BACHELOROPPGAVE:

Ruteplanlegger og Flåtestyring

i

CarAdmin

FORFATTERE:

Else Dalby

Marte Selsjord Bjørseth

Trine Anita Grønvold

07HBINDA

Dato: 19.5.2010

Sammendrag av Bacheloroppgaven

Tittel:	Ruteplanlegger og Flåtestyring i CarAdmin	Nr. :
		Dato : 19.5.2010
Deltakere:	Else Dalby	
	Marte Selsjord Bjørseth	
	Trine Anita Grønvold	
Veileder:	Høgskolelektor Tom Røise	
Oppdragsgiver:	Electric Time Car AS (ETC)	
Kontaktpersoner:	Dag L. Solhaug & Øyvind Flatval	
Stikkord:	CarAdmin, Ruteplanlegging, Flåtestyring, Google Maps API v2, Java	

Antall sider: 168	Antall bilag: 13	Tilgjengelighet: Åpen

Kort beskrivelse av bacheloroppgaven:

Vi har har utviklet to moduler til Electric Time Car AS sitt frontprodukt CarAdmin. CarAdmin er en nettapplikasjon som håndterer administrasjonen av fellesbilene i en bilpark, i forhold til kjøretøyoversikt, nøkkelstyring, service-oversikt, kjørebok og reservering av biler. Modulene vi har utviklet til denne applikasjonen er Ruteplanlegger og Flåtestyring, som begge benytter Google Maps API v2.

Ruteplanleggermodulen har funksjonalitet for å opprette, endre og slette ruter for bilene. Rutene vil kunne vises i et kart med tilhørende veibeskrivelse, og stoppestedsinformasjon. Det er også mulighet for å vise flere ruter i samme kart, for å sammenligne rutene for samkjøring.

Flåtestyringsmodulen inneholder et sanntidskart med markører for sanntidsplasseringen til bilene. Her er det også mulighet for å søke opp en adresse og finne nærmeste bil, og få opp veibeskrivelse og kjøretid fra bilens nåværende posisjon og til adressen. Modulen inneholder også et historiekart. Denne viser hvor en bestemt bil har vært i løpet av de siste 14 dager, eller mindre. Flåtestyringsmodulen får sine data fra en GPS med GSM-sender, som er installert i alle bedriftens biler.

Hovedutfordringene med denne oppgaven har vært å jobbe i den eksisterende koden til ETC, og håndteringen av Google Maps API v2. I tillegg kom arbeidet med å lære seg Java og JavaScript.

Forord

Ruteplanlegger- og Flåtestyringsmodulen til CarAdmin har blitt utviklet av tre engasjerte jenter ved Bachelor i ingeniørfag - data ved Høgskolen i Gjøvik. Oppgaven har vært utrolig spennende og lærerik, og har lært oss mye i forhold til prosessen med å sette oss inn i den eksisterende koden til CarAdmin og i dokumentasjonen til Google Maps.

Java og JavaScript er ikke lenger like foruroligende som de virket i begynnelsen, og den eksisterende koden til CarAdmin har etter hvert blitt forståelig. Utviklingen av små detaljer tar ikke lenger flere dager, men kan kortes ned til noen timer. Alt i alt har det vært en positiv læringsopplevelse!

Veileder for oppgaven har vært Tom Røise. Vi takker for et godt samarbeid og gode tilbakemeldinger gjennom hele semesteret.

Vi vil også gjerne gi en stor takk og en god klem til Øyvind Flatval ved ETC. Tusen takk for at du har holdt ut med oss, og de mange spørsmålene du har måtte tålt gjennom hele prosjektperioden. Det har vært veldig godt å ha noen å spørre, når vi ikke skjønner hvorfor ting ikke fungerer.

Takk til Dag L. Solhaug, daglig leder i ETC, for konstruktiv kritikk underveis, særlig i forhold til brukergrensesnitt.

Gjøvik, 20.5.2010

_____	_____	_____
Else Dalby	Marte Selsjord Bjørseth	Trine Anita Grønvold

Innhold

Figurliste

1 INNLEDNING

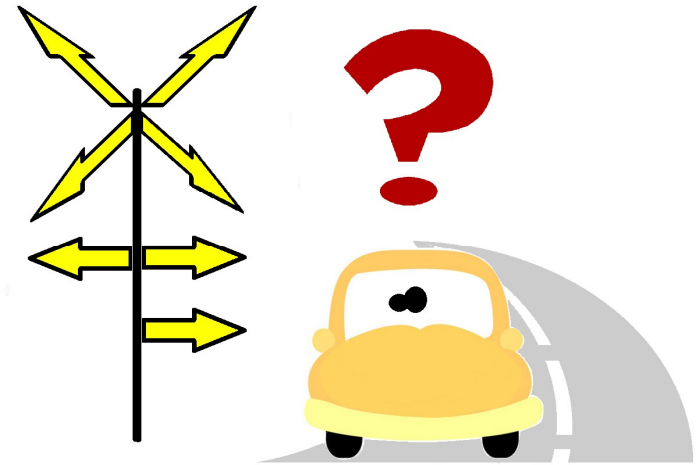
1.1 Introduksjon

1.1.1 Problemområde

I bedrifter og kommuner som opererer med bilparker med fellesbiler, er det ikke greit å vite hvilke biler som er disponible til hvilke tider, hvilke ansatte som bruker de eller hva de blir brukt til, uten et administrativt system. Et slikt system finner vi i Electric Time Car AS (ETC) sin CarAdmin-*applikasjon*, som de stadig utvikler og forbedrer. De har til nå ingen funksjonalitet i CarAdmin for planlegging av kjørerutene til bilene, og det var dette de ga oss som oppgave å lage. I tillegg til dette ønsket de seg også funksjonalitet i forhold til å få en oversikt over hvor bilene til enhver tid befinner seg.

Det er flere yrkesgrupper som benytter seg hyppig av bilparker, som f.eks. hjemmehjelp og leveringsbud.

I slike arbeidssituasjoner er det ikke nødvendigvis lett for arbeidsgiver å vite hvor effektivt ressursene blir utnyttet. Det er ikke gitt at brukerne av bilene er kjent i området de skal kjøre, og det kan gå med mye tid på unødvendige omveier og dårlig planlegging i forhold til den mest ressurssparende *ruten* de kan følge.



Figur 1 Det er ikke alltid gitt at de ansatte er kjent i området, og det kan gå mye tid på dårlig planlegging

Hvis vi tar for oss arbeidsdagen til en ansatt i for eksempel hjemmetjenesten, vil vedkommende kun ha en oversikt over hvem de skal besøke, og i hvilket tidsrom de bør være der. Dette fører til at de i mer eller mindre grad har ansvaret for å planlegge rutene selv. Kanskje kunne to ansatte med mer eller mindre samme rute ha delt bil? Kanskje det er flere folk på jobb enn strengt talt nødvendig?

I forhold til sikkerhet er det også hensiktsmessig for arbeidsgiver å se hvor bilene til enhver tid befinner seg. Kanskje befinner bilen seg i en helt annen kommune enn det den skulle? Hvilken er den nærmeste bilen som kan sjekke opp hva som skjer?

1.1.2 Oppgavedefinisjon og avgrensning

Vi har i forbindelse med disse problemstillingene fått som oppgave å lage en *Ruteplanlegger*- og en *Flåtestyring*smodul til CarAdmin, som ved hjelp av *kartverk* fra Google, skal kunne hjelpe til med planleggingen av kjørerutene og sanntidsplasseringen til bilene i bilparken. Dette blir gjort med tanke på å få en effektiv utnyttelse av bilparken og andre ressurser i bedriften, samt å gi en god oversikt over bruken av bilene til arbeidsgiver.

Ruteplanleggermodulen fungerer som et administrativt støttesystem som tar for seg problemer i forhold til planlegging av ruten, hvilke veier som blir kjørt, og hvor lang tid det tar å fullføre ruten. En overordnet bruker, Administrator, planlegger og plotter inn kjøreoppdragene til brukerne av bilene på forhånd, og legger de inn på bestemte datoer. På hver av rutene vil all nødvendig informasjon om de enkelte *stoppestedene* bli lagt inn, som blant annet arbeidsoppgavene som skal bli gjort og til hvilke tider Bruker skal være på stedet. Selve *modulen* er en kartbasert tjeneste som bruker funksjonalitetene i *Google Maps*, slik at Administrator automatisk får opp den mest effektive ruten til disse stoppestedene kalkulert hvor lang tid ruten vil ta. Bruker kan bare skrive ut kartet med sin respektive rute ut fra kalenderen, og komme i gang med arbeidsdagen.

Administrator har også muligheten til å sammenligne ruter med hverandre, slik at han på en enkel måte ser om det er mulig å samkjøre der det forekommer liknende ruter, eller gjøre eventuelle endringer slik at ruten blir mer effektiv. Dette blir gjort ved hjelp av Google Maps, hvor de valgte rutene blir vist i samme kart slik at Administrator kan velge hvilke ruter han vil sette sammen.

I tillegg til Ruteplanleggermodulen har vi også utviklet en Flåtestyringsmodul, som vil gi arbeidsgiver en generell oversikt over hvor bilene er til enhver tid.

Bilene i bilparken vil bli utstyrt med *GPS med GSM-sender* fra ETC, som hvert minutt sender bilens koordinater til ETC sin server. Ut fra disse koordinatene får Administrator opp et *sanntidskart*, som gir informasjon om hver bil og et kartutsnitt over hvor alle bilene befinner seg.

Det er også mulig å hente ut bilhistorikken til en bestemt bil, noe som gjør det enkelt for arbeidsgiver å spore den faktiske bruken av bilene. Administrator kan her velge rutehistorikken til en bestemt bil inntil 14 dager bakover i tid. Hvis det er noen ruter som har blitt kjørt i denne perioden, vil Administrator få opp et kartutsnitt over rutene som har blitt kjørt i det valgte tidsrommet. De kjørte rutene vil vises med hver sin farge i kartet.

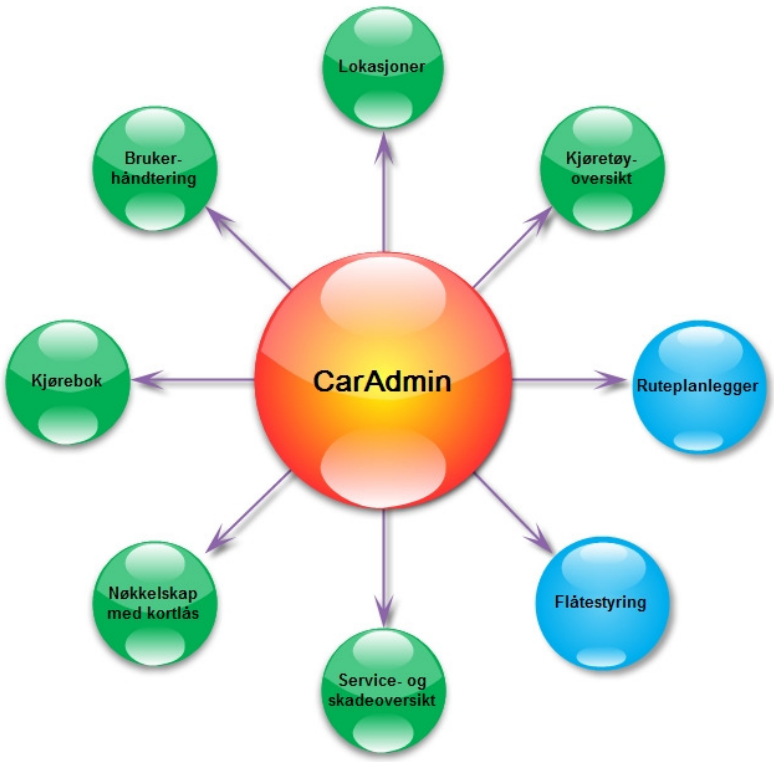
Modulene er integrert i CarAdmin-applikasjonen. Som en forutsetning til denne oppgaven krevde det at vi satte oss inn den eksisterende CarAdmin-koden, ble kjent med filstrukturen i applikasjonen og finne igjen klasser og funksjoner som vi kunne gjenbruke. Dette ble gjort for ikke å få unødig kode som allerede eksisterer i CarAdmin fra før, og for å integrere koden bedre med det gamle, slik at det blir lettere for ETC å endre og videreutvikle koden i fremtiden. Slik ble det også lettere å opprettholde designet til den eksisterende CarAdmin-løsningen. Funksjoner som vi har måtte gjenbruke var blant annet kalenderfunksjonen, tabellfunksjonen og knappene til CarAdmin. Dette viste seg å være en stor del av oppgaven.

Vi fikk som ramme av ETC at vi skulle bruke Google Maps som kartverk, da de mente denne kunne dekke funksjonaliteten de ønsket. Google er en stor aktør, og såpass utbredt, at det ga oss mulighet til å finne god dokumentasjon og eksempler på hvordan vi kunne bruke Google Maps. Vi har dermed

også måtte sette oss inn i funksjonalitetene til Google Maps, og studere hvordan vi skulle få kartverket og koden i modulen til å fungere sammen.

1.1.3 CarAdmin

CarAdmin er et administrativt system, som gir en oversikt over fellesbiler i en bilpark. Applikasjonen holder blant annet detaljert informasjon om de enkelte bilene, slik som kilometerstanden til bilen, om den har sommer- eller vinterdekk, eventuelle feil på bilen, hvilken ansatt som kjørte den sist og i hvilket tidsrom den ble brukt. Den viser også den nåværende tilstanden til de enkelte bilene - om den er på service, i bruk eller ledig. Om bilen er ledig kan Bruker gå inn i kalenderen og reservere bilen på en strukturert måte. Systemet fører også en logg over skadehistorikken og servicehistorikken til bilene, samt aktivitetene til brukerne.



Figur 2 CarAdmin-applikasjonen og dets moduler

Ruteplanlegger- og Flåtestyringsmodulene kommer inn som nye funksjonaliteter til CarAdmin-applikasjonen, og vi har fått tilbakemeldinger fra ETC om at med noen små justeringer vil modulene bli benyttet i CarAdmin når oppgaven er avsluttet.

1.2 Målgrupper

Rapporten og modulene har forskjellige målgrupper.

Rapporten er hovedsakelig rettet mot oppdragsgiver og sensor, men kan også bli benyttet av andre (fortrinnsvis studenter) som har interesse for oppgaven. Vi har dermed gått mer teknisk inn i rapport i forhold til fagbakgrunnen vår som dataingeniører. ETC vil kunne bruke rapporten som dokumentasjon for å videreutvikle modulene, mens den for sensor vil danne grunnlaget for bedømme bacheloroppgaven vår.

Selve modulene er laget for ETC og deres kundegruppe. Vi har dermed tatt hensyn til brukerhåndtering i modulene, slik at de er intuitve å bruke, selv for mindre datakyndige brukere.

1.3 Mål

1.3.1 Resultatmål

Målet med oppgaven er å utvikle to ferdige tilleggsmoduler til ETC sin eksisterende applikasjon CarAdmin. Modulene skal ta for seg Ruteplanlegging og Flåtestyring, og skal fungere som en del av CarAdmin-applikasjonen.

1.3.2 Effektmål

Med produktet er det ønsket å oppnå ressursbesparelser hos brukerbedriftene, hovedsakelig i form av tid for den som skal sette opp rutene, og antall personer som bedriften egentlig trenger på jobb. Vinningen vil variere ut fra brukerbedriftens størrelse og arbeidsform. Modulene skal samkjøre med den eksisterende applikasjonen CarAdmin for å styrke oppdragsgivers posisjon i markedet, ved å gi større konkurransedyktighet. ETC sitt mål med produktet er å kunne gi kundene et bredere tilbud innen håndtering av bilparken, og gi større mulighet for planlegging av bruk, samt muligheten til å se hvordan den faktiske kjøringen ble.

1.4 Gruppens faglig bakgrunn og kompetanse

Samtlige av oss i prosjektgruppen studerer ved dataingeniørlinjen ved Høgskolen i Gjøvik, slik at vi alle har god kompetanse innen blant annet programmeringsspråket C++, databaser og systemutviklingsmetoder. Vi har alle hatt fordypning i Objektorientert Systemutvikling, og det var i dette emnet vi ble først introdusert for ETC og CarAdmin-applikasjonen. Trine har i tillegg til dette tatt fordypning innen programutvikling, og var dermed den eneste i gruppen med noe videre kompetanse i programmeringsspråket Java i forkant av bacheloroppgaven. Med tanke på at to trdjedeler av prosjektgruppen ikke var kjent med utviklingsmiljøet og koden, fikk vi en noe treg start på prosjektet, hvor det var mye å ta inn over oss før vi kunne komme i gang med modulene.

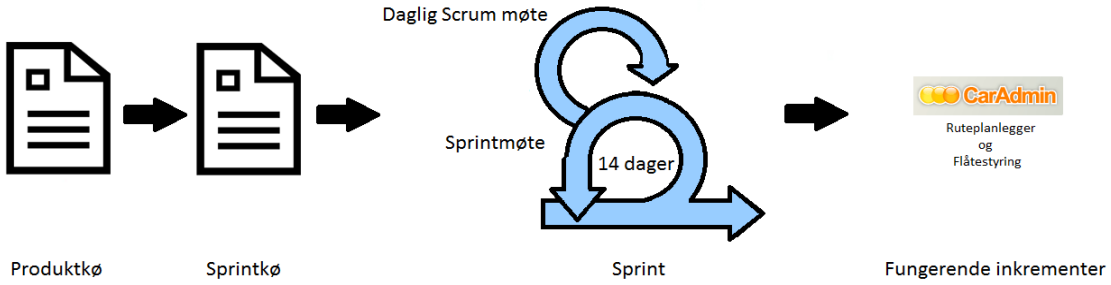
Ingen av oss hadde noe kjennskap til JavaScript, selv om vi har hatt HTML ved tidligere kurs. Dermed har vi måtte lære oss hvordan Java og JavaScript ble brukt om hverandre. Vi har kun hatt mindre prosjekter i programmerings- og systemutviklingsfagene tidligere, slik at dette ble et solid sprang i forhold til omfanget på oppgaven og arbeidsprosessen.

Vi måtte også sette oss inn i den eksisterende koden til CarAdmin. Her måtte vi lære oss hvordan filene var koblet sammen, hva som var i filene, forstå hva som egentlig ble gjort, og ikke minst hva slags funksjoner vi skulle bruke videre i prosjektet. Vi måtte også sette oss inn i Google Maps sin kode. Dette hadde ingen av oss noe kunnskap om, slik at vi måtte lære dette selv ut fra forumer og gode nettsteder.

1.5 Arbeidsprosess

Vi har gjennom de største delene av prosjektperioden jobbet sammen til fastsatte tider. Dette har i hovedsak vært hverdager fra kl 0800, bortsett fra mandager da vi hadde full timeplan med andre fag, se figur 32 og 33. Å jobbe sammen slik, har vært en stor fordel. Det førte ikke bare til at det ble lettere for oss å starte opp arbeidet på morgenen, men vi fikk også diskutert problemer ettersom de oppstod, og hjulpet hverandre med ting vi stod fast på.

Vi valgte å jobbe etter systemutviklingsmodellen *Scrum*, etter begrunnelsene gitt i Forprosjektrapporten, se vedlegg E.



Figur 3 Scrumprosessen

I henhold til scrumprosessen hadde vi daglige Scrum møter, slik at vi startet arbeidsdagen med et kort møte hvor alle i gruppen fikk en oversikt over hva de enkelte gruppemedlemmene hadde gjort, hva vi måtte gjøre videre i oppgaven, hvem som skulle gjøre de forskjellige delene og hvordan vi skulle gjøre dette.

Sprintene våre gikk over to uker, slik at vi hadde sprintmøter med oppdragsgiver annen hver uke, for å vise frem hva vi hadde gjort. Her fikk vi tilbakemelding på det vi hadde gjort med modulene, hva de ville ha gjort annerledes og eventuell ekstra funksjonalitet de kunne ha tenkt seg at modulene skulle ha hatt. Hvis ETC ønsket seg ny funksjonalitet i modulene, undersøkte vi mulighetene for dette og implementerte endringene hvis dette var mulig. Hvis det ikke var mulig, lagde vi et Teknisk Memo som omhandlet ønsket. Vi førte også en *sprintkø* for hver av sprintene, slik at vi hele tiden hadde en oversikt over hva som var gjort, og hva som stod igjen i *produktkøen*, se avsnitt 2.2.

Vi hadde ingen ytterligere rollefordeling i forhold til Scrum. I stedet tok hvert av gruppemedlemmene ansvaret på rundgang i forhold til de daglige Scrummøtene og fremgangen i prosjektet.

Oppdragsgiver var også involvert i prosjektet utover sprintmøtene. Vi hadde blant annet ordinære ukentlige møter, hvor vi fikk tatt opp eventuelle problemer som hadde oppstått i prosessen og stilt kritiske spørsmål i forhold modulene vi skulle lage eller miljøet rundt disse. Vi hadde også kontakt

over e-post og Live Messenger, slik at vi fikk raske tilbakemeldinger hvis det var noe som var uklart i oppgaven eller i koden til CarAdmin.

Vi hadde ukentlige møter med veileder, hvor vi jevnlig ble påminnet om den tunge dokumentasjonen som også ble krevet av oss. Slik fikk vi dermed en balanse i oppgaven, i forhold til at dette ikke bare var et program som skulle bli levert til oppdragsgiver, men en bacheloroppgave som krevde at vi måtte tenke gjennom hele prosessen og måtte argumentere for beslutningene vi tok.

I forhold til dokumentasjonen har vi sett på tidligere bacheloroppgaver, og da gjerne gamle rapporter som gikk på CarAdmin-applikasjonen, samt gode prosjektrapporter som ga oss en retning i forhold til rapportskrivningen. I forhold til testing av modulene har vi brukt familie, som hadde et forhold til slike systemer, venner som ikke hadde fullt så stor kjennskap til slike systemer, og representative folk som arbeider i hjemmetjenesten. Dette ble gjort med tanke på brukerhåndtering og funksjonalitet. Mer om dette under hovedkapittel 5.

Vi har ført logg daglig – en felleslogg, se vedlegg H, som vi la ut på hjemmesiden vår, og en logg som vi førte hver for oss. Vi har også hatt en prosjektdagbok der alle de større beslutningene har blitt skrevet opp. Vi lagde møtereferater fra både veileder og oppdragsgiver etter hvert som de ble gjennomført. Andre arbeidsmetoder vi har brukt er tavlen på grupperommet og nyttige skriv som vi har delt på fellesområdet på serveren.

1.6 Øvrige roller

Oppdragsgiver: Daglig leder i Electric Time Car AS, Dag L. Solhaug
Veileder: Tom Røise, Høgskolelektor ved Høgskolen i Gjøvik

1.6.1 Oppdragsgiver

Oppdragsgiveren for oppgaven er ETC, som er et IT-selskap som er lokalisert i Gjøvik. Bedriften baserer seg på å lage "nyskapende løsninger for administrasjon av fellesbiler, som benyttes av flere sjåfører i både små og store bilparker".[1] CarAdmin er ETC sitt frontprodukt og hovedapplikasjon, hvor man enkelt kan samle fellesbilene under daglig oppfølging av en person. Under denne applikasjonen finner man flere moduler, hvis kombinasjon gjør at deres produkt blir spesielt, og best mulig brukertilpasset i forhold til deres kundegruppe. Ruteplanlegger- og Flåtestyringsmodulen er nettopp slike utvidelser, som ETC ønsker skal gi de et forsprang i forhold til liknende produkter på markedet.

Øyvind Flatval er kontaktpersonen vår i ETC og den vi har forholdt oss mest til under arbeidsprosessen, blant annet på våre ukentlige møter. Han har veiledet oss gjennom hele perioden, hvor han for eksempel i begynnelsen av prosjektet viste oss hvordan CarAdmin-applikasjonen var bygd opp, og gjennomgikk utviklingsverktøyene vi skulle bruke. Han har også vært tilgjengelig til å svare på tekniske spørsmål om programmeringen, og ga oss gjerne innspill på hva vi kunne gjøre annerledes i forhold til brukergrensesnitt og funksjonalitet i modulene, gjennom hele prosjektet.

Dag L. Solhaug er daglig leder i ETC og mannen bak selve oppgaven, og var aktiv i hvordan modulene formet seg i løpet av utviklingen. Han var til stede på flere møter, og hadde en god innsikt i hvordan programmet skulle være oppbygd med hensyn på brukerhåndtering. Han hadde også mange innspill i

[1] http://www.electrictimecar.com/modules/content/index.php?id=12

forhold til nyttige funksjoner som vi kunne ha med i modulene, slik at vi i løpet av utviklingsperioden har utvidet modulene med noen fiffige funksjonaliteter.

1.6.2 Veileder

Veileder i forbindelse med oppgaven er Høgskolelektor Tom Røise. Vi hadde i forrige semesteret den originale Ruteplanleggeroppgaven i forbindelse med kurset Objektorientert Systemutvikling, hvor Tom er emneansvarlig. Siden bacheloroppgaven hadde en nær problemstilling i forhold til denne oppgaven, var han ganske kjent med problemstillingen. Slik kunne han stille oss kritiske spørsmål med tanke på hva vi hadde hatt problemer med tidligere og hva vi hadde gjort med dette.

1.7 Organisering av rapporten

1.7.1 Oppsett av rapporten

Rapporten er delt inn i 8 kapitler, med tilhørende underkapitler. Underkapitlene inneholder en videre inndeling der det er behov for dette. Disse tre nivåene er nummerert, hvor de to øverste nivåene finnes i innholdsfortegnelsen. De innledende sidene benytter romertall for sidetall, for skille disse fra selve rapporten.

Vi bruker en standardfont gjennom hele prosjektet. Ved ord som forekommer i ordforklaringen i vedlegg A, bruker vi kursiv skrift når ordet først forekommer i rapporten.

1.7.2 Kapitteloppsummering

Kapittel 1: Innledning
Her får man en oversikt over grunnlaget til prosjektet. Man får vite hva selve oppgaven går ut på, litt om prosessen med å få til produktet, samt formålet med prosjektet.

Kapittel 2: Kravspesifikasjon
Her settes kravene til modulene. Disse vises i form av *Use Case* diagrammer og -tabeller.

Kapittel 3: Design
Kapitlet tar for seg brukergrensesnittet, lagdelingsstrukturen og filstrukturen til modulene.

Kapittel 4: Implementasjon
Hvordan vi implementerte koden, hva vi brukte av eksisterende kode i CarAdmin og hva vi måtte lage selv, og håndtering av Google Maps sitt API.

Kapittel 5: Testing og kvalitetssikring
Kapitlet omhandler kvalitetssikringen for å gjøre programmet brukervennlig og robust, samt hvordan vi har testet og feilsøkt i modulene.

Kapittel 6: Avslutning
Evalueringen av arbeidet blir omtalt her, sammen med muligheten for videre arbeid på modulene og videre implementering mot CarAdmin, samt oppsummering av prosjektet.

Kapittel 7: Litteraturliste
Inneholder referansene vi har markert gjennom rapporten, hovedsakelig nettsider.

Kapittel 8: Vedlegg
Her ligger alle vedleggene til rapporten.
Vedleggene vi har valgt er henholdsvis terminologiliste, prosjektplaner, statusrapporter, designskjemaer, kode, skjermbilder, forprosjekt, logg, prosjektdagbok, eksempler på møtereferater, kontrakten med oppdragsgiver og innholdet på CD-en.

2 KRAVSPESIFIKASJON

2.1 Brukerbeskrivelser

2.1.1 Omgivelser

For å få tilgang til Ruteplanlegger- og Flåtestyringsmodulen, som skal være en del av CarAdmin-applikasjonen, må brukeren identifisere seg med gyldig brukernavn og passord. CarAdmin er et nettbasert produkt, slik at brukeren må være tilkoblet Internett for å få tilgang til applikasjonen. Modulene skal bruke Google Maps til å generere og vise kart i forhold til rutedataene som er gitt. All informasjon fra modulene skal ligge lagret på CarAdmin sin database, slik at kunden ikke trenger å installere noe programvare for å bruke applikasjonen annet enn en oppdatert nettleser.

2.1.2 Systemets brukere

Vi skal forholde oss til CarAdmin sine brukere av systemet, som er Administrator og Bruker.

Administrator skal ha det overordnede ansvaret for å planlegge rutene. Dette innebærer å plotte inn hvert stoppested til rutene, og føre relevant informasjon for hvert av disse stedene. Planleggingen av rutene blir gjerne gjort med hensyn til samkjøring, slik at man kan spare på ressursene der rutene i større grad er samstemte. Når punktene har blitt plottet inn, skal applikasjonen generere en relevant rute.

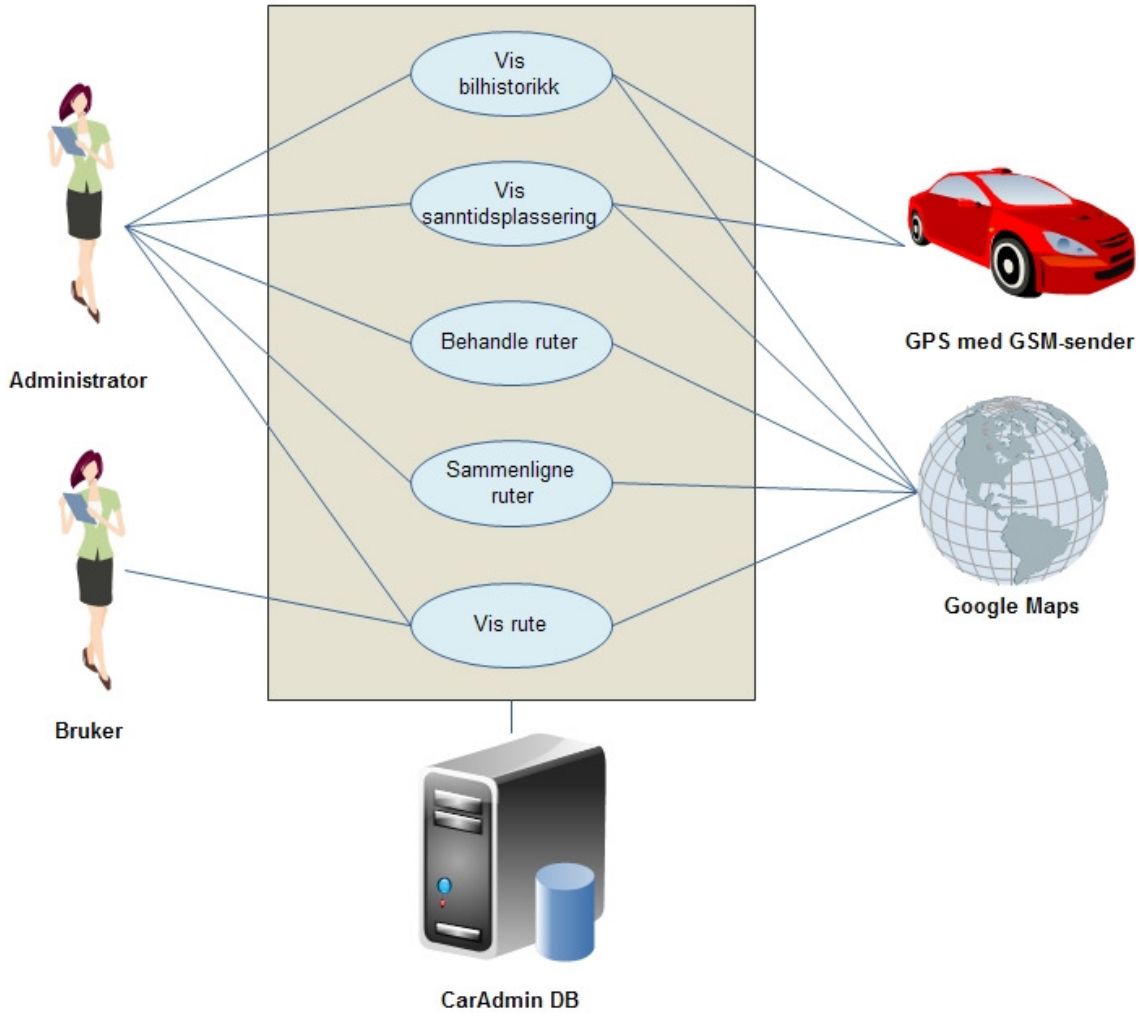
På grunn av personvern, se avsnitt 2.5.7, er det kun Administrator som skal ha tilgang til Flåtestyringsmodulen. I denne modulen skal de få oversikt over hvor hver av de enkelte bilene til bedriften befinner seg, samt se bilhistorikken til de forskjellige bilene. Dette blir brukt som et hjelpemiddel i forhold til fremtidig planlegging av ruter.

Brukerne skal kunne få opp en oversikt over alle rutene, samt kunne skrive ut kartet til ruten de skal kjøre. Dette vil da være de ansatte i bedriften, som kun skal forholde seg til de ferdigplanlagte rutene og arbeidsoppgavene som har blitt satt opp på ruten.

2.2 Funksjonelle krav

Siden Scrum ikke har noen definert måte å lage kravspesifikasjonen på, har vi besluttet å bruke use case til å formidle kravene best mulig. Dette vil da innebære overordnede og detaljerte use case, samt sekvensdiagram for de mer kompliserte modulene.

2.2.1 Use case diagram



Figur 4 Use Case diagram

CarAdmin DB
Vi skal bruke den eksisterende databasen til CarAdmin, til innhenting og uthenting av data. Databasemodellen og lignende ligger i avsnitt 3.3.

Bil med GSM-sender
Fellesbilene i bilparken inneholder en GPS med en GSM-sender, og vil sende utvalgt informasjon, slik som koordinatene og farten til bilene, til databasen. Vi har kun fått som oppgave å lage et grensesnitt til dette, og skal dermed ikke jobbe med det reelle. Informasjonen som blir uthentet her, skal bli brukt i ”Vis Bilhistorikk” og ”Vis sanntidsplassering”.

Google Maps
Karttjenesten som skal brukes er Google Maps, da denne karttjenesten ble ønsket av oppdragsgiver. Mer om Google Maps kommer i kapittel 4.4.

2.2.2 Detaljert use case

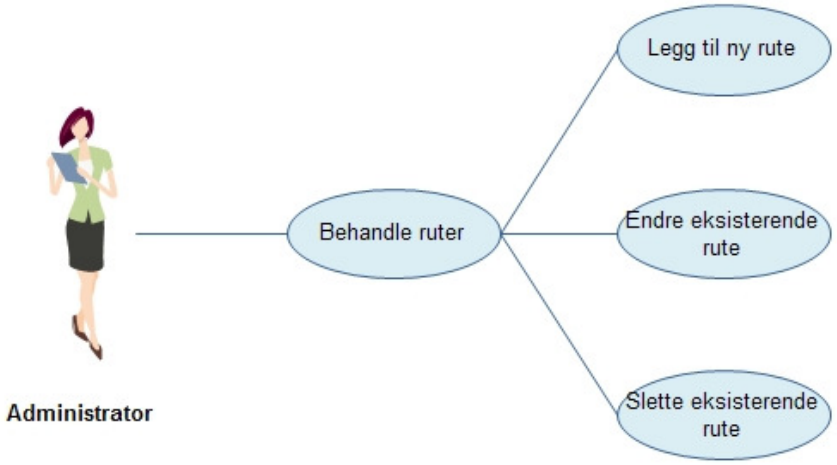
Sammenligne ruter

UseCase	**Sammenligne ruter**
Aktør	Administrator
Mål	Sammenligne ruter i et felles kart
Sammendrag	Administrator angir hvilken type ruter som skal sammenlignes. Her skilles det mellom frie ruter og ruter som blir valgt ut fra et bestemt datointervall. Han får så opp alle rutene som ligger under valgt type, og kan igjen velge hvilke av disse rutene han vil sammenligne i samme kart.
Prekrav	Det må være lagt inn minst en rute i databasen.
Postkrav	
Spesielle krav	

Hendelsesforløp	
Aktør	System
1. Administrator angir hvilke ruter han vil sammenligne ved hjelp av sjekkebokser.	
	2. Systemet viser rutene i henhold til sjekkboksene som Administrator har krysset av, og oppdaterer kart.

Behandle Ruter

For å få bedre oversikt har vi delt opp ”Behandle rute” inn i de tre hendelsesforløpene ”Legg til rute”, ”Endre rute” og ”Slett rute”. Som man ser ut fra diagrammet nedenfor, er dette fortsatt en funksjon, sett fra Administrators side.



UseCase	**Behandle ruter**
Aktør	Administrator
Mål	Legg til/slette/endre rute
Sammendrag	Administrator får opp en oversikt over eksisterende ruter (se ’vis rute’). Her kan han velge om han vil slette en rute, endre en rute eller legge til en ny rute.
Prekrav	For å kunne endre eller slette en rute, må det ligge ruter i CarAdmin sin database på typen rute som Administrator har valgt. Hvis ingen ruter er lagt til i databasen av denne typen, vil det kun vises valget om å legge til en ny rute, samt en beskjed om at ingen ruter finnes for dette valget.
Postkrav	
Spesielle krav	

Hendelsesforløp – *Legg til rute*	
Aktør	System
1. Administrator velger aktivitet: *Legg til rute*	
2. Administrator angir hvilken type rute han skal legge til (*fri rute*, rute ved en bestemt dato), *kjøretøygruppe*n ruten skal gjelde, samt lage et navn på ruten. Når han er ferdig, vil Administrator kunne velge å legge til et nytt stoppested.	3. Hvis Administrator ikke fyller ut alle feltene, vil han få en feilmelding og de manglende feltene vil bli markert med farge. Han vil ikke kunne lagre informasjon før alle felter er fylt ut.

	4. Generer og viser interaktivt kart, hvor Administrator får mulighet til å hente adresse ved hjelp av å sette en markør i kartet.
5. Administrator angir adresse og oppmøtetidspunkt til første stoppested/startstedet til ruten.	
	6. Laster opp oppdatert kart, samt generer tabell over stoppestedene til ruten.
7. Administrator får nå mulighet til å endre, eventuelt slette, det oppgitte stoppestedet. I så fall vil Administrator bli tatt med til punkt 4 igjen. Administrator kan også legge til et nytt stoppested.	
	8. Generer og viser interaktivt kart, hvor Administrator får mulighet til å hente adresse ved hjelp av markør i kartet.
9. Administrator må her angi stoppestedsinformasjon som oppmøtetidspunkt, avreisetidspunkt, oppdragsbeskrivelse, navn på besøkssted og evt. bedriftsnavn eller personnavn.	
	10. Laster opp oppdatert kart, med genererte ruter mellom punktene. Oppdaterer også tabell over stoppestedene til ruten.
11. Hvis Administrator vil legge inn et nytt stoppested, gjentas alle punktene fra 6 til 9. Hvis Administrator vil lagre ruten, trykker han på lagre-knappen, og sekvensen blir avsluttet.	
	12. Ruten legges inn i CarAdmin-databasen.
	13. Laster opp oppdatert tabell, og mulige aktiviteter vises på nytt.
Alternativt hendelsesforløp – Slette rute	
Alternativt hendelsesforløp – Endre rute	

Hendelsesforløp – *Slett rute*	
Aktør	System
	1. Genererer tabell over eksisterende ruter, over valgt type av ruter.
2. Administrator velger aktivitet: *slett rute*	
	3. Ruten slettes fra CarAdmin-databasen.
	4. Laster opp oppdatert tabell.
Alternativt hendelsesforløp – Legg til rute	
Alternativt hendelsesforløp – Endre rute	

Hendelsesforløp – *Endre rute*	
Aktør	System
	1. Viser tabell over eksisterende ruter, over valgt type ruter.
2. Administrator velger aktivitet: *endre rute*	
	3. Laster opp kart, med rutens stoppepunkter, samt genererer tabell over stoppestedene til ruten.
4. Administrator får enten legge til et nytt, endre et eksisterende, eller slette et stoppested.	
Alternativt hendelsesforløp – Legg til nytt stoppested	
1. Administrator velger aktivitet: Legg til nytt stoppested.	
	2. Generer og viser interaktivt kart, hvor Administrator får mulighet til å hente adresse ved hjelp av markør i kartet.
3. Administrator må her angi stoppestedsinformasjon som oppmøtetidspunkt, avreisetidspunkt, oppdragsbeskrivelse, navn på besøkssted og bedriftsnavn eller personnavn.	
	4. Laster opp oppdatert kart, med genererte ruter mellom punktene. Oppdaterer også tabell over

	stoppestedene til ruten.
Alternativt hendelsesforløp – Slett stoppested	
1. Administrator velger aktivitet: Slett stoppested.	
	2. Stoppestedet slettes fra CarAdmin-databasen.
	3. Laster opp oppdatert tabell, over stoppestedene til ruten.
Alternativt hendelsesforløp – Endre stoppested	
1. Administrator velger aktivitet: Endre stoppested.	
	2. Generer og viser interaktivt kart, som viser stoppestedsadressen i form av markør. Laster opp dataene til stoppestedet.
3. Administrator kan her endre stoppestedsinformasjonen.	
	4. Laster opp oppdatert kart, med genererte ruter mellom punktene. Oppdaterer også tabell over stoppestedene til ruten.
Alternativt hendelsesforløp – Legg til rute	
Alternativt hendelsesforløp – Slett rute	

2.2.3 High level use case beskrivelser

UseCase	**Vis rute**
Aktører	Bruker, Administrator (herunder refereres begge til som bruker)
Mål	Vise frem en lagret rute for eventuell utskrift
Beskrivelse	Bruker får opp en oversikt over eksisterende ruter, hvor han først angir hvilken type rute han vil se. Her skilles det mellom frie ruter, *dagsruter* og ruter som blir valgt ut fra en bestemt dato ved hjelp av en kalender. Frie ruter er ikke er knyttet til en bestemt dato eller dag, mens ved dagsruter kan bruker se ruter ut fra ukedagene de er lagt inn på.
Variasjoner	Ingen rute har blitt lagt inn i databasen.

UseCase	**Vis Sanntidsplassering**
Aktører	Administrator
Mål	Se plasseringen til alle bilene til bedriften i sanntid
Beskrivelse	Administrator får opp en oversikt over hvor alle bilene til bedriften befinner seg. Dette blir vist i et kart, ved hjelp av markører. Administrator kan se relevant tilleggsinformasjon til de enkelte bilene, ved å trykke på markøren som tilhører den bilen som Administrator vil inspisere. Dette vil da være informasjon som fart, Administratoren til bilen, og hvor den er på vei. Administrator skal også kunne søke på en adresse, som blir markert i kartet. Administrator kan dermed velge nærmeste bil og få opp veibeskrivelse til adressen. Koordinatene blir sendt til ETC sin database med et minutts intervall, og Administrator selv kan oppdatere siden, slik at det de nyeste data kommer opp.
Variasjoner	GPS fungerer ikke, og ingen eller uriktig informasjon blir gitt.

UseCase	**Vis Bilhistorikk**
Aktører	Administrator
Mål	Se historikken til en valgt bil
Beskrivelse	Administrator ber om å få se historikken til en bestemt bil, og hvilket datointervall rutene skal være på. Et kart med bilens kjørerute i valgt intervall vises, og Administrator kan selv velge hvilke ruter han vil se, og eventuelt se tilleggsinformasjon, slik som Administratoren til bilen, tidspunktet, ruten som blir kjørt, til rutene ved å trykke på markørene.
Variasjoner	Hvis det ikke ligger lagret noen ruter som har blitt kjørt med bilen, eller i valgt datointervall, vil det komme en melding om dette.

2.3 Produktkø

Produktkøen inneholder alle funksjonalitetene vi skal implementere i modulene, og skal bli brukt som utgangspunkt til sprintkøene, se vedlegg K: Sprintkøer. Til en sprintkø velger vi passende mengde av hva vi tror vi blir ferdig med av produktkøen i løpet av en sprint, her må vi også prioritere rekkefølgen på det vi skal implementere.

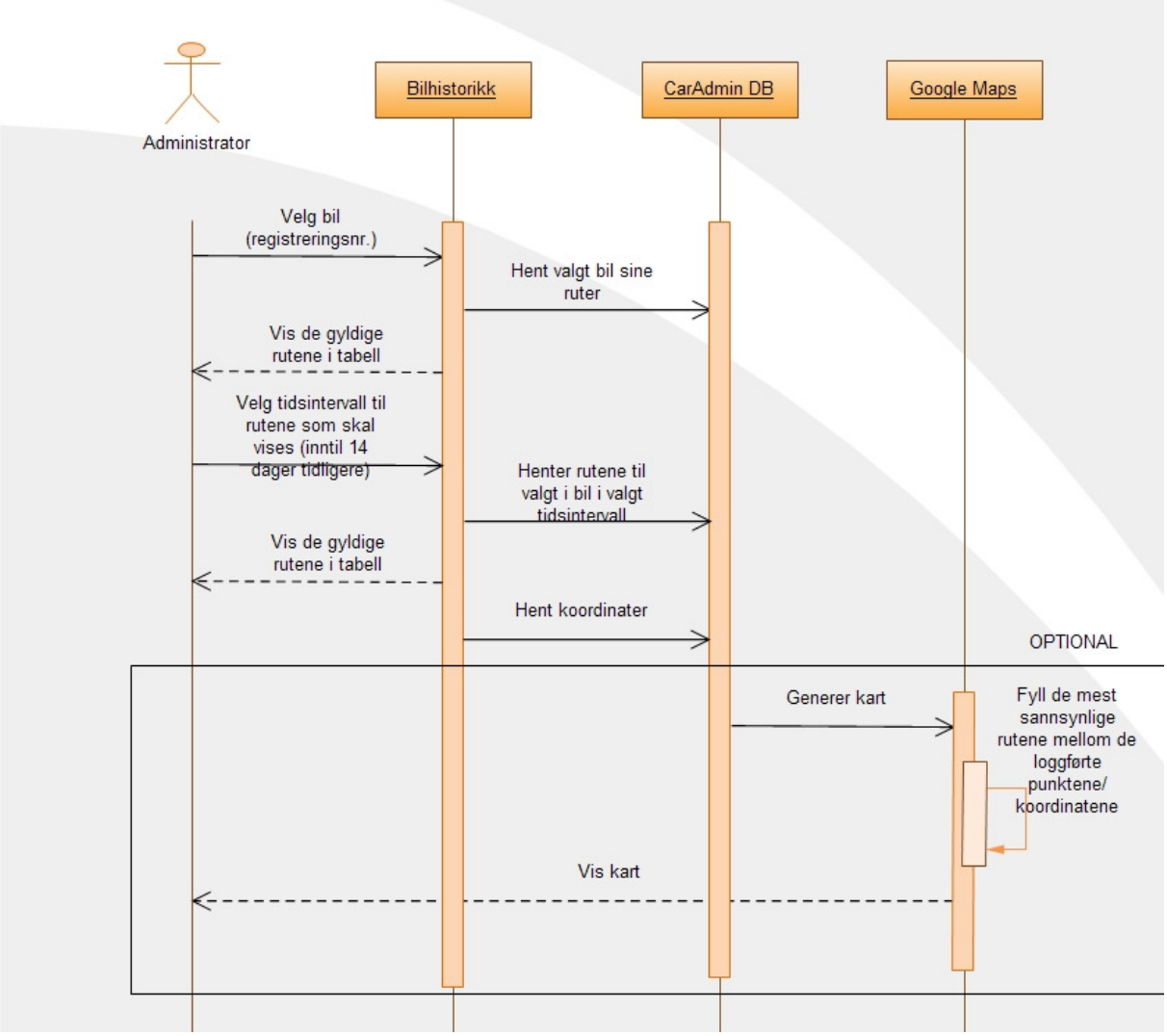
Beskrivelse	Gjenbruk av CarAdmin-kode
Feillogging/sporbarhet	Integrer eksisterende
Ruteplanleggermodulen	
Vis alle rutene:	
Velg type rute:	
Fri rute	
Rute med dato vha. kalender	Integrer eksisterende
Rute ut fra ukedag	
Tabell over eksisterende ruter	Integrer eksisterende
Behandle ruter:	
Velg type rute:	
Fri rute	
Rute med dato vha. kalender	Integrer eksisterende
Velg kjøretøysgruppe	Integrer eksisterende
Tabell over eksisterende ruter	Integrer eksisterende
Legg til rute	
Legg til stoppested med nødvendig stoppestedsinformasjon:	
Kalenderfunksjon ved innelegging av tid	Integrer eksisterende
Generer og vis rute vha. Google Maps	
Endre rute	
Legg til nytt stoppested:	
Kalenderfunksjon ved endring av tid	Integrer eksisterende
Endre eksisterende stoppested:	
Kalenderfunksjon ved endring av tid	Integrer eksisterende
Slette eksisterende stoppested	
Generer og vis rute vha. Google Maps	
Slett rute	
Sammenligne ruter:	
Velg type rute:	
Fri rute	
Rute ut fra datointervall vha. kalender	Integrer eksisterende
Tabell over eksisterende ruter	Integrer eksisterende
Generer og vis ruter vha. Google Maps	

Flåtestyringsmodulen	
Vis sanntidsplassering: Generer og vis rute vha. Google Maps Tabell over bedriftens kjøretøy	 Integrer eksisterende
Vis bilhistorikk: Velg kjøretøygruppe Tabell over bilens ruter Generer og vis rute vha. Google Maps	 Integrer eksisterende Integrer eksisterende

2.4 Sekvensdiagram

Vi har valgt å bruke sekvensdiagram for å illustrere samhandlingen mellom objektene i Bilhistorikk. Dette gjorde vi for å ha en detaljert visuell illustrasjon å forholde oss til, i tillegg til de skriftlige detaljerte use casene. Dette gjør det litt enklere å se hvordan kommunikasjonen mellom de ulike delene i funksjonen blir gjort.

