HØGSKOLEN I ØSTFOLD

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

BACHELOROPPGAVE

Prosjektkategori: Bachelorprosjekt	X	Fritt tilgjengelig
Omfgang i studiepoeng: 20		Fritt tilgjengelig etter
Fagområde: Informasjonsteknologi		Tilgjengelig etter avtale med oppdragsgiver

Tittel: Utvikling av reiseregnignsmodul	Dato: 3. mars 2015
Forfatterere: Robin Furu, Ingvild Bjørlo Karlsen, Christian Jacobsen, Glenn Bjørlo	Veileder: Terje Samuelsen
Avdeling / Program: Avdeling for Informasjonsteknologi	Gruppenummer: BO15-G02
Oppdragsgiver: Infotjenester AS	Kontaktperson hos oppdragsgiver: Petter Ekran

Ekstrakt:
HRessurs skal utvides med en mulighet for å beregne avstand og bompengekostnader under føring av reiseregninger, slik at brukeren ikke skal måtte ta i bruk eksterne verktøy for å gjøre dette. Denne må ta hensyn til statens krav for føring av reiseruter, som innebærer at kjørerutene skal beskrives detaljert, for senere kontroll/etterdokumentasjon av reiseregningene. Brukeren skal kunne angi type fremkomstmiddel, og en detaljert beskrivelse av kjørt rute, i form av start og endepunkt for ruta, og punkter underveis på strekningen. Basert på angitt informasjon, skal programmet beregne avstand og bompengekostnader for ruta. For at brukeren skal kunne kontrollere at programmet har beregnet riktig kjørerute, skal programmet skrive ut en kort veibeskrivelse av ruta.

3 emneord:

HRessurs
Reiseregning
Webutvikling

Innhold

Figurer

Tabeller

Kodeliste

Kapittel 1

Introduksjon

I dette kapitelet vil vi presentere gruppen, oppdragsgiver og selve oppdraget. Vi vil gå gjennom hva som skal gjennomføres, hvilke formål det er med prosjektet, og metoden vi vil bruke

1.1 Prosjektgruppen

Ingvild Karlsen Bjørlo
90204584 - Invgilkb@hiof.no

Robin Furu
41524376 - robinaf@hiof.no

Christian Jacbosen
41758575 - chrisjac@hiof.no

Glenn Bjørlo
47384852 - glennab@hiof.no

Prosjektets Hjemmeside
http://frigg.hiof.no/bo15-g2/

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

Ingvild bor i Halden, hvor hun også er oppvokst. Hun gikk studiespesialisering på Halden videregående, med et utvekslingsår i Tyskland, og jobbet i et år før hun søkte seg til dataingeniørstudiet. Hun har tidligere jobbet 2,5 år på Svinesund Infosenter, som i tillegg til turistinformasjon, fungerer som servicestasjon for bompengeselskapet Svinesundsforbindelsen. De faglige interessene retter seg mot programmering og dokumentasjon.

Robin er oppvokst i Fredrikstad, Gikk Idrettslinjen på videregående, hvor det så bar videre til et års tjeneste i Hans Majestet Kongens Garde. Rett ut fra førstegangstjenesten bar det videre til Tre-semester Dataingeniørkurs hvor det først var en sommer med matte og fysikk. Han har gjennom studiene blitt veldig interessert i HMI/automatisering – Industriell IT, en mer praktisk tilnærming til IT.

Christian er født og oppvokst i Fredrikstad og bor på Gressvik. Han har tidligere erfaring som teknikker i telekomfaget hvor han jobbet i YIT Building systems i 2 år og har fagbrev i telekommunikasjon. Ved siden av studiene jobber han på Expert, og har foto som en stor interesse.

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

1.2 Oppdragsgiver

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

De har en ledende posisjon når det gjelder kurs og nettbaserte fag- og kompetanseverktøy som skal gjøre det enklest mulig for bedrifters ledelse og arbeidere. Deres kompetanseverktøy er egenutviklet og de veileder kundene med dette i tillegg til å tilby konsulenttjenester. Sammensetningen av kompetanse hos de ansatte innen jus, HR og systemutvikling gjør at de kan utvikle gode og praktiske systemer i henhold til regelverk og brukervennlighet. De har et stort nettbasert oppslagsverk med fagsupport, med mer enn 40.000 brukere. Flere hundre tusen ledere og medarbeidere har tilgang til deres nettbaserte håndbøker og systemer. De tilbyr i tillegg rådgiving utover oppslagsverkene, og deres rådgivere er alle enten jurister eller fagspesialister med lang erfaring innen både offentlig og privat sektor. De svarer på mer enn 50 000 spørsmål hvert år.

Et av deres mest populære produkter er det egenutviklede personalsystemet HRessurs, der man kan behandle alt fra personaladministrering, sykefravær, ferie og reiseregninger. Dette er et nettbasert system, slik at kundene abonnerer på tilgang til nettløsningen og ikke trenger å kjøpe og installere programvare. Dette fører til at systemet alltid er oppdatert, uten at kundene må tenke på å laste ned oppdateringer. Kundene kan velge mellom tre ulike pakker ettersom hva bedriften trenger: HRessurs Sykefraværsoppfølging, HRessurs personal- og ledersystem og HRessurs Reise og Utlegg. HRessurs Sykefraværsoppfølging hjelper brukeren gjennom alle krav og oppgaver knyttet til sykefravær. Det sørger for at all dokumentering av fravær skjer på samme måte, uavhengig av hvilken leder som har ansvaret for føringen av fraværet. Systemet inneholder fraværsstatistikker, påminnelser om frister, maler til relevante dokumenter som kan måtte fylles ut, og en trinn for trinn beskrivelse av prosessen utviklet av eksperter.

