. Løsningen ble at vi tok en kopi hver av hovedmappen, slik at vi jobbet i hver vår kopi av prosjektet. Etter at vi hadde utført de endringene vi skulle i vår lokale kopi av hovedmappa, måtte en av oss slå disse sammen til et prosjekt og oppdatere hovedmappen på Dropbox. Dette fungerte fint ved små endringer, men ved store endringer fikk vi en del problemer vi måtte bruke mye tid på å løse. I ettertid ser vi at det ville vært mye bedre å heller bruker mer tid i begynnelsen på å få GitHub til å fungere, og at vi da ville ha spart oss for mye av tiden vi har brukt på å manuelt sette sammen de ulike delene av prosjektet.

Måloppnåelse

I kapittel 1.4.1 gikk vi gjennom målene for prosjektet. Hovedmålet vårt var å gjøre utfylling av reiseregninger i HRessurs både enklere og raskere. Vårt første delmål var å «foreta en nøyaktig beregning av ruten med fokus på bompengekostnader og avstander». Vi har oppnådd dette målet så godt det lar seg gjøre, da det i innspurten dukket opp noen få unøyaktigheter i Ruteplantjenesten. Disse feilene ligger hos Vegvesenet, men vi har vært i kontakt med de og slike feil er stadig under utbedring.

Vårt andre delmål var å «kunne beskrive den beregnede ruta detaljert nok til å etterkomme alle regler for føring av reiseregninger». Vi har kommet fram til en kortfattet beskrivelse av kjøreruta, der både start, stopp og eventuelle viapunkter tas med i beskrivelsen. Denne gir en kort og konkret beskrivelse, og vil være mulig å etterprøve. Det eneste den mangler er stedsnavn underveis på ruta, men at dette ikke med i beskrivelsen, er også en mangel hos Vegvesenet.

Vår tredje og siste delmål var å «implementere vår ferdige applikasjon i HRessurs». Dette er som nevnt i kapittel 4.5 en arbeidsoppgave som ligger utenfor vår kontroll, da det er Infotjenester som må foreta implementeringen. Implementeringen er godt i gang, og det er i hovedsak kun tilpasningen av brukergrensesnittet og en grundig testing igjen før applikasjonen er klar til å publiseres som en del av HRessurs.

Selv om implementeringen ikke er ferdig når vi avslutter vår bachelorrapport, kan vi i all hovedsak fastslå at hovedmålet er oppnådd. Vi har utviklet en applikasjon, som oppfyller de kravene som var fastsatt, og som nå er under implementering etter Infotjenester standard.

Utvikling

Vi har lært mye i dette prosjektet. Da vi startet hadde vi knapt hørt om C# og vi hadde heller ikke så mye erfaring innen HTML, Javascript og CSS. Derfor har det også vært mye «prøv og feil» når vi har programmert. Vi har ikke hatt fullstendig oversikt over hvordan ulik teknologi og løsningsmetoder fungerer i praksis, og for å få oversikt over dette, har vi måtte gjøre en del tester. Dette har vi lært veldig mye av, og vi ser andre bruksmåter for flere av teknologiene og løsningsforslagene vi har forkastet i forhold til vår applikasjon.

Det som for oss var helt nytt i forhold til programmering, var prinsippet bak lagdelingen som er beskrevet i kapittel 3.2.1. I våre programmeringsfag har det stort sett kun vært fokus på å lære det grunnleggende innen et programmeringspråk, og ingen fag tar for seg hvordan programmering stort sett foregår i praksis. Vi forstod raskt fordelene bak en lagdelt struktur, og når oppdragsgiver ga oss en liten innføring i tankegangen bak en lagdelt struktur forsto vi med en gang hvorfor det må være slik i store applikasjoner. Ved å dele applikasjonen i flere lag blir både vedlikehold og utskifting av deler av applikasjonen mye enklere. Vi har derfor fått et større innblikk i programmering og applikasjonsdesign enn det vi har fått i programmeringsfagene vi har hatt. Dette har vært veldig lærerikt for oss alle og vi føler at det har tatt programmeringskunnskapene våre opp flere nivåer.

Vi har diskutert mye frem og tilbake både internt i gruppen, og også med oppdragsgiver rundt hva som er best med hensyn til fremstilling av beregnet kjørerute. Vi har diskutert om det er behov for et kart siden brukeren allerede har gjennomført en reise og ikke trenger kartet for å finne fram. Vi kom til at det vil være raskere og enklere å kontrollere om beregnet rute stemmer ved å ta en rask titt på et kart fremfor og lese en tekstlig beskrivelse av ruten. Vi har derfor valgt å beholde begge deler, slik at brukerne skal kunne kontrollere kjøreruta på den måten de selv foretrekker. Ved utfylling av reiseregningen vil det være den tekstlige beskrivelsen som er aktuell, men for kontroll av reisen, vil de fleste vi har snakket med foretrekke en kartvisning.