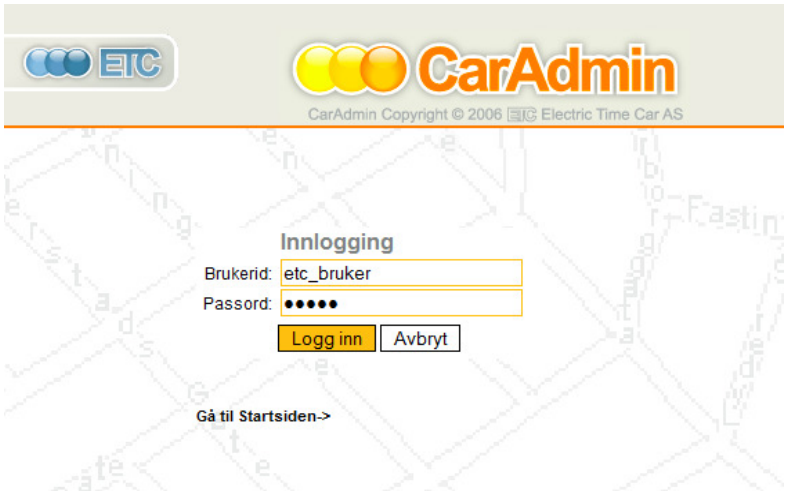
I forhold til optionalboksen i sekvensdiagrammet, vil en slik handlingsgang kun forekomme når Administrator har tilgang til Google Maps, siden nettstedet kan ha nedetid som er utenfor vår kontroll. Funksjonen vil fortsatt fungere uten tilgang til Google Maps, men i dette tilfelle vil kun tabellen bli vist.



Figur 5 Sekvensdiagram for visning av bilhistorikk

2.5 Supplementær spesifikasjon

2.5.1 Sikkerhet



Figur 6 Innlogging til CarAdmin-applikasjonen

Ved pålogging til CarAdmin-applikasjonen som modulene skal ligge på, må brukerne identifisere seg ved hjelp av brukernavn og passord, som vil bli sjekket mot CarAdmin sin database. Siden vi kun lager nye moduler til CarAdmin ligger dette allerede ferdig implementert.

2.5.2 Logg

Ruteplanlegger- og Flåtestyringsmodulen skal logge feil som skjer i modulene, slik at feilene kan hentes frem og analyseres.

Det skal også føres logg over aktiviteter i modulene. Dette vil si at det skal være mulig å spore alle endringer brukerne av modulene har gjort. Sporbarhet av brukerendringer er viktigere enn utviklingslogg, fordi den lar ETC spore endringer i systemet. Den innebygde *feilfanger*en vil plukke opp eventuelle programmeringsfeil. Det skal brukes try-catch i alle spørringer mot databasen, slik at kritiske feil ikke blir lagret.

Vi skal ikke lage noen av disse loggverktøyene, men ta i bruk funksjoner for dette som ETC allerede har i CarAdmin. Dette er henholdsvis Javalogger, og en ”feilfanger” som sender e-post til en forhåndsprogrammert adresse ved feil. Utover dette skal vi, for vår egen del under utviklingen av prosjektet, bruke en ”sporingslogg”; denne logger all informasjon fra *konsollvindu*et. Se 4.5.2 for mer om bruk av konsollvinduet.

2.5.3 Implementasjon

Ruteplanlegger- og Flåtestyringsmodulen skal utvikles med de samme utviklingsverktøy ETC bruker i CarAdmin. Det vil si vi skal bruke *Eclipse* og kode i Java, JavaScript og HTML, mot en MySQL database på en Tomcat server.

Mer om dette under implementasjonsdelen i kapittel 4.1.

2.5.4 Grensesnitt

Software

Ruteplanlegger og Flåtestyring er moduler som skal være integrert i den eksisterende applikasjonen CarAdmin. Dette er en nettapplikasjon, slik at brukerne ikke trenger å installere ekstra programvare. Modulene skal kunne fungere i alle de store nettleserne sine siste versjoner, som Mozilla Firefox 3.6.3, Internet Explorer 8, Google Chrome 4.1, Opera 10.53 og Safari 4.0.5. Modulene har imidlertid ingen støtte for utgåtte nettlesere, da det er brukers eget ansvar å ha oppdatert programvare siden tidligere nettelesere har påvist flere sikkerhetshull. Modulene skal ikke slutte å fungere, men vi kan bare ikke garantere at alt fungerer optimalt.

Siden Ruteplanlegger og Flåtestyring er moduler i CarAdmin-applikasjonen, vil tilgangen til disse være begrenset til CarAdmin sin tilgjengelighet.

CarAdmin er tilgjengelig på norsk, svensk og engelsk, mens Ruteplanlegger -og Flåtestyringsmodulen skal kun være tilgjengelig på norsk og engelsk i første omgang.

Hardware

Ruteplanlegger- og Flåtestyringsmodulen skal fungere optimalt hvis bruker har Bredbånd (1 Mb) eller Fiber. Google Maps kommer til å bruke varierende tid til å lastes inn på sidene. Denne tiden avhenger helt av hvor mye informasjon som finnes i kartet, og hvor rask tilknytningen til Internett datamaskinen har. Vi har ikke optimert Google Maps for at denne skal lastes raskere siden dette ikke kommer som en del av vår oppgave.

Design

Figur 7 Eksempel på hvordan modulene skal bli inkorporert i CarAdmin

Ruteplanlegger- og Flåtestyringsmodulen sitt utseende skal være basert på ETC sin CarAdmin-applikasjon, siden modulene skal være en del av denne. Brukergrensesnittet skal derfor være det samme som brukes i CarAdmin, slik at brukere av denne tjenesten enkelt vil kjenne seg igjen, og slipper å forholde seg til et nytt grensesnitt.

ETC har lagt vekt på et enkelt og intuitivt brukergrensesnitt i sin CarAdmin-applikasjonen slik at folk som i liten grad bruker PC til daglig, skal klare å bruke applikasjonen uten store problemer. Kunder har imidlertid også anledning til å få et skreddersydd brukergrensesnitt til både CarAdmin, Ruteplanlegger og Flåtestyring hvis de vil ha noe annerledes. Mer om dette i kapittel 3.3.

2.5.5 Responstid

Responstiden til modulene går under kravene til responstid for CarAdmin-applikasjonen. Dette innebærer at siden skal lastes innen ett sekund, slik at ingen serverhandlinger må bruke mer tid enn dette. Dette blir gjort med tanke på at brukeren skal ha en følelse av at det skjer noe, og derfor kan det ikke ta særlig mye lengre tid å laste applikasjonen, enn det tar å laste en standard nettside. Det kan ventes at Google Maps trenger noe lengre responstid. Responstiden går utenom oppgaven vår.

Stabilitet

ETC har ikke fastsatt noen krav til stabilitet i forhold til CarAdmin, annet enn at den skal være så høy at alle feil og unntak som skjer skal påvirke brukeropplevelsen så lite som mulig. ETC ønsker at løsningen skal være utformet på den måten at programvaren i seg selv skal garantere 99,7 % oppetid, noe som betyr at det blir litt over 24 timer nedetid per år. Grunnlaget til nedetid kan være nedetid i annen software som blir brukt, f.eks. Windows Update som typisk kan være ca 3 timer nedetid per år.

ETC vil at vi skal se bort fra maskinvare og databaseprogramvare i vår oppgave, dermed skal vi ikke se på stabilitetskravene som er satt til CarAdmin. De krever på sin side at vi skal strebe etter robusthet i koden vår, slik at det som blir gjort i løsningen ikke kan forårsake at databasen eller webserveren blir utilgjengelig. Robusthet innebærer feilsjekk av innlagte data, og feilhåndteringsrutiner for lagrede data. Det er dette som er viktig for å garantere en høy oppetid; at ingenting som kommer fra klienten kan frembringe en uventet feilmelding, og at ingenting som leses fra database (eller ikke kan leses) fører til at en side ikke lastes.

2.5.6 Konfigurasjon

Det skal være lett å tilpasse Ruteplanlegger- og Flåtestyringsmodulen i forhold til ønskene fra kundene. Vi må derfor være påpasselig med kommentering i koden vår, slik at det blir lett for ETC sine utviklere å følge gangen i hva vi har gjort. Vi må også sørge for at det ikke blir hardkoding der vi kan unngå det, og gjenbruke ETC sin kode i mest mulig grad.

2.5.7 Juridiske krav om personvern

Flåtestyringsmodulen krever sensitiv informasjon, slik som farten og posisjonen til bilen. Modulen vil inngå under Arbeidsmiljøloven bestemmelser om styringsrett, som sier at arbeidsgiver kan iverksette tiltak for å effektivisere bedriften. Dette vil si at så lenge denne informasjonen ikke blir lagret for å kartlegge de ansattes kjøremønster, er dette lovlig. ETC har dermed satt på restriksjonen i Vis Bilhistorikk, at bruker kun skal kunne se rutene til bilen i 14 dager bakover i tid. [2]

For å hindre misbruk av disse opplysningene, må vi også plassere en restriksjon på tilgangen til Flåtestyringsmodulen, slik at Administrator er den eneste som får se dette.

Ruteplanleggermodulen vil ikke inneholde noen sensitive opplysninger i forhold til de ansatte, slik at den faller utenom slike krav hvis ikke kundene til ETC har ønske om annet. Dette har vi diskutert nærmere under ”Risiko” i Forprosjektet, se vedlegg G.

Datatilsynet krever en viss sikkerhet i forhold til sikkerheten rundt slik informasjon, men dette gjelder hele CarAdmin-applikasjonen og ETC skal derfor håndtere denne problemstillingen.

2.5.8 Dokumentasjon

Ruteplanlegger og Flåtestyring skal benytte seg av samme brukerstøtte som CarAdmin. Denne er for tiden åpen døgnet rundt, men vil senere bli begrenset til ETC sin arbeidstid. Det vil være en brukermanual til selve applikasjonen som kundene får tildelt av ETC, men denne er ikke en del av vår oppgave.

En mindre form for brukermanualer er også tilgjengelig under menypunktene ”hjelp” i hele CarAdmin-applikasjonen, og skal dermed være tilgjengelig i Ruteplanlegger- og Flåtestyringsmodulen. Disse skal vi lage egen tekst til i våre moduler.

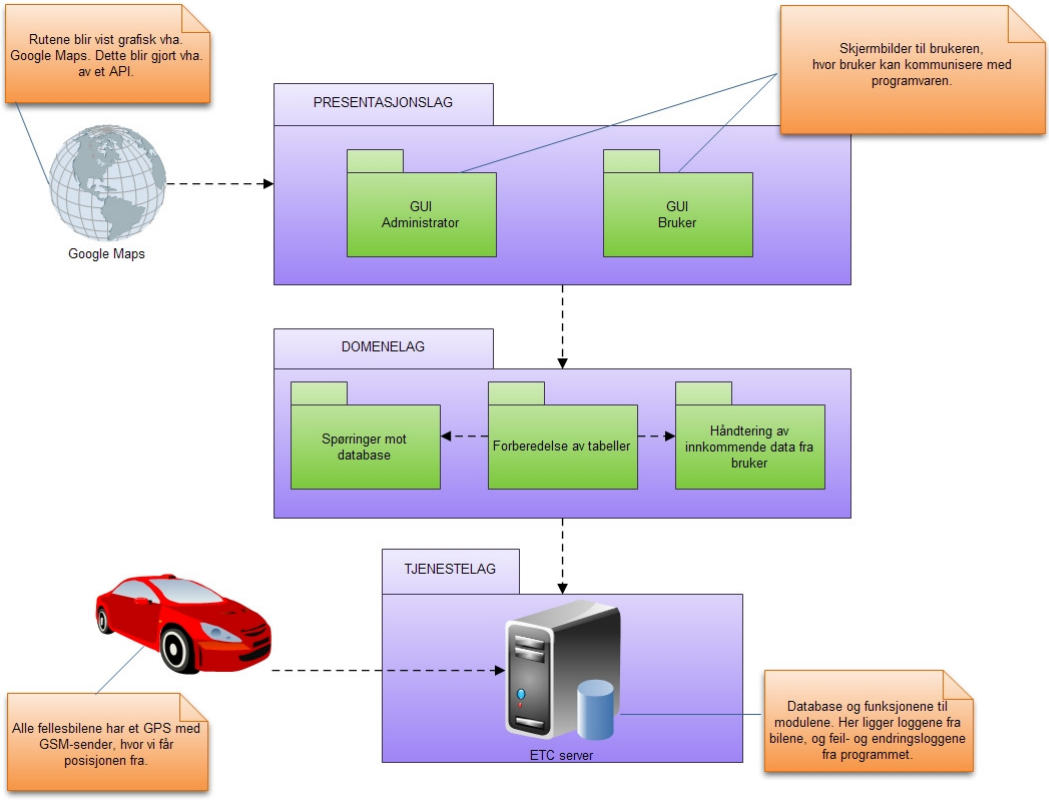
2.6 Utgivelser

Inkrementene skal være på fjorten dagers sprinter, hvor vi skal ha ulike deler av modulene ferdig etter hver sprint. Disse ferdigkodede elementene blir lastet på ETC sin server, slik at oppdragsgiver kan teste ut modulene.

[2] http://www.e-trip.no/?page_id=111, Publisert i 02.03.2010
http://www.datatilsynet.no/templates/Page____1366.aspx,Bekreftet kilde, publisert i 2006

3 DESIGN

3.1 Introduksjon



Figur 8 Pakkediagram

Designet i modulene bygger på det eksisterende designet til ETC i CarAdmin, slik at brukeren til CarAdmin blir møtt med en mest mulig konsistent applikasjon. Vi viderefører dermed bruken av en trelagsstruktur, men får i tillegg innhenting av data fra GPS med GSM-sender i bilene, og fra Google Maps sitt kart og veibeskrivelse.

3.1.1 Generelle mål

Modulene har et intuitivt oppsett, slik at de er enkle å finne frem i, selv for de som er mindre datakyndige. Dette målet har vi håndtert ved hjelp av brukertester, se kapittel 5.2. I forhold til brukergrensesnittet har vi i størst mulig grad gjenbrukt deler fra CarAdmin, slik at det skal være lett for brukerne å finne frem - også i de nye modulene.

Det underliggende designet følger det eksisterende oppsettet til CarAdmin, både med hensyn på lagdelingsstruktur, navngiving og databasestruktur, slik at det er enkelt for programutviklerne i ETC å gå inn for å gjøre endringer og eventuelt videreutvikle modulene.

3.2 Logisk oversikt

3.2.1 Lagdelingsmodellen

Som nevnt tidligere har modulene blitt laget i trelagsstruktur. Dette ble gjort etter ønske fra ETC. Deres eksisterende kode har en flerlagsstruktur, som er et utspring fra trelagsstrukturen.

Figur 9 ETC sin lagdelingsstruktur

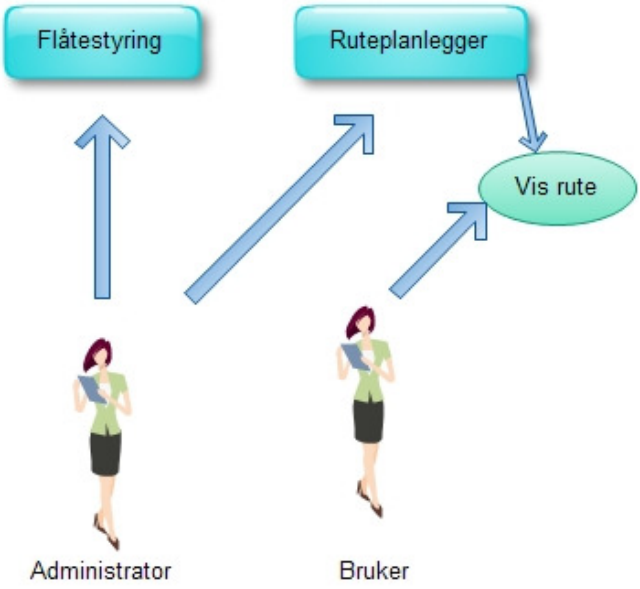
Trelagsstrukturen baserer seg på en visningsdel, presentasjonslaget, hvor det kun foregår visning og den mest nødvendige funksjonaliteten. Domenelaget håndteres på ETC sin server. Her blir alle spørringer mot databasen foretatt, og alle utregninger, logikk og lignende blir utført. For å forenkle arbeidet vårt, er dette laget er en sammenslåing av alle mellomlagene i ETC sin lagdelingsstruktur. Databasen blir tjenestelaget, hvor alle data ligger lagret i logiske tabeller.

Figur 10 Vår trelagsstruktur

Presentasjonslaget

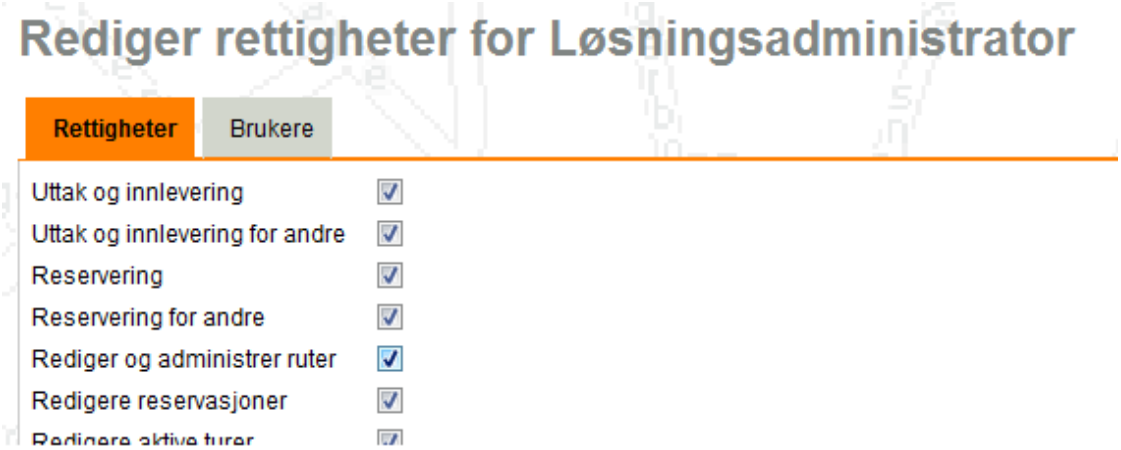
Presentasjonslaget skal ta for seg det som har med brukergrensesnittet å gjøre, der samhandling mellom brukeren og applikasjonen blir visualisert med knapper, menyer, tabeller og lignende, samt hvordan tilgangen til de ulike funksjonalitetene blir håndtert.

Bruken av Google Maps sitt API (Application Programming Interface) bryter med trelagsstrukturen, siden denne koden kjøres og behandles direkte fra presentasjonslaget, se figur 8 (Pakkediagram) og figur 10 (Vår trelagsstruktur). Kodefilene i presentasjonslaget, altså JSP-filene (JavaService Pages), består av tre programmeringsspråk: Java, JavaScript og HTML. Det som ligger av kode i Java i JSP-filene, og tilhørende Javafiler, kjøres på serveren, mens HTML-koden og JavaScript kjøres direkte i nettleseren. Ved lasting av nettapplikasjonen vil derfor Javakoden kjøres først, deretter JavaScriptkoden. Dette medfører at all koden for Google Maps, og all direkte logikk knyttet til blant annet innhenting av data fra bruker, skjer i JavaScript og HTML. Mer om arbeidet med å implementere bruken av Google Maps sitt API kommer i kapittel 4.4.



Figur 11 Brukernes tilgang til funksjonalitetene

Vi skiller mellom to brukere, Administrator og Bruker, som omtalt i avsnitt 2.1.2. Det er kun Administrator som har tilgang til Flåtestyringsmodulen, med tanke på sikkerhetshensyn og personvern. Dette har vi diskutert i kapittel 2.4.7. I forhold til Ruteplanleggermodulen har Administrator tilgang til det overordnede arbeidet med å planlegge rutene i forhold til kalenderen, hvor han kan legge til nye, samt redigere og slette eksisterende ruter. Administrator kan også sammenligne flere ruter for å se om de bør endres eller bli slått sammen for å bedre effektiviteten til bedriften. Bruker har kun tilgang til å se de ferdigplanlagte rutene, og har dermed ikke tilgang til annen funksjonalitet i modulene annet enn muligheten til å skrive ut rutene sine. Med hjelp av ETC, har vi løst de ulike brukertilgangene ved å legge tilgangen til å håndtere eller sammenligne ruter fra Ruteplanleggermodulen og hele Flåtestyringsmodulen som et ekstra tilgangspunkt, ”Rediger og administrer ruter”, i styringen av brukerrettigheter.



Figur 12 Håndtering av brukerrettigheter

Domenelaget

Domenelaget tar for seg funksjonene, og er knyttet opp mot tjenestelaget samt presentasjonslaget. I figur 8 (Pakkediagram), har vi tatt med tre viktige domener som er sentrale i våre funksjoner. Domenet "Sted" har variablene og funksjonaliteten for å håndtere enkelte stoppesteder, samt en tilknytning til en rute. En rute har variable og funksjoner for å håndtere rutespesifikke opplysninger, for eksempel navn på ruten, tiden ruten tar og om det er en fri rute. Domenet "Dato" er laget av ETC, men siden vi har gjenbrukt denne i såpass stor grad i koden vår, har vi valgt å ta den med i diagrammet. Dato og tidspunkt, som dette domenet håndterer, benyttes i alle ruter og stoppesteder som er datospesifikke, mens frie ruter med sine stoppesteder, kun holder på tidspunktet. Mer om dette finnes i kapittel 4.3.1. Domenelaget har ingen eksterne tilkoblinger.

Tjenestelaget

I tjenestelaget finner vi databasen og serveren til ETC, og hvordan de kommuniserer med GPS-enhetene (Global Positioning Service) i bilene. Som man ser av figur 8(Pakkediagram) henter serveren informasjonen fra databasen hvor innkommende data fra GPS-enhetene lagres, og mater informasjonen til databasen en gang i minuttet. Implementeringen av GPS i bilene er ETC sitt ansvar. Siden de ikke har fått dette helt i orden foreløpig, har vi i prosjektet arbeidet mot en selvlaget databasetabell med testverdier. Databasen diskuteres videre under avsnittet 3.2.3 (Lagring av data).

3.2.2 Kommunikasjon

Modulene skal håndtere innkommende data fra eksterne medier, henholdsvis en GPS med GSM-sender i hver bil, og fra Google Maps sitt API. All informasjon, som legges inn via brukergrensesnittet, må håndteres og lagres unna til databasen.

Bruker – server

Ved pålogging til applikasjonen må brukeren skrive inn sitt gitte brukernavn og passord. Hvis passord og brukernavn blir godkjent, vil applikasjonen spørre serveren om hvilken tilgang denne brukeren har. Brukeren får så tilgang til sin respektive meny, med dets restriktive tilgang til data, ettersom hvilken tilgang han har. Hvis brukernavn og/eller passordet ikke blir godkjent, vil det komme opp en feilmelding og et nytt påloggingsvindu. Dette er allerede håndtert i ETC sitt eksisterende system.

Annen kommunikasjon med serveren skjer eksempelvis ved oppretting av en ny rute, da dataene vil bli lagret i databasen. Dette skjer når Administrator har lagt rutens navn i tekstfeltet, krysset av kjøretøygruppe i avkrysningsboksene og valgt dato for ruten i kalenderen.

Bil – server

I alle fellesbilene i bilparken vil det bli installert en GPS med et GSM-sender. GPS-enhetene vil en gang i minuttet sende posisjonen sin til ETC sin server, slik at denne informasjonen blir loggført, og kan senere bli hentet ut til videre bruk i Sanntidskart eller *Historiekart*.

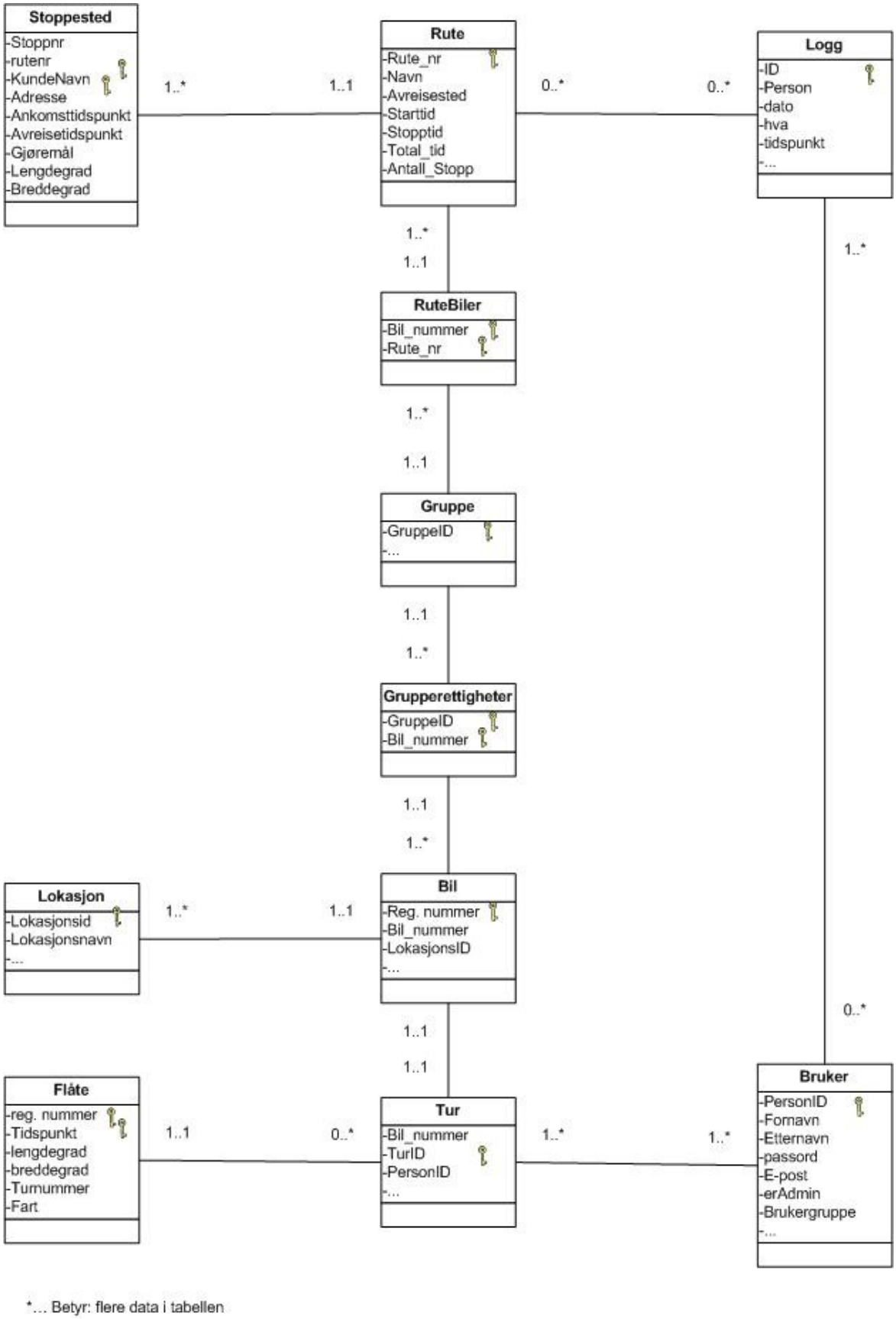
Google Maps – nettleser

Det er flere funksjoner i modulene som krever Google Maps for å vise rutene grafisk. JSP-sidene vil be Google Maps om å vise kartet ut fra koordinatene til stoppestedene i ruten brukeren har bedt om. Dersom Google Maps er nede, vil kun tabellen med informasjonen bli vist. I Sammenlign ruter, Sanntidskart og Historiekart vil det i et slikt tilfelle komme en feilmelding, siden vi ikke har laget noen tabeller i disse funksjonalitetene. Mer om denne kommunikasjonen omtales i kapittel 4.4.2.

3.2.3 Lagring av data

Dataene blir lagret på ETC sin database i logiske tabeller. Tabellene og attributtene vi bruker er representert i databasemodellen, hvor tabellene for ”Lokasjon”, ”Bil”, ”Gruppe”, ”Grupperettigheter”,” Tur”, ”Logg” og ”Bruker” er hentet fra den eksisterende databasen. I tabellene hvor det er merket med ”…” ligger det flere attributter i tabellen, som ikke er relevante for vårt bruk. Posisjonsloggen kommer inn fra GPS-enhetene en gang i minuttet.

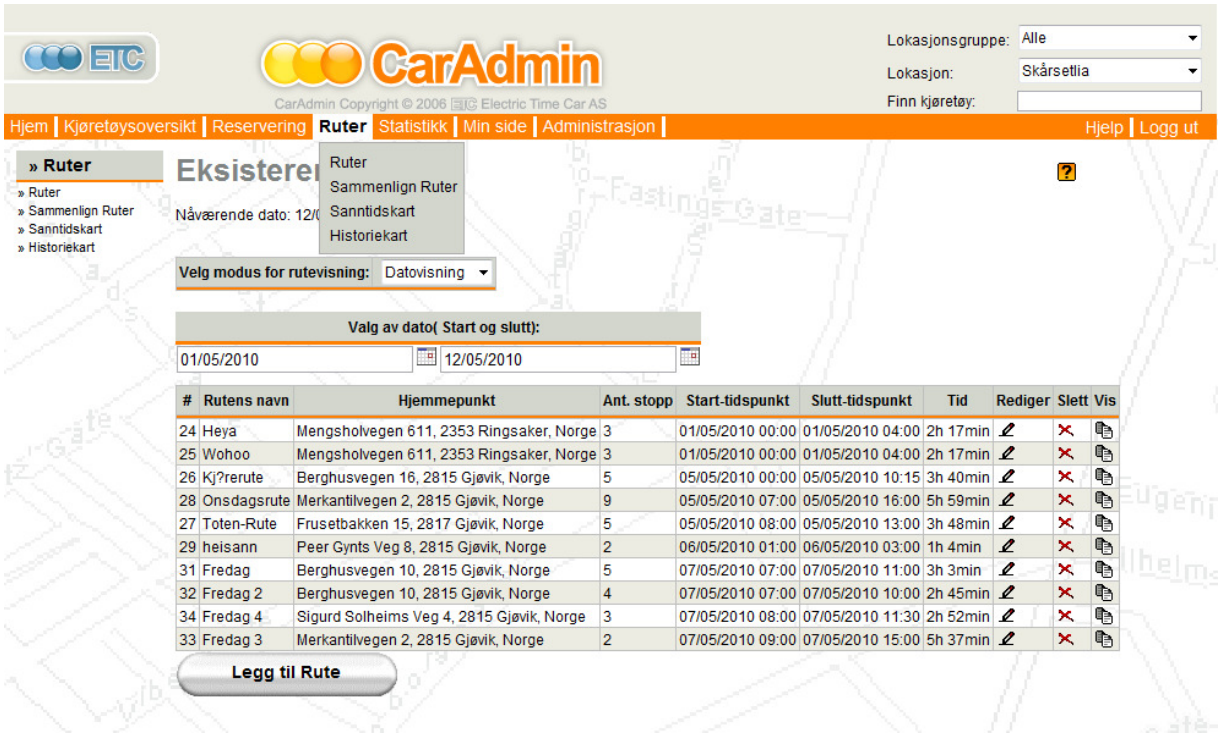
Databasen inneholder ingen fremmednøkler, siden den opprinnelige databasemotoren til ETC ikke inneholdt støtte for relasjonsdatabaser. Disse koblingene løses derfor med noe ekstra kode, som sikrer at forbindelsene fungerer. ”RuteBiler”-tabellen er laget som en mellomtabell mellom ”Bil” og ”Rute”, for å unngå mange til mange-koblinger. Dette gjelder også ”GruppeRettigheter”, som finnes i CarAdmin sin database fra før. ”Lokasjon” er knyttet til bilene, og hentes ut blant annet ut for å kunne foreslå *lokasjonen* som avreisestedet for en ny rute. På denne siden er det kjøretøysgruppen som angis, men denne henter ut lokasjonen til en bestemt bil i kjøretøysgruppen.



Figur 13 Databasemodell

3.3 Design av brukergrensesnitt

Brukergrensesnittet til modulene har tatt utgangspunkt i det eksisterende grensesnittet til CarAdmin-applikasjonen. Som vist i figur 14 (Oppsett av side), finner man Ruteplanlegger- og Flåtestyringsmodulene med deres funksjonaliteter under menypunktet ”Ruter”.



#	Rutens navn	Hjemmepunkt	Ant. stopp	Start-tidspunkt	Slutt-tidspunkt	Tid	Rediger	Slett	Vis
24	Heya	Mengsholvegen 611, 2353 Ringsaker, Norge	3	01/05/2010 00:00	01/05/2010 04:00	2h 17min			
25	Wohoo	Mengsholvegen 611, 2353 Ringsaker, Norge	3	01/05/2010 00:00	01/05/2010 04:00	2h 17min			
26	Kj?rerute	Berghusvegen 16, 2815 Gjøvik, Norge	5	05/05/2010 00:00	05/05/2010 10:15	3h 40min			
28	Onsdagsrute	Merkantilvegen 2, 2815 Gjøvik, Norge	9	05/05/2010 07:00	05/05/2010 16:00	5h 59min			
27	Toten-Rute	Frusetbakken 15, 2817 Gjøvik, Norge	5	05/05/2010 08:00	05/05/2010 13:00	3h 48min			
29	heisann	Peer Gynts Veg 8, 2815 Gjøvik, Norge	2	06/05/2010 01:00	06/05/2010 03:00	1h 4min			
31	Fredag	Berghusvegen 10, 2815 Gjøvik, Norge	5	07/05/2010 07:00	07/05/2010 11:00	3h 3min			
32	Fredag 2	Berghusvegen 10, 2815 Gjøvik, Norge	4	07/05/2010 07:00	07/05/2010 10:00	2h 45min			
34	Fredag 4	Sigurd Solheims Veg 4, 2815 Gjøvik, Norge	3	07/05/2010 08:00	07/05/2010 11:30	2h 52min			
33	Fredag 3	Merkantilvegen 2, 2815 Gjøvik, Norge	2	07/05/2010 09:00	07/05/2010 15:00	5h 37min			

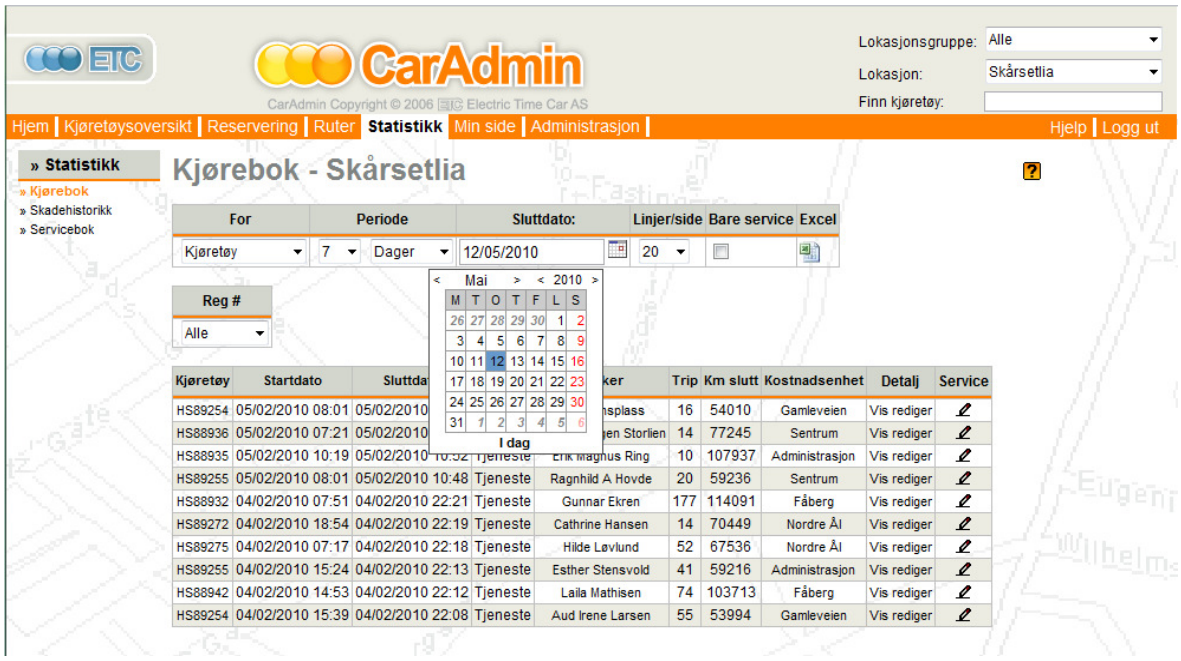
Figur 14 Oppsett av side

CarAdmin er bygd opp med en tittel-linje øverst på siden, med blant annet hyppig brukte søkefunksjoner som ”Finn kjøretøy”. Under tittellinjen, finner vi hovedmenyen markert i oransje. I forhold til figur 14 (Oppsett av side) er vi inne i ”Ruter”, og, som man kan se, får man opp en aktiv undermeny til venste i siden, som viser funksjonalitetene som ligger under valgt menypunkt. Disse funksjonalitetene kommer også opp som rullgardin-vindu når bruker holder musepekeren over menypunktet.

Grensesnittet til CarAdmin er laget for å være enkel og intuitiv å bruke for brukerne, samtidig som det skal kunne tilby mange menyvalg på en enkel og oversiktlig måte. Dette er løst i CarAdmin-applikasjonen ved å dele inn menyen i toppmenyer og undermenyer. Funksjonaliteten som ligger under disse menypunktene har blitt løst ved hjelp av knapper, tekstfelt, tabeller og lignende. Noe av funksjonaliteten har også blitt skjult, i den betydning at hvis brukeren eksempelvis vil endre, vise eller slette innholdet til ruten, må de gjøre det ved hjelp av ikonene til venstre i rutetabellen. Ikoner blir gjerne brukt da dette gjør det mer robust i forhold til språkbarrierer.

Det har aldri tidligere blitt benyttet kartverk i CarAdmin. Her har vi benyttet Google Maps API v2 sin standard. Dette er et enkelt kart med funksjonalitet for zoom og panorering, og mulighet for å velge å vise som satelittbilde eller hybridbilde, som er et satelittbilde med ekstra opptegning av veier, og

noe tilleggsinformasjon. I tillegg kommer også Google sin standard veibeskrivelse. Google Maps er videre omtalt i kapittel 4.4.



Figur 15 Eksempel på CarAdmin sitt brukergrensesnitt

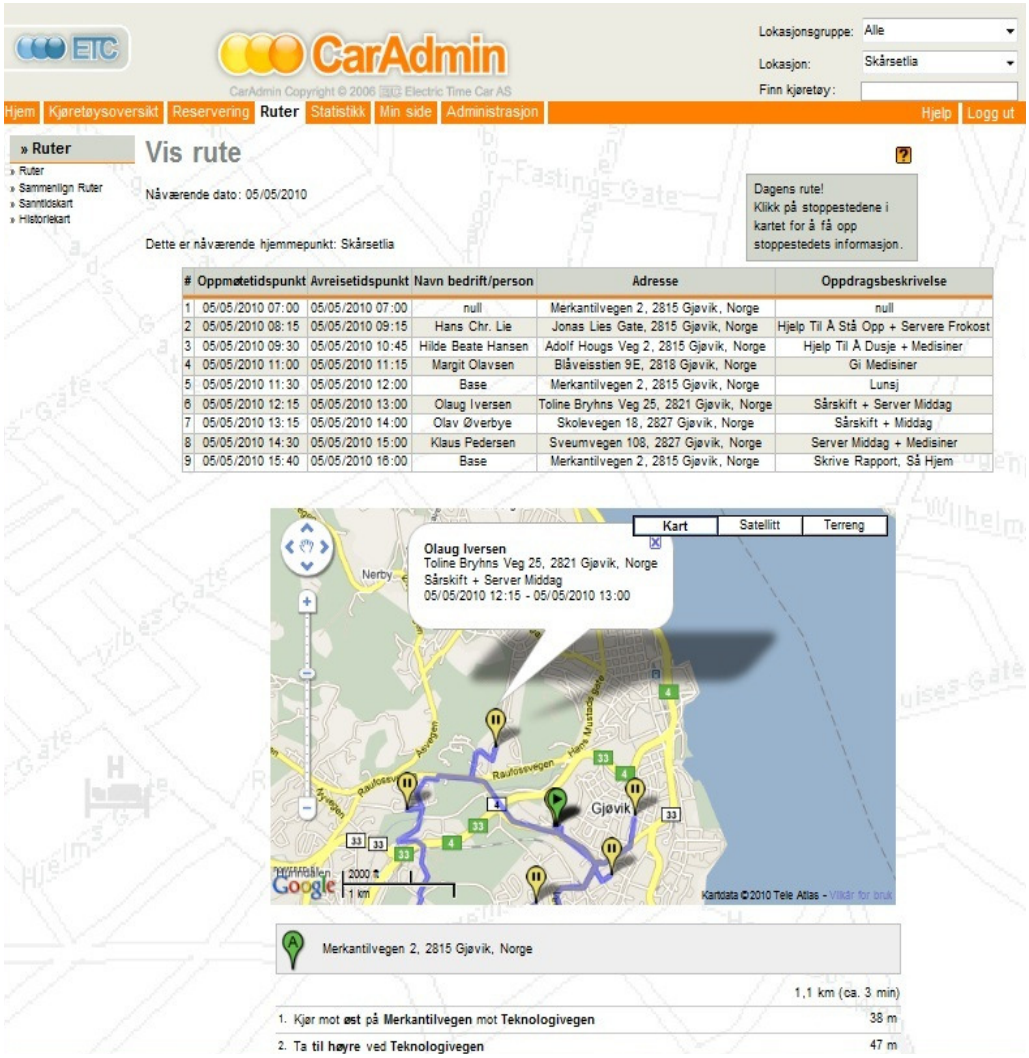
Ovenfor er det gitt et utsnitt av hvordan grensesnittet ser ut når man skal reservere et kjøretøy, som var i den opprinnelige CarAdmin-applikasjonen. Vi har blant annet brukt kalenderen og tabellen herfra i utviklingen av våre moduler.

En av de større utfordringene vi hadde med denne oppgaven, var å få opp et kart, som tok utgangspunkt i tabellen som ble generert med nødvendig data.

Figur 16 Tidlig utkast til forslag av oppbygning for å vise ruten

Før vi begynte med kodingen, skisserte vi et tidlig utkast, se figur 16, i forhold til oppbygningen av visning av en rute. Figur 17 viser den tilsvarende funksjonaliteten som den ble i den ferdige

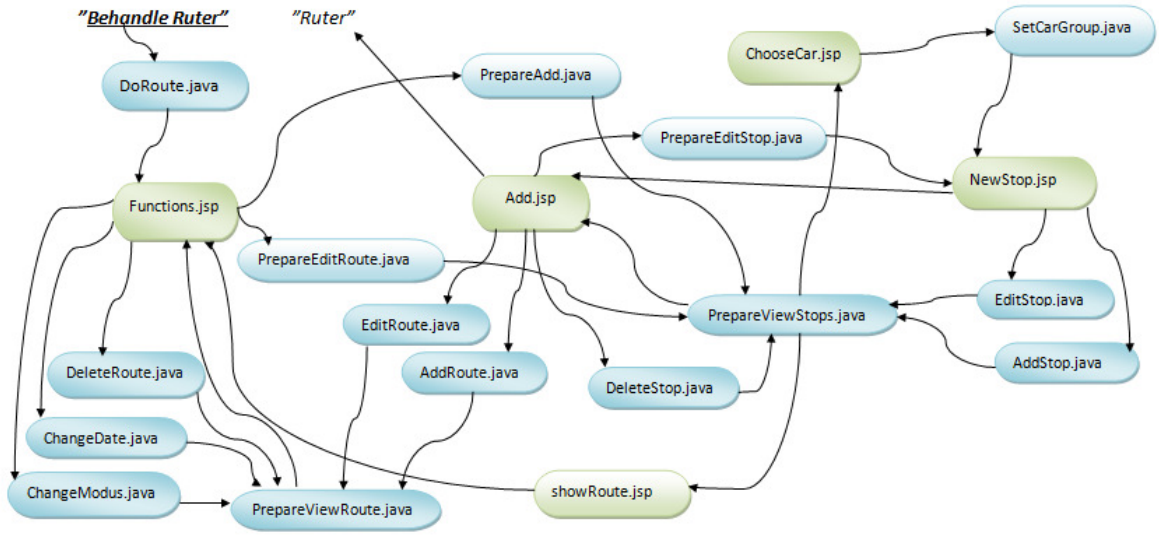
løsningen. Som man ser ut fra disse figurene, har ETC implementert en ny menystyring i CarAdmin underveis i utviklingen.



Figur 17 Viser en ferdig kjørerute

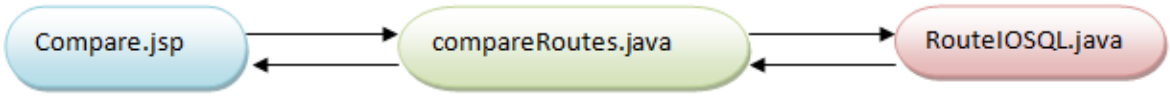
3.4 Filstruktur

Under finner vi et utvalg figurer som illustrerer de filene vi har laget selv, og hvordan de er koblet sammen.



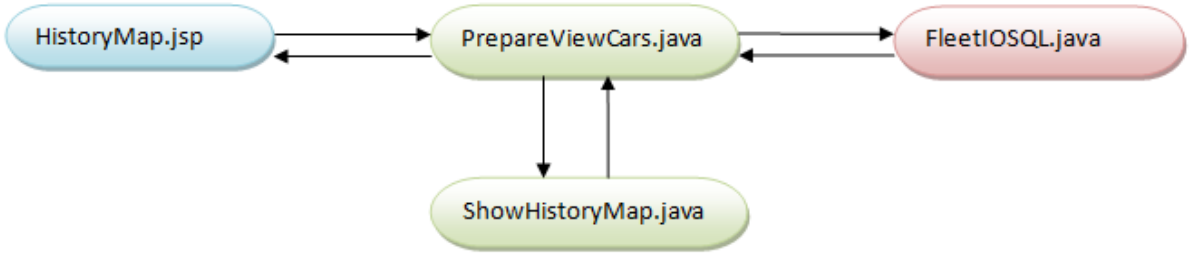
Figur 18 Kall mellom filene under menypunktet "Ruter"

I figur 18 finnerer man alle filene vi har laget for å implementere menypunktet "Ruter". Dette er det største menypunktet til modulene, hvor bl.a. funksjonalitetene for å legge inn, slette og redigere ruter, samt vise rutene blir vist i sin helhet. Alle Java-filene har kall til RouteIOSQL, se vedlegg E (Kode) for å hente eller lagre data i databasen. RouteIOSQL inneholder spørringer mot databasen.



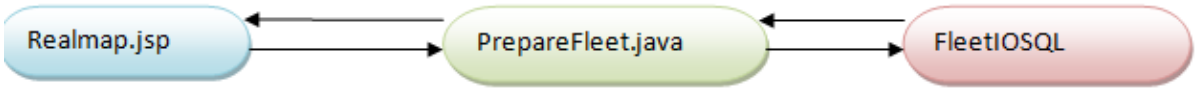
Figur 19 Kall mellom filene i menypunktet Sammenlign Ruter

Som man ser av figur 19, går Sammenlign Ruter sine kall til compareRoutes, uansett om det er visningsformen av rutene, eller datoen til rutene som vises, som skal forandres. Hver gang dette kjøres, går det et kall videre til RouteIOSQL som tar seg av all videre samhandling med databasen. I denne sammenhengen henter RouteIOSQL ut all informasjon om alle lagrede ruter i et visst datointervall.



Figur 20 Kall mellom filene i menypunktet Historiekart

I Historiekartet går kallet rett til ShowHistoryMap fra menypunktet, for så å gå til PrepareViewCars som kaller FleetIOSQL for håndtering av kall til database. Når bruker er inne i HistoryMap og endrer datoer som kjørte ruter skal vises på, eller hvilken bil sine kjørte ruter som skal vises går kallet direkte til FleetIOSQL, via PrepareViewCars. FleetIOSQL returnerer informasjon om de enkelte bilenes posisjoner.



Figur 21 Kall mellom filene i menypunktet Sanntidskart

Sanntidskartet får også informasjon fra FleetIOSQL, men her hentes kun det nyeste punktet for hver bil ut ved hjelp av PrepareFleet. Det vil si at det er en egen spørring i FleetIOSQL som tar seg av å finne ut hvem av alle de lagrede postene om en enkelt bil som er den nyeste.

Ruteplanlegger- og Flåtestyringsmodulen har, som vist ovenfor, en rekke funksjoner som gjør at de skiller seg ut fra hva CarAdmin og Google Maps API kan gjøre alene. Som hovedfunksjonalitet er det mulig å legge inn, endre og slette ruter, som har mange stoppesteder. Her får man opp både tabeller og visning i kart, som inneholder all informasjon om ruten. Det finnes også funksjonalitet for å vise i et sanntidskart hvor bilene befinner seg til enhver tid, og vise et historiekart for hvor en bil har kjørt i løpet av de siste 14 dagene. Noe som spesielt skiller seg ut, er muligheten til å vise flere ruter i samme kart, for å kunne sammenligne disse.

Vi har delt inn våre filer ettersom hvilket Use Case de tilhører. Ruter tar for seg all funksjonalitet beskrevet i Use Caset ”Behandle Ruter”, Sammenlign Ruter tar for seg Sammenligne ruter, Historiekart tar for seg ”Vis Bilhistorikk” og Sanntidskart tar for seg ”Vis Sanntidsplassering”. Use Caset ”Vis rute” er brukt både i Ruter, Sammenlign Ruter og Historiekart.

Vi valgte denne inndelingen fremfor noen annen, fordi vi ville strukturere koden vår ut ifra hva vi utviklet akkurat nå, og ikke hva vi kanskje skulle utvikle i neste sprint.

3.5 Tekniske memoer

De tekniske memoene har blitt brukt til to ting - for å illustrere vårt syn på eventuelle problemer, og løsninger på slike, og for å gi et bilde på hvordan vi ville løst funksjonalitet vi ikke har implementert.

Vi har prøvd å implementere all funksjonalitet ETC har bedt om, men i de siste ukene før innleveringen av bacheloroppgaven har vi måttet nedprioritere koding til fordel for rapportskriving. Som en følge av dette har funksjonaliteten for mulighet til omkjøringer ikke blitt implementert, men dette betyr ikke at vi ikke har tenkt ut en løsning for hvordan dette kan gjøres. Denne løsningen er presentert som et Teknisk Memo.

3.5.1 Problem: Kunder finner forholdet mellom CarAdmin og Ruteplanlegger/ Flåtestyring uklart

Oppsummering av løsning:

Ruteplanlegger og Flåtestyring utgis med samme brukergrensesnitt som CarAdmin.

Faktorer:

- Kundene er kjent med CarAdmin sitt brukergrensesnitt.
- Færre brukergrensesnitt å forholde seg til.
- Gjør implementering mot CarAdmin lettere.

Løsning:

Ruteplanlegger gis samme brukergrensesnitt som CarAdmin har.

Motivasjon:

Dette gjør implementering og videreutvikling av Ruteplanlegger og Flåtestyring i CarAdmin mye enklere, og gir brukerne færre grensesnitt de må lære seg.

Uløste problemer:

- Ingen.

Alternativer:

Lage et nytt grensesnitt til Ruteplanlegger og Flåtestyring med baktanke at den er en selvstendig modul, og gi denne et grensesnitt kun tilpasset sin egen oppgave. Dette vil gi noe mer arbeide, men vil gi et mer oppdatert brukergrensesnitt. Ulempe er at implementeringen mot CarAdmin må gjøres om.

3.5.2 Problem: Håndtering av avbrudd.

Oppsummering av løsning:

En avbrutt handling må utføres på nytt.

Faktorer:

- Hver gang en bruker starter en handling, tildeles vedkommende en *sesjon* (”session”). Denne vil logge ut etter en gitt tid, dersom ikke lagrende handlinger utføres, siden sesjonen avholder tid på serveren og denne vil gis videre til andre som venter. Dersom en sesjon er utgått vil brukeren miste sine ulagrede data.

Løsning:

Dersom en sesjon går ut, kobles serveren fra, og over til neste ventende. Dersom bruker ikke har lagret eller utført handling som gir lenger sesjon, vil bruker logges ut. Den mistede informasjon må angis på nytt.

Håndteringen mot server må derfor inneholde metoder, som gjør at alle transaksjoner lagres unna etter hver endring som blir lagret, slik at minst mulig informasjon går tapt.

Motivasjon:

Målet er å miste minst mulig informasjon ved avbrudd, men samtidig ikke eksponere systemet for unødvendige sikkerhetsrisikoer ved at brukere av Ruteplanlegger og Flåtestyring kan være innlogget hele tiden. Dessuten holder en sesjon på ganske mye av en servers ressurser og det er derfor ønskelig å bytte på å gi brukere tilgang. Alle brukere kan ikke ha tilgang til hele Ruteplanlegger og Flåtestyring samtidig.

Uløste problemer:

- Hva skjer med brukerens data?
- Hvor lang en sesjon skal være, må avklares på fremtidige møter med ETC, samt testes under programmering, slik at denne blir tilpasset både brukervennlighet, og servertilgjengelighet.

Alternativer:

- Det spørres om sesjonen vil forlenges eller om dataene skal lagres, når sesjonen går ut.
- Det lagres automatisk, og vises med spørsmål om fortsettelse ved neste start.

3.5.3 Problem: Hvordan skal vi implementere muligheter for omkjøring i rutene?

Oppsummering av løsning:

Hver rute sine stoppesteder må ha mulighet til å knytte til seg omkjøringer enten før eller etter.

Faktorer:

- Overstyring av Google Maps API sin valgte rute.
- Bruker kan ha bedre kjennskap til veien og div. snarveier enn Google Maps API.
- Google Maps API har ingen kjennskap til hva som skjer lokalt, og da ved spesielle anledninger sånn som ved ny veilegging, asfaltering eller andre grunner til forsinkelser kan ikke API hjelpe bruker med dette.

Løsning:

Når et nytt stoppested opprettes eller når et stoppested er valgt til redigering vil brukeren få valg om han/hun vil sette på en eller flere omkjøringspunkter foran eller etter nåværende stoppested. Bruker skal også kunne slette omkjøringer, men ikke redigere dem siden det bare er adressen og før eller etter variabelen som kan endres, og da er det like greit å opprette en ny omkjøring.

Hvis det opprettes omkjøringspunkter vil disse bli lagret i en egen tabell i databasen med rutenummeret til ruten de tilhører, nummeret til stoppestedet de tilhører, adressen til omkjøringen, omkjøringsnummeret og om omkjøringen skal skje før eller etter stoppestedet.

For å tegne ruten riktig med de nye omkjøringene vil disse bli hentet og lagt som en del av ruten før og etter stoppestedet, alt ettersom hva bruker har spesifisert. Men det vil ikke bli tegnet noen ikoner ved omkjøringene siden dette ikke er stoppesteder i seg selv men, en del av ruten.

Motivasjon:

At brukerne skal kunne bruke sin lokale kunnskap, og evt. nyheter til selv å velge den beste ruten.

Uløste problemer:

- Hva om bruker vil opprette en omkjøring uavhengig at et stoppested?

Alternativer:

Omkjøringene kunne vært laget uavhengig av stoppestedene og blitt knyttet direkte opp mot rekkefølgen til stoppestedsnumrene. Men dette ville skapt problemer siden stoppestedene sorteres etter hvilken dato og tidspunkt de ligger på og da ved en omrokering av denne rekkefølgen vil omkjøringene bli plassert feil.

4 Implementering

4.1 Utviklingsverktøy

Ruteplanlegger- og Flåtestyringsmodulen har blitt utviklet med ETC sine standard utviklingsverktøy for CarAdmin. Modulene har med andre ord blitt programmert i Java, versjon J2ee[3], ved hjelp av kompilatoren Eclipse med tillegget *Subclipse.*

Databasen er av typen MySQL, og serveren er av typen Tomcat. Modulene gjør bruk av ETC sin CarAdmin database, som vi har brukt MySQL Query Browser til å lage SQL-spørringer mot.

Vi har brukt Microsoft Word til å utlede rapporten vår i, samt brukt Microsoft Project til å lage Ganttskjemaene våre for fremdriftsplaner, se vedlegg B, og avslutningsdelen av hovedrapporten. Vi har brukt E-Draw til å lage Use Case diagrammet, sekvensdiagrammene og de fleste andre figurene i rapporten, bortsett fra databasemodellen som vi lagde i Microsoft Visio. Excel er brukt for å lage våre personlige logger.

4.1.1 Eclipse

Vi har brukt Eclipse Galileo versjon 6, av open source programmeringsverktøyet Eclipse. Denne versjonen ble utgitt 24. juni 2009 og har blitt utviklet av The Eclipse Foundation[4].

Eclipse har vært et stort verktøy å sette seg inn i. Det har mange bra funksjonaliteter, som for eksempel å gi klare markeringer der feil oppstår, forslag til løsninger der feilene oppstår og gi oversikt over hvilke funksjoner det er mulig å bruke i en gitt klasse. Ikke alt i Eclipse har fungert optimalt. Noen av problemene vi har slitt med er uventede feilmeldinger hvis server ikke er med i åpningsbildet, siden denne ble lukket sist gang programmet ble brukt, samt falske markeringer av feil. Vi skulle gjerne sett at Eclipse hadde brukt ressurser til å rette opp småfeilene sine bedre. Disse har gitt oss et middelmådig inntrykk av vektøyet, samtidig som de har tatt bort verdifull tid fra programmering da vi har måttet lete etter midlertidige løsninger ved småfeilene til Eclipse.

En annen faktor vi ikke har latt oss nevneverdig begeistre av i Eclipse er feilsøkingsmodulen. Vi opplevde denne som lite brukervennlig, treg og til tider uforståelig, noe som medførte at vi kun brukte denne modulen der det absolutt ikke kunne unngås.

Det er ikke sikkert at vi ville hadde valgt å bruke Eclipse som utviklingsverktøy for oppgaven, hvis vi hadde hatt erfaringene i begynnelsen av oppgaven, som vi har med Eclipse per i dag. Vi har veldig liten kjennskap til hvilke utviklingsverktøy av denne typen som er på markedet i dag utenom Eclipse, og hvor bra de enkelte fungerer. Men hvis vi hadde gjort oppgaven om igjen ville vi nok ha laget en oversikt over hva som fantes, med fordelder og ulemper, ved de enkelte av disse slik at vi kunne ha gjort et informert og gjennomtenkt valg.

4.1.2 Subclipse

Subclipse[5] er et open source tillegg til Eclipse som integrerer *Subversion*[6] inn i Eclipse IDE. Denne brukes til å holde orden på versjoner og forandringer i de ulike filene i oppgaven opp mot en server

[3] java.sun.com/j2ee/overview.html – Java 2 Platform Enterprise Edition; standard for utvikling av applikasjoner.
[4] www.eclipse.org/

der alle versjonene av alle filene ligger lagret. Når en av oss i prosjektgruppen gjør en forandring i en fil, laster vi opp denne til serveren, og de andre i prosjektgruppen får så tilgang til forandringen gjennom å oppdatere filen fra samme server.

Bruken av Subclipse har fungert bra med unntak av noen feil i oppstartfasen, i forhold til sletting av filer. Det eneste vi har slitt med ellers er tilfeller der flere av oss har jobbet i samme fil uten å ha oppdatert hverandres versjon gjennom server. Dette har medført at Subclipse ikke har villet la oss oppdatere filer før vi har lastet ned nyeste versjon av disse filene fra server.

4.2 Egenutviklet kode

Samtidig som vi har brukt mange klasser og funksjoner fra CarAdmin, og mye av funksjonaliteten Google Maps API versjon 2 tilbyr, har mesteparten av jobben i bacheloroppgaven vært å lage egne filer med kode som skal ta for seg hovedimplementasjonen av kravspesifikasjonen vår. Se figur 19. i design for dette.

4.2.1 Vis ruter

Vi har brukt mye tid på å utvikle muligheten for å vise frem rutene på forskjellige måter, siden det er viktig at brukeren skal kunne ha full oversikt over alle rutene i systemet. Visning av ruter skjer primært i Ruter, se figur 19. Muligheten bruker har i forhold til visning av rutene i tabeller, er datovisning, fri visning og dagsvisning. Ved fri visning får bruker en oversikt over alle frie ruter, altså rutene som ikke er knyttet til en dato, mens ved dagsvisning kan bruker velge å se alle rutene ut fra en bestemt dag, for eksempel mandag. Ved datovisning får bruker se ruter ut ifra dato, og velger så hvilket dato-intervallet han vil se rutene i.

Figur 22 Brukers muligheter til å velge visning av ruter

Denne måten å velge visning av rutene på var originalt ikke en del av kravspesifikasjonen, men på et av møtene vi hadde med ETC kom det frem at de ønsket en slik måte å vise rutene. Siden vi har valgt å jobbe etter systemutviklingsmodellen Scrum, var vi åpene for forslag til forandringer i applikasjonen fra ETC sin side, og arbeidet løsningen inn i neste sprint.

Måten vi har løst dette på er ved hjelp av spesifikke spørringer til databasen. Dette foregår i funksjonen getTableRoute() i RouteIOSQL.java, se vedlegg E. Kode.

Eksempel 1:
For de frie rutene spørres det etter ruter der datoen er 0000-00-00

```
if(mod==2){        //Fri visning av ruter
```

[5] subclipse.tigris.org/
[6] subversion.apache.org/ - Subversion er et kjent open source system for versjonskontroll.

```
      ad= "0000-00-00 00:00:00";
      ed= "0000-00-00 23:59:59";
}                 //Lager spørring som henter alle ruter med tidspunkt fra
                  //og med ad, til og med ed.
String Q = "SELECT route_no, homepoint, tot_stops, stop_time, begin_time,
tot_time, name FROM Routes" + " WHERE begin_time >= '"+ ad + "' AND
stop_time <= '" +ed +"'ORDER BY begin_time ASC";
```

Eksempel 2:
For dagsvisningen av ruter spørres det etter alle rutene i databasen, disse blir sjekket ved hjelp av en if-setning og hvis de er samme dag som det spørres etter ja da vises de.

```
Q = "SELECT route_no, homepoint, tot_stops, stop_time, begin_time,
tot_time, name FROM Routes ORDER BY begin_time ASC";
if(mod==3) {                          //Hvis dagsvisning av rutene
if(start.getDayOfWeek() == day) {   //Hvis datoen er på spesifisert dag
```

Eksempel 3:
For ruter i et datointervall kjøres bare spørringen.

```
String Q = "SELECT route_no, homepoint, tot_stops, stop_time, begin_time,
tot_time, name FROM Routes" + " WHERE begin_time >= '"+ ad + "' AND
stop_time <= '" +ed +"'ORDER BY begin_time ASC";
```

Vi hadde originalt tenkt å implementere denne funksjonaliteten ved en enkel spørring mot databasen, og med funksjoner som sorterte ut ønskelig eller ikke ønskelig resultat, for så å legge dette i tabellen. Vi avslo denne ideen da vi fant ut at MySQL allerede hadde funksjonalitet for å sortere ut ønskede resultater i spørringene mot databasen.

En modifisert versjon av denne rutevisningen brukes for å velge ruter som skal sammenlignes i Sammenligningskartet compare.jsp, og for valg av datoer for turene til en bil i Historiekartet historymap.jsp.

I den første versjonen har brukeren mulighet til å velge om han vil se frie ruter eller ruter tilknyttet en dato, men ingen mulighet til å velge ruter basert på dagen i uka. Grunnen til dette er at ruter som er lagt på samme dag i uka ikke nødvendigvis behøver å ha noen sammenheng, slik som ruter som er laget på samme dato. Der dagsvisning-funksjonaliteten er nødvendig i Ruter for å lage oversikt, vil den i dette tilfellet bare være til forvirring.

Den andre versjonen har bare mulighet til å velge dato for å se de kjørte rutene. Det er her ikke noe som heter frie ruter, og ruter vist på dag ville være av liten interesse siden dette er noe som allerede har skjedd og rutene mest sannsynlig vil være like.

4.2.2 Sammenligningskartet

I Sammenligningskartet, se figur 20, i compare.jsp bruker vi mye av Google Maps API sin funksjonalitet, men før denne i det hele tatt kan begynne å utføre sine oppgaver, så er visse nøkkeldata nødt til å være på plass. Slike data vil være alle rutene det er ønskelig å sammenligne, og alle stoppestedene til disse rutene; disse dataene finnes i databasen. Dette blir mange lag med

informasjon som må organiseres på en så strukturert måte at den enkelt kan finnes frem til igjen når den skal brukes i compare.jsp. Måten dette ble gjort på var med et RoutesBO objekt som inneholdt en vektor med RouteBO objekter. Et RouteBO objekt inneholdt igjen en vektor med StopBO objekter og hvert StopBO objekt inneholdt all informasjon om dette stoppestedet.

Dette ble først prøvd implementert med mange arrayer av heltall, flyttall og tekststrenger, men etter mye rot med å vite hvilken informasjon som hørte til hvilken rute, ble det besluttet at vektorer var et mye bedre lagringsmedium for rutene med alle deres stoppesteder.

Eksempel 4.
Slik hentes informasjon ut fra vektorene for ruter og stoppesteder. Her er det koordinater som hentes ut for å kunne tegne kjøreruten.(Tatt fra compare.jsp)

```
Vector<RouteBO> route = null;       //Lager en vektor for rutene
Vector<StopBO> stop = null;         //Lager en vektor for stoppestedene


Vector<String> ves= new Vector<String>(); //Vekor over adressene til rutene
                                    //for linjene
if(ro != null) {                    //Hvis ruter finnes
      route = ro.getRoute();        //Henter rutene

   String st = null;                //String for adresser/koordinater


for(int j =0; j<route.size(); j++) {      //For alle rutene
 stop = route.elementAt(j).getStops();    //Sett stoppene for ruten
 for(int k =0; k<stop.size(); k++) {      //For alle rutens stopp
    if(k==0) {                      //Hvis k er 0 er dette et startpunkt
      st = "from:
"+stop.elementAt(k).getLat()+","+stop.elementAt(k).getLng();
                                    //Henter koordinater
                  }
     else {                         //Dette er et vanlig stopp på ruten
      st +=" to:
"+stop.elementAt(k).getLat()+","+stop.elementAt(k).getLng();
                                    //Henter koordinater
                        }
       }
ves.add(st);                  //Legger til stringen over adr. til ruten
st=null;                      //Nullstiller var

  }
    }
```

4.2.3 Historiekartet

I Historiekartet har vi laget et kart som er veldig likt Sammenligningskartet, med unntak av at vi her har tegnet ruter basert på hvilken rute-ID bilens punkter er registrert på. Dette ble vanskelig siden vi får inn alle punktene en gitt bil har vært på i rekkefølge basert på tiden han var der, og ikke spesifikt sortert på rutens ID.

Måten vi har løst dette på er å legge alle posisjonene bilen har vært på i arrayer(lengdegrad, breddegrad, rute-ID og fart), og deretter skille ut ruter på rute-IDene når det skal lages ikoner i kartet og tegnes linjer.

Eksempel 5.
Slik skiller vi ruter på rute-id i før vi lager ikoner og linjer i kartet.

```
if(rouID[h-1]!=rouID[h]&& h!=0)      //Hvis h er ulik 0, og to rute-IDer
                                     //etter hverandre er ulike hverandre må
                                     //en ny rute lages.
```

Som en alternativ løsning på dette problemet kunne vi ha brukt samme løsning med vektorer som i Sammenligningskartet, men vi valgte bort denne løsningen siden det her ikke er så mange lag data å holde rede. Dataene vi trenger ligger allerede i en tabell i Historiekartet så i stedet for å lage nye funksjoner som henter alle data på nytt i vektorer, henter vi bare ut data fra tabellene. Dette er betydelig enklere.

4.2.4 Stoppesteder

Under utviklingen har diskusjonen rundt håndtering av rekkefølgen på stoppesteder i en rute vært pågående. I utgangspunktet havnet stoppestedene i den rekkefølgen de ble lagt inn, men dette var ikke holdbart siden ETC ønsket seg mulighet for å flytte et stoppested opp eller ned på listen. Da vi begynte å studere muligheten for å implementere dette oppdaget vi at hvis vi gjorde det slik kunne vi få en rute hvor klokkeslettene ikke stemte overens. Dag tok derfor tilslutt en beslutning om at stoppestedene legges etter tidspunkt, og at kan man legge inn stoppesteder mellom eksisterende stoppesteder.

Måten vi løste dette på var med en spørring til databasen der vi ba om at resultatet som returnertes derfra skulle være stigende sortert på startdato.

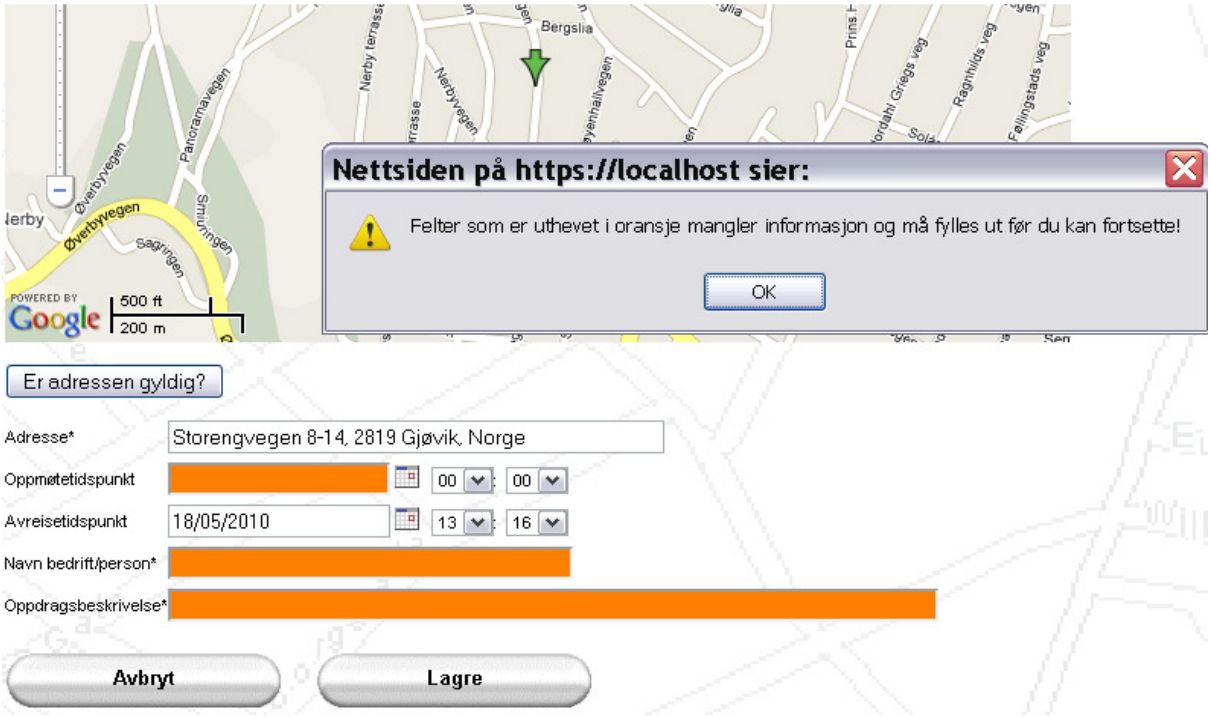
Eksempel 6:
*En slik spørring henter stoppestedene i riktig rekkefølge, legg merke til at spørringen sorteres.(Tatt fram RouteIOSQL(*Se vedlegg E. Kode*) getTableStops())*

```
StringQ = "SELECT * from Stops WHERE r_no=" + r + " ORDER BY meet ASC";
```

4.2.5 Feilsjekking

For å sikre at modulene ikke gir bruker feilmelding og returnerer til startsiden på grunn av feilaktig inntastet eller manglende data i felter, så sjekker vi alle data som er inntastet av bruker før informasjonen blir lagret i databasen. Hvis et eller flere felt ikke er utfylt, eller et valg ikke valgt, så blir disse markert i oransje, og det kommer en feilmelding til bruker om at han må fylle ut feltet/velge før han får mulighet til å lagre.

Måten feilsjekkingen utføres på er at når bruker trykker ”nytt stoppested”-knappen i ChooseCar.jsp eller ”lagre”-knappen i newstop.jsp så blir en funksjon tilkalt, før en eventuell lagring av informasjonen i siden blir gjort. I denne funksjonen blir alle felter i siden gjennomgått, og de felter som ikke er utfylt blir markert og funksjonen returnerer usant uten å gjøre noe lagring av informasjonen. Hvis all informasjon er riktig gitt av bruker returnerer funksjonen sant og en lagring av dataene blir gjort.



Figur 23 Feilsjekk som har returnert negativt svar i newstop.jsp. Noe mangler!

4.3 Gjenbrukt og modifisert kode

Oppgaven vår går ut på å lage to moduler til den eksisterende applikasjonen CarAdmin. ETC ønsket ikke at vi skulle lage funksjoner som eksisterte fra før av på nytt, og vi har dermed gjenbrukt og/ eller modifisert mye av koden fra CarAdmin-applikasjonen. Dette gjelder særlig håndtering av datoer og tidspunkt, tabeller og det logiske i applikasjonen som tar seg av handlingene fra JSP-sidene til Javafiler og tilbake.

4.3.1 DateTime

DateTime.java er en mye brukt Java-fil i modulene våre. Filen brukes primært til å formatere dato og tidspunkt til et ønsket format i en streng, slik at disse kan vises frem på riktig måte, eller lagres i databasen, på det standardformatet som databasen krever. Prosjektgruppen har primært brukt denne i filer som RouteIOSQL(Se vedlegg E. Kode) for å lagre riktig format til database, og i functions.jsp, compare.jsp, realmap.jsp, historymap.jsp, newstop.jsp og add.jsp(Se vedlegg E. Kode) for å vise dem på skjerm.

Eksempel 7.

1. *Det ønskede datoformatet hentes.*
2. *Stringene stop_time og begin_time omgjøres til DateTime type med format av typen som brukes til å hente ut fra database.*
3. *Disse DateTime'ene omgjøres så til Stringer igjen med format df som brukes til å vise frem dato og tidspunkter.*
4. *Stringene legges i hver sin celle for visning i en tabell.*

*(Tatt fra RouteIOSQL(*Se vedlegg E. Kode*) getTableRoute())*

```
LocationSettings ss = ClassFactory.getFactory().getLocationSettings(l);
String df = ss.UI_DATE_FORMAT;            //Setter df til datoformat
DateTime stop = new
DateTime(stop_time,SystemPreferences.get("date_format"));
DateTime start = new
DateTime(begin_time,SystemPreferences.get("date_format"));
                                  //Omgjør DateTime til string og setter i celle
row.addCell(start.getFormatedDate(df));
row.addCell(stop.getFormatedDate(df));
```

Andre nyttige funksjoner fra denne filen, som vi har tatt i bruk, er muligheten for manuelt å kunne styre setting av variablene dag, måned, år, time, minutt og sekund. Dette trengs for å kunne hente ut ruter i datointervaller. Ta for eksempel at bruker vil se alle rutene fra 21/3-2010 til 25/3-2010. I slike tilfeller settes tidspunktene for spørringen til databasen som skal hente rutene fra 00:00:00 til 23:59:59, slik at en skal være sikker på at en får med seg alle rutene i intervallet, uansett når på døgnet de er, så lenge datoen er riktig. Denne muligheten blir også brukt ved opprettelse av frie ruter, dvs. ruter der det ikke blir satt noen spesiell dato. Her settes nemlig datoen manuelt til 0000-00-00.

Prosjektgruppen har laget noen linjer egen kode i denne filen for å kunne vise frem de frie rutene på riktig måte, dvs. at bare tidspunkt delen av DateTime vises. Datoen vil her være lite interessant å se på, siden denne er satt til 0000-00-00. Disse linjene finnes i `exception` nr. 5 i DateTime.java sin `public DateTime(String date, String incDateformat)` –funksjon.

Eksempel 8.
Her kan man se at formatet til datoen skal være HH:mm. Bare timer og minutter skal vises.
(Hentet fra DateTime.java)

```
catch(Exception e5) {
      try {
            sdf.applyPattern("HH:mm");
            d = sdf.parse(date);
      }
```

TableBO, RowBO og CellBO

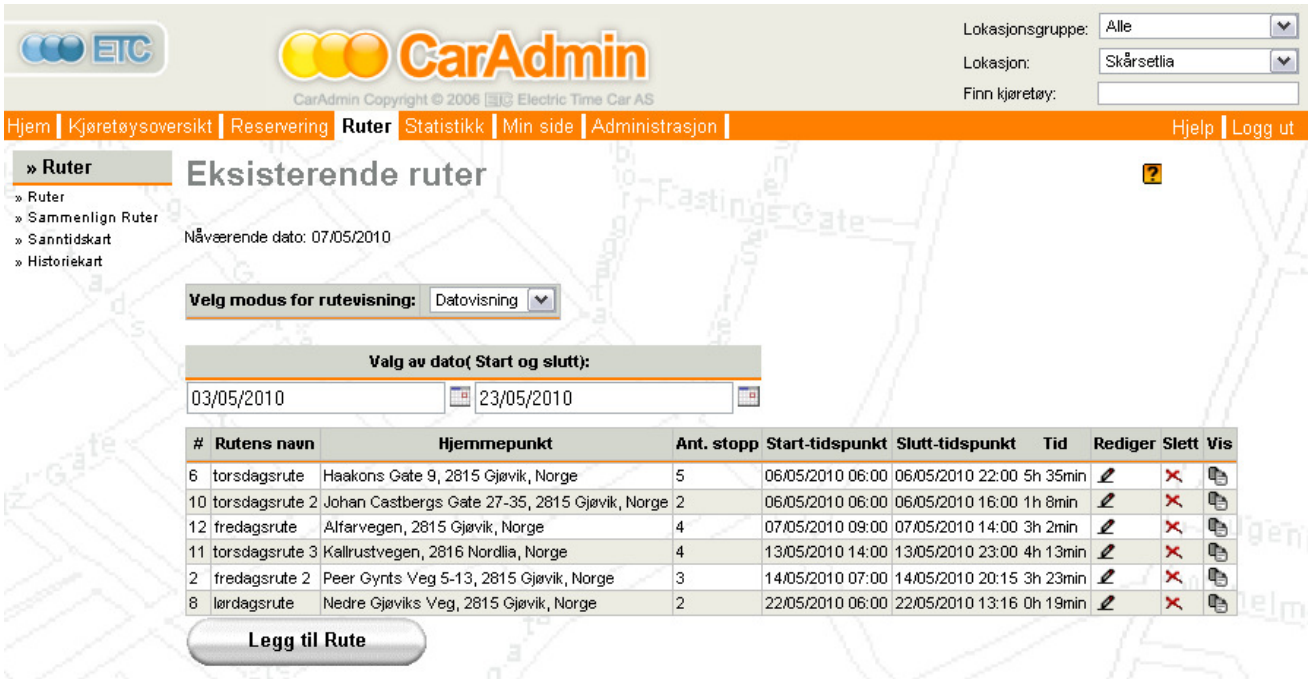
TableBO.java, RowBO.java og CellBO.java er hva ETC bruker til å lage sine tabeller, og vi bruker dermed de samme. Steder der dette er i bruk er i Functions.jsp og Add.jsp(Se vedlegg E. Kode) for visning av tabeller hentet ut av databasen, og i PrepareViewRoute.java, PrepareViewStops.java og RouteIOSQL.java(Se vedlegg E. Kode) for å hente ut informasjon og for å lage tabellene.

Vi kunne alternativt ha laget våre egne tabeller, men dette ville brutt med designet til ETC. Deres tabeller har nemlig et spesielt utseende og dette brukes mange steder i den eksisterende applikasjonen deres.

Eksempel 9.

Slik blir rader og celler i tabellen vist frem på rett sted i tabellen.(Hentet fra linje 414 til 420 i Functions.jsp) Først gås det gjennom alle radene til tabellen, deretter så skrives alle cellene til raden ut.

```
<% for(int j=0;j<table.getSize();j++) {   //For hele tabellen
       RowBO r = table.getRowAt(j);       //Hent rad fra tabellen og vis
       for(int k=0;k<r.getSize();k++) {         //For alle cellene i raden
             CellBO cell = r.getCellAt(k);      //Setter celler
             String link = cell.getLink();
             String cellData = cell.getData();
             String style = cell.getStyleclass();
             if(style == null) { style = ""; }
            String popup = cell.getParamName();%>
            <td><%=cellData %></td>
```



Figur 24 Slik vises tabellen frem. Dette er en tabell over rutene fra og med 3/5-2010 til 23/5-2010 (functions.jsp)

4.3.2 CarPark

CarParkIOSQL.java brukes til å hente ut biler som allerede eksisterer i CarAdmin sin database. Forskjellige biler hentes ut ifra hvilken lokasjon bruker i øyeblikket er registrert på. Denne funksjonaliteten brukes i ChooseCar.jsp for å liste ut bilene som tilhører en spesifikk bilgruppe. Registreringsnummeret til bilene blir også listet ut i realmap.jsp, samt at CarParkIO her også brukes til å hente ut informasjon om bilen som er på tur, selve turen og sjåføren som har turen.

Eksempel 10.

En instans av CarParkIO hentes fra factory, og for hvert av bilgruppeelementene skrives registreringsnumrene til bilene som hører til den aktuelle gruppen ut.

```
                                    //Henter CarParkIO instans fra factory
<% CarParkIO cio = ClassFactory.getFactory().getCarParkIO();
for(int no=0;no<objects.size();no++) {    //For alle bilgruppeobjektene
                    //Henter ett og ett registreringsnummer
 String carS = objects.elementAt(no).toString();
 try {              //Legger regnr til string
   car+=cio.getCarId(Integer.parseInt(objects.elementAt(no).toString()));%>
```

4.3.3 Service

Service.java håndterer alle logiske objekter i CarAdmin. Vår logikk behandles også her, og vi har lagt på vår egen kode bakerst i denne filen. Ting som forespørsler, bevegelser mellom JSP-sider, kjøring av Javakode på et spesielt tidspunkt og ”tilbake”-funksjonalitet håndteres her.

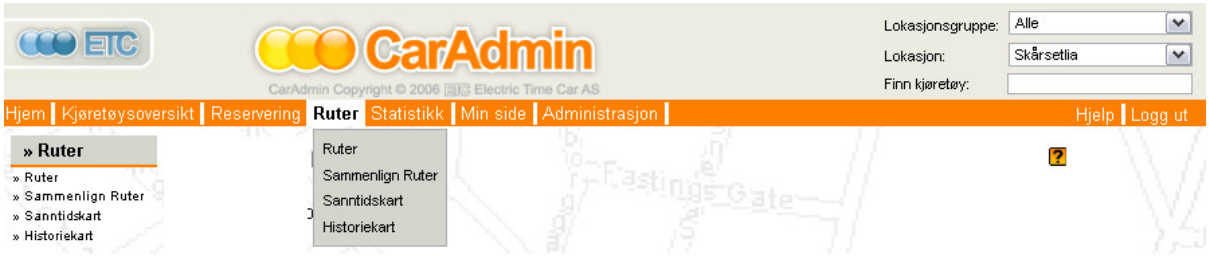
Service.java er en del av flere moduler som for eksempel motoren i applikasjonen. Motoren får inn spørringer som inneholder målfiler og logikk, basert på logikk finnes rett modul og basert på målfilen finnes rett klasse for å ta seg av spørringen. Logikkobjektet traverserer så lista i Service.java til den kommer til en JSP-side, og så går den til dennes oppgitt adresse. Dette en lett måte å knytte .jsp-filer og klassefiler sammen på.

4.3.4 JSP – sider

Java Server Pages[7] eller JSP forkortet, er en Java teknologi som tilbyr mer funksjonalitet mot HTML og XML, og derfor gjør utvikling av nettsider ved hjelp av Java enklere. Vi bruker denne løsningen for å lage våre nettsider, og har opplevd løsningen som enkel og grei å forstå. Her kan vi skrive både, Java, JavaScript og HTML kode i samme dokument, og så lenge vi skriver koden riktig vil ikke dette by på problemer.

Alle JSP-sidene våre er laget med utgangspunkt i en mal av en typisk JSP-side for ETC CarAdmin. Det vil si at i disse sidene ligger koden for deres menyer og funksjonene tilknyttet disse. Slik har vi gjort vårt brukergrensesnitt så likt deres som mulig.

For at vår del av CarAdmin skal gli best mulig inn i helheten til applikasjonen har fire poster i menyen Ruter blitt dedikert til våre Ruteplanlegger- og Flåtestyrings - moduler. Dette er henholdsvis Ruteplanlegger: Rute og Sammenlign Ruter, og Flåtestyring: Sanntidskart og Historiekart.



Figur 25 Standard brukergrensesnitt og meny for CarAdmin, våre sider har også en slik meny

4.3.5 Språkfilene

Språkfilene bygger videre på de som allerede eksisterer i CarAdmin. Her finnes det en fil for norsk, en for svensk og en for engelsk. Vår oppgave har vært å fylle ut språkfilene for norsk og engelsk, men ikke for svensk, da de ikke har så stort behov for denne.

Måten språkfilene implementeres på er at vi har "kodeord" som lagres i begge disse filene, disse har en norsk oversettelse og en engelsk oversettelse. Kodeordene brukes så alle steder i .jsp eller .java filene der den spesifikke setningen eller ordet kodeordet inneholder skal stå. Dette hentes ut ved setningen `m.getString("compare.hide")`, hvor `compare.hide` er kodeordet. Dette kan finnes igjen i språkfilen med linjen `compare.hide = Vil du skjule kartet?.` For å styre om ordene kommer fra den engelske eller den norske språkfilen må en administrator manuelt sette den spesifikke konfigurasjonen av CarAdmin til å være på et av de to språkene. Konsekvensene av dette

[7] www. java.sun.com/products/js

er at CarAdmin er tospråklig, og at administrator på den enkelte lokasjonen selv kan sette opp hvilke brukere som skal ha hvilke språk i sin løsning.

Hvis et engelsk ord er vesentlig lengre enn et norsk ord eller motsatt, og ordet skal beskrive en knapp vil knappen automatisk forandre størrelse slik at knappen passer til ordet. Slike sikringsmekanismer er inkorporert i funksjonen som brukes til å lage knapper. Denne funksjonen har ETC laget, og vi bruker den til å lage alle våre knapper. I andre tilfeller av oversettelser av ord finnes ingen slike sikringsmekanismer, men språkfilene er oversatt til å bety det samme og til å ha ca. samme lengde på setningene slik at dette ikke skal bli noe problem.

4.3.6 Calendar Popup

CalendarPopup.js er laget i JavaScript(forkortet js), og inneholder funksjonaliteten som gjør at datoer kan velges i kalendere som vises når en trykker på ikonet for kalender. Vi bruker dette i newstop.jsp, historymap.jsp, functions.jsp og compare.jsp. CalendarPopup.js og alle funksjoner som hører til kalenderen er laget av Matt Kruse[8] og er et gratis JavaScript for nedlastning. Alle har rett til å bruke dette.

Figur 26 Matt Kruse sin kalender

4.4 Google Maps API

Google driver kontinuerlig forbedring og videreutvikling av sitt Maps API og for øyeblikket er versjon 2 den nyeste implementasjonen deres av kartverket. Det er også denne versjonen av Google Maps API vi bruker. Versjon 3 av Maps API skal være tilgjengelig for utviklere snart, men vi har ikke gjort videre undersøkelser i forhold til denne.

4.4.1 Tilgang og lisenser

For å få tilgang til Google Maps sitt API må utvikler registrere url til sidene der kartet skal tas i bruk, samt e-postadresse og diverse annen nødvendig informasjon. Da dette er gjort blir utvikler tildelt en kode, som må legges inn på de sidene i applikasjonen hvor kartet skal vises eller brukes.

Ifølge lisensen som hører med til Google Maps API er det gratis for utviklere som skal bruke dette i applikasjoner og nettsider som er tilgjengelig for alle på internett. Hvis innsynet til kartet blir begrenset av for eksempel passordgodkjenning, må bruker betale for Google Maps API. Vi som

[8] www.mattkruse.com/ - Nettside for all informasjon om koden i CalendarPopup.js

utviklere av et offentlig skoleprosjekt, behøver derfor ikke å betale Google noe, mens ETC, som skal bruke kartet i kommersiell sammenheng på en restriktiv side, må betale for dette.

4.4.2 Vår bruk av Google Maps API

Vi har tatt i bruk Google sine kart i både Ruteplanlegger og Flåtestyring det vil si i filene add.jsp(Se vedlegg E. Kode), showRoute.jsp, newstop.jsp, compare.jsp, realmap.jsp og historymap.jsp, se figurene 19, 20, 21 og 22 i kapittel 3.4 Filstruktur. I add.jsp bruker vi kartet for å gi Bruker en oversikt over selve ruten han har laget til nå, i showRoute.jsp er kartet identisk til det i add.jsp, mens newstop.jsp sitt kart gir Bruker muligheten til å trykke inn stoppestedet rett på kartet. Kartet i compare.jsp viser mange ruter og stoppesteder i samme kartet, sanntidskartet realmap.jsp viser markører der hvor bilene til brukeren befinner seg i øyeblikket, mens Historiekartet historymap.jsp er veldig lik Sammenligningskartet compare.jsp viser oversikt over rutene og stedene utvalgte biler har vært. Her ser vi flere muligheter for videre arbeid, omtalt i avsnitt 6.3.

For å implementere dette har vi brukt et lite utvalg av den funksjonaliteten Google Maps API tilbyr sine brukere.

Geokoding

En sentral del av vår oppgave har vært å få adresser Bruker skriver inn i modulene vist ut på et kart. For å få til dette må den aktuelle adressen gjøres om til koordinater. Å gjøre om en adresse til koordinater kalles *geokoding*. Funksjonene som gjør dette mulig i Google Maps API heter getLatLng() og getLocations(), disse funksjonen er stort sett like, den eneste forskjellen er at getLocations returnerer noe mer informasjon, slik som status for konverteringen. Mens GetLatLng tilkaller getLocations og henter bare ut koordinatene herfra.

Geokoding har vært ett av de vanskeligste momentene ved Google Maps. Det er ikke vanskelig å bruke det, men det er vanskelig å forstå begrensningene Google har lagt på disse funksjonene, og ikke minst å løse problemene rundt dette når de oppstår. For det er ikke enkelt å vite at det er nettopp geokodingen som forårsaker feilen.

I Sammenligningskartet compare.jsp ville vi geokode adressene til alle de valgte rutenes stoppesteder rett etter hverandre. Dette gikk bra så lenge det var færre enn ti adresser å geokode, men så fort tallet oversteg ti, fikk vi en feilmelding der det stod at Google Maps ikke hadde funnet koordinatene, og derfor bare returnerte null. Vi forstod at for en eller annen grunn kunne ikke Google returnere riktig koordinater til oss, men vi viste ikke hvorfor dette skjedde.

Vi prøvde ut flere alternativer for å løse problemet. Først prøvde vi å bytte ut funksjonen getLatLng () med getLocations(), men dette fungerte ikke noe bedre med over ti geokodinger, og problemet vedvarte. Deretter så vi på løsninger for å statisk geokode adressene, før vi i det hele tatt laget kartet, med en geokoder fra en annen leverandør enn Google. Men vi var sterkt i tvil om disse leverandørenes geokodinger var av like god kvalitet som Google sine, og om de ville gi like eksakte markeringer i kartet som vi ønsket. Til slutt fant vi andre forslag [9]til løsning på internettfora, som tok

[9] http://econym.org.uk/gmap/geomulti.htm

for seg Google Maps API og nettopp dette problemet. Her fant vi ut at getLatLng() og getLocations() ikke var en del av løsningen, men hovedproblemet vårt.

Problemet med geokodingen er at dette er asynkrone funksjoner. Siden asynkrone hendelser kjøres uavhengig av flyten til hovedprogrammet, vil dette forårsake problemer hvis flere geokodinger kjøres rett etter hverandre. Dette var tilfellet i compare.jsp. Google Maps sin geokoder takler nemlig ikke at flere enn ti adresser geokodes samtidig av den grunn at noen av resultatene fra geokodingen da ikke er ferdig konvertert før hovedprogrammet ønsker å bruke dem. For å løse dette problemet bestemte vi oss for å utføre geokodingen av adressen med det samme denne blir lagret i newstop.jsp, og så lagre disse koordinatene i databasen.

Koordinatene ble så hentet ut av databasen når vi trengte å plotte ruter i kartet i compare.jsp. På denne måten vil det bare bli kjørt en geokoding om gangen og vi unngår problemet. Ulempen med denne løsningen er at vi har måttet lage ny kode for å få med koordinatene i AddStop.java og EditStop.java, samt utarbeidet en ny metode for hvordan redigering av stoppesteder skal foregå. Metoden vi har brukt er at koordinater genereres på nytt hver gang et stoppested skal redigeres. Vi ser allikevel på denne løsningen som den beste, siden den lar oss beholde Google Maps API sin geokoder.

Ikoner og linjer i kartet

For å illustrere forskjellen på startsteder, stoppesteder, endepunkter, biler, ruter og hvor bruker har klikket i kartet, har vi tatt i brukt flere typer ikoner og ruter. Utformingen på de ikonene vi har brukt nedenfor er de standardene Google har for sine egne ikoner, heriblant er G_DEFAULT_ICON, G_START_ICON, G_PAUSE_ICON og G_STOP_ICON.

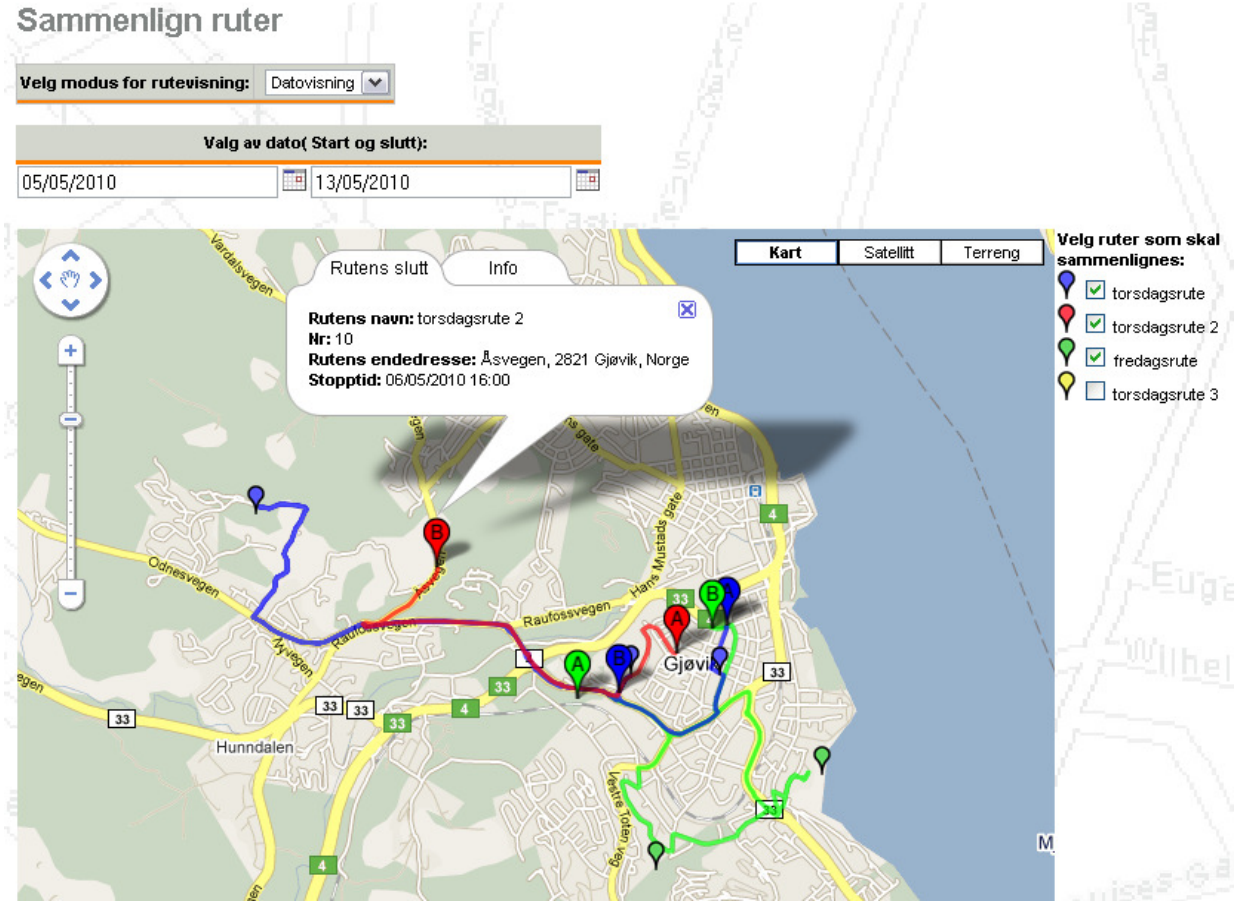
Figur 27 Startikon, endeikon og pilikon for ruter, disse brukes i realmap.jsp og newstop.jsp

For å lage kart med ruter og markører som representerer forskjellige steder og biler, trengte vi litt mer enn Google sine standardikoner. Dette gjelder særlig i sammenligningskartet og i sanntidskartet der flere ruter, og deres tilhørende ikoner, skal vises. Løsningen ble å gi dem forskjellige farger for kunne skille mellom dem.