Som vi har vært inne på i kapittel 3.1.3, har vi hatt en del utfordringer med kartvisningen. Den største utfordringen har vært å få Ruteplantjenesten og Google Maps til å samkjøre. Siden Google Maps ikke styres av landegrenser slik Ruteplantjenesten gjør, har vi måttet tvinge ruten til å holde seg innenfor landegrensene. Vi har valgt å gjøre dette ved hjelp av Polyline, som ikke har noen innebygd funksjon for å endre ruten dynamisk, slik flere testere har hatt et ønske om. Vi har valgt å bruke den tiden vi har hatt til rådighet på å finne en så god løsning som mulig uten interaktivitet i kartet, fremfor for å finne en halvgod løsning med interaktivitet.

Som nevnt i kapittel 3.2.1, opplevde vi at vi noen ganger brukte unødvendig mye tid på å finne ut av feil. Vi hadde blant annet et tilfelle der en funksjonalitet i forbindelse med Web Service var faset ut, og kun Internett Explorer ga en forklarende feilmelding. Vi fikk da en påminnelse om viktigheten av å både sjekke dato for eksemplene og å teste i ulike nettlesere. Å teste i ulike nettlesere har i ettertid vært veldig viktig spesielt med tanke på viapunktene, siden vi her blant annet fikk problemet med å ikke kunne trykke seg inn i et tekstfelt i nettleseren Firefox (se kapittel 3.2.3). Innad i gruppa har vi hatt den fordelen at vi har hatt ulike favorittnettlesere vi har testet i, og vi har på denne måten testet applikasjonen jevnlig i både Firefox, Chrome og Opera. Internett Explorer derimot er det ingen av oss som har hatt som standardnettleser, men denne har vi også testet med jevne mellomrom som følge av feilen vi fikk i forbindelse med Web Service.

Testing

Gjennom hele utviklingsperioden har vi testet applikasjonen og utbedret feil og mangler vi har oppdaget. Da applikasjonen var klar til brukertesting, som beskrevet i kapittel 4, trodde vi derfor at de fleste feil var fjernet. Vi tenkte ikke på at når vi selv har testet, har vi testet slik applikasjonen er ment å brukes. Vi visste for eksempel at beregningen tok tid, og ventet derfor til resultatene ble vist. Vi tenkte ikke over at når en bruker ikke får beskjed om at reiseruta beregner, kom han/hun til å trykke på beregnknappen igjen i den oppfattningen av at det første trykket ikke ble registrert. Dette førte til OVER_QUERY_LIMIT-feilmeldinger, at viapunktene ble slettet og ikke kom med i beregningene og at kjøreruten av den grunn ikke ble beregnet riktig. Med tanke på viapunktene hadde vi vært innom veldig mange ulike løsninger før vi fikk denne til å fungere stabilt. Det kan virke ulogisk å lagre viapunktene før man trykker beregn, men dette var den beste og mest stabile måten å gjøre det på. Der er ikke mulig å hente flere enn ett koordinat av gangen, og derfor ville en løsning der alle viapunktene lagres samtidig ikke fungere. Vi hadde ikke beskrevet denne funksjonaliteten godt nok i brukergrensensnittet, og testerne opplevde blant annet å ikke få med alle viapunktene i beregningene.

En av våre testere oppgav å ha testet både i Google Chrome og Mozilla Firefox. I ettertid ser vi at vi burde hatt dette som et eget spørsmål i brukertesten, da dette ville gitt oss enda bedre forståelse for hvordan applikasjonen fungerte, og om opplevelsen av applikasjonen hadde sammenheng med bruk av nettleser. Vi burde også spurt hvilken versjon av nettleseren det var testet i, da dette også er relevant med tanke både ny og utdatert funksjonalitet. Før vi fikk lastet opp applikasjonen på skolens Windows-server, har vi kun testet applikasjonen lokalt på våre Windows-maskiner. Vi har fått tilbakemeldinger fra testere om at applikasjonen også fungerer på nettleseren Safari på MAC (Apple). Vi har lært til en annen gang at vi bør teste applikasjonen tidligere og beregne mer tid til utbedring etter testing, da utenforstående testere gjerne oppdager helt andre feil og mangler enn det vi har tenkt på. For oss fungerte applikasjonen som den skulle, men med en gang noen ikke trykket på knapper i riktig rekkefølge, eller trykket for mange ganger, oppsto det feil. Vi ble ferdig med applikasjonen i løpet av april, slik vi hadde planlagt. Det at testingen ble foretatt nokså sent er derfor ikke en følge av at vi ble forsinket i utviklingsprosessen, men at vi feilvurderte omfanget av hvilke feil og mangler som ville komme fram under testingen.