Vi har ved hjelp av nettsider, som

`http://chart.apis.google.com/chart?chst=d_map_pin_letter&chld=B|0000FF|000000` for å generere startikoner og stoppikoner i forskjellige farger (sett ønsket bokstav etter chld og den ønskede fargen sin HEX-kode etter |) og ved hjelp av `http://labs.google.com/ridefinder/images/mm_20_purple.png`, laget de små markørene for stoppestedene. Ikonene vi har brukt ligger som .png bilder åpent på nettet og er tilgjengelig for alle som bruker Google Maps API.

Figur 28 Ikoner for rødt startsted, blått stoppested og grønt endepunkt i Sammenligningskartet, (compare.jsp)



Figur 29 Sammenligningskartet (compare.jsp)

Her vises tre ruter i sammenligningskartet, hver rute har sin egen farge på ikonene sine og på selve linjen som representerer ruten. Når bruker beveger peker over en av rutenes linjer så utheves denne ruten med en bredere linje. Dersom bruker klikker på et ikon kommer informasjon om selve stoppestedet frem i et lite vindu. Se for eksempel på torsdagsrute 2 som her har et informasjonsvindu oppe. På høyre side kan bruker velge hvilke ruter som kan sammenlignes, her er blå, rød og grønn ruter valgt til sammenligning, mens den gule ikke er valgt.

Tegning av rute

En annen utfordring i Google Maps ble å få tegnet selve ruten for bilen. Dette dreide seg primært om kartet i add.jsp(Se vedlegg E. Kode), som skal vise ruten som er laget til nå med start, stoppesteder og endepunkt.

Siden dette var den første kartdelen som ble implementert, var den også den mest utfordrende og lærerike, i forhold til kartdelene som fulgte etter. For å lære fremgangsmåten for først å lage et kart og deretter en rute, ble det først benyttet et enkelt eksempel fra Google[10] . I dette eksempelet er det bare å oppgi startsted og stoppsted, og den tegner ruten i mellom. Å få inn flere stoppesteder for en rute i denne koden ble imidlertid en utfordring, siden vi fortsatt hadde minimal kunnskap om hvordan kartet fungerte. Derfor fortsatte vi å se etter et passende eksempel. I Google Maps guiden til

[10] http://code.google.com/intl/no-NO/apis/maps/documentation/examples/directions-advanced.html

Mike, en guru på Google Maps API, fant vi et slikt eksempel[11]. Her hadde vi noe som endelig håndterte stoppesteder mellom *startpunkt* og *endepunkt*, og dette virket derfor som et passende eksempel for oss å gjøre bruk av.

Eksemplet håndterte flere stoppesteder, men det hadde også med aktive og inaktive stoppesteder noe vi ikke trengte. Dette gjorde overgangen til vår kode litt mer komplisert, på grunn av variablene ”active” og ”state”. ”State” er hvor langt den hadde fått informasjon om ruten, men i vårt bruk skal alt dette være innlagt på forhånd. Variabelen ”active” var om stoppestedene skulle være med i beregningen av ruten. Forsøket på å eliminere disse fra koden, uten å endre funksjonaliteten for resten av ruten, endte med en rute som ga veibeskrivelse helt frem, men som ikke ville tegne opp ruten lengre enn til nest siste stoppested. Koden så heller ikke direkte pen ut, etter en slik utrenskning. Vi tok derfor kopi av den gamle koden, for ikke å miste det vi alt hadde fått til, slettet alt i dokumentet og begynte på nytt. Den nye koden ble bedre, med en god del linjer fra det gamle eksempelet og en ny forståelse av hvordan dette faktisk fungerte. Etter noen indeksfeil på arrayer, som vi forsøkte å lese mer ut av enn de faktisk inneholdt, fikk vi endelig ruten ut i en versjon vi var vesentlig mer fornøyd med. Vi hadde klart å lage vårt første kart med en rute nesten helt selv.

4.5 Hjelp

I vår utvikling av Ruteplanlegger- og Flåtestyringsmodulene har vi, som relativt uerfarne programmerere, hatt en del problemer. Det var en stor utfordring for oss å begynne rett på et utviklingsprosjekt i Java, JavaScript, HTML og MySQL, uten å ha noen erfaring i dette fra liknende prosjekter. Av den grunn gikk utviklingen utrolig tregt i starten av utviklingsfasen, og vi trengte mye hjelp til å forstå feilene vi gjorde underveis. Samtidig ga feilsøking i Google Maps API oss mye hodebry, siden denne ikke viser feilmeldinger når noe er galt.

4.5.1 Problemløsning

Vi har fått god hjelp fra vår oppdragsgiver ETC med alle feil som omhandlet CarAdmin, og kodeproblemer rundt dette. Dette vil ikke si at ETC har utført rettinger eller kodet løsninger for oss, men at de har gitt oss tips om hvordan feil kan løses, vist oss beste metoden å utføre kodingen på, og støttet oss med gode svar på våre spørsmål underveis.

ETC kunne ikke hjelpe oss så mye med Google Maps sitt API og våre problemer rundt dette, siden de aldri har brukt dette kartverktøyet før. Løsningen på disse problemene ble i hovedsak funnet gjennom mye søking på internett etter poster og diskusjonsforum om andre som har hatt like eller lignende problemer. Google Maps API har også en egen hjemmeside[12] som er laget for utviklere som bruker deres API, og her finnes både kodeeksempler og dokumentasjon over funksjoner, variable, konstanter, og forslag til bruk. Denne siden har vært en enorm ressurs for oss, og virkelig vist oss hvordan vi kan bygge opp et kart helt fra bunnen av, og hvordan strukturen i koden må være for at kartet skal fungere optimalt, siden mange av funksjonene til kartet er asynkrone.

[11] http://econym.org.uk/gmap/example_multi2.htm - Se vedlegg E. Kode

[12] http://code.google.com/intl/no/apis/maps/

Google Maps API er et populært kartverktøy og det finnes også flere guider[13], utenom Google sin egen, på internett over hvordan funksjoner kan utføres i dette kartet. Vi har også tatt i bruk mange av disse for å lage våre egne kartfunksjoner i applikasjonene våre.

4.5.2 Feilsøking

Feilsøking i koden vår har primært blitt utført når programmet har gitt fra seg en feilmelding eller når et kart som skulle ha blitt vist, har blitt borte. For å finne feil har vi benyttet oss mye av konsollvinduet i Eclipse og utskrifter til denne, men vi har også brukt *alerts*.

Nullpekerfeil er for eksempel en svært vanlig feil vi har hatt i våre moduler, og denne kommer av at en variabel vi tror har blitt satt til en spesifikk verdi forblir null. For å finne slike feil og lignende skriver vi ut variable til konsollvinduet med `System.out.println("tekst:" + variabel)` for å se hvor feilen oppstår. Deretter må vi selv finne ut av hvorfor feilen oppstår. Feil i Google Maps API gir ikke fra seg noen feilmelding annet enn at kartet ikke kommer til syne og dette gjør at feilsøking i kartfunksjonene blir mye vanskeligere. For å løse slike feil bruker vi derfor alerts til å fortelle oss hvor langt koden klarte å kjøre før det oppstod en feil, og vi prøver å nøste opp problemet derfra.

[13] http://econym.org.uk/gmap/geo.htm

5 Testing og kvalitetssikring

5.1 Tekniske tester

For å teste at modulene våre holder de kvalitetene vi har spesifisert i kravspesifikasjonen, slik som brukervennlighet, og at all vår kode fungerer etter de gitte spesifikasjonene i Use casene har vi gjennomført omfattende tester av Ruteplanlegger og Flåtestyrings modulene.

Underveis i sprintene har kode blitt testet etter at hver funksjon har blitt ferdigstilt, slik at vi hele tiden har visst konkret hva som fungerte og hva som ikke fungerte. På slutten av hver sprint har det også vært en større test, i samarbeid med ETC, for å teste om applikasjonene inneholder den funksjonaliteten ETC ønsker.

På slutten av oppgaven har vi tre i prosjektgruppen holdt en stor test av Ruteplanlegger og Flåtestyring i alle de nettleserne vi har spesifisert at modulene skal fungere i. Her har vi testet at alt ser likt ut, at alt fungerer i alle nettleserne, at ”hjelp”-knapper vises riktig, at data lagres til riktig sted i databasen på riktig format, at feilsjekkingene våre fungerer, at knappene fungerer og at ”avbryt”-knappene går tilbake til riktig sted i modulene. Dette kan vi nå si med sikkerhet at fungerer.

Siden ETC har oppgradert utseende til CarAdmin, og dette også har blitt implementert i våre moduler, fikk vi en del kompatibilitetsproblemer fra nettleser til nettleser. Modulene våre fungerte strålende i Opera, Safari og Google Chrome, men vi fikk en hel del ukjente feilmeldinger ved bruk av Mozilla Firefox og Internet Explorer. Disse feilmeldingene omhandlet at vi prøvde å hente ut elementer basert på id, men verken Firefox eller IE klarte å finne disse elementene. Dette kom av at vi hadde gitt alle våre elementer navn, men vi hadde ikke gitt alle id. De andre nettleserne godtok navn som det samme som id, men både IE og Firefox krever en egen id på elementer. Dette har vi nå rettet opp, og modulene våre fungerer i alle disse nettleserne.

Når vi kjører CarAdmin-applikasjonen i nettleseren Internett Explorer vil det komme opp advarsler før hvert kart vises, og vi får ikke gjort noe med dette i vår del av applikasjonen. Grunnen til dette er at en sikker nettside prøver å vise usikkert innhold, som er kartet. Denne advarselen vil ikke komme opp når applikasjonen kjøres fra ETC, siden de har betalt Google for retten til å bruke kartene, og det innholdet som før var usikkert nå vil komme på HTTPS-form, altså sikker form.

5.2 Utførte brukertester

For å teste om våre moduler er brukervennlige, enkle å forstå, at de ikke feiler under vanlig bruk ,og at de har en naturlig flyt i måten funksjonene våre er implementert på, har utvalgte personer med gjennomsnittelig erfaring med datamaskiner testet modulene. Disse testene vil sikre at kvaliteten på modulene skal tilfredsstille potensielle brukere.

Vi spør våre testere om de opplever applikasjonen som treg, om applikasjonen er enkel å forstå i både oppbygging og flyt, om tester har noen tips til hva som kan forbedre brukeropplevelsen, og hva tester tror han vil få mest bruk for. I tillegg oppfordret vi våre testere til å fortelle oss hvis noe virket ulogisk, hvis det var noe i modulene de ikke forstod, eller om det var noe som trakk deres brukeropplevelse ned.

Vår første tester er ansatt i hjemmetjenesten og bruker ikke pc i jobbsammenheng, men kunne tenke seg å ha en applikasjon som CarAdmin med Ruteplanlegger og Flåtestyring på jobben.

Test 1.

Opplever du applikasjonen som treg?	Nei, absolutt ikke. Dette er en stor forbedring fra kommunens system.
Som enkel å forstå?	Ja, dette programmet og oppsettet virker lett forståelig og logisk. Knapper, valg og tabeller er godt navngitt og gangen i applikasjonen er naturlig.
Har du tips til forbedring?	Mulighet for å skrive ut vising av rute. Mulighet for å gjemme veibeskrivelsen i visning av rute.
Hva vil du få mest brukt for i modulene?	Tiden hver rute tar å gjennomføre, hvor lang tid det tar å komme seg til et sted, hvor lang tid det skal brukes på et visst sted, hva som skal gjøres på stedet, veibeskrivelsene, og sanntidskartets mulighet til å finne nærmeste bilen til en viss adresse.

Brukeropplevelsen trekkes litt ned av at det fortsatt finnes feil ved lagring av en rute og at det kommer opp uforklarlige feilmeldinger ved klikking i kart og kalender.

Oppsummering: Første tester virker fornøyd med applikasjonen, og påpeker momenter vi ikke har tenkt på før, sånn som mulighet til å gjemme kjørebeskrivelsen og skrive ut all informasjonen om en rute.

Test 2.

Vår andre tester er en av kontaktpersonene våre fra ETC. Han har kjørt gjennom våre moduler for å se etter om vi har igjen noen store stygge feil. Siden vi allerede har fått tilbakemelding på de spørsmålene vi stiller våre vanlige testere slik som tips til forbedringer og lignende, kommer vi her til å fokusere bare på eventuelle feil. Hans tilbakemelding var følgende:

Feil:

#1
a. Rediger en rute
b. Rediger stoppested
c. Trykk avbryt
d. Trykk legg til stoppested
→ Detaljer for redigert stoppested ligger fortsatt i ”session”

#2
Nullpeker når jeg trykker lagre på rediger stoppested: Jeg er usikker på

akkurat HVA jeg må gjøre for at dette skal skje
WARN 102326,684 Next nullpekerfeil!

Oppsummering: Vi har tatt feilene til følge og rettet de opp etter beste evne. Den første feilen rettet vi ved å avslutte sessions når avbryt knappen får bekreftet at bruker vil avbryte. Den andre feilen gjaldt at koordinater ikke blir generert for en adresse hvis bruker ikke forandrer den foreslåtte adressen ved oppretting av et nytt stoppested, eller ved redigering av stoppested. Dette ga en nullpekerfeil siden stoppestedene som skulle lagres ikke fikk medsendt noen koordinater. Dette løste vi ved å implementere generering av koordinater uansett om adressen er ny eller ikke. Ulempen med dette er at det skjer flere geokodinger enn nødvendig, men vi fant likevel ut at dette er den beste løsningen da problemet gjelder både nye stoppesteder med forslag til adresse og gamle stoppesteder som allerede har en adresse. Et alternativ ville vært å ha en rutine for geokoding for de nye adressene, og en rutine for å hente koordinatene til de gamle adressene fra databasen. Vi valgte å bruke den ene løsningen som løste begge problemene.

Test 3.

Den tredje testen har blitt gjennomført av en mindre rutinert databruker, og av den grunn er det ikke med noen forslag til forbedringer eller svar på ”hva du vil få mest bruk for i modulene…”

Testet:	Ruteplanlegger og Flåtestyring
Opplever du applikasjonen som treg?	Nei, jeg opplever applikasjonen som flytende.
Som enkel å forstå?	Overraskende nok er dette programmet enkelt og smart i bruk, jeg trodde det ville være mer komplisert. Så fort jeg fikk satt meg ned og sett på applikasjonen og lest ordentlig hva det stod på knapper og overskrifter etc. var det ikke noe problem å finne frem eller bruke modulene Ruteplanlegger og Flåtestyring. Selv om jeg ikke har nevneverdig erfaring med slike programmer fra tidligere.

Oppsummering: Tester liker programmet, og synes det var enkelt å lære seg å bruke det. Selv for personer som ikke bruker datamaskin til daglig.

Test utført av Astrid K. Dalby.

6 Avslutning

6.1 Resultater

Da vi begynte på oppgaven, fikk vi avklart med ETC at vi skulle lage våre moduler direkte implementert i deres CarAdmin-applikasjon og at vi skulle bruke en trelagsstruktur. Dette anbefalte de for å spare oss for noe arbeidsmengde og for å holde kompleksiteten i oppgaven på et riktig nivå for oss.

Hvis vi skulle ha laget vår versjon av innlogging, brukerhåndtering, datobehandling og håndtering av dataflyt mellom de forskjellige lagene ville ETC hatt lite utbytte av vårt arbeide, siden dette da ville blitt dobbelkodet. De ville også sittet igjen med vesentlig mindre av den funksjonaliteten de ønsket i våre moduler slik som Sammenligning av ruter, Historiekart og Sortering av ruter på dato siden vi da hadde fått mindre tid til dette. På den annen side har vi ikke fått prøvd oss på implementering av slike metoder som nevnt over, men vi kan likevel si med sikkerhet at vi har hatt mye å bryne oss på i oppgaven.

Kodingen opp mot Google Maps sitt API har vært en spennende oppgave. Ingen av oss hadde forsøkt dette tidligere, og vi måtte derfor sette oss inn i dette helt fra begynnelsen. Mye bruk av Google Maps-dokumentasjonen[14], og forskjellige andre fora om Google Maps API, dannet et godt utgangspunkt for oss til å prøve å feile med dette verktøyet. Dette ble likevel en utfordring for oss p.g.a. tidspresset, hvordan kartverket i følge kravspesifikasjonen vår skulle fungere sammen med øvrig kode. Programmering opp mot et API fra en ekstern aktør forårsaket også usikkerhet omkring hvordan dette så skulle kobles sammen med andre funksjoner. Vi kan derfor med stor sikkerhet si at Google Maps API var den mest tidkrevende delen av oppgaven, ikke minst fordi det som tidligere nevnt, se kapitlene 4.4 og 4.5.2 ikke finnes noe godt feilsøkingsverktøy for API'et.

En overraskende stor del av tiden avsatt til programmering ble brukt til å implementere modulene våre inn i CarAdmin og implementere funksjoner fra CarAdmin inn i våre moduler. Å få til en sømløs funksjonalitet rundt dette er viktig, slik at bruker ikke merker om han er inne i en av de modulene vi har laget eller en av de originale modulene fra CarAdmin, eller om det er funksjoner fra CarAdmin eller våre filer som blir kjørt. I tillegg til dette ble det brukt mye tid på studering av den koden ETC hadde fra før, finne ut hva vi ønsket å gjenbruke herfra, hva vi kunne modifisere og hva vi måtte lage fra bunn av selv.

Vi begynte utviklingen med å skrive egen kode, og det var det bra vi gjorde, slik at vi fikk et godt tak på de nye programmeringsspråkene vi skulle bruke før vi måtte begynne å håndtere utenforstående API'er og kode. Ut i fra hva vi har beskrevet i de personlige loggene våre har det gått med mye mer tid enn det vi hadde avsatt i begynnelsen til å fikse usle småfeil i koden vår. Den første sprinten vår ble utvidet med sprint 1.5 slik at vi skulle bli ferdig med sprintloggen for perioden. Utover i loggene ser en godt at vi har utviklet oss som programmerere underveis i oppgaven. Oppgaver som i begynnelsen kunne ta en uke ble i senere perioder gjort på en dag eller mindre.

6.2 Kritikk av oppgaven

Da vi startet på oppgaven i januar, tok det noe tid før det ble avklart hva vi egentlig skulle lage som modul nummer to. Vi var derfor litt sene med å komme i gang med kravspesifikasjonen, som vi i det

[14] http://code.google.com/intl/nb-NO/apis/maps/documentation/index.html, sist 18.5.2010

minste kunne ha påbegynt for den første modulen. Oppgaveteksten ble klarlagt innen vi skulle levere forprosjektet i slutten av januar, og vi tok opp arbeidet med kravspesifikasjonen og designet for fullt. 3. februar begynte vi på første sprint, men siden vi hadde fått koden fra ETC kun dager i forveien, hadde vi ikke rukket å sette oss så godt inn i denne. Her kunne vi ha strukturert oss bedre i jakten på klassene og filene vi hadde størst behov for. Dette fant vi i stedet ut av underveis i kodingen. Google Maps sitt API kunne også med fordel vært utforsket litt mer på et tidligere stadie. Vi fikk tidlig opp et lite kart i en testside, og ble såpass fornøyd med det vi hadde prestert, at det tok litt lenger tid enn nødvendig med tanke på videre utforskning.

Første sprint startet med godt mot og store ambisjoner, men mangelen på kjennskap til koden, gjorde behovet for en ekstra sprint gjeldende. Oppstarten gikk med andre ord noe senere enn vi selv forventet, men ETC var allikevel godt fornøyde med det vi hadde. Det ble mindre å presentere enn vi hadde planlagt, men vi fikk i alle fall presentert noe i slutten av første sprint. Senere utover våren har vi levert det som ble lovet for sprintene, selv om noe har blitt forbedret i flere runder, ettersom Dag og Øyvind har kommet med forslag til forbedringer og ekstra funksjonalitet, særlig i forhold til brukerhåndtering. Vi ser her fordelen av å jobbe med en smidig metode.

I forhold til rapportskriving hadde vi planlagt å begynne med dette så tidlig som mulig, slik at vi slapp å ta et skippertak med fristen hengende over oss. Dette gikk ikke helt som vi hadde håpet. Vi fant ut at vi skulle vente med rapporten til etter Høgskolelektor Frode Haug sitt kurs for rapportskriving. Men det var mye å gjøre på koden, og det var stadige problemer med å vise programmet i de ulike nettlesere som ble forårsaket av det nye grensesnittet til CarAdmin, forbedring av kode og ikke minst feilretting. Rapportskrivingen ble dermed ikke påbegynt så tidlig som vi ønsket, men koden ble levert i tide, med mange av endringene og ønskene ETC hadde kommet med underveis.

Selve produktet vårt; modulene Ruteplanlegger og Flåtestyring lider av lite feillogginger i forhold til hva Bruker taster inn i tekstfelter. Her kunne vi ha implementer metoder for og fange opp slike feil og logget dem før vi sendte data videre til metodene som tar seg av lagring til databasen, men vi ble ikke klar over dette før sent i utviklingsperioden og derfor er ikke dette med. Det samme gjelder renvasking av tekst som Bruker skriver inn i tekstfelt, slik at det ikke kommer inn SQL-setninger her som prøver å kjøre ulovlige skript. ETC har for så vidt senere fortalt oss at vi skulle nedprioritere dette siden vi antar at kunde, når han først er logget inn, er en ordentlig bruker, og ikke bevist ønsker å ødelegge noe.

Vi er selv meget godt fornøyde med resultatet av arbeidet, med hensyn på utgangspunktet vi hadde. Vi ser allikevel at det er flere punkter som kunne vært forbedret, eller gjort annerledes, og har omtalt disse under neste punkt; 6.3 Videre arbeid.

6.3 Videre arbeid

Underveis i utviklingen av modulene har vi funnet noen punkter som kan utvides i forhold til hva vi har rukket å implementere. Noen av punktene har vært forslag fra veileder eller oppdragsgiver, mens andre har vi sett selv at kan videreutvikles. Siden vårt fokus har vært på utvikling av funksjonalitet, er det flere muligheter for forbedring av brukergrensesnitt. Dette gjelder spesielt å legge lenker til andre funksjonaliteter i programmet på steder det er naturlig å ha dette. Eksempel på dette kan være å kunne redigere et stoppested direkte fra boblene som kommer opp når man trykker på et stoppested i kartet i add.jsp(Se vedlegg E. Kode), eller å kunne trykke i kartet, og det kommer

mulighet for å lage et nytt stoppested her og lignende. Av utvikling av funksjonalitet ser vi blant annet behovet for å kunne endre hvilken vei man ønsker å kjøre mellom to stoppesteder.

I veibeskrivelsen kan også navn og gjøremål for stoppestedet legges inn, sammen med adressen, hvor det står at stoppestedet skal være. I siden for nytt/rediger stoppested kan ruten med fordel tegnes opp i kartet, slik at bruker ser hvor langt i ruten man har kommet. Her kan han også få opp hvor lang tid det vil ta å kjøre fra det forrige stoppestedet til det nye, eventuelt også til neste stoppested, dersom det nye/redigerte stoppestedet legges mellom to eksisterende. En jobb vi anser til å være noe større, vil være å implementere muligheten for Google sin Street View-funksjonalitet, slik at en bruker som ikke er kjent i området lettere kan finne frem til riktig hus, ved å ha sett på gatebildene for stedene.

Da en ny rute legges inn, velges nå en bilgruppe. Her er det mulighet for å sammenkoble dette til å reservere en enkelt bil for ruten, med reservasjon fra første avreise, til og med siste stoppested.

For å forenkle brukergrensesnittet mer, kan også siden med nytt/rediger stoppested bli fjernet, og funksjonaliteten flyttes direkte inn i tabellen og kartet som finnes i add.jsp. Dersom dette gjøres, kan man i sammenligne ruter legge inn funksjonalitet for å slå sammen ruter ved hjelp av denne omgjøringen. Uten denne er jobben med å slå sammen ruter automatisk for tungvint, og må gjøres manuelt.

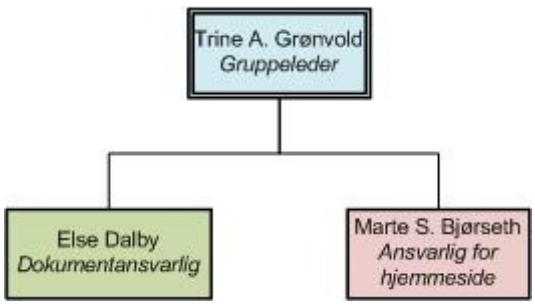
Ved å overføre rutene til GPS - og ikke bare hente plasseringsdata fra disse – er det også en mulighet for et større arbeid, som kanskje kan egne seg som en ny oppgave?

6.4 Evaluering av gruppas arbeid

6.4.1 Innledning

Prosjektgruppen består av tre jenter som går Bachelor ingeniørfag Data ved Høgskolen i Gjøvik. I Objektorientert systemutviklingskurset som vi hadde i fjor høst, hadde vi et mindre prosjekt med ETC hvor vi jobbet med dokumentering for et prosjekt med Ruteplanlegger som en selvstendig modul, som skulle kunne integreres i CarAdmin. Vi valgte å gå videre med å videreutvikle denne oppgaven som bacheloroppgave. Det tok litt tid før det ble avklart mellom oppdragsgiver, veileder og oss om at vi skulle lage to direkte implementerte moduler til CarAdmin - Ruteplanlegger og Flåtestyring.

6.4.2 Organisering



Figur 30 Ansvarsfordeling

I begynnelsen av oppgaveperioden fordelte vi ansvaret i gruppen. Etter anbefaling fra veileder bestemte vi oss for å ha en prosjektleder. Siden Trine var den eneste i gruppen som hadde Javakunnskaper i forkant (se kapittel 1.4), ble hun valgt til prosjektleder. På prosjektet i høst fungerte Else som referent på møtene vi hadde med ETC, og det var derfor naturlig å utnevne henne som dokumentansvarlig. Marte fikk ansvaret for innholdet på hjemmesiden til bacheloroppgaven vår, noe som har inkludert daglig felles loggføring, oppdatering av forsiden i blant, og ellers holde siden oppdatert med møtereferater og lignende.

Denne organiseringen har fungert veldig godt. Dersom et gruppemedlem har vært syk eller av annen grunn fraværende, har resten av gruppa tatt jobben de aktuelle dagene, uten noen problemer.

6.4.3 Fordeling av arbeidet

Vi har jobbet en god del sammen tidligere i studiene, og vi var derfor ganske trygge på motivasjonen og samarbeidet innbyrdes i gruppen, og det å gjøre en bacheloroppgave sammen. For ordens skyld etablerte vi grupperegler og andre formalia i oppstarten av oppgaven, slik at det ikke skulle oppstå konflikter underveis. Dette har vi forøvrig klart å unngå særdeles bra! Arbeidsmengde har blitt fordelt underveis, og kommer ut forholdsvis jevnt, slik at alle til en hver tid har hatt sitt å gjøre, uten for mye kollisjon med hverandres arbeid. Vi har derfor jobbet med flere deler parallelt, noe som har gjort fremdriften noe mindre synlig på kort hold, men har gjort arbeidsprosessen mye lettere. Arbeidet har i all hovedsak foregått på vårt tildelte grupperom på skolen, med noen dagers hjemmearbeid dersom en har vært fraværende av ulike, godkjente årsaker.

Dokumentasjon vi har laget underveis i utviklingen, som prosjektdagboka og møtereferater, både med veileder og oppdragsgiver, ligger også ute på hjemmesiden, med deres godkjenninger. Vi har også hentet ut skjermbilder gjennom utviklingen og lagt ut på hjemmesiden, med tillatelse fra ETC. Hjemmesiden vår har dermed vært en løpende oppdatering på arbeidet vårt, med logg, møtereferater og forside med hyppigste oppdateringer.

I begynnelsen av oppgaven laget vi en arbeidsplan i form av et Gantt-skjema, se vedlegg B, over hvor lang tid vi skulle bruke på de forskjellige delene av oppgaven. Dette sprakk imidlertid på første sprint, da det å komme i gang krevde sin egen sprint. I den opprinnelige tidsplanen hadde vi satt av ekstra tid mot slutten av oppgaven for fullføring av koding, og vi tok derfor av denne tiden for å lage en ekstra sprint. Ut over dette har planen vår fungert godt, noe som kommer frem ved sammenligning av det opprinnelige og det reviderte Gantt-skjemaet.

6.4.4 Prosjekt som arbeidsform

Vi i gruppen har som nevnt tidligere også arbeidet sammen i prosjekter og annet skolearbeid, og det har derfor ikke vært noen større problemer med å arbeide sammen på denne bacheloroppgaven. Vi har etter beste evne forsøkt å følge utviklingsmetoden Scrum, som nevnt i kapittel 4.1, selv om noen deler, som egen Scrummaster, har vi kuttet bort. På grunn av noe liten tid, valgte vi ikke å låse sprintene helt, slik at vi kunne forsøke å få med ETC sine ønsker om endringer direkte.

Prosjektarbeid er en arbeidsform som blir stadig mer vanlig i arbeidslivet, og vi syns det har vært en god erfaring å kunne ta med seg. Vi har hatt noen mindre prosjekter i andre fag tidligere i studiet, men bacheloroppgaven har vært det desidert største og mest lærerike prosjektet for oss. Vi var veldig spente på hvordan dette ville være, men arbeidet har fungert meget bra totalt sett, noe vi også har fått skryt for av oppdragsgiver.

6.4.5 Subjektiv opplevelse av bacheloroppgaven

Vi begynte på oppgaven med godt mot i januar, etter å ha hatt en liten forsmak på oppgaven i høst. Arbeidet fra i høst måtte riktignok mer eller mindre skrinlegges, siden det ble en del endringer på hva vi skulle utvikle, men vi hadde fått en viss grad av innsikt i hva ETC ønsket seg av løsningen. Dokumentasjonen måtte vi lage på nytt, siden vi nå skulle lage to implementerte moduler til CarAdmin, i stedet for en ekstern. Mye av dette fikk vi på plass i løpet av januar, hvor vi leverte forprosjektet, og skrev et nytt utkast til kravspesifikasjon og designdokument.

Da kodingen skulle begynne tok det noe tid å få all koden fra ETC, og få installert Eclipse og Tomcat på våre datamaskiner, men med god hjelp fra Øyvind i ETC, lot det seg gjøre. Vi fikk også en rask innføring i hvordan systemet i bunn fungerte. Det tok allikevel mye tid å sette seg inn i deres kode, og komme skikkelig i gang med modulene vi skulle lage, men vi snublet oss litt frem i starten, og med litt mer hjelp fra Øyvind, fikk vi da levert noe i slutten av første sprint. Et innskudd med sprint 1,5 gjorde at vi fikk ferdig det som opprinnelig var planen i sprint 1, og vi fortsatte videre med godt mot. Den opprinnelige planen på fem sprinter gjorde at vi fremdeles hadde tid til de resterende planlagte sprintene. Disse sprintene gikk forholdsvis greit, uten større endringer, annet enn endringer som kom underveis fra ETC.

Rapportskrivingen tok seg sakte men sikkert opp igjen etter rapportkurset til Høgskolelektor Frode Haug den 17. mars. Vi var nå glade for at vi hadde tatt en god del av arbeidet med å nedtegne kravspesifikasjonen og designet allerede i januar, slik at vi nå hadde noe å jobbe videre på. Etter påskeferien tok arbeidet med rapportskrivingen seg gradvis opp igjen, selv om det fortsatt var kodingen som stod i fokus. Da sprint 4 var ferdig, tok vi oss sammen og fikk litt mer fart på skrivingen. Da sprint 5 var ferdig hadde vi også litt under halvparten av teksten klar, og fikk litt hjelp med en midlertidig gjennomlesning av veileder. Da første hele utkast var klart, fikk veileder og oppdragsgiver komme med tilbakemelding på om det var noen feil eller mangler som vi burde rette opp før levering.

Arbeidet med utviklingen av våre to moduler har vært både spennende og særdeles lærerikt for oss.

6.5 Konklusjon

Etter et halvt års arbeid, som avslutning på våre tre år her på Høgskolen i Gjøvik, har vi kommet i mål med en oppgave vi selv er godt fornøyde med. Vi har nå fått testet ut hvordan det er å gjøre et fullverdig oppgave i sin helhet, ved å benytte alle de små aspektene vi har lært i forskjellige emner gjennom studiet.

Vi har i tillegg måttet lære oss JavaScript-programmering i JSP-sider, og å håndtere kode i en trelagsstruktur, med både JavaScript og HTML, Java og SQL-koding. Dette har vært en meget lærerik periode, og vi har fått prøvd oss på å finne ut av mange elementer innen Google Maps API og å måtte sette oss inn i den eksisterende koden til ETC.

Når oppgaven nå er ferdig, er vi meget fornøyd med resultatet vi har kommet frem med. Vi har møtt på en del utfordringer underveis, men søking på nettsidene til Google, en opplæringsside for bruken av Google Maps, og diverse nettfora har hjulpet oss med å få opp funksjonaliteten vi har ønsket oss. Nå håper vi ETC får god bruk for modulene vi har laget, eventuelt med noe videreutvikling og finpussing.

7 Litteraturliste

7.1 Nettsider

Google Maps API Concepts 19/05-2010
http://code.google.com/intl/nb-NO/apis/maps/documentation/index.html

Java™ Platform, Standard Edition 6 API Specification 19/05-2010
http://java.sun.com/javase/6/docs/api/

Asynchrony 19/05-2010
http://en.wikipedia.org/wiki/Asynchrony

Arrays 19/05-2010
http://java.sun.com/docs/books/tutorial/java/nutsandbolts/arrays.html

Vector 19/05-2010
http://java.sun.com/j2se/1.4.2/docs/api/java/util/Vector.html

HTML tutorial 19/05-2010
http://www.w3schools.com/html/default.asp

SQL tutorial 19/05-2010
http://www.w3schools.com/sql/default.asp

Google Maps API References 19/05-2010
http://code.google.com/intl/nb-NO/apis/maps/documentation/reference.html

JExamples 19/05-2010
http://www.jexamples.com/srchRes/java.util.Vector

Google Maps API tutorial by Mike Williams 19/05-2010 http://econym.org.uk/gmap/

- http://econym.org.uk/gmap/mouseo.htm
- http://econym.org.uk/gmap/didyoumean.htm
- http://econym.org.uk/gmap/snap.htm
- http://econym.org.uk/gmap/basic1.htm
- http://econym.org.uk/gmap/basic2.htm
- http://econym.org.uk/gmap/basic3.htm
- http://econym.org.uk/gmap/basic14.htm
- http://econym.org.uk/gmap/basic15.htm
- http://econym.org.uk/gmap/directions.htm

Google Marker Icons 19/05-2010
http://jg.org/mapping/icons.html

Mapki 19/05-2010
http://mapki.com/wiki/Available_Images

Google Web Toolkit 19/05-2010
http://groups.google.com/group/Google-WebToolkit/browse_thread/thread/f87f3e292d0dce71/20f5f308b2162c96

Drawing multiple routes in Google Maps 19/05-2010
http://osdir.com/ml/GoogleMapsAPI/2009-01/msg01392.html

Routes from waypoint 19/05-2010
http://www.geocodezip.com/multiple_routes.asp

Google Maps JavaScript API example 19/05-2010
http://www.marschellsmusic.com/API/Map_API__EXAMPLE_01.html

Polylines color change 19/05-2010
http://groups.google.com/group/Google-Maps-API/msg/005590fd03a901c6

Chart types for maps 19/05-2010
http://groups.google.com/group/google-chart-api/web/chart-types-for-map-pins?pli=1

Turnings markers on and off… 19/05-2010
http://www.geocodezip.com/v2_MarkerCheckBoxes.asp

Goolge Maps return null… 19/05-2010
http://groups.google.com/group/google-maps-api/browse_thread/thread/cacc6cf7e4061e9c/bebb8a0a4c4b66ad?hl=en&lnk=gst&q=getLatLng+returns+null#bebb8a0a4c4b66ad

Web Color Chart 19/05-2010
http://html-color-codes.com/

Travelling by boat, ferries and cruises… 19/05-2010
http://australiantravel.info/boat.html

7.2 Gamle bacheloroppgaver

”Kontrollsystem for Autoklave” av Klundby, Konstad og Øverjordet 24/05-2006

”iSlideS Slideshowgenerator ” av Gulla og Sporsheim 14/05-2004

”Klientbasert reservering av standard kalenderapplikasjoner” av Haziraj, Kultom og Klundby 19/05-2008

”BilBooking CarAdmin” av Ensrud, Gjernde og Solberg 19/05-2005

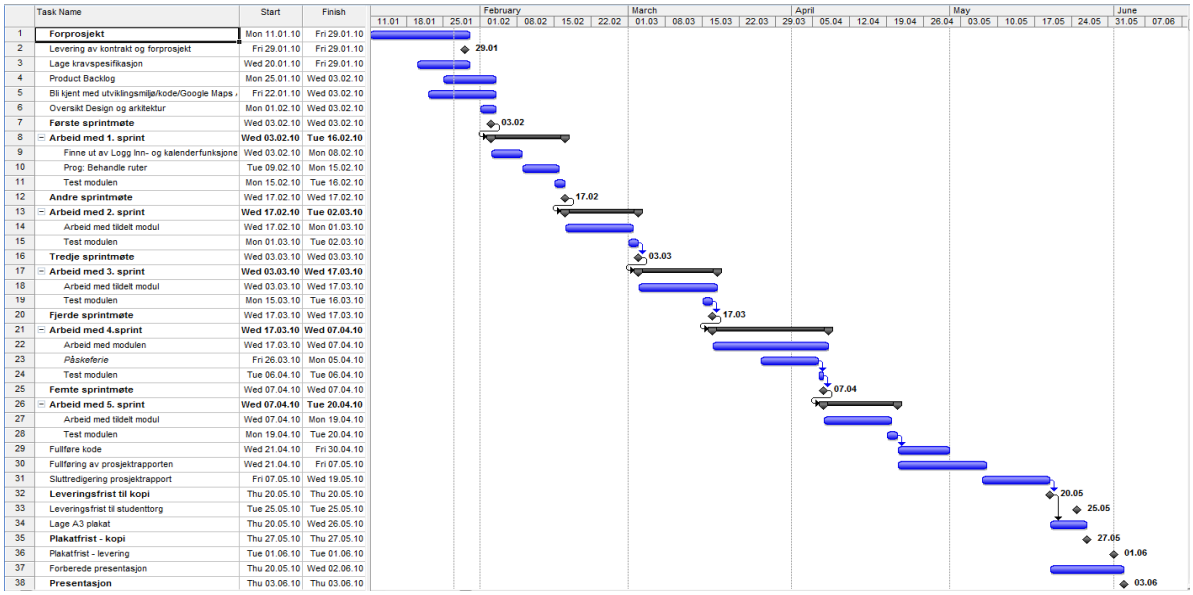
”Styresaksdatabase” av Lunde, Falstad og Backstrøm 20/05-2009

8 Vedlegg:

A. Definisjoner

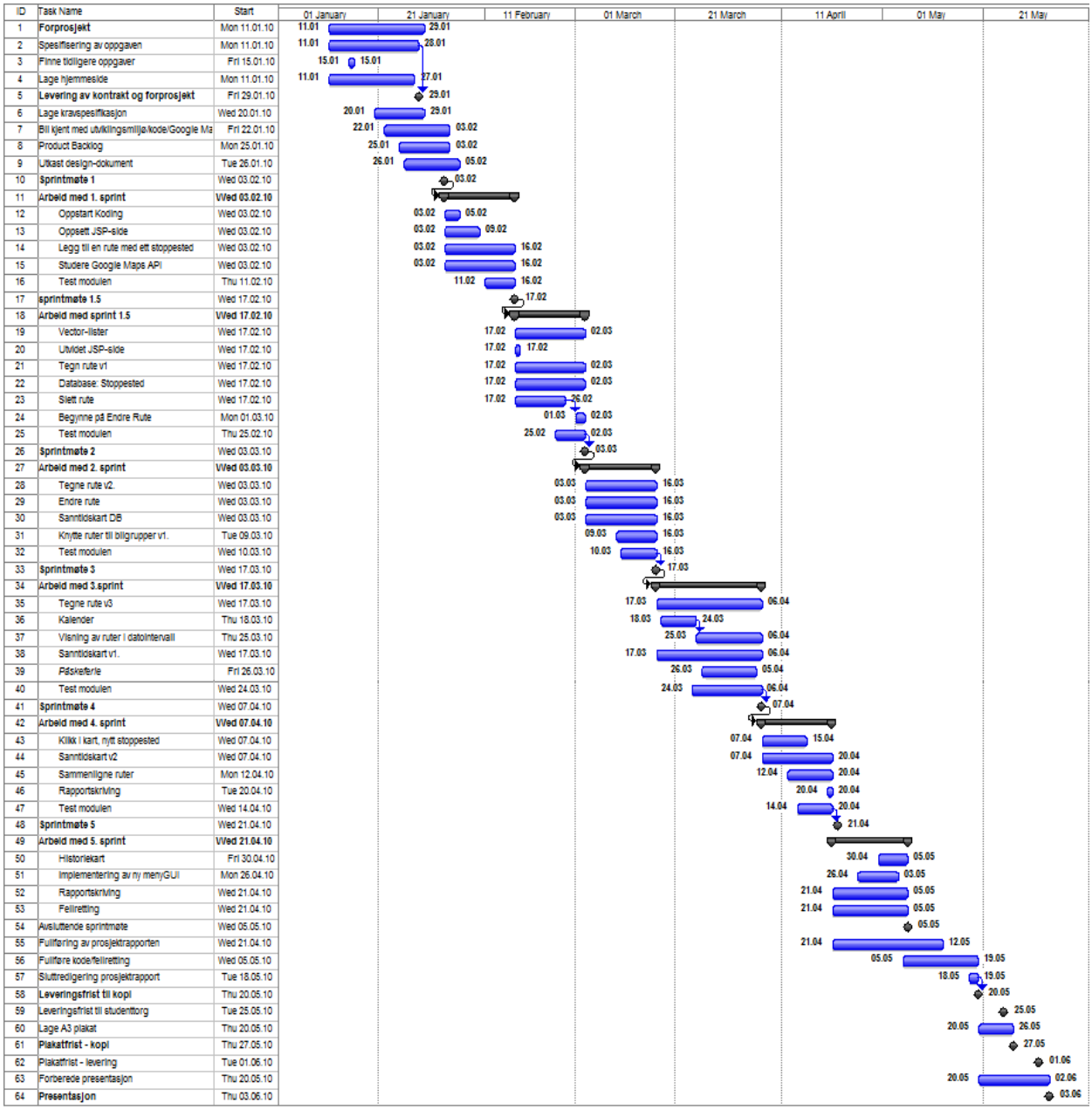
- Eclipse: En utviklingsplattform, som er fri til bruk
- Utviklingsplattform: En applikasjon hvor programkode skrives, testes og håndteres på andre måter.
- Subversion: Et versjonskontrollsystem som holder styr på endringer i filer.
- Subclipse: Et tillegg til Eclipse, som er et versjonskontrollsystem.
- Scrum: En smidig systemutviklingsmodell som sier noe om arbeidsprosessen i et utviklingsprosjekt.
- Google Maps API: Et kartverk med åpen kildekode
- Kartverk: Elektronisk karttjeneste som kan vise og håndtere kart i en nettside.
- Sprint: En arbeidsperiode hvor det arbeides med en bestemt del av prosjektet. Et prosjekt består av flere sprinter, som skal ha en fast lengde.
- Produkktkø: Alle deler av prosjektet som skal utvikles.
- Sprintkø: De delene av prosjektet som er arbeidet med under den valgt sprinten.
- Lokasjon: Tilholdssted(er) for bedriften. Eksempelvis kontoret til hjemmetjenesten.
- Stoppested: Et punkt i ruta som sjåføren skal innom.
- Startpunkt: En rutes første punkt i kartet.
- Endepunkt: En rutes siste punkt i kartet.
- Alerts: Advarsler som kommer opp i vinduer på skjermen og ber bruker bekrefte noe eller informerer om noe.
- Fri rute: En rute som ikke er registrert på noen spesifikk dato. Disse rutene har bare tidspunkt.
- Sanntidskart: Kart over hvor bilene er akkurat nå.
- Historiekart: Kart over hvor en valgt bil har vært.
- Rute: Mange stoppesteder satt sammen til å utgjøre en rute.
- Ruteplanlegger: En del av det vi skal lage for ETC
- Flåtestyring: En del av det vi skal lage for ETC
- Modul: En del av et system.
- Applikasjon: Et helt system.
- GPS med GSM-sender: Hardwaren som sender data om bilens posisjon tilbake til
- Kjøretøygruppe: En gruppe med kjøretøyer som hører til en spesifikk lokasjon.
- Dagsruter: Visning av ruter på en gitt dag.
- Feilfanger: Kode implementert i CarAdmin av ETC for å rapportere feil når disse oppstår.
- Konsollvindu: Vinduet i Eclipse der systemet skriver ut sine meldinger.
- Geokoding: En metode for å gjøre om en adresse til koordinater.
- Sesjon: Tiden en bruker har på serveren.

B. Prosjektplaner



	Task Name	Start	Finish
1	**Forprosjekt**	Mon 11.01.10	Fri 29.01.10
2	Levering av kontrakt og forprosjekt	Fri 29.01.10	Fri 29.01.10
3	Lage kravspesifikasjon	Wed 20.01.10	Fri 29.01.10
4	Product Backlog	Mon 25.01.10	Wed 03.02.10
5	Bli kjent med utviklingsmiljø/kode/Google Maps	Fri 22.01.10	Wed 03.02.10
6	Oversikt Design og arkitektur	Mon 01.02.10	Wed 03.02.10
7	**Første sprintmøte**	Wed 03.02.10	Wed 03.02.10
8	**Arbeid med 1. sprint**	**Wed 03.02.10**	**Tue 16.02.10**
9	Finne ut av Logg Inn- og kalenderfunksjone	Wed 03.02.10	Mon 08.02.10
10	Prog: Behandle ruter	Tue 09.02.10	Mon 15.02.10
11	Test modulen	Mon 15.02.10	Tue 16.02.10
12	**Andre sprintmøte**	Wed 17.02.10	Wed 17.02.10
13	**Arbeid med 2. sprint**	**Wed 17.02.10**	**Tue 02.03.10**
14	Arbeid med tildelt modul	Wed 17.02.10	Mon 01.03.10
15	Test modulen	Mon 01.03.10	Tue 02.03.10
16	**Tredje sprintmøte**	Wed 03.03.10	Wed 03.03.10
17	**Arbeid med 3. sprint**	**Wed 03.03.10**	**Wed 17.03.10**
18	Arbeid med tildelt modul	Wed 03.03.10	Wed 17.03.10
19	Test modulen	Mon 15.03.10	Tue 16.03.10
20	**Fjerde sprintmøte**	Wed 17.03.10	Wed 17.03.10
21	**Arbeid med 4.sprint**	**Wed 17.03.10**	**Wed 07.04.10**
22	Arbeid med modulen	Wed 17.03.10	Wed 07.04.10
23	*Påskeferie*	Fri 26.03.10	Mon 05.04.10
24	Test modulen	Tue 06.04.10	Tue 06.04.10
25	**Femte sprintmøte**	Wed 07.04.10	Wed 07.04.10
26	**Arbeid med 5. sprint**	**Wed 07.04.10**	**Tue 20.04.10**
27	Arbeid med tildelt modul	Wed 07.04.10	Mon 19.04.10
28	Test modulen	Mon 19.04.10	Tue 20.04.10
29	Fullføre kode	Wed 21.04.10	Fri 30.04.10
30	Fullføring av prosjektrapporten	Wed 21.04.10	Fri 07.05.10
31	Sluttredigering prosjektrapport	Fri 07.05.10	Wed 19.05.10
32	**Leveringsfrist til kopi**	Thu 20.05.10	Thu 20.05.10
33	Leveringsfrist til studenttorg	Tue 25.05.10	Tue 25.05.10
34	Lage A3 plakat	Thu 20.05.10	Wed 26.05.10
35	**Plakatfrist - kopi**	Thu 27.05.10	Thu 27.05.10
36	Plakatfrist - levering	Tue 01.06.10	Tue 01.06.10
37	Forberede presentasjon	Thu 20.05.10	Wed 02.06.10
38	**Presentasjon**	Thu 03.06.10	Thu 03.06.10

Figur 31 Orginalt Gantt-skjema



ID	Task Name	Start
1	**Forprosjekt**	Mon 11.01.10
2	Spesifisering av oppgaven	Mon 11.01.10
3	Finne tidligere oppgaver	Fri 15.01.10
4	Lage hjemmeside	Mon 11.01.10
5	**Levering av kontrakt og forprosjekt**	Fri 29.01.10
6	Lage kravspesifikasjon	Wed 20.01.10
7	Bli kjent med utviklingsmiljø/kode/Google Ma	Fri 22.01.10
8	Product Backlog	Mon 25.01.10
9	Utkast design-dokument	Tue 26.01.10
10	**Sprintmøte 1**	Wed 03.02.10
11	**Arbeid med 1. sprint**	**Wed 03.02.10**
12	Oppstart Koding	Wed 03.02.10
13	Oppsett JSP-side	Wed 03.02.10
14	Legg til en rute med ett stoppested	Wed 03.02.10
15	Studere Google Maps API	Wed 03.02.10
16	Test modulen	Thu 11.02.10
17	**sprintmøte 1.5**	Wed 17.02.10
18	**Arbeid med sprint 1.5**	**Wed 17.02.10**
19	Vector-lister	Wed 17.02.10
20	Utvidet JSP-side	Wed 17.02.10
21	Tegn rute v1	Wed 17.02.10
22	Database: Stoppested	Wed 17.02.10
23	Slett rute	Wed 17.02.10
24	Begynne på Endre Rute	Mon 01.03.10
25	Test modulen	Thu 25.02.10
26	**Sprintmøte 2**	Wed 03.03.10
27	**Arbeid med 2. sprint**	**Wed 03.03.10**
28	Tegne rute v2.	Wed 03.03.10
29	Endre rute	Wed 03.03.10
30	Sanntidskart DB	Wed 03.03.10
31	Knytte ruter til bilgrupper v1.	Tue 09.03.10
32	Test modulen	Wed 10.03.10
33	**Sprintmøte 3**	Wed 17.03.10
34	**Arbeid med 3.sprint**	**Wed 17.03.10**
35	Tegne rute v3	Wed 17.03.10
36	Kalender	Thu 18.03.10
37	Visning av ruter i datointervall	Thu 25.03.10
38	Sanntidskart v1.	Wed 17.03.10
39	*Påskeferie*	Fri 26.03.10
40	Test modulen	Wed 24.03.10
41	**Sprintmøte 4**	Wed 07.04.10
42	**Arbeid med 4. sprint**	**Wed 07.04.10**
43	Klikk i kart, nytt stoppested	Wed 07.04.10
44	Sanntidskart v2	Wed 07.04.10
45	Sammenligne ruter	Mon 12.04.10
46	Rapportskriving	Tue 20.04.10
47	Test modulen	Wed 14.04.10
48	**Sprintmøte 5**	Wed 21.04.10
49	**Arbeid med 5. sprint**	**Wed 21.04.10**
50	Historiekart	Fri 30.04.10
51	Implementering av ny menyGUI	Mon 26.04.10
52	Rapportskriving	Wed 21.04.10
53	Feilretting	Wed 21.04.10
54	Avsluttende sprintmøte	Wed 05.05.10
55	Fullføring av prosjektrapporten	Wed 21.04.10
56	Fullføre kode/feilretting	Wed 05.05.10
57	Sluttredigering prosjektrapport	Tue 18.05.10
58	**Leveringsfrist til kopi**	Thu 20.05.10
59	Leveringsfrist til studentorg	Tue 25.05.10
60	Lage A3 plakat	Thu 20.05.10
61	**Plakatfrist - kopi**	Thu 27.05.10
62	Plakatfrist - levering	Tue 01.06.10
63	Forberede presentasjon	Thu 20.05.10
64	**Presentasjon**	Thu 03.06.10

Figur 32 Oppdatert Gantt-skjema

<table>
<tr><th></th><th>Mandag</th><th>Tirsdag</th><th>Onsdag</th><th>Torsdag</th><th>Fredag</th></tr>
<tr><td>08.00</td><td rowspan="3"></td><td rowspan="3">B.oppg</td><td>Møte med ETC</td><td rowspan="3">B.oppg</td><td></td></tr>
<tr><td>09.00</td><td rowspan="2">B.oppg</td><td rowspan="3">B.oppg</td></tr>
<tr><td>10.00</td></tr>
<tr><td>11.00</td><td>statistikk</td><td></td><td></td><td></td></tr>
<tr><td>11.45</td><td>Lunsj</td><td>Lunsj</td><td>Lunsj</td><td>Lunsj</td><td>Lunsj</td></tr>
<tr><td>12.30</td><td rowspan="3">k.ledelse</td><td>statistikk</td><td rowspan="3">B.oppg</td><td>Møte med Tom</td><td rowspan="3">B.oppg</td></tr>
<tr><td>13.30</td><td rowspan="2">B.oppg</td><td rowspan="2">K.ledelse</td></tr>
<tr><td>14.30</td></tr>
<tr><td>15.30</td><td rowspan="2">statistikk</td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td></tr>
<tr><td>16.30</td></tr>
</table>

Figur 33 Timeplan for semesteret

C. Statusrapporter

Høgskolen i Gjøvik 18.02.10

Statusrapport nr. 1 ved bacheloroppgave: Kjørerute

Status:

Planlegging/ fremdrift:
Vi har holdt oss til fremdriftsplanen så langt, men har bestemt oss for å legge inn en ekstra sprint, før vi begynner på den planlagte sprint to.

Organisering av gruppens arbeid og ansvarsområder:
Arbeid og ansvarsområder holdes som i forprosjektet.

Rapportskriving:
Vi har laget kravspesifikasjon og designdokument som skal danne et grunnlag for videre rapportskriving, men venter med videre arbeid på hovedrapporten til vi er kommet lengre i kodingen. Av annen rapportering skrives møtereferat etter hvert av våre ukentlige møter med hhv. veileder og oppdragsgiver.

Trusler/Problemer/Muligheter:
Det har tatt lenger tid å sette seg inn i den eksisterende kode, enn først antatt, men er nå begynt å komme inn i den, men med litt hjelp som vi får fra oppdragsgiver skal ikke dette utgjøre noe stort problem.

Hva er avsluttet?
Kravspesifikasjon og designdokument er avsluttet, men skal flettes inn i sluttrapport.
Sprint 1 er ferdig, hvor vi har fått opp det meste av brukergrensesnittet til sluttløsningen.
Forprosjektet ble levert til fristen, og tilbakemeldingene er rettet opp.

Hva er under arbeid?
Begynner nå på neste sprint, hvor vi vil ta for oss de resterende emner fra sprint 1, i tillegg til å begynne på kartdelen av hjemmesidene.

Tidsfrister:
Alle tidsfrister er så langt overholdt.

Motivasjon i gruppa:
Gruppen samarbeider godt, og diskuterer problemer som kommer opp underveis. Arbeidet blir fordelt forholdsvis jevnt mellom gruppens medlemmer. Det er til tider frustrerende med uforståelige feilmeldinger under koding, men hittil har dette løst seg forholdsvis greit med god hjelp fra oppdragsgiver. Det oppleves som motiverende for gruppen å sitte sammen med kodingen, og ha tett kontakt med oppdragsgiver som kan hjelpe oss når vi står fast. Det er også motiverende å se resultater, dersom det er ting vi har strevd med.

Veilederkontakt:
Ved å holde ukentlige møter med veileder synes vi at det er blitt god kontakt, og vi får tilbakemeldinger på det vi lurer på.

På vegne av gruppen:

Trine Anita Grønvold, gruppeleder

Høgskolen i Gjøvik 17.03.2010

Statusrapport nr. 2 ved bacheloroppgave: Kjørerute

Status:

Planlegging/ fremdrift:
Vi ligger forholdsvis i rute på sprint 2, men har måttet endre innholdet i denne noe, for å unngå å jobbe på samme filer.

Organisering av gruppens arbeid og ansvarsområder:
Arbeid og ansvarsområder holdes som i forprosjektet. Har hver vår programmeringsdel vi har ansvaret for.

Rapportskriving:
Vi har laget kravspesifikasjon og designdokument som skal danne et grunnlag for videre rapportskriving, men venter med videre arbeid på hovedrapporten til vi er kommet lengre i kodingen. Av annen rapportering skrives møtereferat etter hvert av våre ukentlige møter med hhv. veileder og oppdragsgiver. Planlegger å legge inn oppstart av rapport som en del i neste sprint.

Trusler/Problemer/Muligheter
Har fått noen endringer fra ETC som vi har forsøkt å flette inn. Har enda ikke fått data fra GPS for bruk i flåtestyring. Dersom vi ikke får disse dataene snart, må vi lage en testdatabase.

Hva er avsluttet?
Kravspesifikasjon og designdokument er avsluttet, men skal flettes inn i sluttrapport.
Sprint 1 er ferdig, hvor vi har fått opp det meste av brukergrensesnittet til sluttløsningen.
Sprint 1,5 og 2 er også avsluttet. Forprosjektet ble levert til fristen, og tilbakemeldingene er rettet opp. Modulen med legg til/slett rute er ferdig.

Hva er under arbeid?
Arbeider med å flette sammen ruter og biler/bilgrupper. Arbeider med starten på sanntidskartet i flåtestyringen, samt å knytte kart til ruter.

Tidsfrister
Alle tidsfrister er så langt overholdt.

Motivasjon i gruppa:
Motivasjonen i gruppa er fortsatt bra, på tross av noe sykdom. Alle er flinke til å møte til avtalt tid, og vi får arbeidet forholdsvis effektivt.

Veilederkontakt:
Vi finner fremdeles veilederkontakten som veldig god. Har hatt ukentlige møter hver uke, med ett unntak.

På vegne av gruppen:

Trine Anita Grønvold, gruppeleder

Høgskolen i Gjøvik 21.04.2010

Statusrapport nr. 3 ved bacheloroppgave: Kjørerute

Status:

Planlegging/ fremdrift:
Vi begynner på Sprint 5 i dag, som blir den siste. Sprint 4 ble avsluttet i dag morges. Er i gang med rapportskrivingen.

Organisering av gruppens arbeid og ansvarsområder:
Arbeid og ansvarsområder holdes som i forprosjektet. Har hver vår programmeringsdel vi har ansvaret for.

Rapportskriving:
Rapportskrivingen er oppstartet. Har begynt på delen med kravspesifikasjon og designdokumentet.

Trusler/Problemer/Muligheter
Måtte tidlig legge inn en ekstra sprint. Denne har tatt noe tid fra rapportskrivningen, men trur ikke det skal bli noen stor hindring. Måtte lage testdatabase ift. sanntidskartet, da vi ikke har fått noen data fra ETC.

Hva er avsluttet?
Sprint 4 er nå avsluttet. Her er mesteparten av ”Behandle ruter” gjort ferdig, og begynnelsen av ”Sammenligne ruter” og ”Sanntidskart” ordnet.

Hva er under arbeid?
Finpussing på ”behandle ruter” og ”sammenligne ruter”, og arbeid på ”sanntidskartet” med historie.

Tidsfrister
Alle tidsfrister er så langt overholdt.

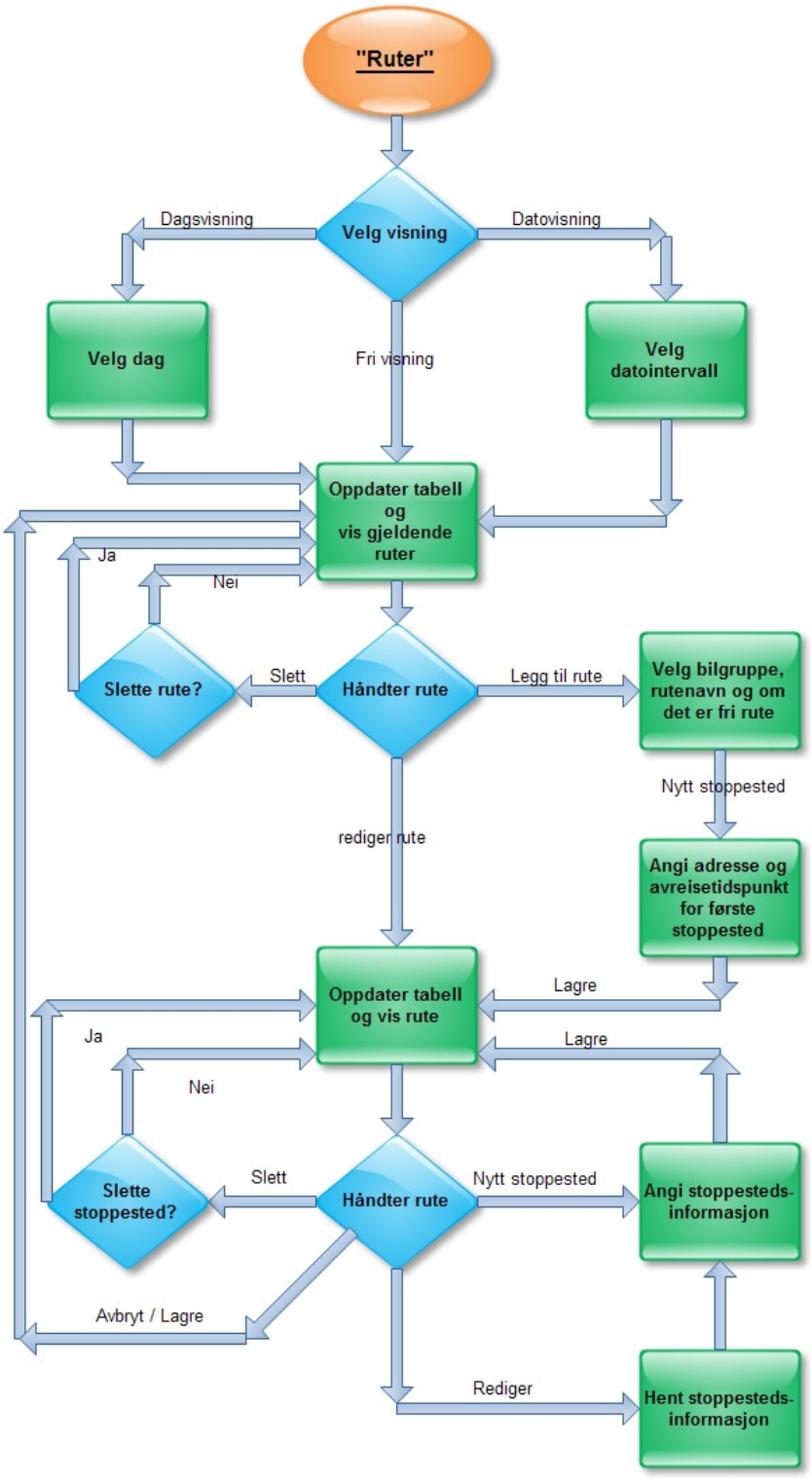
Motivasjon i gruppa:
Motivasjonen i gruppa er fortsatt bra. Alle er flinke til å møte til avtalt tid, og vi får arbeidet forholdsvis effektivt.

Veilederkontakt:
Vi finner fremdeles veilederkontakten som veldig god. Har hatt ukentlige møter hver uke, med to unntak.

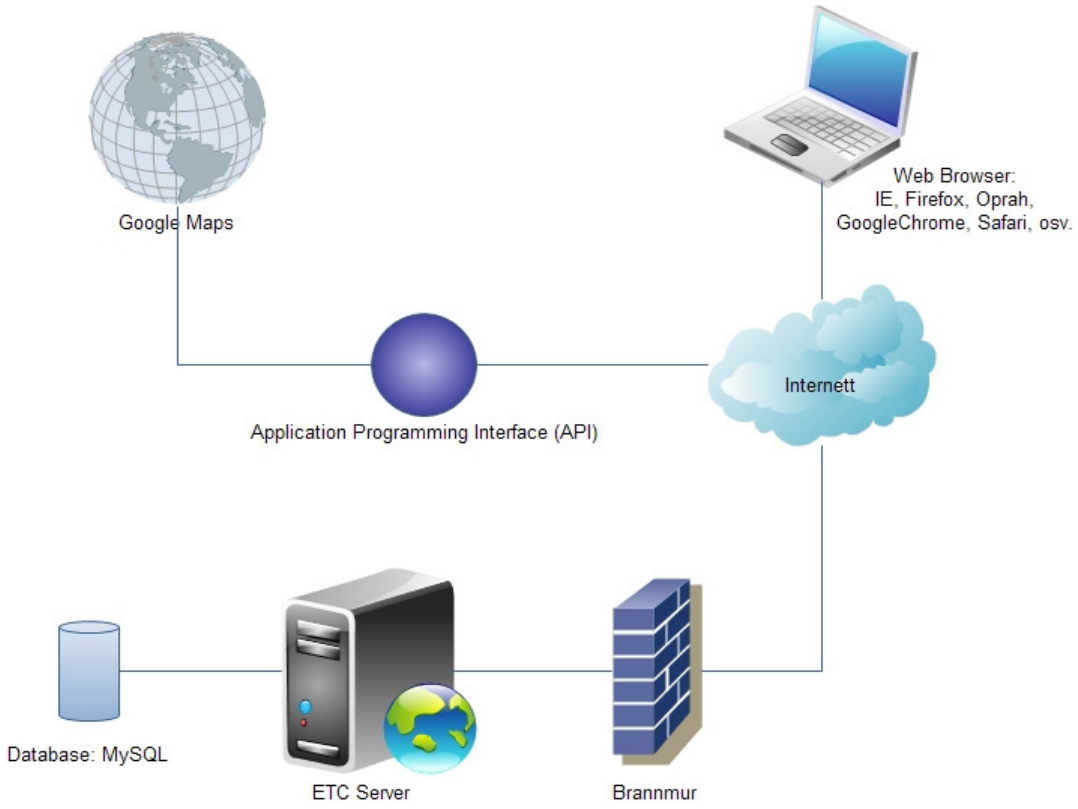
På vegne av gruppen:

Trine Anita Grønvold, gruppeleder

D. Designskjemaer



Figur 34 Flytdiagram for håndtering av ruter



Figur 35 Deploymentmodell

Oppsummering av løsning:

Den faktiske ruten til en bil vil vises som en farget strek over veien den har kjørt i kartet, mellom punktene som er logget. Dersom det er flere muligheter av veier, vil den mest sannsynlige veien bli vist som en stiplet linje.

Faktorer:

- Hver bil har en GPS utstyrt med SIM-kort, som sender bilens koordinater til ETC sin server. Disse koordinatene danner grunnlaget for karttegningen.

Løsning:

Når planleggeren velger å vise en bestemt bil sin faktiske kjørerute, åpnes en ny side med et kart. På kartet settes alle de loggede koordinatene for bilen inn, og applikasjonen tegner en strek mellom hvert punkt, langs etter veiene. Dersom det mellom to punkter er flere mulige veier å kjøre, tegnes den mest sannsynlige veien inn i kartet.(MER!)

Motivasjon:

Kunne se hvordan en rute er kjørt, for å kunne optimere rutene ut i fra dette.

Uløste problemer:

- Hvordan vite hvilken vei som er mest sannsynlig kjørt, der flere veier er mulige?
- Skal flere biler kunne vises i samme Historiekart? Hvordan skal i så fall disse vises?

Alternativer:

Der hvor det er flere mulige veier å kjøre, mellom to loggede punkter, hentes ruten fra planleggingen inn, og tas i betraktning i forhold til hvilken vei som velges. Denne kan enten vises som hel, eller stiplet linje.

Kartet kan eventuelt også vise stoppestedene ruten består av.

Problem: Hvordan vise bilhistorikk i kart?

E. Kode

Kildekode til Mike sitt Google Maps eksempel på en rute med flere stoppesteder. (http://econym.org.uk/gmap/example_multi2.htm)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8"/>
    <title>Google Maps Multi-Point Routing</title>
    <script
src="http://maps.google.com/maps?file=api&amp;v=2&amp;sensor=false&amp;key=
ABQIAAAAPDUET0Qt7p2VcSk6JNU1sBSM5jMcmVqUpI7aqV44cW1cEECiThQYkcZUPRJn9vy_TWx
WvuLoOfSFBw" type="text/javascript"></script>
  </head>
  <body onunload="GUnload()">
  <h2>Multi-point routing</h2>
    <ul>
    <li>Specify the start location. You can either enter an address at the
bottom or click on the map.
    <li>Specify the destination. You can either enter an address at the
bottom or click on the map.
    <li>Three inactive intermediate points will appear.
    <li>When you drag an inactive intermediate point it becomes active.
    <li>You can also drag the start and end points.
    <li>Click on "get directions".
    <li>If the route cannot be calculated, or is not suitable, adjust the
        positions of the points, and try again.
    </ul>
    <table>
      <tr>
        <td width=700>
          <div id="map" style="width: 700px; height: 600px; border:1px
solid black"></div>
        </td>
        <td width=400>
          <div id="path" style="width: 400px; height: 600px; overflow:auto;
border:1px solid black"></div>
        </td>
      </tr>
    </table>
    <div id="start">
      Enter the address of the starting point or click the map.
      <form onsubmit="showAddress(); return false" action="#">
        <input id="search" size="60" type="text" value="" />
        <input type="submit" value="Find start address" />
      </form>
    </div>
    <div id="end">
      Enter the address of the destination point or click the map.
      <form onsubmit="showAddress(); return false" action="#">
        <input id="search2" size="60" type="text" value="" />
        <input type="submit" value="Find destination address" />
      </form>
    </div>

    <div id="drag">
```

```
       Drag the markers as required.
       <input type="button" value="Get Directions" onclick="directions()"
/>
    </div>

    <a href="directions.htm">Back to the tutorial page</a>
    <noscript><b>JavaScript must be enabled in order for you to use Google
Maps.</b>
      However, it seems JavaScript is either disabled or not supported by
your browser.
      To view Google Maps, enable JavaScript by changing your browser
options, and then
      try again.
    </noscript>

    <script type="text/javascript">
    //<![CDATA[

    if (GBrowserIsCompatible()) {

      var map = new GMap(document.getElementById("map"));
      map.addControl(new GLargeMapControl());
      map.addControl(new GMapTypeControl());
      map.setCenter(new GLatLng(40,-100),5);

      var bounds = new GLatLngBounds();

      // ====== Create a Client Geocoder ======
      var geo = new GClientGeocoder(new GGeocodeCache());

      // ====== Array for decoding the failure codes ======
 var reasons=[];
      reasons[G_GEO_SUCCESS]            = "Success";
      reasons[G_GEO_MISSING_ADDRESS]    = "Missing Address: The address was
either missing or had no value.";
      reasons[G_GEO_UNKNOWN_ADDRESS]    = "Unknown Address:  No
corresponding geographic location could be found for the specified
address.";
      reasons[G_GEO_UNAVAILABLE_ADDRESS]= "Unavailable Address:  The
geocode for the given address cannot be returned due to legal or
contractual reasons.";
      reasons[G_GEO_BAD_KEY]            = "Bad Key: The API key is either
invalid or does not match the domain for which it was given";
      reasons[G_GEO_TOO_MANY_QUERIES]   = "Too Many Queries: The daily
geocoding quota for this site has been exceeded.";
      reasons[G_GEO_SERVER_ERROR]       = "Server error: The geocoding
request could not be successfully processed.";
      reasons[G_GEO_BAD_REQUEST]        = "A directions request could not
be successfully parsed.";
      reasons[G_GEO_MISSING_QUERY]      = "No query was specified in the
input.";
      reasons[G_GEO_UNKNOWN_DIRECTIONS] = "The GDirections object could not
compute directions between the points.";

      // ====== Geocoding ======
      function showAddress() {
        if (state==0) {
          var search = document.getElementById("search").value;
          addresses[0] = search;
        }
```

```
        if (state==1) {
          var search = document.getElementById("search2").value;
          addresses[4] = search;
        }
        geo.getLatLng(search, function (point)
          {
            if (point) {
              if (state==1) {doEnd(point)}
              if (state==0) {doStart(point)}
            }
            else {
              var result=geo.getCache().get(search);
              if (result) {
                var reason="Code "+result.Status.code;
                if (reasons[result.Status.code]) {
                  reason = reasons[result.Status.code]
                }
              } else {
                var reason = "";
              }
              alert('Could not find "'+search+ '" ' + reason);
            }
          }
        );
      }

      function swapMarkers(i) {
        map.removeOverlay(gmarkers[i]);
        createMarker(path[i],i,icon2);
      }

      var baseIcon = new GIcon(G_DEFAULT_ICON);
      baseIcon.iconSize=new GSize(24,38);

      var icon1 = G_START_ICON;
      var icon2 = G_PAUSE_ICON;
      var icon3 = G_END_ICON;
      var icon4 = new
GIcon(baseIcon,"http://labs.google.com/ridefinder/images/mm_20_white.png");
          icon4.shadow =
"http://labs.google.com/ridefinder/images/mm_20_shadow.png";
          icon4.iconSize = new GSize(12, 20);
          icon4.shadowSize = new GSize(22, 20);
          icon4.iconAnchor = new GPoint(6, 20);
          icon4.infoWindowAnchor = new GPoint(5, 1);


      function createMarker(point,i,icon) {
        var marker = new GMarker(point, {draggable:true,icon:icon});
        gmarkers[i]=marker;
        GEvent.addListener(marker, "dragend", function() {
          path[i] = marker.getPoint();
          if (!active[i]) {
            setTimeout('swapMarkers('+i+')', 1000);
          }
          active[i] = true;
          addresses[i] = "";
        });
        map.addOverlay(marker);
      }
```

```
// ===== Array to contain the points of the path =====
var path = [];
var active = [true,false,false,false,true];
var gmarkers = [];
var addresses = [];

// ===== State Driven Processing =====
var state = 0;

function handleState() {
  if (state == 0) {
    document.getElementById("start").style.display = "block";
    document.getElementById("end").style.display = "none";
    document.getElementById("drag").style.display = "none";
  }
  if (state == 1) {
    document.getElementById("start").style.display = "none";
    document.getElementById("end").style.display = "block";
    document.getElementById("drag").style.display = "none";
  }
  if (state == 2) {
    document.getElementById("start").style.display = "none";
    document.getElementById("end").style.display = "none";
    document.getElementById("drag").style.display = "block";
  }
}

handleState();

GEvent.addListener(map, "click", function(overlay,point) {
  if (point) {
    if (state == 1) { doEnd(point) }
    if (state == 0) { doStart(point) }
  }
});

function doStart(point) {
  createMarker(point,0,icon1);
  path[0] = point;
  state = 1;
  handleState();
}

function doEnd(point) {
  createMarker(point,4,icon3);
  path[4] = point;
  state = 2;
  handleState();
  for (var i=1; i<4; i++) {
    var lat = (path[0].lat()*(4-i) + path[4].lat()*i)/4;
    var lng = (path[0].lng()*(4-i) + path[4].lng()*i)/4;
    var p = new GLatLng(lat,lng);
    createMarker(p,i,icon4);
    path[i] = p;
  }
  bounds.extend(path[0]);
  bounds.extend(path[4]);
  map.setZoom(map.getBoundsZoomLevel(bounds));
```

```
        map.setCenter(bounds.getCenter());
      }

      var gdir=new GDirections(null, document.getElementById("path"));

      GEvent.addListener(gdir,"error", function() {
        var code = gdir.getStatus().code;
        var reason="Code "+code;
        if (reasons[code]) {
          reason = "Code "+code +" : "+reasons[code]
        }
        alert("Failed to obtain directions, "+reason);
      });

      var poly;
      GEvent.addListener(gdir, "load", function() {
        if (poly) map.removeOverlay(poly);
        poly = gdir.getPolyline();
        map.addOverlay(poly);
      });


      function directions() {
        if (addresses[0]) {var a = addresses[0] + "@" +
path[0].toUrlValue(6)}
          else {var a = path[0].toUrlValue(6)}
        if (addresses[4]) {var b = addresses[4] + "@" +
path[4].toUrlValue(6)}
          else {var b = path[4].toUrlValue(6)}
        for (var i=3; i>0; i--) {
          if (active[i]) {
            b = path[i].toUrlValue(6) +" to: "+b;
          }
        }
        var a = "from: "+a + " to: " + b;
        gdir.load(a, {getPolyline:true});
      }
    }

    // display a warning if the browser was not compatible
    else {
      alert("Sorry, the Google Maps API is not compatible with this
browser");
    }

    // This Javascript is based on code provided by the
    // Community Church Javascript Team
    // http://www.bisphamchurch.org.uk/
    // http://econym.org.uk/gmap/

    //]]>
    </script>
  </body>

</html>
```

Add.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<%@ page language="java" %>
<%@ page import="etc.automat.logic.Parameters" %>
<%@ page import="etc.automat.logic.Parameter" %>
<%@ page import="java.util.Vector" %>
<%@ page import="etc.beans.table.TableBO" %>
<%@ page import="etc.automat.ui.jsp.JSPConstants" %>
<%@ page import="etc.beans.table.RowBO" %>
<%@ page import="etc.beans.table.CellBO" %>
<%@ page import="etc.lang.Messages" %>
<%@ page import="etc.lang.MessageFactory" %>
<%@ page import="etc.beans.location.LocationBO" %>
<%@ page import="etc.options.LocationSettings" %>
<%@ page import="etc.io.ClassFactory" %>
<%@ page import="etc.beans.user.Rights"%>
<%@ page import="etc.util.DateTime"%>
<%@ page import="etc.options.settings.SettingSections"%>
<%@ page import="etc.io.location.LocationIO"%>
<%@ page import="etc.route.RouteBO"%>
<%@ page import="etc.route.RoutesBO"%>
<%@ page import="etc.io.car.CarParkIO"%>
<%@ page import="etc.automat.logic.service.ServiceLogicElement"%>
<%@ page import="etc.automat.access.AccessControlList" %>
<%@ page import="etc.automat.access.Subject" %>
<%@ page import="etc.automat.access.GroupType" %>
<%@ page import="etc.automat.ui.jsp.SessionInfo"%>
<%@ page import="etc.io.user.CostUnitIO" %>
<%@ page import="etc.beans.user.CostUnitBO"%>
<%@ page import="etc.util.QuickSort"%>
<%@   page import="java.util.Enumeration" %>
<%@ page import="etc.automat.access.AccessControlMatrix" %>
<%@ taglib prefix="etc" uri="/WEB-INF/tags/tags.tld" %>
<%!
JSPConstants con = JSPConstants.getInstance();

public String makeButton(String text, String action,int size) {
      //Standardmal for aa lage knapper
    String siz = size == 1 ? "nor" : "Xbig";
    String img = size == 1 ? "buttonCA" : "buttonXbig";
    String t = "<div class=\"relposx\"><div class=\"abspostxt" + siz + "\"
onclick=\"javascript:" + action + ";\">" + text + "</div>";
    t += "<div class=\"absposx\"><img onclick=\"javascript: " + action +
";\" class=\"" + siz + "\" src=\"" + con.BASEURL + "images/" + img +
".gif\"></div>";
    return t;
}
%>
<%
       Parameters params = (Parameters) session.getAttribute("params");

Object siO=session.getAttribute("sessionInfo");
SessionInfo si = null;
if(siO != null) {
      if(siO instanceof SessionInfo) {
            si = (SessionInfo) siO;
      }
```

```
}
	Messages m = si.getJspMsg();
	JSPConstants con = JSPConstants.getInstance();
	Parameter testParam = params.getParam("Rutedata");
	String s = "";
	if(testParam != null) {
		s = testParam.getDataS();
	}

	String l = "", ladr = "";
	Integer n = 0;
	l = si.getLocation().getLocation(); //Henter string med location
	n = si.getLocation().getLocID();    //Henter int med location id
	ladr = si.getLocation().getPhysicalAdress();
		//Henter ut string med location sin adresse

	Parameter adr = params.getParam("Adresse");
		//Lager array for bilene adm. kan bestemme ov
	String a ="";
	if(adr != null)   {
		a=adr.getDataS();
	}

	Parameter sortByP = params.getParam("sortBy");
		//Henter rekke å sortere på

	 String sortByS = "0";
	 if(sortByP != null) { sortByS = (String) sortByP.getData(); }

	 int sortBy = Integer.parseInt(sortByS);

	 String sortMethod = "ASC";
	 if(params.getParam("sortMethod") != null) {
		sortMethod = (String) params.getParam("sortMethod").getData();
	}

	 boolean desc = false;
	 if(sortBy < 0 ) {
		desc = true;
		sortBy *= -1;
	 }

	 TableBO table = new TableBO();
	 Parameter tableP = params.getParam("ta");

	 if(tableP != null) {
		 if(tableP.getData() != null) {
			table = (TableBO) tableP.getData();
		 }
	 }
			//Arrayer for å hente ut data fra tabellen:
	 Integer antStops = table.getSize(); //Antall stoppesteder i tabellen
	 String stops[];                     //Array med stoppestedene
	 stops = new String[antStops];
	 String tasklist[];                 //Array med gjøremål
	 tasklist = new String[antStops];
	 String namelist[];                 //Array med navn
	 namelist = new String[antStops];
	 String start[];              //Array med ankomsttidspunkter
	 start = new String[antStops];
```

```
        String stop[];                      //Array med avresetidspunkter
        stop = new String[antStops];
        String lat_kol[];                   //Arrray med lengdegrader
        lat_kol = new String[antStops];
        String lng_kol[];                   //Array med breddegrader
        lng_kol = new String[antStops];

        String nvn ="";
        nvn= table.getName();

        RoutesBO roubo = new RoutesBO();//Lager et temp RoutesBO objekt
      Parameter par = params.getParam("Routes");
            //Henter objektet fra DoRoute
      if(par != null) {              //Hvis objekt ikke er null
            roubo = (RoutesBO) par.getData();
            //Henter selve data fra objektet
       }
 %>

<head>
      <meta http-equiv="Content-Language" content="no-bok">
      <jsp:include page="/theme/themestyle.jsp"/>
      <meta http-equiv="content-type" content="text/html" ; />
     <script src="http://maps.google.com/maps?file=api&amp;
                  v=2&amp;
                  hl=no&amp; <!-- Norsk -->

      key=ABQIAAAA8KOsuvYhoFQfghm1YnDV7RTiqzI4mvhSfaT5XWRJCcZO71UsTRSXoRKaP
ZI_3CyujQ4wJqkUCS_zAw"
            type="text/javascript"></script>
</head>

<body onload="initialize();" onunload="GUnload()">

<div id="help" class="help">
      <script language="javascript" type="text/javascript">
            initHelp();
            function changeLocation(e) {
                  document.changeLocationF.reportLocation.value =
                  e.options[e.selectedIndex].value;
                  document.changeLocationF.submit();
            }
      </script>
</div>

<!--LAGER TIME-OUT FUNKSJON -->
<script language="javascript">
      initHelp();
      function timeoutmethod() {
            alert('<%=m.getString("settings.timeoutwarning")%>');//Viser et
vindu med advarsel
            window.location.href = '<%=con.BASEURL%>';
      //Gaar tilbake til start-side for CarAdmin
      }
      //setTimeout("timeoutmethod()",<%=5*60*1000%>);
      //Setter hvor lenge til timeout skjer
</script>

<!--LAGER KNAPP FOR Å LAGRE  -->
<script language="javascript" type="text/javascript" >
```

```
	initHelp();

				//Informasjonen i tekstfeltene blir lagret

	function addSave() {
		document.goRegRoute.submit();
	}
</script>

<form name="goRegRoute" action="<%=con.BASEURL%>" method="post" >
	<input type="hidden" name="logic" value="Service" />
	<%if( roubo.getrouteModus()==false) {%>		<!-- Hvis dette
er en gammel rute som har blitt redigert -->
	<input type="hidden" name="target" value="comRouteE" /> <%} else { %>
	<!-- Hvis dette er en ny rute -->
	<input type="hidden" name="target" value="comRoute" />	<% } %>
	<input type="hidden" name="durations" value="0"/>
	<input type="hidden" name="stays" value =""/>
</form>

<!-- LAGER ALERT TIL KNAPPEN AVBRYT -->
<script language="javascript" type="text/javascript" >
	initHelp();
		//Bruker blir advart om han/hun vil avslutte
	function sure() {
		if(confirm('<%=m.getString("handleroute.AreYouSure")%>'))

			document.goBackFunctions.submit();
			//Går tilbake til startside
	}
</script>
<form name="goBackFunctions" action="<%=con.BASEURL %>" >		<!--
form navnet det samme som document....submit() -->
	<input type="hidden" name='logic' value='Service' /> <!-- dette er
standard på alle -->
	<input type="hidden" name="target" value="goBackFunctions" />	<!--
value her skal være det samme som i Service.java -->
</form>

<!-- LAGER KOBLING TIL KNAPPEN NYTT STOPPESTED -->
<script language="javascript" type="text/javascript">
	function addStop() {  //Funksjon kjører ved trykk på knapp addStop
		document.goStop.submit()		//Går videre til goStop
	}
</script>

<form name="goStop" action="<%=con.BASEURL %>" >
<!-- form navnet det samme som document....submit() -->
	<input type="hidden" name='logic' value='Service' />
<!-- dette er standard på alle -->
	<input type="hidden" name="target" value="GotoStopN" />
<!-- value her skal være det samme som i Service.java -->
</form>

<jsp:include page="/theme/themeheader.jsp"/>
<form action="<%=con.BASEURL%>" method="post" name="changeLocationF">
	<input type="hidden" name="target" value="InitTableMenu" />
	<input type="hidden" name='logic' value='Service' />
	<input type="hidden" name="reportLocation" value="0" />
</form>
```

```
<etc:LeftMenu></etc:LeftMenu>
<div class="heading"
id="heading"><h1><%=m.getString("handleroute.AddRoute") %></h1>
</div>

<div class="pagehelpStyle" id="pagehelp">
      <A href="#" onMouseOver="popup('<FONT
color=black><%=m.getString("add.help")%><FONT>')"
onMouseOut="removeBox()">
            <IMG border="0" width="15px" height="15px" src="<%=con.BASEURL
%>images/help.gif">
      </A>
</div>

<div class="main" id="main"> <%=m.getString("handleroute.datenow")%>  <%=s
%>  <br/>

<!-- FORBEREDER STOPPESTEDSTABELL -->
<% RowBO headerRow = new RowBO();%>
<% if(table != null) { %>
<% if(table.getSize() == 0) { %>
      <%=m.getString("drivingbook.Nostatisticdataforselectedperiod")%>
<% } else {

%>

<!-- SLETTER ET STOPPESTED -->
      <form name="delStop" action="<%=con.BASEURL %>" method="post" >
            <input type="hidden" name="logic" value="Service" />
            <input type="hidden" name="thisStop" value="-1" />
            <input type="hidden" name="target" value="DeleteStop" />
      </form>

<!-- FORANDRER ET STOPPESTED -->
      <form name="chStop" action="<%=con.BASEURL %>" method="post" >
            <input type="hidden" name="logic" value="Service" />
            <input type="hidden" name="thisStop" value="-1" />
            <input type="hidden" name="target" value="PrepareEditStop" />
      </form>

<!-- LAGER TABELL OVER STOPPESTEDENE TIL RUTEN -->
      <p><%=m.getString("homepoint")%> <%= l + " (" + ladr + ") " %></p>

      <table class="report" align ="center">
            <tr>
                  <td class="TableHeaderOverview"> # </td>

                  <td class="TableHeaderOverview">
<%=m.getString("handleroute.time")%></td>
                  <td class="TableHeaderOverview">
<%=m.getString("handleroute.stopTime") %></td>
                  <td class="TableHeaderOverview">
<%=m.getString("handleroute.stopName") %></td>
                  <td class="TableHeaderOverview">
<%=m.getString("handleroute.adress") %></td>
                  <td class="TableHeaderOverview">
<%=m.getString("handleroute.task") %></td>
                  <td class="TableHeaderOverview"><%=m.getString("Edit")
%></td>
```

```
                <td class="TableHeaderOverview"><%=m.getString("Delete")
%></td>
            </tr>
            <tr>
                <td class="headerline" colspan="12"></td>
            </tr>
                <% int iee = 0, iea=0, iei=0, ieo=0, ieu=0;  //mulig å
bruke bare en?
                for(int j=0;j<antStops;j++) { %>
                    <% RowBO r = table.getRowAt(j); %>

                    <%
                //farge?
                    String bgcolor = "";
                    String firstColor = "";
                    String route_no = r.getCellAt(0).getParamValue();
                    String rll = r.getCellAt(r.getSize()-1).getData();
                //String mul = r.getCellAt(11-offset2).getParamValue();
                    lat_kol[j] = r.getCellAt(6).toString();
                    lng_kol[j] = r.getCellAt(7).toString();
                    if(j % 2 == 1) {
                        bgcolor = "bgcolor=\"EDECE2\"";
                    }
                    %>

            <tr class="report" <%=bgcolor%> align='center'>
                    <%
                    for(int k=0;k<6;k++) {
                        CellBO cell = r.getCellAt(k);

                //Henter ut stoppestedsinformasjon:
                        if(k==1){
                //henter ut ankomststidspunktet
                            start[ieo] = cell.toString();
                            ieo++;
                        }
                        if(k==2){
            //Henter ut avreisetidspunktet
                            stop[ieu] = cell.toString();
                            ieu++;
                        }
                        if( k == 3){
            //Henter ut navn på person/bedrift som besøkes
                            namelist[iei] = cell.toString();
                            ++iei;
                        }
                        else if( k == 4){
            //Henter ut adressen til stoppestedet
                            stops[iee] = cell.toString();
                            ++iee;
                        }
                        else if( k == 5){
            //Henter ut adressens gjøremål
                            tasklist[iea] = cell.toString();
                            ++iea;
                        }%>
                    <% String link = cell.getLink(); %>
                    <% String cellData = cell.getData(); %>
                    <% String style = cell.getStyleclass(); %>
```

```
                                    <% if(style == null) { style = ""; } %>
                                    <% String popup = cell.getParamName(); %>
                        <td><%=cellData %></td>
                                     <%} %>
                        <td><a href="#"
onMouseOver="popup('<%=m.getString("Edit")%>')"
onMouseOut="removeBox()"onclick="javascript: askChange(<%=
r.getCellAt(0).getData() %>);">
                        <img width="16" height="16"
src="<%=con.BASEURL%>images/edit.gif" ></a> </td>

                  <td><a href="#"
onMouseOver="popup('<%=m.getString("Delete")%>')"  onMouseOut="removeBox()"
onclick="javascript: askDelete(<%=r.getCellAt(0).getData() %>);">
                  <img width="16" height="16"
src="<%=con.BASEURL%>images/delete.gif" ></a> </td>
                  </tr>
                                    <% } %>
            </table>
                  <% } } %>
                  <% if(table == null) { %>
            <table>
                  <tr><td><%=m.getString("NoRightOnThisPage")%></td></tr>
                  <tr><td class="spaceCellHalf"></td></tr>
                  <tr><td class="spaceCellHalf"></td></tr>
            </table>
<% } %>

<!--LAGER OG SETTER KNAPPENE ETTER HVERANDRE-->
            <table class="twosmallbutton" align= "center">
                  <tr>
                        <td><div>
<%=makeButton(m.getString("Cancel"),"sure()",1)%></div></td>     <!-- Lager
knapp -->
                        <td class="spaceCell"></td>
                        <td><div>
<%=makeButton(m.getString("Save"),"addSave()",1)%></div></td> <!-- Lager
knapp -->
                        <td class="spaceCell"></td>
                        <td><div>
<%=makeButton(m.getString("handleroute.newStop"),"addStop()",1)%>
</div></td>  <!-- Lager knapp -->
                        <td class="spaceCell"></td>
                  </tr>
            </table><br></br>

<!-- LAGER KART -->
            <table class="map" align ="center" >
                  <tr><td><div id="map_gjovik" style="width: 600px; height:
350px"></div></td></tr> <!--  Kartet -->
                  <tr><td><div id="path" style="width: 600px; height:
350px"></div> </td></tr>           <!--  veibeskrivelsen -->
            </table>

<!-- SCRIPT TIL KARTET  -->
            <script type="text/javascript">
            if (GBrowserIsCompatible()) { //sjekker om nettleseren kan vise kartet
                  var addresses = [];     //Array med stoppestedenes adresse
                  var path = [];          //Array med stoppestedenes koordinater
                  var taskList = [];      //Array med stoppestedenes gjøremål
```

```
            var nameList = [];      //Array med stoppestedenes navn
            var startstop = [];//Array med stoppestedenes ankomst og
avreisetidspunkter
            var departure = [];//Array med stoppestedenes avreisetidspunkt
            var arrival = []; //Array med stoppestedenes ankomsttidspunkt
            var geo = new GClientGeocoder(new GGeocodeCache());//Geocoder:
gjør om mellom adresse og koordinat
            var bounds = new GLatLngBounds();//Grense: Hva skal kartet
zoomes og sentreres på
            var numberOfAddresses = <%= antStops %>; //Antall stoppesteder,
hentet fra tabell
            var map;

            var reasons=[];         //Array med feilmeldinger
reasons[G_GEO_SUCCESS]='<%=m.getString("G_GEO_SUCCESS")%>';
reasons[G_GEO_MISSING_ADDRESS]='<%=m.getString("G_GEO_MISSING_ADDRESS")%>';
reasons[G_GEO_UNKNOWN_ADDRESS]='<%=m.getString("G_GEO_UNKNOWN_ADDRESS")%>';
reasons[G_GEO_UNAVAILABLE_ADDRESS]='<%=m.getString("G_GEO_UNAVAILABLE_ADDRE
SS")%>';
reasons[G_GEO_BAD_KEY]= '<%=m.getString("G_GEO_BAD_KEY")%>';
reasons[G_GEO_TOO_MANY_QUERIES]='<%=m.getString("G_GEO_TOO_MANY_QUERIES")%>
';
reasons[G_GEO_SERVER_ERROR]= '<%=m.getString("G_GEO_SERVER_ERROR")%>';
reasons[G_GEO_BAD_REQUEST]= '<%=m.getString("G_GEO_BAD_REQUEST")%>';
reasons[G_GEO_MISSING_QUERY]= '<%=m.getString("G_GEO_MISSING_QUERY")%>';
reasons[G_GEO_UNKNOWN_DIRECTIONS]='<%=m.getString("G_GEO_UNKNOWN_DIRECTIONS
")%>';

var baseIcon = new GIcon(G_DEFAULT_ICON); //ikoner for merking på kart:
baseIcon.iconSize=new GSize(24,38);       //ikonenes størrelse
            var icon1 = G_START_ICON;

            var icon2 = G_PAUSE_ICON;
            var icon3 = G_END_ICON;
            var stay = "";

function createMarker(point,i, icon, html) {    //Tegner ikon på kartet
      var marker = new GMarker(point, {draggable:false, icon:icon});
      //lager markøren, med riktig ikon
      GEvent.addListener(marker, "click", function() {//dersom markøren
klikkes
      marker.openInfoWindowHtml(html);//åpne boble med informasjon om
stoppestedet
                });
                map.addOverlay(marker);   //Legger ikonet på kartet
            }

function initialize() {
      map = new GMap2(document.getElementById("map_gjovik"));//initierer
kartet
              //map.addControl(new GLargeMapControl());
            //Mulig for annet brukergrensesnitt på zoom osv.
                  map.setUIToDefault();
            //Legger på brukergrensesnitt; zoom og hybrid/satelittbilde
                  var k = 0;
                                    //Tellevariabel, k er erstatter for i.

<% String adre = "", tl = "", name ="", st ="",strt="", stp="", lt="",
lg="";      //Dummyvariable for kopiering av array mellom java og
javascript
```

```
                  for( int i=0; i<antStops; i++){
                  //går igjennom stoppestedene, henter ut fra javaarrayene:
                        adre = stops[i];        //Adressen
                        tl = tasklist[i];       //gjøremål
                        name = namelist[i];     //navn
                        st = start[i] + " - " + stop[i];//ankomst- og
avreisetidspunkt
                        strt = start[i];        //ankomst
                        stp = stop[i];          //avreise
                        lt = lat_kol[i];        //lengdegrad
                        lg =lng_kol[i];
                                          //Breddegrad
                        %>

                                          //Legger inn i javascriptarrayene
                        addresses[k] = '<%=adre%>';

                        taskList[k]= '<%=tl%>';

                        nameList[k] = '<%=name%>';
                        startstop[k] = '<%=st%>';
                        arrival[k] = '<%=strt%>';
                        departure[k] ='<%=stp%>';
                        var lt = '<%=lt%>';
                        var ln = '<%=lg%>';
                        pont = new GLatLng(lt, ln);
                        if (k==0){
                        //for første stoppested: starticon, legg inn
koordinater i path-array
                              createMarker(pont, k, icon1, "<b>" +
'<%=m.getString("add.startplace")%>'+ ": </b><br/>" + addresses[0] +
"<br/>" + departure[0] );
                              path[k] = pont;
                        }
                        else if (k==numberOfAddresses-1) {
            //for siste stoppested: stoppicon, legg inn koordinater i path-
array, håndter grenser og kall på tegningen av ruten
                              createMarker(pont, k, icon3, "<b>" +
'<%=m.getString("add.stopplace")%>' +": " + nameList[k] + "</b><br/>" +
addresses[k] + "<br/>" + taskList[k] + "<br/>" + startstop[k]);
                              path[k] = pont;
      bounds.extend(path[0]); //setter grensenes første hjørne
      bounds.extend(path[numberOfAddresses-1]);//setter grensenes andre
hjørne
      map.setZoom(map.getBoundsZoomLevel(bounds)-1);       //setter kartets
zoomnivå
      directions();
      //Tegner streken for kjøreruten og henter veibeskrivelsen
      }
      else{
      //For alle stoppesteder mellom start og stopp
            createMarker(pont, k, icon2, "<b>" + nameList[k] + "</b><br/>"
+ addresses[k] + "<br/>" + taskList[k] + "<br/>" + startstop[k]);
                              path[k] = pont;
            }
            if(bounds)map.setCenter(bounds.getCenter());//dersom grensen er
definert, sentrer kart på ruten
```

```
            else map.setCenter(point,14);            //ellers: sentrer kart
på siste punkt
                    stay = stay + arrival[k] + "@" + departure[k];
      //legger stoppestedenes tidspunkter i en streng. skilles med @
                    if(k <numberOfAddresses-1) stay = stay + "@";
      //dersom ikke siste, legg på skilletegn.
                          k++;  //øk javascript-løkka sin tellevariabel
                          <%
                    }
                                //slutt for-løkke
                    %>



          function directions(){    //Tegner kjøreruten
            var a = "", b = "" , i = 0;
            //dummystrenger for vei, legges inn baklengs, og teller
                if (addresses[numberOfAddresses-1]) {
      //for siste stopp: legg inn i rekka, avhengig av addresse
                          b = addresses[numberOfAddresses-1] + "@" +
path[numberOfAddresses-1].toUrlValue(6);
                    }
                else {
                          b = path[numberOfAddresses-1].toUrlValue(6);
                    }
                if (addresses[0]) {
            //for første stopp: legg inn i rekka, avhengig av addresse
                          a = addresses[0] + "@" + path[0].toUrlValue(6);
                }
                else{
                          a = path[0].toUrlValue(6);
                }
                for (i=(numberOfAddresses-2); i>0; i--)
            //for mellomstopp: legg inn i rekka.
                        b = path[i].toUrlValue(6) +" to: "+b;
                a = "from: " + a + " to: " + b;
            //samler alt i en streng

gdir = new GDirections(null, document.getElementById("path"));   //Googles:
finner ruten ut fra stoppestedene + veibeskrivelse
                    gdir.load(a, {getPolyline:true});     //tegner rute-streken
                      var poly;                       //variabel for rute-streken
                      var time;         var sec=0;  //variabel for rutens
kjøretid
             GEvent.addListener(gdir,"error", function() { //Dersom ruten ikke
kan finnes:
            var code = gdir.getStatus().code;   //feilmelding
            var reason='<%=m.getString("gdir.error.code")%>' +" "+code;
            if (reasons[code]) {
                  reason ='<%=m.getString("gdir.error.code")%>' +" "+code
+" : "+reasons[code];
                        }
                        alert('<%=m.getString("gdir.error.alert")%>'+", " +
reason);
                    });

      GEvent.addListener(gdir, "load", function() {  //tegner ruten på
kartet
            if (poly) map.removeOverlay(poly);  //fjerner evt gammel strek
                    poly = gdir.getPolyline();  //henter den nye streken
```

```
                    map.addOverlay(poly);       //legger ny rute på kartet
                    time = gdir.getDuration();  //henter ut tiden det tar å
kjøre ruten
                  if(time.html != "undefined"){
                 //alert(time.html + "<br/>" + time.seconds);
                        sec = time.seconds;    //antall sekunder kjøretiden
varer
                        document.goRegRoute.durations.value = sec;
//legger inn i skjult tekstfelt, hentes ut i addRoute.java eller
editRoute.java
                        document.goRegRoute.stays.value = stay;
//legger inn stoppestedenes tidsintervaller som en streng. hentes ut i
addRoute og editRoute.
                      }
                      else {

alert('<%=m.getString("gdir.load.alert.time")%>');
                      }
                  });
          }
                      //end directions()


      }     }
                            // end: if (GBrowserIsCompatible())
      // display a warning if the browser was not compatible
      else
                      //Kan ikke vise kartet
            alert('<%= m.getString("gmap.error")%>');
      // This JavaScript is based on code provided by
      // http://econym.org.uk/gmap/

      </script>
      <script>
            function askDelete(deleteID)
            {

      if(confirm('<%=m.getString("handleroute.VerifyDeleteStop")%>')) {
                        document.delStop.thisStop.value= deleteID;
                        document.delStop.submit();
                  }
            }

            function askChange(changeID)
            {
                        document.chStop.thisStop.value= changeID;
                        document.chStop.submit();
            }
      </script>
</div>
</body>
</html>
```