Vedlegg

Gjennom hele prosessen har vi dokumentert hva vi har testet, hva som ikke fungerte som det skulle og hvordan vi har løst det, eventuelt hva vi har gjort istedenfor. Hele utviklingsfasen har vært preget av testing av ulike metoder for å løse problemstillingen, og mange av metodene har blitt forkastet underveis. Vi har beholdt noen av metodene vi testet i begynnelsen, mens enkelte deler har vi først løst helt i innspurten. Vi har hele tiden hatt et stort fokus på å få til en best mulig applikasjon, og ettersom det å teste ulike metoder å løse oppgaven på har vært en veldig stor del av prosjektet, har vi valgt å legge ved den utfyllende beskrivelsen som vedlegg, vedlegg B - Notater fra utviklingsprosessen. Her er alle valg som er tatt underveis dokumentert. Vi føler at vi har hatt et veldig stort læringsutbytte av å systematisk jobbe oss gjennom ulike måter å løse oppgaven på for å få resultatet slik vi ønsker. Dette har bidratt til at vi har lært veldig mye med tanke på programmering.

7. Konklusjon

Prosjektets hovedmål var å gjøre utfylling av reiseregning enklere og raskere for brukere av HRessurs. De skal kun måtte angi start, stopp, eventuelle viapunkter og kjøretøy for å få avstand og bompengekostnader beregnet. På bakgrunn av tilbakemeldingene på vår frittstående applikasjon, konkluderer vi med at dette målet i all hovedsak er nådd. Når implementasjonen er ferdigstilt og publisert slik at den integrerte applikasjonen blir tilgjengelig for brukerne av HRessurs, er vi av den oppfatningen at utfyllingen av reiseregninger vil bli både enklere og raskere enn det er i dag.

Målet har vi oppnådd ved å foreta en grundig gjennomgang av tilsvarende tjenester og ved å sette oss nøyne inn i tilgjengelige teknologi. Vi har gjennomført et godt planleggingsarbeid slik at vi har fått en best mulig oversikt over hva slags teknologi og løsninger som kunne være aktuelle for vårt prosjekt. Det gikk med mye tid i starten på å tilegne oss kunnskap både i forhold til ruteberegnning og reiseregninger, samt å lære oss teknologiene vi har benyttet i prosjektet.

Vi har utviklet applikasjonen i små steg av gangen, etter standarden som er ønsket for programvare hos Infotjenester. Vi har utviklet applikasjonen i en lagdelt struktur slik at alle deler av applikasjonen er utskiftbare. I vår applikasjon kan man bytte ut deler av applikasjonen uten å måtte gjøre endringer i de resterende delene, forutsatt at den forhåndsdefinerte dataflyten er oppfylt. Man kan bytte ut Ruteplantjenesten med Ferd ruteberegnning, ved å kun endre klassen der beregningene mot Ruteplantjenesten foregår. Hvis applikasjonen skal vedlikeholdes eller leverandør av API skal byttes ut, vil det med tanke på lagdelingen være mye enklere å gjøre dette. Uten en lagdeling ville store deler av applikasjonen måtte endres for å muliggjøre et leverandørbytte. Vi har lagt stor vekt på tankegangen rundt lagdeling, da det har vært et felles fokus fra både oss og oppdragsgiver om at vi skal få et innblikk i hvordan applikasjonsutvikling i all hovedsak foregår i arbeidslivet. Vi har også hatt et ønske om å utvikle en applikasjon Infotjenester kan bruke i sitt eksisterende system.

Vi kan derfor konkludere med at vi har levert en brukervennlig applikasjon som med stor sannsynlighet vil forenkle hverdagen til veldig mange av HRessurs sine brukere når implementeringen er fullført. Vi er også stolte over at vi har utviklet en applikasjon som er på et nivå som er godt nok for å bli implementert i HRessurs. Vi har tilegnet oss mye kunnskap gjennom dette samarbeidet, og vi har lært veldig mye om ulike aspekter ved programmering som vi med stor sannsynlighet aldri ville kommet borti i vanlig skolesammenheng.