RouteIOSQL

```
public class RouteIOSQL extends RouteIO {

   /**
    /**
     * @return objekt med informasjon om rute.
     * @throws IOException hvis noe går galt under lesingen.
     * @see etc.io.car.RouteIO#getRoute(String route_no)
     */
    public RouteBO getRoute(int route_no)  {

      RoutesBO rsbo = new RoutesBO();           //Lager en ny RoutesBO
            RouteBO rbo = new RouteBO();        //Lager en ny RouteBO

      //Lager en spørring
            String Q = "SELECT homepoint, begin_time, stop_time, tot_time,
tot_stops FROM Routes WHERE route_no=" + route_no + " LIMIT 1";


        Connection c = SQLConnector.getConnection(); //Oppretter kobling
        ResultSet rs = null;                         //Lager resultatsett
        ResultSet stop = null;

        try {
            rs = c.createStatement().executeQuery(Q);//Kjører spørring

            if(rs.first() ) {       //Tar det førstei resultatsettet
                String homepoint = rs.getString("homepoint");//Leser inn
                String begin_time = rs.getString("begin_time");
                String stop_time = rs.getString("stop_time");
                int tot_time = rs.getInt("tot_time");
                int tot_stops = rs.getInt("tot_stops");

            //Setter data til riktig sted
                rbo.setHomepoint(homepoint);
                rbo.setBegin_time(new
DateTime(begin_time,SystemPreferences.get("date_format")));
                rbo.setStop_time(new
DateTime(stop_time,SystemPreferences.get("date_format")));
                rbo.setTot_time(tot_time);
                rbo.setTot_stops(tot_stops);
            }

                return rbo;                          //Returnerer objekt
        }catch(SQLException sqle) {             //Logger feil
            logging.warn("SQL feil!",sqle);
            return rbo;
        } catch(NullPointerException npe) {
            logging.warn("Nullpekerfeil",npe);
            return rbo;
        }
        finally {                   //Lukker koblinger
            try { rs.close(); } catch(Exception e) {}
            try { stop.close(); } catch(Exception e){}
            try { c.close(); } catch(Exception e) {}
            rs = null;
            stop = null;
            c = null;
```

```
        }

    }

/**
     * <p> Putter en ny rute i databasen </p>
     * @param RouteBO ro
     * @return tilstandsobjekt
     * @see etc.route.io.RouteIO#putRoute(etc.route.RouteBO)
     */
    @Override
    public IOState putRoute(RouteBO ro, Boolean mod) {
      String df = "HH:mm";          //Datoformat
      RouteBO o = getDatesR(ro.getRouteNo(), mod);
      Connection c = SQLConnector.getConnection();
//Får kontakt med databasem
      String statement ="";
        try {

              if(ro == null)        //Hvis det ikke er sendt med noen rute
                  return IOState.IO_REMOVED;

              if(mod==true) {
               statement = "INSERT INTO Routes " +
//Lager spørring for å legge inn rute i tabell
              "(homepoint, begin_time, stop_time, tot_time, tot_stops,
route_no , name)" +
              "VALUES('";
              statement += o.getHomepoint() + "','" +

o.getBegin_time().getFormatedDate(SystemPreferences.get("date_format")) +
"','" +

o.getStop_time().getFormatedDate(SystemPreferences.get("date_format")) +
"','" +
                           ro.getTot_time() + "','" +
                           ro.getTot_stops() + "'," +
                           ro.getRouteNo() + " , '" +
                           ro.getName() +
                           "')";
              }
              else {
                  String begin_time="0000-00-00 " +
o.getBegin_time().getFormatedDate(df);
                  String stop_time ="0000-00-00 " +
o.getStop_time().getFormatedDate(df);
                   statement = "INSERT INTO Routes " +
            //Lager spørring for å legge inn rute i tabell
                    "(homepoint, begin_time, stop_time, tot_time,
tot_stops, route_no , name)" +
                    "VALUES('";
                    statement += o.getHomepoint() + "','" +
                                 begin_time + "','" +
                                 stop_time + "','" +
                                 ro.getTot_time() + "','" +
                                 ro.getTot_stops() + "'," +
                                 ro.getRouteNo() + " , '" +
                                 ro.getName() +
                                 "')";
              }
```

```
            logging.info("statement: " + statement);//Logger spørringen
            c.createStatement().execute(statement);
      //Kjører spørringen mot databasen


            return IOState.IO_OK;       //Returnerer ok
        }
        catch(SQLException sqle) {  //Logger ect. feil
            logging.warn("SQL exception oppsto i putRoute!",sqle);
        } catch(NullPointerException npe) {
            logging.warn("Nullpekerfeil i putRoute!",npe);
        }
        finally {
            try { c.close(); } catch(Exception e) {} //Lukker spørringen
            c = null;
        }

        return IOState.IO_PUT_ERROR;
    }



    /**
     * <p>Henter et stoppested fra databasen </p>
     * @return objekt med generell informasjon om stoppested.
     * @throws IOException hvis noe går galt under lesingen.
     * @see etc.io.car.RouteIO#getStop(String stop_no)
     */
    @Override
      public StopBO getStop(int sid, int rid) {
      StopBO sbo = new StopBO();
      //Lager et stopp objekt

            //Lager en spørring
            String R = "SELECT meet, stay, notes, adress, name, lat, lng
FROM Stops WHERE stop_no=" + sid + " AND r_no=" + rid;

          Connection c = SQLConnector.getConnection();
//Kobler til databasen
        ResultSet rs = null;  //Lager et resultatsett

        try {

            rs = c.createStatement().executeQuery(R);
//Kjører spørring for å hente resultater

            if(rs.first() ) {
      //Henter ut fra resultatsett
                System.out.println("Inne i getStop try");
              String meet = rs.getString("meet");
              String stay = rs.getString("stay");
              String notes = rs.getString("notes");
              String adress = rs.getString("adress");
              String name = rs.getString("name");
              Float lat = rs.getFloat("lat");
              Float lng = rs.getFloat("lng");

      //Konverterer til DateTime i riktig format
```

```
                sbo.setMeet(new DateTime(meet,
SystemPreferences.get("date_format")));
                sbo.setStay(new DateTime(stay,
SystemPreferences.get("date_format")));

      //Setter dataene i stoppe objektet
                sbo.setNotes(notes);
                sbo.setAdress(adress);
                sbo.setName(name);
                sbo.setStopNo(sid);
                sbo.setRouteNo(rid);
                sbo.setLat(lat);
                sbo.setLng(lng);

            }

            return sbo;       //Returnerer stoppestedet

        } catch(SQLException sqle) {      //Logger
            logging.warn("Feil med sql i getStop, kanskje med" + R, sqle);
        } catch(NullPointerException npe) {
            logging.warn("Nullpekerfeil I getStop", npe);

        }
        finally {       //Lukker kobling mot database
            try { rs.close(); } catch(Exception e) {}
            try { c.close(); } catch(Exception e) {}
            rs = null;
            c = null;
        }

        return null;    //Ellers returneres null
    }

       /**
       * <p> Putter et stoppested i databasen </p>
     * @param StopBO so
     * @return tilstandsobjekt IOState
     * @see etc.route.io.RouteIO#putStop(etc.route.StopBO)
     */

    public IOState putStop(StopBO so, Boolean mod) {
      Connection c = SQLConnector.getConnection();         //Oppretter
kobling mot database
      String statement ="";

        try {
              if(so == null)  //Hvis Stopp objekt so er lik null return feil
                  return IOState.IO_REMOVED;

              if(mod==true) { //Hvis dette er en rute som er datobasert
               statement = "INSERT INTO Stops " +
//Lager spørring for å legge til stop
              "(meet, stay, notes, adress, name, r_no, lat, lng, stop_no )"
+
              "VALUES('";     //Tar med hele datoene og tidspunktet
              statement +=
so.getMeet().getFormatedDate(SystemPreferences.get("date_format")) + "','"
+
```

```
so.getStay().getFormatedDate(SystemPreferences.get("date_format")) + "','"
+
                            so.getNotes() + "','" +
                            so.getAdress() + "','" +
                            so.getName() + "'," +
                            so.getRouteNo() + "," +
                            so.getLat() + "," +
                            so.getLng() + "," +
                            so.getStopNo() + ")";
                }
                else {    //Hvis dette er en fri rute
                          //Sett datoer til null og ta bare med tidspunkter
                    String tida = "0000-00-00
"+so.getMeet().getHours()+":"+so.getMeet().getMinutes();
                    String tidb = "0000-00-00
"+so.getStay().getHours()+":"+so.getStay().getMinutes();

                    statement = "INSERT INTO Stops " +       //Lager spørring
                      "(meet, stay, notes, adress, name, r_no, lat, lng,
stop_no )" +
                      "VALUES('";
                      statement += tida + "','" + tidb + "','" +

                                  so.getNotes() + "','" +
                                  so.getAdress() + "','" +
                                  so.getName() + "'," +
                                  so.getRouteNo() + "," +
                                  so.getLat() + "," +
                                  so.getLng() + "," +
                                  so.getStopNo() + ")";
                }


                logging.info("statement: " + statement);//Logger spørring
                c.createStatement().execute(statement);      //Kjører
spørring mot database
                return IOState.IO_OK;       //Returnerer OK
            }
            catch(SQLException sqle) {       //Logger hvis noe er feil
                logging.warn("SQL exception oppsto i  putStop!",sqle);
            } catch(NullPointerException npe) {
                logging.warn("Nullpekerfeil i putStop!",npe);
            }
            finally {
                try { c.close(); } catch(Exception e) {} //Lukker kolbing
                c = null;
            }

            return IOState.IO_PUT_ERROR;      //Returnerer feil
        }


        /**
           * <p> Genererer en tabell over ruter på en viss dato </p>
         * @param Session info, String date
         * @return TableBO
         * @see etc.route.io.RouteIO#putStop(etc.route.StopBO)
         */
```

```
    public TableBO getTableRoute(SessionInfo sio, DateTime ada, DateTime
eda, Integer mod, Integer day) {
      String ad =  null;//Lager tidsstring, fra-tidspunktet for intervallet
      String ed = null; //Lager tidsstring, til-tidspunkt for intervallet

  //Intervallet er fra kl 00:00 første dag til 23:59 siste dag
    if(mod==1 || mod==3) {
//Hvis dette er en visning av ruter med dato eller dag
      ada.setHour(00); ada.setMinute(00); ada.setSecond(00);
      ad = ada.getFormatedDate(SystemPreferences.get("date_format"));
      eda.setHour(23); eda.setMinute(59); eda.setSecond(59);
      ed = eda.getFormatedDate(SystemPreferences.get("date_format"));
    }

    if(mod==2) {        //Fri visning av ruter
      ad= "0000-00-00 00:00:00";
      ed= "0000-00-00 23:59:59";

    }
//Lager spørring som henter alle ruter med tidspunkt fra og med ad, til og
med ed.
    String Q = "SELECT route_no, homepoint, tot_stops, stop_time,
begin_time, tot_time, name FROM Routes"
             + " WHERE begin_time >= '"+ ad + "' AND stop_time <= '" +ed
+"'ORDER BY begin_time ASC";

    if(mod ==3) {       //Lager spørring som henter alle rutene i databasen
    Q = "SELECT route_no, homepoint, tot_stops, stop_time, begin_time,
tot_time, name FROM Routes ORDER BY begin_time ASC";
    }

    TableBO t = new TableBO();                  //Lager en ny tabell
    t.setName("Route");                         //Setter navnet til tabellen

    Connection c = null;                        //Lager variable
      ResultSet brs = null;
      String totTime ="";


    c = SQLConnector.getConnection();     //Kjører en connect mot database

    LocationBO l;       //Får tak i formatet på datovisning fra LocationBO
      l = sio.getLocation();
    LocationSettings ss = ClassFactory.getFactory().getLocationSettings(l);
      String df = ss.UI_DATE_FORMAT;            //Setter df til datoformat
      String dfs = "HH:mm";


      try {
            brs = c.createStatement().executeQuery(Q);//Kjører en spørring

            if(brs.first() ) {      //Setter peker til første linje i tabell
                  do {

                        int route_no = brs.getInt("route_no");
//Leser fra resultat ev spørring
                        String homepoint = brs.getString("homepoint");
                        int tot_stops = brs.getInt("tot_stops");
                        String stop_time = brs.getString("stop_time");
                        String begin_time = brs.getString("begin_time");
```

```
                        int tot_time = brs.getInt("tot_time");
                        String routename = brs.getString("name");


                        Integer hour=0, min=0, sec=0;
//Dummyvariable for omregning av tid
                        hour= tot_time / 3600;  //Regner ut antall timer
                        min   =(tot_time -(hour*3600))/60;
//Regner ut antall minutter
                        sec   = tot_time - (hour*3600) - (min*60);//antall
sekunder
                        if (sec >=30) min+=1;
      //Runder av til nærmeste minutt
      totTime = hour.toString() + "h " + min.toString() + "min";
      //konverterer til tekststreng //språkfil på bokstavene??

//Lager DateTime av tidsstringene som har blitt lest inn over
DateTime stop = new
DateTime(stop_time,SystemPreferences.get("date_format"));
DateTime start = new
DateTime(begin_time,SystemPreferences.get("date_format"));


                        RowBO row = new RowBO();      //Lager ny linje

                        if(mod == 1 || mod==2) {
//Hvis rutene skal vises etter dato eller fritt
                        row.addCell(Integer.toString(route_no));
//Legger til celle i tabell
                        row.addCell(routename);
                        row.addCell(homepoint);
                        row.addCell(Integer.toString(tot_stops));
//Omgjør integer til string og setter i celle

                        if(mod==1) {
//Hvis dette er en datovisning av ruter vises hele tidspunktet samt dato

                              row.addCell(start.getFormatedDate(df));
//Omgjør DateTime til string og setter i celle
                              row.addCell(stop.getFormatedDate(df));
                        }

                        if(mod==2) {
//Hvis dette er en fri visning av ruter vises bare tidspunktene
                              row.addCell(start.getFormatedDate(dfs));
                              row.addCell(stop.getFormatedDate(dfs));
                        }

                        row.addCell(totTime);
                        t.addRow(row);    //Legger til rad i tabell
                        }
                        if(mod==3) {//Hvis dagsvisning av rutene
                              if(start.getDayOfWeek() == day) {
            //Hvis datoen er på spesifisert dag
                                    row.addCell(Integer.toString(route_no));
      //Legger til celle i tabell
                                    row.addCell(routename);
                                    row.addCell(homepoint);

      row.addCell(Integer.toString(tot_stops));
```

```
      //Omgjør integer til string og setter i celle
                                    row.addCell(start.getFormatedDate(df));
      //Omgjør DateTime til string og setter i celle
                                    row.addCell(stop.getFormatedDate(df));
                                    row.addCell(Integer.toString(tot_time));

                                    t.addRow(row);
                        //Legger til rad i tabell
      }
                        }
                  }
            } while(brs.next());
            //Mens det fremdeles ligger linjer i resultat


                  }
            logging.info("Rute tabell: "  +Q); //Logger spørringen
            return t;
            //Returnerer tabell-objekt
            }

                        catch (Exception e) {
      //Exception til try
            // TODO Auto-generated catch block
            //e.printStackTrace();
            logging.error("En feil oppsto under oppretting av
rutetabell",e);         //Logger feil
            return null;

      }
                        finally {   //Stenger spørringene til databasen
                              try { brs.close(); brs = null; }
catch(Exception e) {}
                              try { //Stenger databasen
                                    c.close();
                                    c = null;
                              } catch(Exception e) {}
                        }
    }

/**
       * <p> Genererer en tabell over alle stoppestedene til databasen </p>
     * @param SessionInfo sio, Integer r (rute_nummer)
     * @return TableBO
     * @see etc.route.io.RouteIO#putStop(etc.route.StopBO)
     */
      @Override
      public TableBO getTableStops(SessionInfo sio, Integer r, Boolean mod,
Integer sa) {
            TableBO ta = new TableBO();               //Lager en ny tabell
          ta.setName("Stops");//Setter navnet til tabellen

            String Q = "SELECT * from Stops WHERE r_no=" + r + " ORDER BY
meet ASC";              //Lager en spørring

          Connection c = null;
            ResultSet brs = null;         //Lager resultatsett

          c = SQLConnector.getConnection();//Kjører en connect mot database
```

```
          LocationBO l; //Får tak i datoformat fra LocationBO
            l = sio.getLocation();
          LocationSettings ss =
ClassFactory.getFactory().getLocationSettings(l);
            String df = ss.UI_DATE_FORMAT;//Setter df til datoformat
            String dfs = "HH:mm";

            try {
                  brs = c.createStatement().executeQuery(Q);
                        //Kjører en spørring

                  if(brs.first() ) {
      //Setter peker til første linje i tabell
                        do {
                              RowBO row = new RowBO();//Lager ny linje
                              int stop_no = brs.getInt("stop_no");//Leser
                                    //     fra resultat av spørring
                              String notes = brs.getString("notes");
                              String adress = brs.getString("adress");
                              String name = brs.getString("name");
                              String meet = brs.getString("meet");
                              String stay = brs.getString("stay");
                              float lat = brs.getFloat("lat");
                        float lng = brs.getFloat("lng");

                              DateTime stop = new
DateTime(stay,SystemPreferences.get("date_format"));
                              DateTime start = new
DateTime(meet,SystemPreferences.get("date_format"));

                              row.addCell(Integer.toString(stop_no));
      //Legger til celle i tabell

                              if(mod==true) {
      //Hvis dette er en rute med dato vises stoppstedenes dato også
                                    row.addCell(start.getFormatedDate(df));
      //Omgjør DateTime til string og legger i cell
                                    row.addCell(stop.getFormatedDate(df));
                                    }

                              else {
//Hvis dette er en fri rute vises bare tidspunktene for stoppestedene.
                                    row.addCell(start.getFormatedDate(dfs));
                                    row.addCell(stop.getFormatedDate(dfs));
                                    }


                              row.addCell(name);
                              row.addCell(adress);
                              row.addCell(notes);
                              row.addCell(Float.toString(lat));
                              row.addCell(Float.toString(lng));

                              ta.addRow(row);//Legger til raden i tabellen
                        } while(brs.next());
            //Mens det fremdeles ligger linjer i resultat


                        }
```

```
                return ta;
                //Returnerer tabell-objekt
                }

                            catch (Exception e) {   //Logger hvis feil
logging.error("En feil oppsto under oppretting av stopptabell",e);
                return null;

            }
                            finally {   //Lukker kolbing til database
                                  try { brs.close(); brs = null; }
      catch(Exception e) {}
                                  try {
                                        c.close();
                                        c = null;
                                  } catch(Exception e) {}
                            }
          }

      @Override
      public boolean isRoute(int stopNo, int routeNo) {
            // TODO Auto-generated method stub
            return false;
      }

       /**
        * <p> Fjerner en rute med alle dets stoppesteder fra databasen </p>
      * @param Integer id (rute nummer)
      * @return true-> fjernet, false-> ikke fjernet
      */
      @Override
      public boolean RemoveRoute(Integer id) {
            String Q = " DELETE FROM routes WHERE route_no =" + id;
            //Lager spørring mote rute
            String R = " DELETE FROM stops WHERE r_no =" + id;
            //Lager spørring mote stoppesteder
            Connection c = SQLConnector.getConnection();//Oppretter kobling

              try {     //Prøver å kjøre spørring mote rute
                  c.createStatement().executeUpdate(Q);//Sletter en rute
              } catch(Exception e) {      //Logger
                  logging.warn("Klarte ikke fjerne rute (" + Q + ")",e);
              }
              finally { //Lukker denne koblingen
                  try { c.close(); } catch(Exception e) {}
                  c = null;
              }

              c = SQLConnector.getConnection(); //Oppretter ny kolbing

              try {     //Pørver å kjøre spørring mot stoppesteder
                  c.createStatement().executeUpdate(R);     //Sletter
stoppestedene til ruten
                  return true;
              } catch(Exception e) {                        //Logger
             logging.warn("Klarte ikke fjerne stoppesteder(" + R + ")",e);
              }
              finally {
                        //Lukker kobling
                  try { c.close(); } catch(Exception e) {}
```

```
                c = null;
            }

            return false;

    }

     /**
     * <p> Fjerner et stoppested fra databasen </p>
    * @param Integer sid ( stoppestedsnummer), Integer rid ( rutenumer)
    * @return true-> fjernet, false-> ikke fjernet
    */
     public boolean RemoveStop(Integer sid, Integer rid) {
String R = " DELETE FROM stops WHERE r_no =" + rid + " AND stop_no =" + sid
;     //Lager spørring
      Connection c = SQLConnector.getConnection();//Oppretter kobling


            try {
                 logging.info("statement: " +      R);   //Logger
                c.createStatement().executeUpdate(R);
          //Kjører spørring mote database, fjerner stoppestedet
                return true;
            } catch(Exception e) {
                                //Logger evt. feil
           logging.warn("Klarte ikke fjerne stoppestedet(" + R + ")",e);
            }
            finally {
                                //Lukker kobling
                try { c.close(); } catch(Exception e) {}
                c = null;
            }

            return false;

    }


     /**
     * <p> Fjerner alle stoppesteder fra ruten i databasen </p>
    * @param Integer rid (rutenummer)
    * @return true-> fjernet, false-> ikke fjernet
    */
     public boolean RemoveStops(Integer rid) {
           String R = " DELETE FROM stops WHERE r_no =" + rid;
           //Lager spørring
            Connection c = SQLConnector.getConnection();
           //Oppretter kobling mot databasen

            try {
                 logging.info("statement: " +      R);
                //Logger spørring
                c.createStatement().executeUpdate(R);
          //Kjører spørring, fjerner alle stoppesteder til ruten
                return true;
                      //Returnerer sant
            } catch(Exception e) {
                      //Logger feil
                logging.warn("Klarte ikke fjerne stoppestedene(" + R +
")",e);
```

```
            }
            finally {
                    //Lukker kobling
                try { c.close(); } catch(Exception e) {}
                c = null;
            }

            return false;
                    //Returnerer ikke sant

    }


     /**
      * <p> Redigerer et stoppested i databasen </p>
     * @param StopBO s
     * @return tilstandsobjekt IOState
     * @see etc.route.io.RouteIO#putStop(etc.route.StopBO)
     */
      @Override
      public IOState EditStop(StopBO s, Boolean mod) {
            String Q= "";
            if(mod==true) {

                  //Lager spørring for å redigere stoppested
Q = "UPDATE stops SET notes='" + s.getNotes() + "', adress='" +
s.getAdress() + "', name='" + s.getName() +"', meet='" +
s.getMeet().getFormatedDate(SystemPreferences.get("date_format")) +
                  //Får tak i tidspukt i riktig format for sql
"',stay='"+s.getStay().getFormatedDate(SystemPreferences.get("date_format")
) +"', lat=" + s.getLat() + ", lng=" + s.getLng() +
" WHERE stop_no =" + s.getStopNo() + " AND r_no= " + s.getRouteNo();
            }
            else {
String tida = "0000-00-00
"+s.getMeet().getHours()+":"+s.getMeet().getMinutes();
String tidb = "0000-00-00
"+s.getStay().getHours()+":"+s.getStay().getMinutes();

Q = "UPDATE stops SET notes='" + s.getNotes() + "', adress='" +
s.getAdress() + "', name='" + s.getName() +
            "', meet='" + tida +
            "', stay='" + tidb +
            "', lat=" + s.getLat() + ", lng=" + s.getLng() +
            " WHERE stop_no =" + s.getStopNo() + " AND r_no= " +
s.getRouteNo();

            }
      Connection c = SQLConnector.getConnection();//Oppretter kobling

            try {
               int innt = c.createStatement().executeUpdate(Q);
                                 //Kjører spørring for å oppdatere

               if(innt == 0)      //Hvis feilet
                   return IOState.IO_PUT_ERROR;     //Returner error

               return IOState.IO_OK;          //Ellers returner ok
            } catch(Exception e) {            //Logger evt. feil
        logging.warn("Noe gikk galt med oppdatering av stoppested!" + Q,e);
```

```
                }
                finally {             //Lukker kobling
                    try { c.close(); } catch(Exception e) {}
                    c = null;
                }

                return IOState.IO_PUT_ERROR;//Return error
        }

         /**
          * <p> Redigerer en rute i databasen </p>
        * @param RouteBO r
        * @return tilstandsobjekt IOState
        */
         @Override
         public IOState EditR(RouteBO r) {
               Boolean mod=false;
               RouteBO o = getDatesR(r.getRouteNo(), mod);
         //Henter nyest oppdatert info fra stoppestedene til ruten

               //Lager spørring for å oppdatere rute
String Q = "UPDATE routes SET tot_stops="+ r.getTot_stops() + ", tot_time="
+ r.getTot_time() +", begin_time='" +
o.getBegin_time().getFormatedDate(SystemPreferences.get("date_format"))+
"', stop_time='"+
o.getStop_time().getFormatedDate(SystemPreferences.get("date_format")) +
"',  homepoint='" + o.getHomepoint() + "' WHERE route_no =" +
r.getRouteNo();


         Connection c = SQLConnector.getConnection();    //Oppretter kobling

                 try {
         //Kjører spørring
                    int innt = c.createStatement().executeUpdate(Q);

                    if(innt == 0)                    //Hvis feilet returner error
                        return IOState.IO_PUT_ERROR;

                    return IOState.IO_OK;            //Ellers returner ok
                } catch(Exception e) {               //Logger feil
                logging.warn("Noe gikk galt med oppdatering av ruten!"+ Q,e);
                }
                finally {                            //Lukker kobling
                    try { c.close(); } catch(Exception e) {}
                    c = null;
                }

                return IOState.IO_PUT_ERROR;      //Returnerer error.
         }

         @Override
         public RouteBO getDatesR(Integer rid, Boolean mod) {
               String df="HH:mm";
               String A ="";

               RouteBO route = new RouteBO();      //Oppretter RoutesBO objekt

String R = "SELECT MIN(meet) meet FROM Stops WHERE r_no=" + rid;
                                  //Henter minst starttid fra stoppestedene
```

```
String Q = "SELECT MAX(stay) stay FROM Stops WHERE r_no=" + rid;
                              //Henter ut max starttid fra stoppestedene

        Connection c = SQLConnector.getConnection();//Kobler til databasen
        ResultSet rs = null;        //Lager resultatsett
        ResultSet s = null;
        ResultSet r =null;

        try {

            rs = c.createStatement().executeQuery(R);       //Kjører
spørring R

            if(rs.first() ) {       //Henter ut fra resultatsett
                String meet = rs.getString("meet");
                                    //Henter starttiden til ruten
DateTime start = new DateTime(meet,SystemPreferences.get("date_format"));
                                    //Omgjør til riktig format
         route.setBegin_time(start);//Setter objektet

            }
        } catch(Exception e) {      //Logger
            logging.warn("Klarte ikke finne  meet (" + R + ")",e);
        }
      finally {
                                    //Lukker denne koblingen
                try { rs.close(); } catch(Exception e) {}
                try { c.close(); } catch(Exception e) {}
                rs=null;
                c = null;
              }

      c = SQLConnector.getConnection();   //Oppretter ny kolbing

                  //Henter adressen for stoppestedet med minst tidspunkt
      if(mod==true) {   //Hvis dette er en rute med datoer
     A = "SELECT adress FROM Stops WHERE meet='" +
route.getBegin_time().getFormatedDate(SystemPreferences.get("date_format"))
+"'";
     logging.info("Kjører spørring: " + A);

     try {
            s = c.createStatement().executeQuery(A); //Kjører spørring
            if(s.first() ) {              //Hvis funnet
                   String adress = s.getString("adress");  //Henter adresse
                   route.setHomepoint(adress);       //Setter adressen
            }
            } catch(Exception e) {        //Logger
                  logging.warn("Klarte ikke finne adress(" + A + ")",e);
                  }

             finally {                    //Lukker denne koblingen
                  try { s.close(); } catch(Exception e) {}
                  try { c.close(); } catch(Exception e) {}
                  rs=null;
                  c = null;
                }
      }
      else {                              //Hvis dette er en fri rute
        String m = route.getBegin_time().getFormatedDate(df);
```

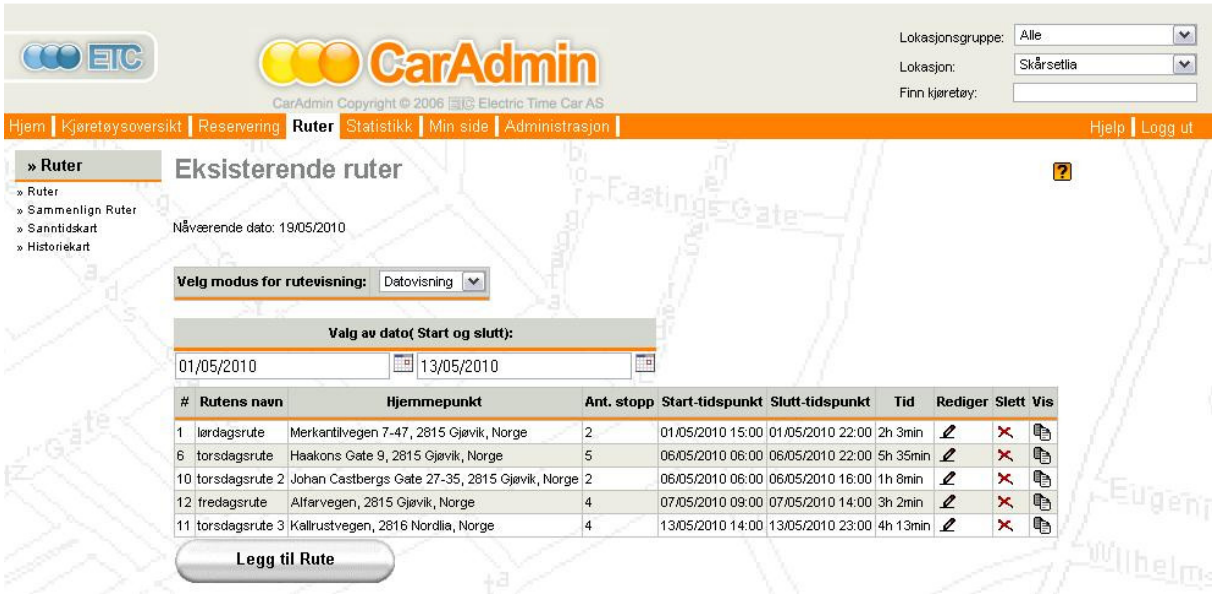
```
                                        //Gjør om BeginTome til HH:mm
        A ="SELECT meet, adress FROM Stops WHERE r_no=" + rid;
                                        //Lager spørring

      logging.info("Kjører spørring: " + A);
    try {
            s = c.createStatement().executeQuery(A); //Kjører spørring
            if(s.first() ) {              //Hvis funnet
                  String meet = s.getString("meet"); //Henter meet
                   DateTime start = new
DateTime(meet,SystemPreferences.get("date_format")); //Lager ny DateTime
meet
                   String me = start.getFormatedDate(df);
                                        //Formaterer dato til form HH:mm
                   String adress = s.getString("adress");  //Henter adresse

                   if(me.equals(m)) {     //Hvis datoene er like
                         route.setHomepoint(adress);
      }//Setter adressen
            }
            } catch(Exception e) {
                  //Logger
                  logging.warn("Klarte ikke finne adress(" + A + ")",e);
                  }

             finally {  //Lukker denne koblingen
                  try { s.close(); } catch(Exception e) {}
                  try { c.close(); } catch(Exception e) {}
                  rs=null;
                  c = null;
              }

      }

              c = SQLConnector.getConnection();//Oppretter ny kolbing

          try {         //Pørver å kjøre spørring mot stoppesteder
                  r = c.createStatement().executeQuery(Q);
            //Kjører spørring Q
                  if(r.first()) {               //Hvis finnes
                  String stay = r.getString("stay"); //Setter stopp
                  DateTime stop = new
DateTime(stay,SystemPreferences.get("date_format")); //Formaterer
                  route.setStop_time(stop);     //Setter objekt

                  }
              } catch(Exception e) {
                        //Logger
                  logging.warn("Klarte ikke finne stay(" + R + ")",e);
              }
              finally {
                        //Lukker kobling
                  try { r.close(); } catch(Exception e) {}
                  try { c.close(); } catch(Exception e) {}
                  r = null;
                  c = null;
              }

              return route;   //Returnerer ruteobjekt med ny informasjon
```

```
	}

@Override
	public RoutesBO getRoutesStops(String st, String ed, Boolean mo,
SessionInfo sio) {
			RoutesBO rs = new RoutesBO();
			Vector<RouteBO> r = new Vector<RouteBO>();
			Vector<StopBO> s = new Vector<StopBO>();


			DateTime start = null;
			DateTime stop = null;


	if(mo == true) {		//Hvis dette er en visning av ruter med
					//dato eller dag
				start = new DateTime(st,
SystemPreferences.get("date_format"));
			start.setHour(00); start.setMinute(00); start.setSecond(00);

			st =
start.getFormatedDate(SystemPreferences.get("date_format"));

			stop = new DateTime(ed, SystemPreferences.get("date_format"));
			stop.setHour(23); stop.setMinute(59); stop.setSecond(59);
			ed =
stop.getFormatedDate(SystemPreferences.get("date_format"));
		}

		if(mo==false) {			//Fri visning av ruter
		  st= "0000-00-00 00:00:00";
		  ed= "0000-00-00 23:59:59";

		}

String Q ="SELECT route_no, homepoint, tot_stops, stop_time, begin_time,
tot_time, name FROM Routes"+ " WHERE begin_time >= '"+ st + "' AND
stop_time <= '" +ed +"'ORDER BY begin_time ASC";


	Connection c = SQLConnector.getConnection();//Oppretter kobling

	ResultSet res = null;		//Lager resultatsett

	try {
	    res = c.createStatement().executeQuery(Q);//Kjører spørring

	    if(res.first() ) {		//Tar det førstei resultatsettet
		  do {
			RouteBO route = new RouteBO();
		    Integer rno = res.getInt("route_no");
		    String homepoint = res.getString("homepoint"); //Leser inn
		    String begin_time = res.getString("begin_time");
		    String stop_time = res.getString	("stop_time");
		    int tot_time = res.getInt("tot_time");
		    int tot_stops = res.getInt("tot_stops");
		    String rname = res.getString("name");

	    //Setter data til riktig sted
		    route = getRStops(rno);
```

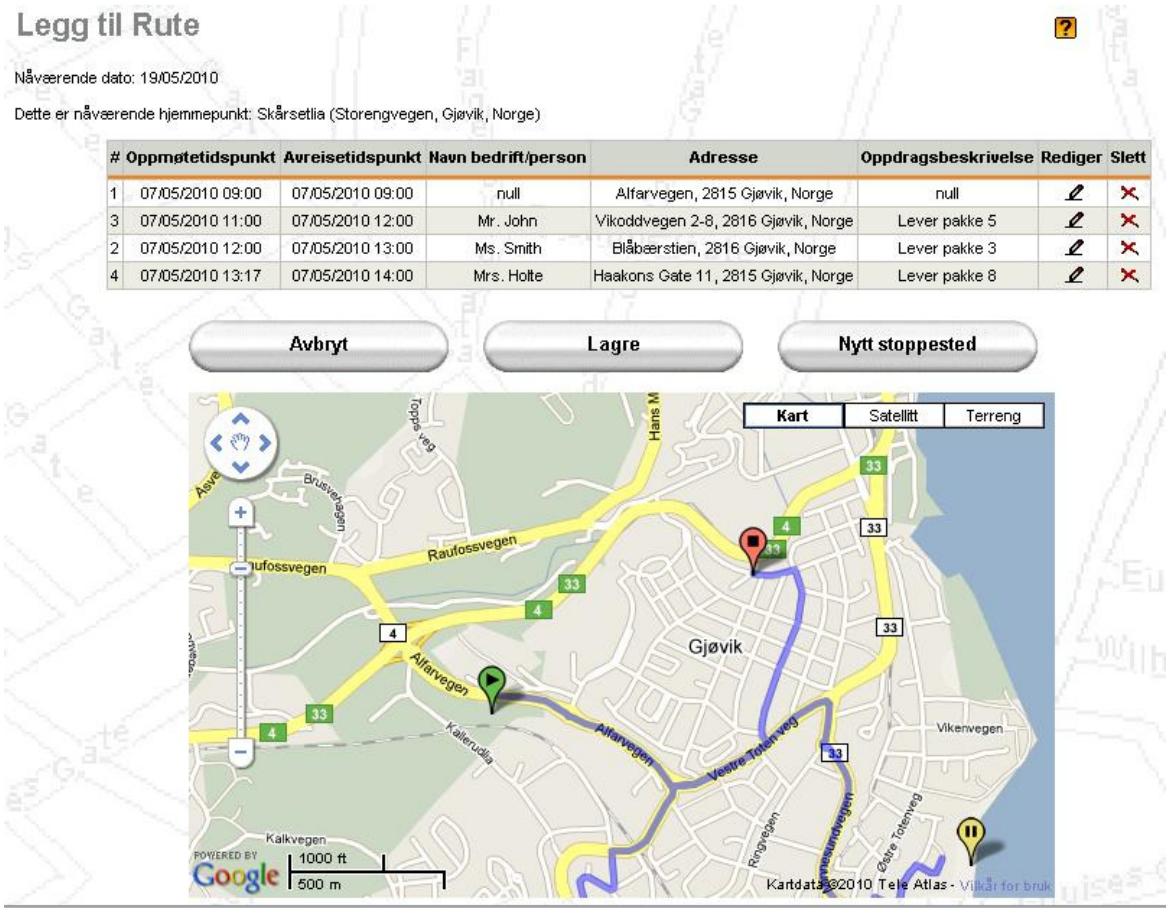
```
                route.setRouteNO(rno);
                route.setHomepoint(homepoint);
                route.setBegin_time(new
DateTime(begin_time,SystemPreferences.get("date_format")));
                route.setStop_time(new
DateTime(stop_time,SystemPreferences.get("date_format")));
                route.setTot_time(tot_time);
                route.setTot_stops(tot_stops);
                route.setName(rname);

                  r.addElement(route);
                  route = null;
                  } while(res.next());
            }
            rs.setRoute(r);
            return rs;
        }catch(Exception e) {                    //Logger
                logging.warn("Klarte ikke hente ruten(" + Q + ")",e);
            }
          finally {                              //Lukker kobling
                  try { res.close(); } catch(Exception e) {}
                  try { c.close(); } catch(Exception e) {}
                  res = null;
                  c = null;
            }

          return null;
      }

      public RouteBO getRStops(int rno) {
             RouteBO route = new RouteBO();
            Vector<StopBO> s = new Vector<StopBO>();

             Connection c = SQLConnector.getConnection();
             ResultSet st = null;
           String QS = "SELECT stop_no, meet, stay, notes, adress, name,
lat, lng FROM Stops WHERE  r_no=" + rno + " ORDER BY meet ASC";

           try {
            st = c.createStatement().executeQuery(QS);
//Kjører spørring for å hente resultater

            if(st.first() ) {
      //Henter ut fra resultatsett
                do {
                int stop_no = st.getInt("stop_no");
                String meet = st.getString("meet");
                String stay = st.getString("stay");
                String notes = st.getString("notes");
                String adress = st.getString("adress");
                String name = st.getString("name");
                float lat = st.getFloat("lat");
                float lng = st.getFloat("lng");

                StopBO stop = new StopBO();

              //Konverterer til DateTime i riktig format
                stop.setMeet(new DateTime(meet,
SystemPreferences.get("date_format")));
```

```
                stop.setStay(new DateTime(stay,
SystemPreferences.get("date_format")));

      //Setter dataene i stoppe objektet
                stop.setNotes(notes);
                stop.setAdress(adress);
                stop.setName(name);
                stop.setStopNo(stop_no);
                stop.setRouteNo(rno);
                stop.setLat(lat);
                stop.setLng(lng);
// System.out.println("Stop: "  + stop.getStopNo() + stop.getNotes());

                s.addElement(stop);
                stop=null;
                 }while(st.next());
             }
            route.setStops(s);
            return route;

         } catch(Exception e) {           Logger
             logging.warn("Klarte ikke å hente stoppet(" + QS + ")",e);
         }
         finally {      //Lukker denne koblingen
                  try { st.close(); } catch(Exception e) {}
                  try { c.close(); } catch(Exception e) {}
                  st=null;
                  c = null;
              }
                  return null;
      }
}
```

F. Skjermbilder

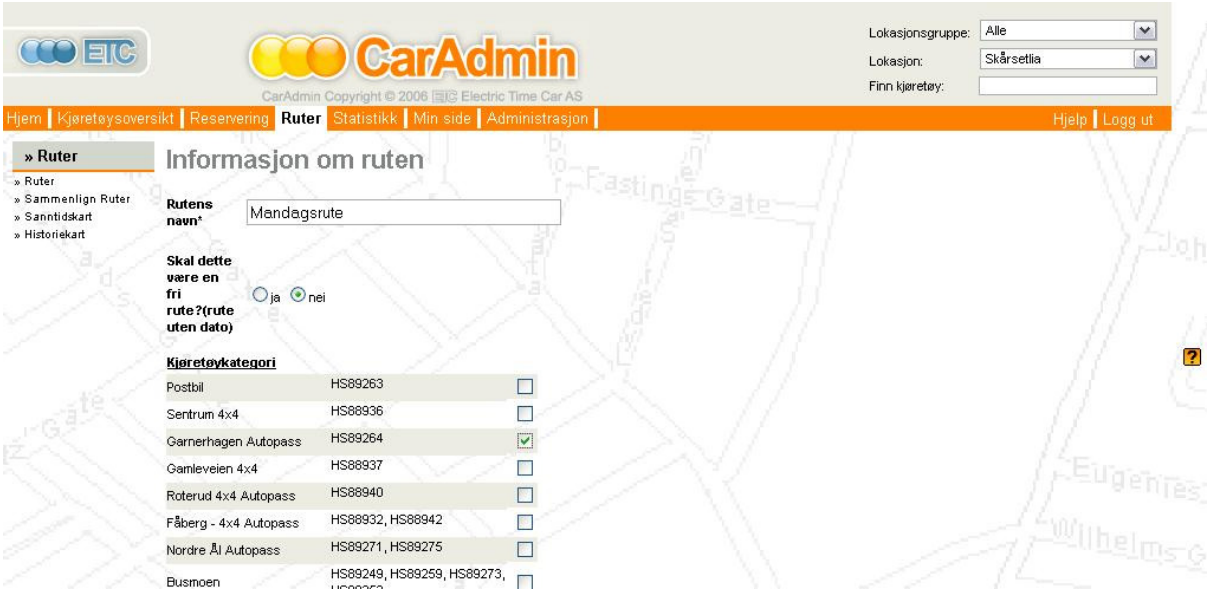


Figur 36 Functions.jsp

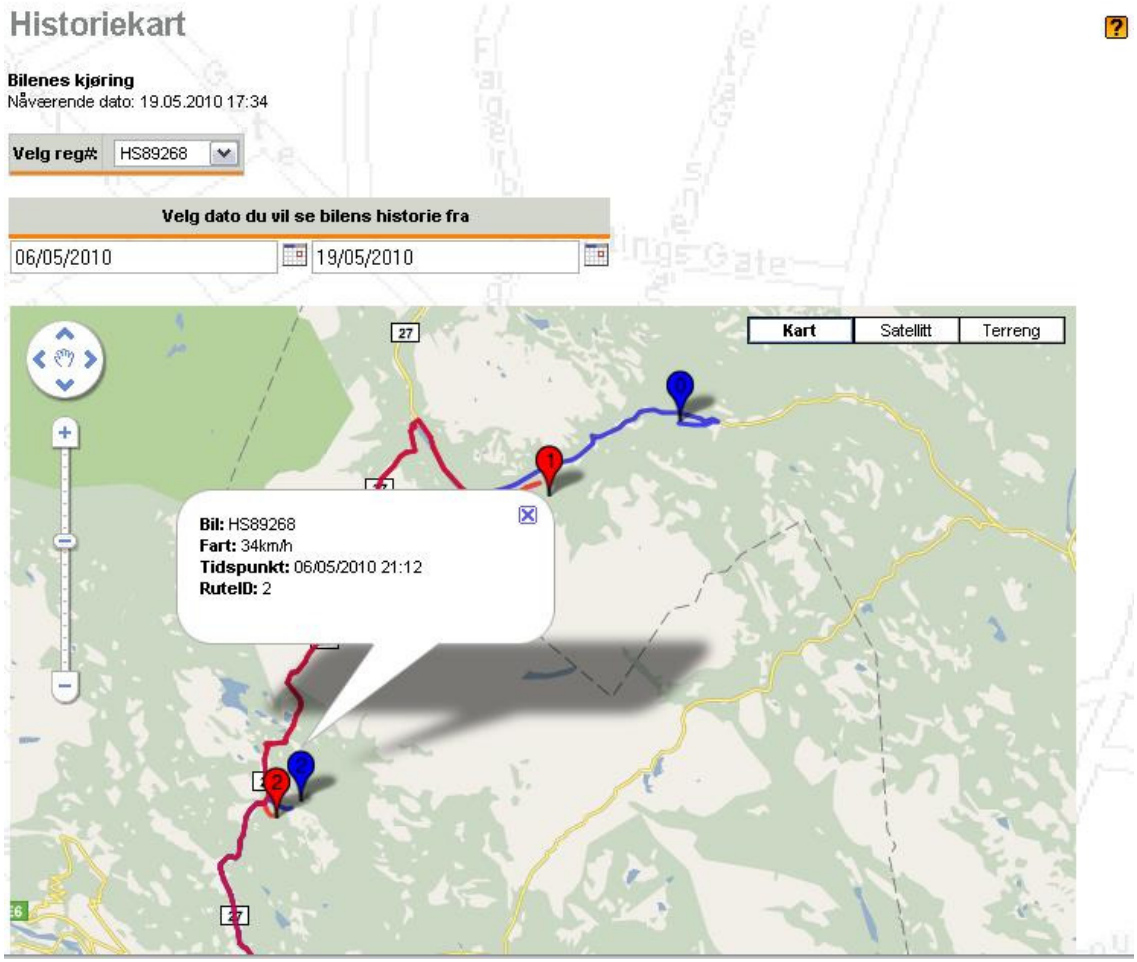


Figur 37 Add.jsp

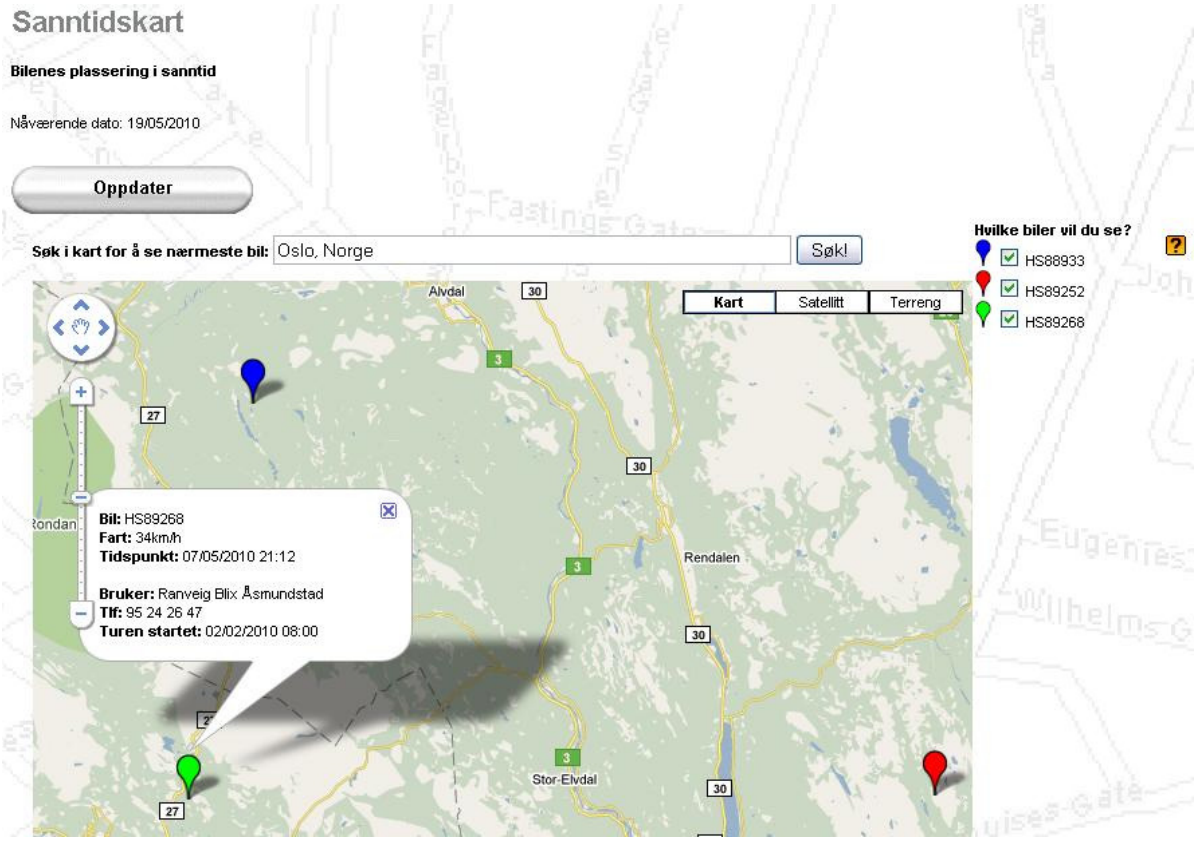
Figur 38 Vegbeskrivelse fra ShowRoute.jsp



Figur 39 CarChooser.jsp



Figur 40 Historymap.jsp



Figur 41 Realtimemap.jsp

G. Forprosjekt

Forprosjekt

Ruteplanlegger & Flåtestyring

For

Else Dalby

Marte S. Bjørseth

Trine A. Grønvold

07HBINDA

29.01.2010

Innhold

1. MÅL OG RAMMER

1.1. Bakgrunn

I bedrifter og kommuner som opererer med bilpark med fellesbiler, er det ikke greit å vite hvilken bil som er disponibel til hvilke tider, hvilke ansatte som bruker den eller hva den blir brukt til, uten et overordnet administrativt system. Et slikt system finner vi i ETC sin CarAdmin applikasjon. For å kunne spare tid og ressurser er det også hensiktsmessig å se hvor bilene er, finne den kjappeste veien til stoppestedene og se om det er muligheter for samkjøring mellom de ulike rutene. Vi har fått som oppdrag å lage en Ruteplanlegger- og en Flåtestyringsmodul til denne eksisterende applikasjonen, som skal gjøre det mulig for brukeren å administrere og planlegge en mest mulig effektiv utnyttelse av bilparken og gi en god oversikt til arbeidsgiveren.

Oppdragsgiveren i forbindelse med dette er Electric Time Car AS (ETC), som er et IT-selskap som er lokalisert i Gjøvik. Bedriften baserer seg på å lage "nyskapende løsninger for administrasjon av fellesbiler, som benyttes av flere sjåfører i både små og store bilparker".[15] CarAdmin er ETC sitt frontprodukt og hovedapplikasjon, hvor man enkelt kan samle fellesbilene under daglig oppfølging av en person. Under denne applikasjonen finner man flere moduler, hvis kombinasjon gjør at deres produkt blir spesielt, og best mulig brukertilpasset i forhold til deres kundegruppe. Ruteplanleggeren og Flåtestyringen er nettopp slike utvidelser, som ETC ønsker at vil gi de et forsprang i forhold til liknende produkter på markedet.

Vi har tidligere arbeidet med ETC og Ruteplanleggermodulen ved kurset Objektorientert Systemutvikling (IMT3102) ved HiG, hvor vi allerede har sett på hvordan en eventuell forprosjektrapport kunne ha sett ut. Dette vil være en sterkt revidert utgave av denne, hvor vi har forbedret og tatt hensyn til den nye modulen; Flåtestyring, som vi skal utarbeide i oppgaven, i tillegg til Ruteplanlegger.

1.2. Prosjektmål

Resultatmål

Målet med oppgaven er å utvikle to ferdige tilleggsmoduler til ETC sitt eksisterende produkt CarAdmin. Når vi er ferdig skal de kunne tilbys direkte til ETC sine kunder. Disse modulene skal ta for seg ruteplanlegging, samt flåtestyring. Modulene skal også være testet opp mot CarAdmin, og kunne fungere som en del av denne.

Effektmål

Med produktet er det ønsket å oppnå ressursbesparelser hos brukerbedriftene, hovedsakelig i form av tid for den som skal sette opp rutene. Vinningen vil variere ut fra brukerbedriftens størrelse og arbeidsform. Modulene skal samkjøre med den eksisterende applikasjonen CarAdmin for å styrke oppdragsgivers posisjon i markedet, ved å gi større konkurransedyktighet. For ETC sin del, er det ønsket å kunne gi kundene et bredere tilbud

[15] http://www.electrictimecar.com/modules/content/index.php?id=12

innen håndtering av bilparken, og gi større mulighet for planlegging av bruk, samt muligheten til å se hvordan faktisk kjøring ble.

1.3. Rammer

- Modulene skal programmeres i utviklingsmiljøet Eclipse.
- Versjonshåndtering av kildekoden gjøres med Subversion.
- Programmeringsspråkene som skal brukes er Java, JSP, JavaScript, HTML, XML og SQL.
- Dokumentering vil foregå i Microsoft Office 2007, hovedsakelig Word og Excel.
- Ruteplanlegger skal bruke Google Maps som et adapter til å vise kart.
- Løsningen skal bygge videre på CarAdmins eksisterende kodebase.

2. OMFANG

2.1. Oppgavebeskrivelse

Det er flere yrkesgrupper som bruker mye av arbeidsdagen til å kjøre bil, som f.eks. hjemmetjenesten og leveringstjenester. I slike arbeidssituasjoner er det ikke nødvendigvis lett å få en helhetlig oversikt over hva de ansatte gjør, eller vite hvor effektivt arbeidskraften blir utnyttet.

Hvis vi tar for oss arbeidsdagen til en ansatt i f.eks. hjemmetjenesten, vil vedkommende kun ha en oversikt over hvem de skal besøke, og i hvilke tidsrom de bør ha vært innom der. Dette fører til at de i mer eller mindre grad har ansvaret for å planlegge rutene selv. Det er ikke nødvendigvis gitt at vedkommende er kjent i området, og vi kan dermed anta at det kan gå med mye tid til unødvendige omveier og dårlig planlegging i henhold til den mest ressurssparende ruten de kan følge. Kanskje kunne til og med to ansatte med mer eller mindre samme rute ha delt bil?

Ruteplanleggermodulen skal fungere som et administrativt støttesystem som vil ta for seg disse problemene. Dette betyr i praksis at en overordnet planlegger skal kunne planlegge og plotte inn kjøreoppdragene til de ansatte på forhånd, for så å legge inn de ferdigplanlagte rutene i kalenderen til Ruteplanlegger. På hver av rutene vil all nødvendig informasjon om de enkelte stoppestedene bli lagt inn, som bl.a. arbeidsoppgavene som skal bli gjort og til hvilke tider de ansatte skal være der. Selve modulen er en kartbasert tjeneste som baserer seg på en adapter mot Google Maps, slik at planleggeren automatisk får opp den mest effektive ruten til disse stoppestedene. De ansatte kan dermed bare skrive ut kartet med sin respektive rute ut fra kalenderen, og komme i gang med arbeidsdagen.

Planleggeren skal også kunne sammenligne enkelte ruter mot hverandre, slik at han på en enkel måte kan se om det er mulig å samkjøre ansatte med liknende ruter, eller gjøre eventuelle endringer slik at ruten blir mer effektiv. Dette blir gjort ved hjelp av Google Maps, hvor de valgte

rutene blir transparert over hverandre, og planleggeren kan velge hvilke ruter han vil sette sammen.

For ikke å bruke unødvendig mye tid på å lage kode som eksisterer fra før kommer vi til å bruke opp noen av funksjonene som allerede eksisterer i CarAdmin-applikasjonen. Av disse funksjonene vil vi benytte kalenderfunksjonen, brukerhåndtering i forhold til brukerens rettigheter og oppretting av profiler samt logg-inn funksjonen, og loggføringsmulighetene til CarAdmin, slik som feillogging. Vi kommer også til å integrere Google Maps i koden vår, og det er naturlig å anta at det allerede finnes en god eksisterende adaptermekanisme til Google Maps som vi kan gjenbruke. Dette gjør at vi står igjen med å lage selve hovedinnholdet til Ruteplanlegger, slik som det å kunne sammenligne ruter, lage en ny rute, skrive ut en rute etc. Dette betyr også at en stor del av oppgaven består av at vi må sette oss inn i mye eksisterende kode, og modifisere disse slik at de passer vår bruk.

I tillegg til Ruteplanleggermodulen skal vi også utvikle en Flåtestyringsmodul, som skal gi arbeidsgiver en generell oversikt over hvor bilene til enhver tid er, og kunne spore de faktiske rutene sjåførene tar. Denne modulen vil henge sammen med Ruteplanleggeren, siden denne vil benytte seg av kartdelen vi skal utvikle under Ruteplanleggermodulen. Bilene vil bli utstyrt med GPS med SIM-kort fra ETC, som med jevne mellomrom skal sende sine koordinater tilbake til ETC sin server. Ut fra disse koordinatene skal brukeren få muligheten til å få opp et sanntidskart, som gir informasjon og et kartutsnitt over hvor alle bilene til enhver tid befinner seg. Biler som ikke er i bruk, vil dermed logge at de står parkert, eller at de er et sted i nærheten av parkeringen. Hvor nøyaktige opplysninger sanntidskartet gir, avhenger alt etter hvor hyppig signalene skal bli gitt og loggført.

Det skal også være mulighet for å hente ut bilhistorikken til en valgt bil, slik at det skal bli enkelt å spore den faktiske bruken av bilene. Her vil brukeren kunne velge en bil, og få opp et kartutsnitt over ruten bilen sist har kjørt, eventuelt den nåværende ruten den er på. Brukerene kan også hente opp tidligere bruk av bilen i en viss tid bakover.

I forhold til denne modulen, kommer vi til å se på eksisterende flåtestyringssystemer til andre bedrifter. Her kommer vi til å vurdere å ta med funksjonaliteter som vi synes er smarte å ha med, alt ettersom hvor god tid vi har utover oppgaven og hvor mye vi føler at de vil gi modulen. Dette er kun et moment som vi skal se på rundt modulen, og er dermed ikke spesifikt gitt at vi må ha med i forhold til oppgavebeskrivelsen fra arbeidsgiver. Siden en flåtestyring allerede er meget utbredt, kommer vi å undersøke om det finnes eksisterende kode, og eventuelt se om det er muligheter for å gjenbruke dette.

2.2. Avgrensning

- Ruteplanlegger- og Flåtestyringsmodulen skal ikke være mulig å bruke uavhengig av ETC sin CarAdmin-applikasjon, siden modulene er en del av dette programmet, og de bruker samme innloggingsfunksjon. Vi trenger derfor ikke å lage noen egne innloggingsfunksjoner for våre applikasjoner.

- Ruteplanleggeren trenger ikke å ta forbehold for alternative kjøreruter, slik som å ta høyde for f.eks. bilulykker på ruten eller andre variable i trafikken som varierer daglig.

- ETC har bedt oss benytte Google Maps. Vi har derfor ikke med noen vurdering av andre kartverk, da ETC allerede har sjekket ut dette.

3. PROSJEKTORGANISERING

3.1. Ansvarsforhold og roller

Prosjektleder i gruppen er Trine Anita Grønvold. Ansvarlig for dokumentasjon, deriblant notater under møter, er Else Dalby. Ansvarlig for nettsiden og loggen er Marte Selsjord Bjørseth.

Underveis i prosjektgjennomføringen vil ansvarsforholdet fordeles mellom gruppemedlemmene. Det innebærer at et gruppemedlem får ansvaret for en modul, slik at dette er utført innen avtalt tid. Den ansvarlige fordeler oppgaver til alle i gruppa, for å sørge for at alle deler av modulen er gjennomført.

3.2. Rutiner og regler i gruppa

Rutiner:

- Arbeidet foregår hovedsakelig på grupperom A030. Det vil også foregå noe arbeid i ETC sine lokaler i Energihuset.
- Alle møter opp til de tider oppsatt på felles timeplan. Denne ligger på prosjektets hjemmeside. Det må i tillegg påregnes å kunne arbeide noe utover dette.
- Det avholdes ukentlige møter med hhv. veileder og oppdragsgiver. Denne hyppigheten er åpen for endring når hoveddelen av prosjektet er i gang.
- Logg føres på slutten av hver dag. Alle er ansvarlig for sin personlige logg.

Grupperegler:

- Alle møter til avtalt tidspunkt, eller gir beskjed til de andre på gruppa.
- Fravær skal meldes til de andre i rimelig tid med god begrunnelse.
- Alle utfører sitt arbeid til avtalt tid.
- Arbeidsoppgaver skal fordeles mest mulig jevnt mellom gruppemedlemmene. Dette gjøres av gruppeleder eller den ansvarlige for gjeldende modul.
- Avtaler kan kun inngås med alles samtykke.
- Det er felles ansvar å holde grupperommet i orden, alle rydder etter seg selv.
- Det skal skrives logg etter hver arbeidsdag med oppgaven. Alle er felles ansvarlig for dette.
- Gruppen er felles ansvarlig for å overholde tidsfrister fra Høgskolen.
- Gjentatte brudd blir først tatt opp innad i gruppen, og dersom ikke forbedring kontaktes veileder og gruppemedlemskap kan vurderes.

3.3. Oppdragsgiver og veileder

Oppdragsgiver er Electric Time Car AS, hovedsakelig omtalt som ETC. Kontaktperson er Dag L. Solhaug, mens systemutvikler og programmerer Øyvind Flatval er ansvarlig for alle tekniske detaljer.

Veileder ved Høgskolen i Gjøvik er Tom Røise.

4. PLANLEGGING, OPPFØLGING OG RAPPORTERING

4.1. Hovedinndeling av prosjektet

Vi har valgt å bruke den agile systemutviklingsmetoden Scrum til å utvikle vårt hovedprosjekt. Vi valgte en av de agile systemutviklingsmetodene fordi vi vet med sikkerhet at det kommer til å bli endringer i kravspesifikasjonen underveis. Dette vil hjelpe oss til å håndtere disse endringene, samtidig som en slik metode tar forbehold om en eventuell utvikling av programvaren senere. Av de agile metodene var XP, Scrum, FDD og Lean alternativene vi vurderte. Disse alternativene har sine sterke og svake sider, og kombinasjonen av dette gjorde en av disse bedre egnet for vårt prosjekt enn de andre.

Vi eliminerte raskt Lean fra disse alternativene, siden denne modellen er mer en liste med gode prinsipper en skal jobbe etter, enn et faktisk metodeverk for hvordan en skal jobbe. Dette er første gang vi begir oss ut i et så stort prosjekt, og vi føler derfor at vi trenger et solid rammeverk og en tydelig metodikk for hvordan vi burde gå frem.

I utgangspunktet virket XP sin framgangsmåte å passe oss bra. Grunnen til at vi tilslutt måtte forkaste denne, var parprogrammeringsaspektet i modellen. Det er nemlig en veldig viktig del av denne modellen at to og to sitter og programmerer sammen slik at koden blir bedre. Dette passet ikke så godt for en liten gruppe på tre. Hvis to av oss skulle sitte sammen og programmere, eller alle tre sammen for den del, ville dette ha ført til et tregere arbeidstempo, og en særdeles lite optimal arbeidssituasjon. FDD er tjenestebasert, noe som i og for seg passer bra, men vi syntes at metodikken i Scrum er bedre å jobbe med.

Til slutt endte vi opp med Scrum. Grunnen til dette er fordi at samtidig som modellen gjør det mulig å introdusere nye krav hele veien gjennom utviklingen, så er selve modulen vi jobber på til enhver tid låst for endringer i den perioden vi jobber med den. Dette er nødvendig da ETC kan komme med nye krav, og samtidig som vi skal rette oss etter disse trenger vi også tid til å jobbe med eksisterende krav. En annen ting vi likte godt med Scrum var Product Backlog og Sprint Backlog; de prioriterte listene over hva som må gjøres; overordnet, og detaljert i sprinten. På denne måten kan vi være sikker på at vi til enhver tid jobber med de modulene som er mest kritiske for prosjektets suksess, og beholder oversikten over hvor langt vi har kommet og hva vi har igjen.

De argumentene som til slutt solgte oss helt og holdent til Scrum, var at uansett om vi ikke blir helt ferdig med prosjektet vårt til i mai, vil vi ha noen av modulene i applikasjonen ferdig som vi kan presentere og vise frem. Det er et godt sikkerhetsnett at vi etter hver sprint har en modul

som ferdigprogrammert. Inndelingen, eller sprintene, av arbeidsperioder passer også bra i Scrum. Disse kan altså deles inn i alt fra en uke til en måned, og vi kan tilpasse dette helt til oss og vår situasjon. Samtidig vet vi at ETC har tidligere jobbet sammen med prosjektgrupper som har brukt Scrum, og har derfor nyttig erfaring fra denne arbeidsmetodikken som kan være behjelpelig for både de og oss.

Vi vil bruke mesteparten av Scrum sine karakteristiske egenskaper i dette prosjektet: Product Backlog for alle krav, Sprint Backlog for alle krav som skal innfris i den enkelte sprint, sprinter på 14 dager, 15 minutters sprintmøter hver dag, planleggingsmøter der ny Sprint Backlog blir utarbeidet for hver ny iterasjon, og vi ser på oss selv som et Scrum-team.

Det vi derimot mangler er en Scrum-master, men siden vi bare er tre stykker så antar vi at vi klarer oss uten dette. Siden Scrum ikke har noen spesielle krav til dokumentering, har vi valgt å hente dette fra de mer tradisjonelle utviklingsmetodene, og vil ta med kravspesifikasjon, arkitektur og design. Dette er hovedsakelig til bruk i den endelige prosjektrapporten.

4.2. Plan for statusmøter og beslutningspunkter

Med jevne mellomrom kommer vi til å levere statusrapporter til veileder, som vil bli diskutert nærmere med veileder på våre ukentlige møter. Der vil også eventuelt manglende fremgang, dårlig tidsbruk, overskriding av tidsplan, og andre problemer som kan ødelegge for gruppens suksess bli tatt opp.

På våre daglige sprintmøter vil dagens fremdriftsplan blir diskutert, hvorpå videre kurs for morgendagen staket ut, og grunner til at planlagt arbeid eventuelt ikke har blitt gjort vil bli tatt opp. Korte møtereferater fra disse møtene vil bli publisert i loggen på web-siden vår.

På en større skala så tas beslutningene primært i møtene med ETC foran begynnelsen på en ny sprint. Her setter vi oss ned og går i gjennom Product Backlog for så å overføre de høyest prioriterte modulene til Sprint Backlogen, disse modulene blir så denne periodens overordnede kravspesifikasjon.

5. ORGANISERING AV KVALITETSSIKRING

5.1. Dokumentasjon, standardbruk og kildekode

Alle skriftlige dokumenter som blir skrevet av oss skal være lagret i docx format, da vi alle bruker Microsoft Word 2007. Kildekoden vil bli skrevet i utviklingsverktøyet Eclipse, siden ETC bruker dette verktøyet. Kildekoden versjonshåndteres på ETC sin utviklingsserver.

Vi skal loggføre alle timer vi bruker på prosjektet. I denne loggen skal det være med tidsbruk, dato, og hva det ble jobbet med. Dette skal både gjøres for gruppen samlet, slik at dette kan publiseres på websiden vår, og individuelt slik at dette kan innarbeides i sluttrapporten i mai.

Det skal lages møtereferater fra alle møter vi har med ETC og veileder, samt våre daglige sprint-møter. Disse skal inngå i en prosjektdagbok sammen med loggen som er nevnt ovenfor, og

notater om alle viktige beslutninger gruppen har tatt angående ansvarsforhold og annet av vesentlig betydning for prosjektet og prosjektgruppen. I forkant av hvert møte med ETC sendes en enkel møteinnkallelse med agenda for møtet.

All kode som blir skrevet av oss skal kommenteres med Javadoc fortløpende, eller så fort den aktuelle kodedelen er ferdig til innsjekking i ETC sin kildedatabase. Forøvrig vil også kodesnuttene kommenteres der dette er nødvendig for beskrivelse av algoritmer, presiseringer eller lignende.

5.2. Konfigurasjonsstyring

For å holde kontinuerlig orden på kildekoden vil vi bruke Subclipse, et versjonskontrollverktøy, slik at vi unngår usikkerhet rundt hva som har blitt gjort på koden og sikrer den nyeste versjonen.

Våre navnekonvensjoner er som følger .Vi vil gi alle våre rapporter et navn og versjonsnummer bak, alle spørsmål til veileder og ETC vil skrives som <Spørsmå til … dato>, og alle møtereferater <Møtereferat … dato> .

Vi har valgt å ikke bruke Subversion på de skriftlige dokumentene, siden vi jobber sammen på grupperommet mesteparten av tiden. Med hyppig kommunikasjon og sikring av lokale kopier på hver av datamaskinene til gruppemedlemmene har vi kommet i en rutine som sikrer at vi ikke overlagrer hverandres endringer, og sikrer at den nyeste versjonen hele tiden er intakt. Dermed følte vi at Subversion ble overflødig å bruke her.

5.3. Risikoanalyse

Problem: Flåtestyringsdelen av oppgaven kan bli kuttet bort på grunn av lovendring

Risiko: Høy

Beskrivelse:

Per dags dato finnes det ingen konkrete lover som forbyr en bedrift å holde oversikt i sanntid over alle sine biler, annet enn de generelle lovene om personvern. Det er derfor i dag fullt lovlig å lage et slikt system, og for en bedrift å ta det i bruk så lenge arbeidstakerne samtykker i at systemet med all dets funksjoner kan brukes. Det er imidlertid en viss risiko for at dette kan forandre seg, da Datatilsynet og diverse fagforeninger med tillitsvalgte har rettet kritikk mot denne praksisen og mener at slike systemer i for stor andel overvåker privatlivet til arbeidstaker.

Datatilsynet og Skattetaten er uenige om slik praksis da Datatilsynet er bekymret over personvernet til arbeidstaker, mens Skattetaten helst vil ha så god oversikt over bruken av bilene som mulig, da noen skatteregler avhenger av om jobbens bil har blitt brukt privat av arbeidstaker.

Løsning:

Vi ønsker å avvente og se om slike lovendringer vil skje. Hvis eventuelle nye lover om slike systemer blir gitt, vil vi se an i hvilken grad vi blir rammet og ta en avgjørelse ut i fra det. Vi tar derfor ingen konkrete beslutninger om saken på dette tidspunktet.

Hvis et eventuelt lovforslag kommer igjennom som vil føre til at Flåtestyringsmodulen ikke kan brukes, vil et eventuelt tiltak være å kutte bort denne delen fra prosjektet, og be ETC utrede muligheter for annet ekstrainnhold til Ruteplanlegger eller CarAdmin som vi kan utvikle i stedet. Vi vil da be veileder om råd om det som angår oppgavens nye størrelse og innhold. Hvis loven kommer etter at vi allerede har begynt utvikling av Flåtestyring, må det tas en avgjørelse om fortsatt utvikling, for læringens del, ellers vil valg av en annen ekstraoppgave bli diskutert og beslutning tatt i samarbeid med både ETC og veileder.

Helt konkret vil dette si at har vi to valg hvis det kommer lov om dette: fortsette utviklingen av Flåtestyring som et eksempel på hvordan teknologien kunne ha fungert eller begynne på en annen tilleggsmodul til Ruteplanlegger. Men inntil noe annet er bestemt fortsetter vi å jobbe med å ferdigstille Flåtestyringsmodulen.

Problem: Vi blir ikke ferdig med verken Ruteplanlegger eller Flåtestyring

Risiko: Middels

Beskrivelse:

Vi har aldri utviklet et så stort prosjekt som Ruteplanlegger og Flåtestyring tidligere, og vet derfor lite om hvor lang tid dette kommer til å ta oss. To av oss har heller aldri brukt programmeringsspråket Java tidligere. Dette er to usikkerhetsmomenter som kan gjøre at vi overskrider tidsplanen vår og ikke kommer i mål med disse to delene av prosjektet.

Løsning:

For å minimere risikoen for at vi ikke har noe konkret program som fungerer til leveringsdato har vi bestemt at prosjektet skal utvikles i inkrementer. På denne måten er vi sikker på at vi har noe å vise frem uansett om hele Ruteplanlegger og Flåtestyring er ferdig eller ikke. Vi vil også være i nær konsultasjon med både veileder og ETC angående vårt tidsbruk gjennom våre daglige loggføringer på hjemmesiden, våre ukentlige møter og gjennom tre statusrapporter.

Problem: Vi klarer ikke å koble oss opp mot Google Maps

Risiko: Middels

Beskrivelse:

Store deler av Ruteplanlegger handler om å kunne koble seg opp mot Google Maps sitt API og generere kartene gjennom denne. Hvis vi ikke klarer å få generert kart kommer oppgaven til å bli altfor forenklet.

Løsning:

En løsning hvis vi ikke klarer å koble oss opp mot Google Maps vil være å velge et annet kartredskap for å vise kartene våre slik som for eksempel GuleSider, KartNorge, FinnKart, BingMaps eller lignende.

Problem: Klarer ikke vise flere ruter i samme kart

Risiko: Middels

Beskrivelse:

Vår løsning for sammenligning av ruter er å vise rutene i samme kart, slik at planlegger selv kan se hvilke ruter han eventuelt vil slå sammen. Hvis vi ikke klarer å vise ruter i samme kart blir dette et problem, siden rutenes likhet da bare vil bli gitt i en tabell med ulike sammenligningsgrunnlag, og ikke grafisk.

Løsning:

Siden sammenligningsfunksjonen er sentral for Ruteplanlegger er vi i så fall nødt til å se etter mulige alternativer. Vi kan for eksempel vise de forskjellige rutene i kart ved siden av hverandre i stedet for alle rutene i ett kart.

Problem: Virus

Risiko: Lav

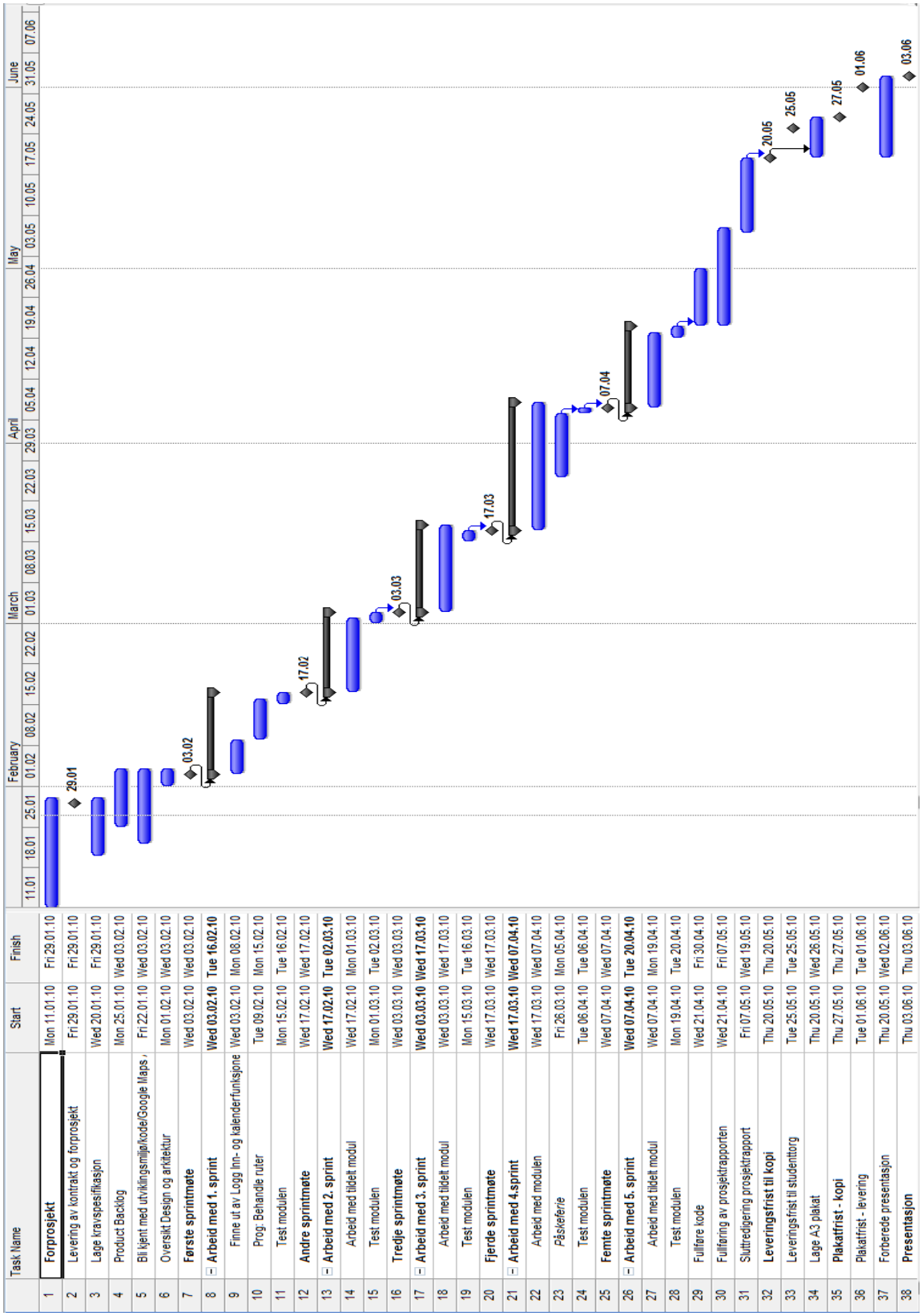
Beskrivelse:

På grunn av en virusinfeksjon på en eller flere av våre datamaskiner kan vi miste sentrale filer og dokumenter i prosjektet vårt.

Løsning:

Siden alle filene som tilhører kode-delen av prosjektetogså ligger lagret på ETC sin server kan vi hente ut disse igjen herfra uten store problemer. Vi kan også anta at dette er de seneste versjonene av filene. Når det gjelder dokumenter er de seneste versjonene av disse lagret på serveren til skolen og markert med versjonsnummer, så dette løser seg også.

6. PLAN FOR GJENNOMFØRING (GANNT-SKJEMA)



	Task Name	Start	Finish
1	**Forprosjekt**	Mon 11.01.10	Fri 29.01.10
2	Levering av kontrakt og forprosjekt	Fri 29.01.10	Fri 29.01.10
3	Lage kravspesifikasjon	Wed 20.01.10	Fri 29.01.10
4	Product Backlog	Mon 25.01.10	Wed 03.02.10
5	Bli kjent med utviklingsmiljø/kode/Google Maps ,	Fri 22.01.10	Wed 03.02.10
6	Oversikt Design og arkitektur	Mon 01.02.10	Wed 03.02.10
7	**Første sprintmøte**	Wed 03.02.10	Wed 03.02.10
8	**Arbeid med 1. sprint**	**Wed 03.02.10**	**Tue 16.02.10**
9	Finne ut av Logg Inn- og kalenderfunksjone	Wed 03.02.10	Mon 08.02.10
10	Prog: Behandle ruter	Tue 09.02.10	Mon 15.02.10
11	Test modulen	Mon 15.02.10	Tue 16.02.10
12	**Andre sprintmøte**	Wed 17.02.10	Wed 17.02.10
13	**Arbeid med 2. sprint**	**Wed 17.02.10**	**Tue 02.03.10**
14	Arbeid med tildelt modul	Wed 17.02.10	Mon 01.03.10
15	Test modulen	Mon 01.03.10	Tue 02.03.10
16	**Tredje sprintmøte**	Wed 03.03.10	Wed 03.03.10
17	**Arbeid med 3. sprint**	**Wed 03.03.10**	**Wed 17.03.10**
18	Arbeid med tildelt modul	Wed 03.03.10	Wed 17.03.10
19	Test modulen	Mon 15.03.10	Tue 16.03.10
20	**Fjerde sprintmøte**	Wed 17.03.10	Wed 17.03.10
21	**Arbeid med 4.sprint**	**Wed 17.03.10**	**Wed 07.04.10**
22	Arbeid med modulen	Wed 17.03.10	Wed 07.04.10
23	*Påskeferie*	Fri 26.03.10	Mon 05.04.10
24	Test modulen	Tue 06.04.10	Tue 06.04.10
25	**Femte sprintmøte**	Wed 07.04.10	Wed 07.04.10
26	**Arbeid med 5. sprint**	**Wed 07.04.10**	**Tue 20.04.10**
27	Arbeid med tildelt modul	Wed 07.04.10	Mon 19.04.10
28	Test modulen	Mon 19.04.10	Tue 20.04.10
29	Fullføre kode	Wed 21.04.10	Fri 30.04.10
30	Fullføring av prosjektrapporten	Wed 21.04.10	Fri 07.05.10
31	Sluttredigering prosjektrapport	Fri 07.05.10	Wed 19.05.10
32	**Leveringsfrist til kopi**	Thu 20.05.10	Thu 20.05.10
33	Leveringsfrist til studenttorg	Tue 25.05.10	Tue 25.05.10
34	Lage A3 plakat	Thu 20.05.10	Wed 26.05.10
35	**Plakatfrist - kopi**	Thu 27.05.10	Thu 27.05.10
36	Plakatfrist - levering	Tue 01.06.10	Tue 01.06.10
37	Forberede presentasjon	Thu 20.05.10	Wed 02.06.10
38	**Presentasjon**	Thu 03.06.10	Thu 03.06.10

6.1. Kommentar til Gannt-skjema

I Gannt-skjemaet har vi satt opp fem sprinter, hver på 14 dager. Før den første sprinten starter 3 Februar, regner vi med å levere forprosjektet, sette opp kravspesifikasjonen samt design og arkitektur. I tillegg vil vi også i denne tidsperioden bruke på å sette oss inn i den eksisterende CarAdmin–koden og Google Maps. Vi har lagt inn en uke påskeferie, slik at sprinten som pågår da, ser ut til å vare en uke ekstra. Den siste sprinten slutter 20. april, så vi har rundt en måned på oss, til prosjektrapporten skal leveres for kopiering. Denne tiden regner vi med at vi skal bruke til finpussing på koden, eventuelt å utvikle små biter som mangler, samt å sy sammen og fullføre prosjektrapporten. A3 plakaten regner vi med å kunne sette i stand på i underkant av en uke. Deretter vil resten av tiden gå til forberedelse av prosjektpresentasjonen som er den 3 juni.

7. TERMINOLOGILISTE

- Utviklingsplatform: En applikasjon hvor programkode skrives, testes og håndteres på andre måter.
- Eclipse: En utviklingsplatform, som er fri til bruk.
- Subversion: Et versjonskontrollsystem som holder styr på endringer i filer.
- Subclipse: Et tillegg til Eclipse, som er et versjonskontrollsystem.
- Scrum: En systemutviklingsmodell som sier noe om arbeidsprosessen i et utviklingsprosjekt.
- Google Maps: Et kartverk med åpen kildekode
- Sprint: En arbeidsperiode hvor det arbeides med en bestemt del av prosjektet. Et prosjekt består av flere sprinter, som skal ha en fast lengde.
- Product Backlog: Alle deler av prosjektet som skal utvikles
- Sprint Backlog: De delene av prosjektet som er arbeidet med under den valgt sprinten.

8 KILDER

- Gamle bacheloroppgaver (Autoklav (2006), Bilbooking CarAdmin (2005), Klientbasert reservering av standard kalenderapplikasjoner (2008), Styresaksdatabase (2009))
- Informasjon fra ETC ved Dag Solhaug og Øivind Flatval
- Tidligere egne oppgaver fra kurset i systemutvikling (USLA og Hytteservice)
- Prosjektet med Ruteplanlegger-modulen ved kurset i Objekt Orientert Systemutvikling

H. Logg

Totalt tidsbruk: 428,5 t

Uke 20

Tidsbruk: 13,0 t

DATO	TID	HVA
20.05.2010		Utskrift av rapport!
19.05.2010	13,0	Møte med ETC, hvor vi gikk gjennom punkter vi var usikre på i rapporten. Vi diskuterte litt hvordan progresjonen og arbeidsinnsatsen vår hadde vært i perioden. Vi fikk ypperlige skussmål. Vi gjorde ferdig/rettet opp de siste delene i rapporten, hvor vi hadde fått familie og venner til å lese over og gi tilbakemelding, slik at vi fikk rettet den opp i forhold til det de ulike leserne merket seg.
18.05.2010	13,0 t	I dag jobbet vi med rapportskriving. Vi fikk tilbakemelding av Tom før helgen, slik at store deler av dagen gikk i retting av rapporten i forhold til dette. Tom kom innom senere på dagen, slik at vi fikk stilt han noen spørsmål vedrørende tilbakemeldingen. Vi leste gjennom hverandres dokumenter, og rettet opp der det trengtes. Det ble gjort en del forandringer i forhold til oppsettet i rapporten. Vi gjorde også ferdig den engelske språkfilen (slik at modulene kan bli vist på engelsk, såvel som norsk).
17.05.2010		17 mai!

Uke 19

Tidsbruk: 24,0 t + hjemmearbeidsdager

DATO	TID	HVA
14.05.2010		Hjemmearbeidsdag.
13.05.2010		Kristi himmelfartsdag.
12.05.2010	10,0 t	Møte med ETC. Her fikk vi ordnet opp i kalenderfunksjonen i "Historiekart", hvor kalenderen er nå fryst, slik at bruker kun kan velge dager fra 14 dager tidligere. Diskuterte også litt rundt dokumentasjonen, og leverte rapporten til gjennomlesing til Øyvind. Ellers gikk dagen i rapportskriving til den store gullmedaljen, hvor vi alle leste gjennom rapporten. Vi fikk også tilbakemelding av Øyvind. Etter at vi rettet opp det han hadde pekt på, sendte vi rapporten videre til Tom for gjennomlesing. Vi jobbet også med kommentering av kode, og retting av "Historiekart" slik at den ble kompitabel med Firefox.
11.05.2010	7,0 t	Dagen i dag gikk med til rapportskriving, og vi vedtok at vi skulle være mest mulig ferdig til i morgen, slik at vi får sendt rapporten til Tom for gjennomlesning. Dette har ført til en meget effektiv dag med tanke på rapportskriving, og vi er nå snart ferdig med introduksjonen, avslutningen, tekniske memo og testing.
10.05.2010	7,0 t	Rapportskriving.

Uke 18

Tidsbruk: 31,0 t + hjemmearbeid

DATO	TID	HVA
07.05.2010		Hjemmearbeidsdag.
06.05.2010	8,5 t	Etter at Øyvind hadde nevnt at det var noen bugs i databasen, testet vi denne, og rettet opp det som manglet i SQL-koden. Vi fant også ut at etter opplastingen av ETC sitt nye interface (de hadde oppdatert error-finner), at det kun kom opp feilmeldinger når vi prøvde å kjøre modulene i Firefox og IE. Dagen gikk derfor i hovedsak med til å finne ut hva som som gjorde at funksjonene ikke var kompitabel med disse nettleserne, og rettet opp dette. Vi hadde også et møte med Tom, hvor vi gikk gjennom kravspesifikasjon, design og implementasjon i hovedrapporten, og fikk tilbakemelding på dette.
05.05.2010	8,0 t	Møte med ETC, hvor både Dag og Øyvind var tilstede. Her gikk vi igjennom det vi hadde gjort med modulene, hvor de pekte på hva de ville ha gjort annerledes, samt ønsket noen elementer de ville ha i tillegg. Vi diskuterte også litt rundt hva vi skulle ha i rapporten, hvor de understreket at vi burde ha med det vi hadde gjort i bakgrunnen (undersøkt, funnet eksempler o.l.) i forhold til ting som ikke ble implementert i modulen (om det så ikke var tid til å gjøre det, eller om det ikke var mulig). Vi oppdaterte til det nye interfacet til ETC, og jobbet videre med endringene som ETC ville ha. Dette innebar bl.a. kunne se resten av ruten i 'legg til nye stoppesteder' i "Behandle ruter", slik at ikke bare stoppestedet kom opp på kartet, se street view i kartet, samt legge opp til at ruten har et egen ruteID, slik at man kan stykke opp rutene i "Historiekart".
04.05.2010	9,0 t	I dag sa vi oss ferdige med hoveddelene i kodingen, selv om vi fortsatt hadde noen problemer som vi trenger hjelp med av ETC i morgen. Vi jobbet med punktene i kartet i "Behandle ruter", og fikset "Sanntidskart" slik at den nå henter kun de nyeste koordinatene til bilene. Vi jobbet også med feilsjekk i forhold til "Behandle ruter" slik at bruker må fylle inn alle nødvendige data. "Historiekart" har også blitt ferdig, hvor den nå har en kalender og viser alle punktene som bruker har bedt om.
03.05.2010	5,5 t	I dag jobbet vi med "Historiekart" (tabell, riktig visning) og rapporten i forhold til design, logiske koblinger mellom filene og hvordan vi bruker Google Maps i koden vår. Vi lagde også et flytskjema over hvordan funksjonen "Ruter" fungerer (tidligere kalt "Behandle ruter").

Uke 17

Tidsbruk: 39,0 t

DATO	TID	HVA
01.05.2010	6,5 t	Vi fant ut at det var greit å få unna litt mer, så vi tok lørdagen på skolen denne uken. Her jobbet vi med hovedrapporten, "Historiekart", og fikset "Sammenligne ruter" slik at ruten blir tegnet opp i forhold til koordinater og ikke adresser.
30.04.2010	7,0 t	I dag jobbet vi med "Behandle ruter", slik at man kan lagre unna adressen som koordinater hvor adressen ikke er god nok for kartet (markørene har en tendens til å komme utenfor vegen i kartet når man lagrer på adresse). Jobbet også med "Sanntidskart", slik at brukeren kan søke på en adresse og finne ut hvilken bil som er nærmest hvis det skal være noe akutt behov for dette. Begynte også på "Historiekart".
29.04.2010	6,5 t	Vi har i dag jobbet med å få opp flere enn 10 markører i "Sanntidskart", siden Google sin Geocoder har en grense på dette, og vi tidligere ikke fikk opp flere enn dette i kartet. Vi fikset også at det er nå mulig å hente ut hvor lang tid en rute tar, dvs. tiden det tar å kjøre + tiden man er oppholdt på stoppestedene. Dette er kun mulig å få ut i sekunder for øyeblikket, så nå mangler vi bare å få ut dette i timer og minutter. Vi hadde et kjapt møte med Tom, og fikk innspill på hvordan noen ting bør implementeres i rapporten, samt ga han en oppdatering på hvordan vi lå an. Vi har fortsatt nok å jobbe med og videreutvikle, men nærmer oss ferdigstilling av det opprinnelige produktet, med endignene vi har blitt bedt om å få gjøre.
28.04.2010	8,0 t	Møte med ETC. Vi er nå halvveis i sprint 5, og har som smått begynt på de siste tingene vi trenger i de ulike modulene - slik som hente ut tiden vi bruker på rutene, feilsjekk i forhold til celler, samt hente ut bruker til bil i "Sanntidskart". Vi jobbet videre med rapportskriving, hvor vi forbedret den ytterligere i forhold til samtale med ETC.
27.04.2010	6,5 t	Vi fikk opp markørene opp i "Sanntidskart", og begynte i med funksjonaliteten 'Vis rute', som skal integreres under "Behandle ruter". Vi begynte også med hovedrapporten, og da henholdsvis kravspesifikasjonen, design og implementasjon av koden. Dette sendte vi til ETC for tilbakemelding.
26.04.2010	4,5 t	I dag jobbet vi med retting av tidsvisning i boblene som kommer opp når man trykker på markørene i kartet i "Sammenligne ruter", samt sidetabellen i "Sanntidskart". Vi jobbet også med hjemmesiden vår, hvor vi oppdaterte linker, loggen og møtereferat.

Uke 16

Tidsbruk: 15,5 t + hjemmearbeidsdager

DATO	TID	HVA
23.04.2010		Hjemmearbeidsdag.
22.04.2010	6,5 t	Dagen i dag har gått med til å rette JSP-sider i forhold til "Behandle ruter", og rette opp koden i sidetabell i "Sanntidskart". Vi hadde statusmøte 3 med Tom, og han syntes at prosessen vår var grei i forhold til planen vi hadde lagt opp.
21.04.2010	5,0 t	Sprintmøte 4 med ETC, hvor vi presenterte det vi hadde gjort, og la opp en slagplan for den siste sprinten. Her bestemte vi at skulle dele opp sprinten i to deler, hvor vi skal være "ferdig" med koding til neste onsdag, slik at vi kan presentere programmet som en helhet for ETC. På den måten kan de se om vi noe de savner i modulene, eller om det er noe vi kan forbedre. Neste delen av sprinten satte vi av til å forbedre programmet ytterligere (etter ETC sine tilbakemedlinger), finpussing av koden, samt begynne med rapporten for fullt. Øyvind syntes at vi var godt i gang i forhold til planen vår, selv om vi har funnet ut at vi ikke kan legge sammen rutene ved klikk i "Sammenligne ruter", slik at bruker må gjøre det manuelt i "Behandle ruter". Han påpekte at vi hadde fortsatt fått til en wow-faktor ved oppgaven, og kanskje spesielt med tanke på hva vi har fått ut av Google Maps. Dagen gikk med til rydding av layout i forskjellige JSP-sider, fikset funksjonalitet og plassering til ulike knapper, lagde statusrapport til i morgen, samt få inn dataene i sidetabellen i sanntidskartet.
20.04.2010		Hjemmearbeidsdag. I dag stod "Sammenligne ruter" og "Sanntidskart" i fokus. Siste dag av sprint 4.
19.04.2010	4,0 t	Vi brukte litt tid på morgenkvisten til å finne ut hvor vi står i forhold til sprint 4, og sett at det er noen småting som ETC har nevnt som vi burde ha hatt med i modulene. Det er noen vanskelige funksjoner som står igjen, som vi må nok bruke litt mer tid på, men alt i alt står vi ganske greit i forhold til planen med tanke på at dette er en stor oppgave, som har blitt litt større og annerledes enn det vi hadde først antatt. Vi fortsatte med hvert vårt ulike problem, og fikk fikset stoppested og tabellen med regisreringsnummerene i "Sammenligne ruter".