Bibliografi

- [1] Bomstasjoner langs E6
<http://www.e6bompenger.no/Bomstasjonene-2.aspx>
- [2] Bomstasjoner i norge:
<http://www.infotjenester.no/kompetanse/lonn/2014/10/bompengekalkulator-alle-norske-bomstasjoner/>
- [3] Artikkel om bompenger Halden:
http://www.ha-halden.no/_vi_kommer_ikke_utenom_bompenger-5-20-19908.html
- [4] Autopass:
<http://www.autopass.no/obligatoriskbrikke/om-obligatorisk-brikke>
- [5] Takster Bompenger:
http://www.vegvesen.no/_attachment/181865/binary/348996?fast_title=Takster+i+bompengeanlegg.pdf
- [6] Bompengeselskaper:
<http://www.autopass.no/Bompengeselskap>
- [7] Bomstasjoner Østfold:
<http://www.ostfold-bompengeselskap.no/service/#oversikt-bomstasjoner>
- [8] Takster Østfold:
<http://www.ostfold-bompengeselskap.no/takster/>
- [9] Svinesundsforbindelsen, autopass:
<http://www.svinesundsforbindelsen.no/autopass.html>
- [10] Rabatter:
<http://bompenger.no/Takster/Nyrabattstruktur.aspx>
- [11] Miljøpakken:
<http://miljopakken.no/om-miljoepakken/om-organisasjonen>
- [12] Miljøpakkens mål:
<http://miljopakken.no/om-miljoepakken/maal>
- [13] Miljøpakken bompunkter:
<http://miljopakken.no/om-miljoepakken/bompunkter>
- [14] Haugesunds regler:
<http://www.haugalandspakken.no/service/#timesregel>
- [15] Reiseregningssregler i bedrift
<https://www.altinn.no/no/Starte-og-drive-bedrift/Drive/Arbeidsfordold/Lonn/Reiseutgifter/Dekning-etter-regning-/>

- [16] Vis veg, ruteplanlegger:
<http://visveg.vegvesen.no/Visveg/>
- [17] NAF's ruteplanlegger:
<https://www.naf.no/tjenester/ruteplanlegger>
- [18] 1881's ruteplanlegger:
<http://www.1881.no/Kart/Veibeskrivelse/>
- [19] Google Maps:
<https://www.google.no/maps/dir/>
- [20] Gulesiders ruteplanlegger:
<http://kart.gulesider.no/veibeskrivelse>
- [21] Kvasir:
<http://kart.kvasir.no/>
- [22] Universiell utforming - regler:
<http://uu.difi.no/regelverk/tidsfristar-ny-og-eksisterande-ikt>
- [23] Universiell utforming - regler:
<http://uu.difi.no/regelverk/kven-skal-folgje-krava>
- [24] Universiell utforming - krav til nettsider:
<http://uu.difi.no/veiledning/nettsider/krav-til-nettsider/oppbygging-av-wcag-20>
- [25] Nettsted for utarbeiding av spørreundersøkelser
<http://www.surveymonkey.com>
- [26] docs.google.form - vår spørreundersøkelse
<https://docs.google.com/forms/d/1chpr-aATjuDkkL1mjzicnJ6U4XouJZ193H5ikZ-FcWo/viewform>
- [27] Hva er UTM?
http://en.wikipedia.org/wiki/Universal_Transverse_Mercator_coordinate_system
- [28] Esri, Geotilbyder
<http://www.esri.com/>
- [29] Koordinatforskjeller
<http://www.differencebetween.net/miscellaneous/geography-miscellaneous/difference-between-wgs84-and-etrs89/>
- [30] Vegvesenet, dokumentasjon Ruteplantjenesten
https://www.vegvesen.no/ruteplan/routingservice_v1_0/routingservice/
- [31] Nvdb Vegreferanse-Api
<https://www.vegvesen.no/nvdb/api/dokumentasjon/vegreferanse>

- [32] c#-konvertering:
<http://stackoverflow.com/questions/2689836/converting-utm-wsg84-coordinates-to-latitude-and-longitude>
- [33] Java script konvertering:
<http://www.uwgb.edu/dutchs/usefuldata/ConvertUTMNoOZ.HTM>
- [34] Dekomprimering:
http://resources.esri.com/help/9.3/arcgisengine/ArcObjects/esriinetworkanalyst/INACompactStreetDirection_CompressedGeometry.htm
- [35] Reactive
<http://www.ractivejs.org/>
- [36] LazyFrameWork er et åpent FrameWork tilgjengelig på GitHub:
<https://github.com/petterek/lzy>
- [37] rute og trafikktjenesten dit.no
<http://dit.no>
- [38] Vegvesenets ruteplantjeneste Visveg
<http://visveg.no>
- [39] Statstikk, bilgodtgjørelse og kilometergodtgjørelse
<http://www.skatteetaten.no/no/Tabeller-og-satser/Bilgodtgjorelse-kilometergodtgjorelse/>
- [40] Finn.no Kart tjeneste
<http://kart.finn.no/>
- [41] Triona:
<http://www.triona.no/>
- [42] Norkart:
<http://www.norkart.no/>
- [43] Ferd:
<https://www.norkart.no/produkt/ruteberegnung-ferd-viser-veg/>
- [44] Arbeidsdager:
<http://www.skatteetaten.no/no/Person/Selvangivelse/Finn-post/3/2/8/>