Uke 15

Tidsbruk: 22,5 t + hjemmearbeid

DATO	TID	HVA
16.04.2010		Hjemmearbeidsdag.
15.04.2010	6,5 t	Etter mye slit fikk vi opp adressen automatisk i adresse-skriveruten ved klikk i kartet i "Behandle ruter". Vi har jobbet med kartdelen i "Sammneligne ruter", samt prosessen med å hente ut data i forhold til "Sanntidskart". Det var ingen møte med Tom i dag.
14.04.2010	7,0 t	Møte med ETC. Øyvind var back in buisness etter sykdom, og vi fikk svar på flere av de mange "små" tekniske problemene vi har slitt med. I dag jobbet vi med "Sammenligne ruter", rettet opp tabellen i "Sanntidskart" hvor vi modifiserte hvordan man hentet ut data fra DB, samt jobbet med hvordan man får opp adressen i 'legg til nytt stoppested' i "Behandle ruter".
13.04.2010	6,0 t	Vi brukte litt på morgenkvisten til å se hvor vi stod i forhold til sprintene. Vi gjorde ferdig rettelsene vi fikk påpekt fra ETC, fikk opp lister over biler i "Sanntidskart" samt jobbet med hvordan man fikk opp markører og koordinater ved hjelp av klikk i kart i 'legg til rute' under "Behandle ruter". Vi begynte også på funksjonen "Sammenligne ruter".
12.04.2010	3,0 t	I dag jobbet vi med rapporten, samt rettelser i koden etter tilbakemelding fra ETC.

Uke 14

Tidsbruk: 16,0 t + hjemmearbeid

DATO	TID	HVA
09.04.2010		Hjemmearbeidsdag.
08.04.2010	6,5 t	I dag begynte vi på hovedrapporten. Det gikk fortsatt i rettelser i koden etter tilbakemelding fra ETC. Vi hadde også et kort møte med Tom, hvor han fikk sett hvor langt vi hadde kommet.
07.04.2010	7,0 t	Vi startet dagen med et to-timers sprintmøte 3 hos ETC. Øyvind var syk, så i dag fikk vi feedback fra Dag, som hadde litt å peke på med tanke på brukerhåndtering. Noen av disse tingene er ikke bare-bare, så vi har en liten nøtt å knekke i forhold til sprint 4 som vi påbegynte i dag. Vi jobbet med rettelsene i forhold til tilbakemelding og hjelp fra Dag, og vi klargjorde hva vi skulle gjøre i sprint 4.
06.04.2010	2,5 t	Første dag på skolen etter påskeferien. Formen var langt i fra topp etter all kvikk lunsjen vi hadde fått i oss i løpet av ferien, så man kan vel si at det var heller en laber gjeng som måtte begynne dagen med statistikkprøve. Vi gikk så løs på kodingen, og jobbet oss sakte men sikkert mot målene til sprint 3, som vi skal presentere i morgen for ETC.

Uke 13

PÅSKEFERIE! :D
Nå trengtes det et lite pusterom før de siste to intensive månedene! ;-)

Uke 12

Tidsbruk: 20,0 t + hjemmearbeid

DATO	TID	HVA
26.03.2010		Hjemmearbeidsdag.
25.03.2010	6,5 t	I dag fortsatte vi med løsningsendringene som Dag foreslo på møtet i går. Vi jobbet med å tegne eksisterende ruter i kartet i "Behandle ruter", og med databasen i forhold til "Sanntidskart". Vi gjorde også noen filendringer med tanke på Flåtestyrings-modulen. Vi jobbet også med å sortert rutene og stoppesteder på dato i "Behandle ruter" og få visningen av rutene i et datointervall (vha. kalenderfunksjon). Hadde også et møte med Tom.
24.03.2010	7,5 t	Vi hadde møte med Dag i ETC i dag. Vi fikk en god innsikt i hvordan vi skulle løse en del funksjoner, og da gjerne med tanke på sluttresultatet og da gjerne med tanke på brukerhåndtering. Dagen gikk med til å løse disse nye problemstillingene.
23.03.2010	4,0 t	I dag jobbet vi med å få inn forskjellige datoer på rutene i "Behandle ruter", slik at det kan hentes ut ruter for den/de valgte datoene. Vi forsøkte også å få tegnet inn de eksisterende rutene i et statisk kart. Vi hadde en bedriftspresentasjon med Accenture i dag, så en del tod gikk bort til dette.
22.03.2010	2,0 t	Koding.

Uke 11

Tidsbruk: 20,0 t

DATO	TID	HVA
19.03.2010		Hjemmearbeidsdag.
18.03.2010	6,5 t	I dag gikk vi gjennom møtereferatene, og samlet opp alle ønskene og bestemmelsene til ETC. Det var ikke mye vi hadde "glemt" å ta med av dette, men det er litt smått som vi har tenkt å ha med i senere sprinter. Vi begynte med kodingen av kalenderfunksjonen, samt jobbe med kartet i 'legg til rute' i "Behandle ruter", og hente data i "Sanntidskart".
17.03.2010	7,0 t	Vi hadde sprintmøte 2 med ETC på morgenkvisten i dag. De var litt bekymret over at vi hadde for store/spredte sprinter, men etter at vi fikk avklart at dette var gjort med tanke på at vi skulle slippe å jobbe med samme filer, og dermed slippe å bruke mye tid på bl.a. synkroniseringskonflikter, syntes de at det var et greit opplegg. Vi presenterte det vi hadde gjort, der de kun hadde noe småplukk å ta oss på. Vi fant ut etter møtet av vi skulle gå gjennom alle møtereferatene, og se på hva annet "smått" ETC hadde ønsket i applikasjonen. Vi hadde også statusmøte 2 med Tom. Han syntes at vi hadde jobbet bra med det ETC har bedt oss om å gjøre, men påpekte at vi manglet fortsatt wow-faktoren i oppgaven. Med andre ord manglet vi (men har forøvrig all intensjon om å gjøre) "Sammenligne ruter", og evt. gjøre noe sprell med "Sanntidskart" . Dette kan f.eks. være å finne de nærmeste bilene i tilfelle en sjåfør brekker et ben e.l. Møtereferat fra denne seansen. Både ETC og Tom stresset forøvrig med at vi må kommentere beslutningene våre i sluttrapporten, noe som vi opprettholder ved hjelp av prosjektdagboka. Vi hadde også et rapportkurs med Frode i dag, hvor vi fikk gjennomgått hvordan rapporten skulle skrives. Ellers gikk dagen til koding med tanke på innspill og retting fra ETC, samt sprintmålene i sprint 3.
16.03.2010	6,5 t	I dag jobbet vi med kartverket i 'legg til nytt stoppested' i "Behandle ruter", og databasen til "Sanntidskart". Siden ETC ikke hadde noe ferdige testdata i forhold til Flåtestyrings-modulen, vil vi nå ta utgangspunkt i en testtabell. Vi fortsatte også med tabellen i forhold til kjøretøysgruppe i "Behandle ruter". Vi kom frem at vi ikke var ajour i forhold til det vi hadde satt opp vi skulle gjøre i sprint 2, men bestemte oss for å ta med det tapte i sprint 3. Vi skisserte dermed opp løpet i sprint 3, som vi vil ta med til ETC for godkjenning. Vi lagde også en statusrapport til statusmøte 2 i morgen, og sendte den til Tom.

Uke 10

Tidsbruk: 26,5 t

DATO	TID	HVA
12.03.2010	6,0 t	I dag fortsatte vi med kartverket, samt avkrysningstabellen til kjøretøysgruppe i "Behandle ruter".
11.03.2010	6,5 t	I dag fortsatte vi med kartverket i "Behandle ruter". Vi jobbet også med å tilpasse kartzoom i forhold til markørene, i "Sanntidskart", og få opp linkene i den tilhørende sidetabellen til å gjøre det den skulle. Vi jobbet også med å knytte opp kjøretøysgruppe med ruter i "Behandle ruter", slik at avi ble nesten ferdig med avkrysningstabellen. Møte med Tom, hvor vi så litt på hvor langt vi hadde kommet, og diskuterte litt rundt rapportskrivingen. Han var igrunn meget fornøyd med progresjonen vår.
10.03.2010	8,0 t	Vi hadde ikke møte med ETC i dag, da vi følte at vi ikke hadde noe spesielt å ta opp med de, og motsatt. Vi jobbet videre med kartverket i både Flåtestyring og Ruteplanlegger, da vi hadde kommet frem til at vi skulle jobbe på forskjellige deler av modulene, slik at vi unngikk å jobbe på samme filer og slite med synkroniseringskluss. Vi fortsatte også med å knytte biler til rutene i "Behandle ruter". Vi la også til menypunktet "Galleri" på hjemmesiden, med noen bilder av det vi hadde fått til så langt.
09.03.2010	6,0 t	Vi jobbet med kartverk, og å knytte kjøretøysgruppe til rutene i "Behandle ruter".

Uke 9

Tidsbruk: 27,0 t

DATO	TID	HVA
05.03.2010	6,0 t	Koding.
04.03.2010	6,5 t	Dagen gikk med på feilretting i forhold til vektorer og 'rediger rute' i "Behandle ruter", og se på hvordan vi skulle inkorporere ønskene til ETC. Vi laget også sidene "Sanntidskart" og "Historiekart", med fungerende basiskart. Vi hadde også et møte med Tom, hvor vi presenterte det vi hadde gjort til nå, og diskuterte litt rundt hva vi burde ta med i forhold til koding i rapporten, og framføring vi skal ha i juni.
03.03.2010	5,5 t	Sprintmøte 1,5 med ETC. Vi presenterte det vi hadde gjort, herunder 'legg til'/'slett'/'rediger rute' i "Behandle ruter" (dog med litt feil), med flere ruter og stoppesteder. Vi hadde også fått til en god del på kartfronten, slik at vi hadde et funksjonelt kart som tegnet en rute etter vegen (dog kartet kun tok utgangspunkt i hardkodede koordinater). Øyvind var storfornøyd, men ønsket å knytte kjøretøysgruppe til ruten, og at stoppestedene ble vist kronologisk i forhold til tiden.
02.03.2010	6,0 t	Vi fikk ferdigstilt 'rediger rute' i "Behandle ruter", slik at vi kom mer eller mindre i mål med sprint 1,5.
01.03.2010	3,0 t	I dag fikk vi ferdigstilt 'slett rute' i "Behandle ruter", og begynte med 'rediger rute' i "Behandle ruter". Vi jobbet også med kartvert, og vektorer.

Uke 8

Tidsbruk: 26,0 t

DATO	TID	HVA
26.02.2010	6,0 t	I dag trøblet vi en del med tanke på synkronisering av filer. Siden flere gruppemedlemmer jobbet med samme filer, ble det mye kluss i koden slik at det gikk mye tid til dette. Vi jobbet med kartverket i 'legg til rute' i "Behandle ruter", samt databasen og vektorer i forhold til å få til en fullverdig databasetabell med flere ruter/stoppesteder istedet for bare en rute/ett stoppested som vi hadde som utgangspunkt til nå.
25.02.2010	6,5 t	I dag jobbet vi med kartverket, vektorer og database i forhold til stoppested i 'legg til rute' i "Behandle ruter".
24.02.2010	7,0 t	Vi hadde et møte med ETC på morgenenkvisten, hvor vi bl.a. fikk avklart noen loggproblemer. Resten av dagen gikk med til koding, hvor vi bl.a. fikk en funksjonell, dog ikke en fullstendig, database.
23.02.2010	6,5 t	Vi fikset feilen vi hadde i forrige sprint i forhold til null-pekere. Vi jobbet videre med kartverket, og fikk opp stoppestedstabellen. Vi hadde en statistikkprøve i dag, så det gikk med en del tid til dette.

Uke 7

Tidsbruk: 23,5 t

DATO	TID	HVA
19.02.2010		Hjemmearbeidsdag.
18.02.2010	4,5 t	Koding. Første statusmøte med Tom, hvor han syntes at fremgangen vår virket grei.
17.02.2010	7,0 t	I dag gikk vi gjennom koden, og prøvde å finne ut hvorfor ikke dataene vil lagre seg. Vi fastsatte at vi måtte utvide prosessen med en ny sprint, sprint 1,5, hvor vi besluttet at vi skulle fokusere på kartverket og fortsette med funksjonalitetene i "Behandle ruter" slik at denne blir ferdigstilt. Vi hadde sprintmøte 1 med ETC, hvor vi presenterte det vi hadde gjort, og fikk hjelp til å få opp databasetabellen.
16.02.2010	8,5 t	Vi hadde en effektiv dag med tanke på koding. Vi kom nesten i mål med 'legg til rute' i "Behandle ruter", men vi hadde fortsatt problemer med å få delene til å passe sammen og få opp databasen. Vi besluttet at vi til å begynne med kun tar utgangspunkt i en rute med ett stoppested.
15.02.2010	3,5 t	Koding i forhold til 'legg til rute' i "Behandle ruter".

Uke 6

Tidsbruk: 25,5 t

DATO	TID	HVA
12.02.2010	6,0 t	I dag diskuterte vi hvordan vi skulle løse brukergrensesnittet på en best mulig måte. Vi jobbet med å få lagret unna data, og fikk implementert en eksisterende database fra ETC som vi kunne teste mot. Herunder installerte vi også MySQL slik at vi fikk en oversikt over databasen og SQL-koden vi kjørte. Vi bestemte oss for å kutte ned sprinten vi var i, og fokuserte på å fullføre 'legg til rute' i "Behandle ruter", slik at det blir mer tid på å lære oss gangen i den eksisterende koden i CarAdmin og komme inn i gode koderutiner.
11.02.2010	6,5 t	Vi kom godt i gang med funksjonen 'legg til rute' i "Behandle ruter", slik at rammeverket kom på plass selv om funksjonaliten ikke fungerte. Vi hadde et møte med Tom, hvor vi gikk gjennom kravspesifikasjonen. Han hadde ikke noe spesielt å peke på, og vi diskuterte litt om arbeidsprosessen vår i forhold til Scrum.
10.02.2010	7,0 t	Vi hadde et møte med ETC på morgenkvisten, hvor vi så litt på kravspesifikasjonen, og fikk ordnet opp i feilmeldingene i debug-vindet til Ecplise. Vi jobbet også med JSP-filene, javafiler og database-opplegget i 'legg til rute' til "Behandle ruter".
09.02.2010	6,0 t	I dag jobbet vi med JSP-filene i forbindelse med 'legg til rute' i "Behandle rute", samt kartverket.

Uke 5

Tidsbruk: 24,0 t

DATO	TID	HVA
05.02.2010	6,0 t	Etter kommentarene fra Tom fortsatte vi med designdokumentet, jobbet litt med hjemmesiden og begynte med utforskning av Google Maps for fullt.
04.02.2010	8,0 t	I dag begynte vi å se på første sprint, og delte opp sprinten i delene vi skulle lage og fordelte ansvar over hvem som skulle gjøre hva av disse. Vi begynte så å jobbe på JSP-filene (som gir utseende til sidene), og fikk opp ulike knapper og menyer. Vi fant kjapt ut at det var mye kode sette seg inn i, med tanke på det som eksisterer i CarAdmin. Vi begynte også å se litt på Google Maps sitt API, og hvordan vi kunne bruke koden deres videre. Vi oppdaterte også designdokumentet, slik at vi ble ferdige med grunnstrukturen og det vi kunne gjøre for denne gangen. Vi hadde også et møte med Tom, hvor vi fikk tilbakemelding på forprosjektet, samt litt kommentarer til designdokumentet og arbeidsprosessen videre.
03.02.2010	4,0 t	Vi hadde et møte med ETC på morgenen. Vi planla så hvordan vi skulle dele opp sprintene, og så litt mer på det som manglet i designdokumentet. Mye av dagen gikk bort til karrieredagen som ble holdt på skolen, hvor vi så på stands og flikket på CVen.
02.02.2010	6,0 t	I dag gjorde vi oss så og si ferdig med designdokumentet, og begynte å se på koden.

Uke 4

Tidsbruk: 22,0 t

DATO	TID	HVA
29.01.2010	5,5 t	I dag leste vi over kravspesifikasjon, rettet den ytterligere og sa oss ferdige med den. Vi fortsatte så med designdokumentet. Vi fikk også en innføring i utviklingsmiljøet og filsystemet i CarAdmin med Øyvind.
28.01.2010	6,5 t	Vi leste gjennom forprosjektrapporten, og la den ut for offisiell vurdering på hjemmesiden. Vi jobbet så med kravspesifikasjonen (herunder UseCase, sekvensdiagrammer, teknisk memoer, domenemodell og litt småpirk), og fortsatte med designdokumentet.
27.01.2010	5,0 t	Vi hadde et kort møte med ETC. Her fikk vi avklart noen viktige punkter i forhold til kravspesifikasjonen. Vi fikk også tilbakemelding fra Tom på forprosjektet, som vi fikk forbedret ytterligere. Det ble en kort dag i dag pga andre møter.
26.01.2010	5,0 t	I dag fikk vi hentet de ferdigsignerte prosjektavtalene (til arbeidsgiver og oss) fra Hilde Bakke, og fått bekreftet at prosjektavtalen hadde blitt arkivert i skolens system. Vi jobbet med kravspesifikasjonen og produktkøen. Vi så at det var endel som stod igjen før kravspesifikasjonen stod ferdig, men vi fikk ikke gjort noe mer før etter møtet vi skulle ha med ETC i morgen. Vi påbegynte også designdokumentet, slik at skjelettet kom på plass.

Uke 3

Tidsbruk: 20,5 t

DATO	TID	HVA
22.01.2010	5,0 t	Vi hadde et møte med Tom i dag, men kommentarer til forprosjektet måtte påventes til Tom fikk tid til å se den over. Vi fikk derimot tilbake en kommentert utgave av forprosjektrapporten fra ETC, og begynte å rette den deretter. Vi fikk også besøk av Øyvind, hvor vi fikk satt opp utviklingsmiljøet og installert nødvendig programvare slik at Eclipse og databasen vår samhandlet riktig med ETC sin server.
21.01.2010	3,5 t	I dag fikk vi de siste delene av forprosjektet på plass, slik at vi fikk levert førsteutkastet av rapporten til oppdragsgiver og veileder for tilbakemelding.
20.01.2010	7,0 t	I dag hadde vi et møte med Øyvind i ETC. Vi fikk underskrift av arbeidsgiver og hvert av gruppemedlemmene på prosjektavtalen, som vi igjen fikk levert til Hilde Bakke, slik at vi kunne avhente den ferdigsignert av dekan neste uke. Vi arbeidet også med forbedring av forprosjektrapporten, og gjorde oss nesten ferdig i forhold til å sende førsteutkastet til vurdering til ETC og Tom. Vi påbegynte også kravspesifikasjonen, og fortsatte med å forbedre hjemmesiden.
19.01.2010	5,0 t	I dag fortsatte vi med å forbedre forprosjektrapporten, og planla morgendagensmøte med ETC.

Uke 2

Tidsbruk: 19,5 t

DATO	TID	HVA
15.01.2010	5,0 t	I dag jobbet med forprosjektet, og oppdaterte hjemmesiden vår ytterligere. Vi fikk også installert utviklingsmiljøet vi skulle ta i bruk. Under forprosjektet diskuterte/undersøkte vi hvilken utviklingsmodell vi skulle bruke fremover, og kom fram til at scrum var den mest optimale modellen for oss og vår oppgave. Vi var også på biblioteket, og så gjennom gamle bacheloroppgaver for dataingeniører. Vi lånte med oss noen relevante oppgaver, og fikk sett litt på hva vi likte og ikke likte med de andre oppgavene.
14.01.2010	6,0 t	I dag fikk vi gjort ferdig store deler av forprosjektrapporten. Herunder spikret vi fast rollene til hvert av gruppemedlemmene og fikk gjort klar prosjektavtalen til underskriving. Vi hadde også et møte med Tom, og avtalte videre ukentlige møter på torsdager kl 1300. Vi oppdatert også hjemmesiden vår, med blant annet bilder av oss.
13.01.2010	5,0 t	I dag hadde vi et lynkurs med Tom Røise, hvor vi fikk tips i forhold til bacheloroppgaven. Vi diskuterte og planla så aktiviteter for uken, og begynte på forprosjektet og fortsatte arbeidet på hjemmesiden.
12.01.2010	2,0 t	Vi hadde et møte med Øyvind i ETC, hvor vi bl.a. fastsatte videre ukentlige møter på onsdager kl 0800. Vi jobbet så litt videre med hjemmesiden.
11.01.2010	1,5 t	Oppstart av bacheloroppgaven! :-) Vi diskuterte litt rundt formalia, og utformet grupperegler som samtlige av gruppemedlemmene underskrev. Vi påbegynte også hjemmesiden.

Logg for Else **Sum: 487 timer**

17	26.04.2010	9	Behandle ruter: Laget ny patch med forandring i adressen og hompoint i stops og routes; disse måtte bli lengre. Rettet opp feil med de frie rutene i rediger ruter og satte på pil i newstop.jsp sitt kart. Sammenligne kart: Sammenligningskart har fått sidebar med valg over hvilken ruter som skal vises i kartet.
	27.04.2010	9	Flåtestyring: Gjorde ferdig realmap.jsp. (6t) Skrev spm. Til ETC. Hovedrapport: Skrev om Gjennbrukt og modifisert kode under pkt. 4 (3 t)
	28.04.2010	5	Møte med ETC. Skrev møtereferat. Gjorde ferdig feilsjekking i newstop.jsp og ChooseCar.jsp. Sammenligne ruter: Undersøkte hvorfor Google Maps ikke vil geokode flere enn 10 adr. etter hverandre. Og fant løsning på problemet.
	29.04.2010	13	Flåtestyring: Laget søkefunksjon for å kunne se nærmeste bil og ruten dit i realmap.jsp. Hadde møte med veileder. Sammenligne rute: Gjorde om compare.jsp til å tegne markørene ut ifra koordinater.
	30.04.2010	6	Behandle Ruter: Laget mulighet for å vise flere resulateter for adressen hvis disse finnes, bruker kan da velge den korrekte. (Feilsjekking av adressens gyldighet)
	01.05.2010	7	Flåtestyring: La til informasjon om bilenes tur i popupbobblen. Sammenligne ruter:Gjorde om compare.jsp til å tegne rutene ut ifra koordinater. Skrev møtereferat fra møtet med veileder.

Figur 42 Logg for Else

Logg for Marte 441,5

Uke	Dato	Antall timer	Arbeidet med
17	26.04.2010	3,5	Sanntidskart: Testing i forhold til kodeproblemet med tabellen. La på feilsjekk, og fikk testet databasen, og fikk rettet opp noen feil i forhold til tabellen. Fungerer fortsatt ikke helt som den skal. Vedlikehold av hjemmeside (logg og oppdatering av møtereferat med tilhørende linker).
	28.04.2010	8	Møte med ETC, rettet kravspesifikasjon ytterligere, kjørte gjennom patchene i Query Browser, slik at DB ble
	29.04.2010		Bortreist. Møte med Bjørknes!
	30.04.2010	2,5	Oppdaterte meg på hva som ble gjort, begynte på Historiekart, og skisserte opp hvordan den skulle
	01.05.2010	3	Historiekart: Kopierte nødvendig kodebegynte å kopiere kode som jeg trengte
	02.05.2010	2,5	Historiekart: Tabell, sidetabell, problemer med kart
18	03.05.2010	7,5	riktig visning, samt fått opp tabell. Rettet opp en del i databasen (realtime har nå blitt PK, slik at vi kan legge inn biler med flere punkter, samt ordnet litt på floatene som har blitt lengre).
	04.05.2010	10,5	Historiekart: Fått opp kalenderen, hvor bruker kun kan velge fjorten dager tidligere (feil: kan fortsatt velge alle datoer etter dagens dato), fått opp kart, hvor alle punkter til valgt bil i riktig datointervall blir plottet, og tegnet rute mellom. Fått opp informasjonsrute i fhrold
			Gikk gjennom DB, hvorav jeg fikset noen feil. Fant endel problemer vedrørende koden vår i forhold til firefox. Fikk spurt og rettet opp en del ting i forhold til dette.

Figur 43 Logg for Marte

Logg for Trine 450

Uke	Dato	Antall timer	Arbeidet med:
	29.04.2010	6,5	Tidsberegning, møte med Tom, oversetting
	30.04.2010	7,0	omgjøring av tid, begynt på koordinater fra newStop til DB
	01.05.2010	6,0	på skolen!, Fikset lagring av koordinater. Rapport: nummerering og designdok. Pluss domenemodell og begynt å se på flytdiagram.
	02.05.2010		
18	03.05.2010	5,5	flytskjema, punkt-orientering i add
	04.05.2010	9,0	planlegging, kodefiks
	05.05.2010	8,0	Møte med ETC, fikset slik at showRoute viser ut i fra koordinater ++ + rapport.
	06.05.2010	8,0	Rapport og feilfiks etter menykonvertering.
	07.05.2010	4,5	Rapport, design
	08.05.2010		
	09.05.2010		
19	10.05.2010	7,0	Rapport
	11.05.2010	11,0	skrevet kapittel 6! :)
	12.05.2010	10,0	Møte med Øyvind + Rapport for harde livet
	13.05.2010		Kristi Himmelfart- konfirmasjon
	14.05.2010	6,0	Rapport
	15.05.2010		
	16.05.2010		
20	17.05.2010		17-mai-feiring
	18.05.2010	13,0	Rapport!

Figur 44 Logg for Trine

I. Prosjektdagbok

Prosjektdagbok – Beslutninger og ansvarsforhold

13/1-2010

Prosjektgruppen beslutter at Trine skal være prosjektleder og Else sekretær, mao. ansvarlig for dokumentasjon. Ukentlig Timeplan er laget og det er enighet om at denne skal følges så godt det lar seg gjøre ut vårsemesteret. Se Timeplan på hjemmesiden. Under møtet med ETC ble det enighet om ukentlige møter hver onsdag kl. 08.00, se Møtereferat ETC 120110.

14/1-2010

Prosjektgruppen beslutter at Marte skal være ansvarlig for websiden og den felles loggen som ligger her. Det er videre besluttet at enhver ny informasjon vi mottar fra ETC og evt. Veileder som vil ha innvirkning på sluttrapporten skal innarbeides i denne så fort som mulig. Under møtet med veileder ble det enighet om ukentlige møter hver torsdag kl. 13.00, se Møtereferat veileder 140110.

15/1-2010

Prosjektgruppens valg av systemutviklingsmodell har falt på Scrum. Else har fått ansvaret for de lånte Hovedprosjektene fra Biblioteket, hun har også ansvaret for å levere tilbake disse eller fornye lånefristen på dem før 12. Februar. Vi har lånt: BilBooking CaraAdmin, Klientbasert reserving av standard kalenderapplikasjoner og Kontrollsystem for Autoklav.

19/1-2010

Trine har ansvaret for å skrive ut og ta med riktig antall kopier av kontrakten til møtet med ETC.

20/1-2010

I samarbeid med ETC har vi kommet fram til at det vil bli utarbeidet en møteinnkallelse med agenda før hvert av våre møter med dem slik at begge parter er klar over hva som skal tas opp på møtet, og partene kan da også vurdere på forhånd om et møte virkelig er nødvendig. Prosjektgruppen har besluttet å ikke bruke Subversion, siden kravet til en versjonskontroll i Prosjektet allerede er tilfredstilt med Subclipse.

21/1-2010

Har sendt forprosjekt til forvurdering til ETC og Tom.

28/1-2010

Forprosjekt og oppgavebeskrivelse er publisert på nettsiden. Tom vil også hente den herfra.

10/2-2010

Vi har bestemt oss for å bare fokusere på å bli ferdig med Behandle Ruter denne sprinten og derfor ta Generer og Vis Kart i neste omgang, siden Scrum promoterer å bli ferdig med aktuell sprint over alt annet.

12/2-2010

I denne sprinten skal Legg til Kart under Behandle Ruter være ferdig. Vi ser at vi ikke klarer og bli ferdig med hele Behandle Ruter og avbryter derfor denne sprinten for å begynne på ny med bare Legg til Rute.

16/2-2010

I tillegg til avgrensningene nevnt overfor velger vi å korte ned Legg til rute til bare å lage en rute med ett stoppested slik at vi har noe å levere på onsdag ettermiddag.

17/2-2010

I dag ble hva vi har gjort til nå vist frem for ETC. De var fornøyd med vår fremgang selv om vi ikke kom helt i mål med sprinten vår. Vi mangler slett og rediger både for ruter og stoppesteder men, vi har klart å lage en rute med et stoppested. Og dermed nådd vårt redigerte mål.

18/2-2010

I vår 1 ½. Sprint velger vi å fokusere videre på å gjøre ferdig ”Legg til ruter” samt ”Endre rute” og ”Slett rute”. Samtidig vil vi fortsette å jobbe med ”Generer og vis kart” hvis vi står fast på noen av de andre punktene over, slik at vi hele tiden har noe å gjøre. Vi har besluttet å ha rutenummeret til rutene i tabellen for stoppesteder slik at vi på en enkel og grei måte kan hente ut de som tilhører en spesifisert rute.

23/2-2010

Under ”Behandle Ruter” i menyen vil bruker først måtte velge dato fra en kalender og de forskjellige rutene vil ligge herunder. På denne måten vil alt rundt dato bli håndtert her og ikke i verken stopp eller ruter.

3/3-2010

Målene for sprint 1½ ble nådd til fristen. Behandle Ruter med Legg til Ruter, Rediger Ruter, Slett Ruter, Legg til Stoppested, Rediger Stoppested og Slett Stoppested er nå ferdig. Samt at både rutene og stoppestedene vises i egne tabeller. Vi har også rukket å fortsette litt med ”Generer og Vis Ruter”, samt sett mer på vektorene vi trenger i ”Sammenlign Ruter”. Målet for sprint 2 er å bli ferdig med ”Generer og Vis Rute”, samt integrere mulighet for å sette tilhørighet for en rute til en bruker v.h.a. en bil eller flere biler i ”Behandle Ruter”. Samt å integrere mulighet for å bytte om plassene på stoppestedene i rekken.

Det er besluttet at oppmøtetidspunkt og avreisetidspunkt i stoppesteder skal være av typen datetime, og inneholde ca. tider som bruker taster inn. (Uten sjekker på om dette er relevant for ruten.)

17/3-2010

Sprint 2 ble i dag ferdig, og målene har blitt nådd.

24/3-2010

Etter å ha snakket med Dag fra ETC i dagens møte har vi besluttet å endre måten funksjonaliteten vår blir presentert på i applikasjonen, slik at dette blir mer instinktivt og lett for brukeren. Dette vil si at vi skiller visning av rutene v.h.a. klikking i kalender fra det å lage en ny rute. At det første bruker får lov å gjøre i en ny rute er å velge bilgruppe ruten skal tilhøre, og at bruker så deretter kommer rett til inntastningen av informasjon til det nye stoppestedet(dette er starten av ruten). I denne inntastingen skal lokasjonen til bilen komme som forslag til startadresse.

For denne sprinten vil dette si at vi må modifisere noen av målene vi har satt oss og redigere noe gammel kode fra forrige sprint. Det er tross alt liten vits i å utvikle funksjonalitet ETC ikke trenger likevel, selv om dette er et lite brudd med prinsippene for Scrum der målene for sprinter skal være låst.

Fokuset i applikasjonen vår skal være på funksjonaliteten og ikke på designet, dette er nedprioritert.

26/3-2010

Prosjektgruppen har påskeferie til 6/4-2010.

7/4-2010

Prosjektgruppen har besluttet at hver deltager skal bruke ca en dag i uka på å skrive prosjektrapport, slik at vi gradvis kan gå fra bare koding av applikasjonen til mer og mer skriving av rapport etter som 20. mai nærmer seg.

21/4-2010

I sammenligne ruter kan vi allikevel ikke slå sammen ruter direkte, da dette fort vil føre til usammenhengende og ulogiske ruter. Lar derfor sammenslåingsfunksjonaliteten ligge på is.

28/4-2010

Fant ut Google Maps ikke vil geokode flere enn ti adresser om gangen, og har dermed besluttet å gjøre om lagringen vår, slik at vi også lagrer unna koordinatene til hvert stoppested, i tillegg til adressen, slik at vi slipper å utføre ”reverse geocoding” for å tegne rutene. Dette implementeres i alle rute-tegnende sider.

J. Møtereferater

Møtereferat ETC 270110

Deltagere: Marte S. Bjørseth, Trine A. Grønvold, Øyvind Flatval og Else Dalby

Referant: Else Dalby

Tidspunkt: 08.00 27/1-2010

Tema:

- Bruk av funksjoner som CarAdmin allerede har og som vi trenger i Ruteplanlegger/ Flåtestyring.
- Det er et system for rettighetsstyring allerede i CarAdmin så dette slipper vi å lage. Men Øyvind anbefaler oss at vi starter med å tenke at den bruker som til hver tid er innlogget er administrator. Og at rettighetsgradering blir en tilleggssak som tas til slutt.
- Feillogging finnes også i CarAdmin, og dette skal vi bruke videre i Ruteplanlegger/Flåtestyring. Loggene vi skal bruke heter Javalogger; skriver ut til fil og dos-vindu. En automatisk "feilfanger" som automatisk sender mail med all info om feilen når dette inntreffer. Utover dette er det normalt å ha en sporingslogg; denne skal logge alt som skjer i debug-vinduer etc. Dette er fint å ha i forhold til utviklingen vår, men dette blir også noen ganger brukt til de andre loggene.
- Det er flere kalendere i CarAdmin som vi kan bruke.
- Vi skal bruker ETC sin database.

Vi har avtalt møte med Øyvind på fredag 29. januar kl. 09.30. for å gjennomgå resten av innføringen i utviklingsverktøyet og noe av koden i CarAdmin.

Øyvind sier at vi kan anta at vi har fått med oss det meste i forprosjektrapporten og at dette er i orden, så får vi heller snakke litt med Dag senere. Han er nemlig opptatt i dag.

Øyvind foreslår at bare administrator har tilgang til Flåtestyring. Dette gjør det enklere i forhold til personvern og sensitive opplysinger.

Når det gjelder domenemodellen er denne fin slik den fremstår i Mappe 3 OOSU fra i høsten 2009. Får å få med Flåtestyring her må vi legge til Bil og Posisjon knyttet til Rute.

I sanntidskartet skal alle biler som tilhører en hjemmesentral vises. Når det gjelder historie-funksjonen skal siste turen for valgte bil vises i kart med en meny på siden, som innholder andre ruter bilen har kjørt den siste arbeidsdagen/6-8 timene/dagen. Hvis mange biler skal vises på en gang i historiefunksjonen burde disse vises som prikker. Øyvind anbefaler oss å vente med konkrete beslutninger her til vi begynner å kode de enkelte delene.

Når det gjelder Produktkøen så sett opp dette i en logisk rekkefølge, begynn på begynnelsen og fortsett steg for steg med de aktivitetene som trengs.

Møtereferat veileder 100429

Deltagere: Marte S. Bjørseth, Trine A. Grønvold, Tom Røise og Else Dalby

Referent: Else Dalby

Tidspunkt: 1300 29/4-2010

Agenda: Rapportskriving

Det er smart å skrive om hvordan vi har brukt forum og div. guider på internett for å få tips, hjelp og å lære nye metoder som kan brukes i Google Maps.

Det er et spennende aspekt at vi klarer å bruke en ekstern kilde, og derved overstyre Google Maps. Et slikt aspekt vil være omkjøringer i rutene. Vi har Techincal Memos på dette, men har ikke nok tid til å implementere dette. Relevant problemstilling. Det er den samme problemstillingen med å få beskrivelse av stoppested inne i kjørebeskrivelse.

Vi må selv finne ut hva som er hensiktsmessig her.

Lurt å ha noen utenforstående til å teste programmet slik at det ser og oppfører seg logisk.

Et førsteinntrykk kan aldri bli omgjort. Derfor må dette være topp. Spesielt i de første to-tre sidene er det viktig med en bra fremførelse.

Vi må ha med hvordan vi har brukt, tilpasset og omgjort Google sine elementer og våre egne. Ikke kjempe mye eks. men nok til å vise det godt.

Vi har avtalt med ETC at de skal lese rapporten vår.

Ikke vær redd for å kaste ting. Rapporten leses i sin helhet av sensor, og hvis det er mye gjentagelser her så blir dette kjedelig. Vis et spekter av det vi har og legg resten som vedlegg. Dette gjelder også de tekniske memoene. Saftig diskusjon er viktig.

Samme på bruk av figurer og skjermbilder, ikke bruk så veldig mye likt her. Diskusjon på hvordan vi har valgt å implementere ETC sin brukerdesign. Problematikk? Vi har hatt en ramme her… Bra? Dårlig? i vår anvendelse av dette.

Fargekode på hva som er vårt, hva som er Google sitt og hva som er ETC sitt…

K. Sprintkøer

Sprint 1:

- Behandle ruter
    - Legg til rute
    - Velg hjemmepunkt
    - Legg til stoppested
        - Lengde og breddegrader
        - Adresse
        - Klikk i kart
    - Legg til stoppestedtid
    - Til
    - Fra
    - Legg til annen stoppestedsinformasjon
    - Legg til sluttpunkt
    - Sett forbindelser(Tegn rute)
    - Sett som ”aktiv”
    - Vis i enkeltkart

- Javakode legg til: felles
- Slett rute: Marte
    - Velg rute
    - Sett som ”inaktiv” → Fjern fra kart
    - Fjern forbindelser?
    - Fjern punkter
    - Fjern fra hjemmepunkt
- Endre rute: felles
    - Velg rute
    - Sett ruta som ”inaktiv”
    - Velg punkt

  - Plukk punkt ut av ruta
  - Endre punkt
  - Legg punkt tilbake i ruta
  - Lagre ruta. → Sett som ”rute”
  - Eller: Slett ruta.

- Annet
  - Språk til slutt i sprinten
  - Sette seg inn i alt eksisterende
  - JSP-fil: knapper og utseende (Else)
  - Gjenstår til senere sprinter:
  - Legge rutene ifm kalender
  - Brukerrettigheter
  - Kart
- Forutsetninger:
  - Alle er administrator
  - Det finnes bare en dag

- Evt. tar for oss kun ”legg til rute”, hvis det viser seg at vi får for dårlig tid på sprinten.

Sprint 1,5

Få til Vector - lister!

- Ruter[]
- Stoppesteder[]

Slett rute

- Velg rute
- Sett som ”inaktiv” → Fjern fra kart
- Fjern forbindelser?
- Fjern punkter
- Fjern fra hjemmepunkt

Endre rute

- Velg rute
- Sett ruta som ”inaktiv”
- Velg punkt
- Plukk punkt ut av ruta
- Endre punkt
- Legg punkt tilbake i ruta
- Lagre ruta. → Sett som ”rute”
- Eller: Slett ruta

Extra:

- kart

Sprint 2

Generer og vis kart

Generer og vis rute

- Få opp kartet
- Få opp kartpunkt ved hjelp av:
  - klikk i kart
  - koordinater
  - adresse
- Tegn i kartet
  - Mellom to steder
  - For hele ruta
- Legg inn ruteinfo i tabell (database)
  - Lagre fra kartklikk,
  - Eller koordinater.

Legg inn hjemmepunkt

- Slik at riktig hjemmepunkt/lokasjon vises i kartet

Begynne å se på sammenligne ruter

- Dette for å forhindre at vi ikke må endre på eksisterende kode alt for mye

Endre på ”håndtere stoppested”

- Stoppesteder kan legges til mellom eksisterende stoppesteder
- Stoppesteder kan endres rekkefølge

Knytte ruter til biler

- Knytte ruter til bilgrupper
- Knytte ruter til biler

Sprint 3

Generer og vis kart

Generer og vis rute

- Få klikk i kart til adresse
- Legg inn ruteinfo i tabell (database)
    - Lagre fra kartklikk
    - Eller koordinater

Legg inn hjemmepunkt

- Slik at riktig hjemmepunkt/lokasjon vises i kartet

Sider

- Add.jsp
- newStop.jsp
- editStop.jsp

Flåtestyring

Sanntidskart

- Lage testdatabase
- Hente ut data fra testdatabasen
- Få sidetabell klikkbar
- Zoom

Kalender

- Legg til ruter i kalender
    - Valg: periode eller dag (Se kjørebok)

Endre på ”håndtere stoppested”

- Stoppesteder kan legges til mellom eksisterende stoppesteder
- Stoppesteder kan endre rekkefølge

Dokumentasjon

- Begynne å skrive rapport

Til senere sprint:

- Historiekart
- Sammenligne ruter
- Brukerhåndtering
- Kopiering av ruter?
- Klokkeslett oppstart av rute
- Tilpass hjelp [?]
- Tilpass sider til overskriftkanten
- Legg til ikoner til slett/rediger/osv.?
- Skrive ut vegbeskrivelser med kartet.?

Sprint 4

Generer og vis kart

Generer og vis rute

- Hente ut tidsforbruk til ruten
- Få klikk i kart til adresse
- Lagre fra kartklikk
- Eller koordinater

Legg inn hjemmepunkt

- Slik at riktig hjemmepunkt/lokasjon vises i kartet

Sanntidskart

- Hente ut data fra testdatabasen
- Få sidetabell klikkbar

Dokumentasjon

- Hvordan organisere rapporten
- Forbedre forprosjektrapport
- Kravspesifikasjonen
- Gjøre ferdig designdokument
- Innledning av rapport
- Fordele arbeidet

Sammenligne ruter

- Begynne å se på denne funksjonen
- Else sitt forslag:
  RouteIOSQL – lese en og en linje fra rutene, lage en funksjon som legger rutene i vektor/array. Ta en rute, les den ruta, be stoppestedet registrere seg, ta vektoren som vi har i RouteBO, og legg over.
  Getroute (returnerer ruteobjekt)
  Getstop (returnerer stoppeobjekt)

Historiekart

- Hente data fra kjørebok
- Tegne rutene

Til senere sprint:

- Brukerhåndtering
- Kopiering av ruter?
- Klokkeslett oppstart av rute
- Tilpass hjelp [?]
- Tilpass sider til overskriftkanten
- Legg til ikoner til slett/rediger/osv.?
- Skrive ut vegbeskrivelser med kartet.? Ruteliste?

Sprint 5

DEL 1 (Ferdig 28/4-2010)

Da skal dette være ferdig, og kunne bli presentert for ETC, slik at de kan se på evt. forbedringer i forhold til brukerhåndtering eller andre ting de savner. Her vil de vurdere om det skal gjøres noe mer.

Generer og vis kart

Generer og vis rute

- Hente ut tidsforbruk til ruten
- Gjøre om til en hånd-ikon, ved der det er mulighet til å klikke i stoppesteder

Sanntidskart

- Få sidetabell klikkbar
- Hente ut siste timene av ruten
- Brukerinformasjon, ruteinformasjon
- Oppdater-knapp
- Får kun siste gps-koor som markør

Vis rute

- Bli implementert under behandle rute, slik at denne overskriften går bort
- Ta bort funksjonalitet (får ikke velge noen)

Dokumentasjon – skal leveres til ETC for retting

- Forbedre forprosjektrapport
- Forbedre kravspesifikasjonen
- Forbedre designdokument
- (Innledning av rapport)

JSP-sider

- Feilsjekking newstop.jsp (ikke sender inn tomme celler)
- Rydding i layout i functions.jsp og compare.jsp (noen flere?)
- Implementering av OnChange i functions.jsp og compare.jsp sine kalendere

Historiekart

- Hente data fra kjørebok
- Tegne rutene
- Velg dag

DEL 2 (Ferdig 5/5-2010)

Her vil få tilbakemelding fra ETC, om spesielt brukerhåndtering, og se om det er noe vi mangler.

- Brukerhåndtering
- Tilpass hjelp [?]
- Tilpass sider til overskriftkanten
- Legg til ikoner til slett/rediger/osv.?
- Kommentering

L. Kontrakten med oppdragsgiver

HØGSKOLEN I GJØVIK

PROSJEKTAVTALE

mellom Høgskolen i Gjøvik (HiG) (utdanningsinstitusjon), Electric Time Car AS (oppdragsgiver), og Else Dalby, Marte Selsjord Bjørseth og Trine Anita Grønvold (studenter).

Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studenten(e) skal gjennomføre prosjektet i perioden fra 11.01.2010 til 20.05.2010.

   Studentene skal i denne perioden følge en oppsatt fremdriftsplan der HiG yter veiledning. Oppdragsgiver yter avtalt prosjektbistand til fastsatte tider. Oppdragsgiver stiller til rådighet kunnskap og materiale som er nødvendig for å få gjennomført prosjektet. Det forutsettes at de gitte problemstillinger det arbeides med er aktuelle og på et nivå tilpasset studentenes faglige kunnskaper. Oppdragsgiver plikter på forespørsel fra HiG å gi en vurdering av prosjektet vederlagsfritt.

2. Kostnadene ved gjennomføringen av prosjektet dekkes på følgende måte:
   - Oppdragsgiver dekker selv gjennomføring av prosjektet når det gjelder f.eks. materiell, telefon/fax, reiser og nødvendig overnatting på steder langt fra HiG. Studentene dekker utgifter for trykking og ferdigstillelse av den skriftlige besvarelsen vedrørende prosjektet.
   - Eiendomsretten til eventuell prototyp tilfaller den som har betalt komponenter og materiell mv. som er brukt til prototypen. Dersom det er nødvendig med større og/eller spesielle investeringer for å få gjennomført prosjektet, må det gjøres en egen avtale mellom partene om eventuell kostnadsfordeling og eiendomsrett.

3. HiG står ikke som garantist for at det oppdragsgiver har bestilt fungerer etter hensikten, ei heller at prosjektet blir fullført. Prosjektet må anses som en eksamensrelatert oppgave som blir bedømt av faglærer/veileder og sensor. Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.

4. Den totale besvarelsen med tegninger, modeller og apparatur så vel som programlisting, kildekode, disketter, taper mv. som inngår som del av eller vedlegg til besvarelsen, gis det en kopi av til HiG, som vederlagsfritt kan benyttes til undervisnings- og forskningsformål. Besvarelsen, eller vedlegg til den, må ikke nyttes av HiG til andre formål, og ikke overlates til utenforstående uten etter avtale med de øvrige parter i denne avtalen. Dette gjelder også firmaer hvor ansatte ved HiG og/eller studenter har interesser.

   Besvarelser med karakter C eller bedre registreres og plasseres i skolens bibliotek. Det legges også ut en elektronisk prosjektbesvarelse uten vedlegg på bibliotekets del av skolens Internett-sider. Dette avhenger av at studentene skriver under på en egen avtale hvor de gir biblioteket tillatelse til at deres hovedprosjekt blir gjort tilgjengelig i papir og nettutgave (jfr. Lov om opphavsrett). Oppdragsgiver og veileder godtar slik offentliggjøring når de signerer denne prosjektavtalen, og må evt. gi skriftlig melding til studenter og dekan om de i løpet av prosjektet endrer syn på slik offentliggjøring.

5. Besvarelsens spesifikasjoner og resultat kan anvendes i oppdragsgivers egen virksomhet. Gjør studenten(e) i sin besvarelse, eller under arbeidet med den, en patentbar oppfinnelse, gjelder i forholdet mellom oppdragsgiver og student(er) bestemmelsene i Lov om retten til oppfinnelser av 17. april 1970, §§ 4-10.

6. Ut over den offentliggjøring som er nevnt i punkt 4 har studenten(e) ikke rett til å publisere sin besvarelse, det være seg helt eller delvis eller som del i annet arbeide, uten samtykke fra oppdragsgiver. Tilsvarende samtykke må foreligge i forholdet mellom student(er) og faglærer/veileder for det materialet som faglærer/veileder stiller til disposisjon.

7. Studenten(e) leverer 3 - tre - eksemplarer av oppgavebesvarelsen med vedlegg til Studenttorget. I tillegg leveres et eksemplar til oppdragsgiver. HiG kan stille til disposisjon ytterligere eksemplar(er) for oppdragsgiver mot at denne godtgjør produksjonskostnadene.

8. Denne avtalen utferdiges med et eksemplar til hver av partene. På vegne av HiG er det dekan som godkjenner avtalen.

9. I det enkelte tilfelle kan det inngås egen avtale mellom oppdragsgiver, student(er) og HiG som nærmere regulerer forhold vedrørende bl.a. eiendomsrett, videre bruk, konfidensialitet, kostnadsdekning og økonomisk utnyttelse av resultatene.

   Dersom oppdragsgiver og student(er) ønsker en videre eller ny avtale, skjer dette uten HiG som partner.

10. Eventuell uenighet vedrørende forståelse av denne avtale løses ved forhandlinger avtalepartene i mellom. Dersom det ikke oppnås enighet, er partene enige om at tvisten løses av voldgift, etter bestemmelsene i tvistemålsloven av 13.8.1915 nr. 6, kapittel 32.

11. Deltakende personer ved prosjektgjennomføringen:

HiGs veileder (navn): Tom Røise

Oppdragsgivers kontaktperson (navn): Dag L. Solhaug

Student(er) (signatur): Else Dalby dato 20/1-10

Trine A. Grønvold dato 20/1-10

Marte S. Bjørseth dato 20/1-10

_____ dato _____

Oppdragsgiver (signatur): Dag L Solhaug dato 20/01-10

Dekan (signatur): M Ly dato 25/01-10

Revidert 11.10.07, Ivar Moe

M. CD-ROMens innhold

Mappestruktur:

- Kode
  - Java
    - Fleet
      - IO
        - SQL
    - Route
      - IO
        - SQL
  - JSP
    - Fleet
    - Manage
    - Route
- Bacheloroppgaven uten vedlegg
- Bacheloroppgaven med vedlegg
- Møtereferater
  - ETC
  - Veileder