Det legges vekt på at god oppfølging bidrar til at syke medarbeidere kommer tilbake i jobb tidligere enn uten oppfølging. HRessurs personal og ledersystem sikrer at personal-oppfølgingen skjer på best mulig måte, og gir både ledere og HR trygghet rundt oppfølging av ansatte. Dette gjør at istedenfor å bruke mye tid på personaladministrering, kan HR bruke mer tid på langsiktig og strategisk arbeid. Systemet gir ledere en steg for steg-veiledning gjennom prosesser der det e viktig at ting gjøres i riktig rekkefølge og etter gjeldene regler. Dette innebærer veiledning om hvordan følge opp nyansatte, hvordan foreta gode medarbeidersamtaler for å bidra til god trivsel på arbeidsplassen, og god hjelp til oppsigelse og alle regler knyttet til ulike grunner til dette. I denne oppgaven er det HRessurs Reise og Utlegg, som behandler reiser og alle utgifter knyttet til dette, vi vil bruke. Regler knyttet til reiseregninger er alltid oppdatert, og de tilbyr valutakalkulator og kilometergodtgjørelse for flere typer fremkomstmidler. **Mer om Reise og Utlegg i kapittel xxxxx – oppdraget.**

Utover HRessurs tilbyr Infotjenester i tillegg avviksoppfølgning, et system for å registrere avvik, kategorisere disse og fordele hvem som skal behandle disse. Den som får ansvar for å behandle avviket, må vurdere tiltak og hva som må gjøres for å avslutte saken, samt en frist dette må oppnås innen. Dersom fristen ikke ikke holdes, vil et varslingssystem over e-post settes i gang. Raportene fra dette avvikssystemet vil gi en god oversikt over hvilke kategorier avvikene fordeler seg i, samt hvilke avdelinger disse har oppstått. Dette vil bidra til å kunne sette opp både kortsiktige og langsiktige planer for å unngå framtidige avvik.

1.3 Oppdraget

Infotjenester lover på sine nettsider et *≪brukervennlig og intuitivt grensesnitt som gjør den ansatte i stand til å registrere/behandle reiser og utlegg uten opplæring. Brukeren trenger ingen kunnskaper om regler og satser≫*. Infotjenester har derimot fått tilbakemeldinger på at modulen ikke er brukervennlig nok, blant annet med hensyn til bompenger. Vår utfordring er å utbedre en av manglene som er årsaken til misnøyen blant enkelt kunder.

HRessurs skal utvides med en mulighet for å beregne avstand og bompengekostnader under føring av reiseregninger, slik at brukeren ikke skal måtte ta i bruk eksterne verktøy for å gjøre dette. Denne må ta hensyn til statens (?) krav for føring av reiseruter, som innebærer at kjørerutene skal beskrives detaljert, for senere kontroll/etterdokumentasjon av reiseregningene. Brukeren skal kunne angi type fremkomstmiddel, og en detaljert beskrivelse av kjørt rute, i form av start og endepunkt for ruta, og punkter underveis på strekningen. Basert på angitt informasjon, skal programmet beregne avstand og bompengekostnader for ruta. For at brukeren skal kunne kontrollere at programmet har beregnet riktig kjørerute, skal programmet skrive ut en kort veibeskrivelse av ruta.

Ettersom dette er en tilleggsmodul til en eksisterende løsning, er det med tanke på fremtidig drift og vedlikehold, en stor fordel at løsningen er bygd opp med samme struktur som det eksisterende. Det at HRessurs er en løsning firmaene betaler for tilgang til, setter også en del ekstra krav til standarden det må følge. HRessurs forutsetter ikke at brukeren har oversikt over til en hver tid gjeldene regler for reiseregninger, så dette må også tas hensyn til. Hvordan dette skal løses, og hvilke krav løsningen må følge, vil vi komme nærmere til i analysedelen i kapittel 2.1.

1.4 Formål, Leveranser og Metode

Videreutvikling av forprosjektrapporten.

1.4.1 Metode

For å ferdigstille produktet vårt samt å få erfaring fra arbeidslivet, har vi benyttet en arbeidsmetode som heter SCRUM. Denne er videre forklart her (. . . . .), Vi har hatt arbeidsperioder på 2 uker hvor vi hver andre onsdag hadde et møte med arbeidsgiver for å kartlegge hvilke arbeidsoppgaver som skulle prioriteres videre, disse arbeidsperiodene kalles Sprinter. De arbeidsoppgavene vi ikke fikk fullført i perioden eller ikke ble prioritert ble lagt i en backlog. Det som er fint med dette er at det er sjeldent alle parter har samme syn på ønsket resultat og når man jobber med kontinuerlig evaluering og prioritering så blir dette mest mulig samkjørt til et tilfredsstillende produkt. Når vi har hatt tidligere versjoner av produktet har vi benyttet et diverse utvalg av personer til å teste funksjonaliteten og komme med tilbakemeldinger på hva som kunne vært gjort annerledes, dette for å få et annet perspektiv på ting enn å kun være låst til et programmerings synspunkt. For samkjøring av dokumenter og for å gi arbeidsgiver et innblikk i hva vi har jobbet med har vi benyttet Github til alt materiale som er under utvikling, vi har også benyttet Issues under Github til å kartlegge Sprintene våre og for å ha backlog oversikt. Dette også for å ha en best mulig dialog med arbeidsgiver. Til programmeringen har vi benyttet Visual Studio 2013.

1.5 Rapportstruktur

En kort oppsummering av hva hvert av kapitilene omhandler, skrives når Rapportstruktren er ferdiglagd.

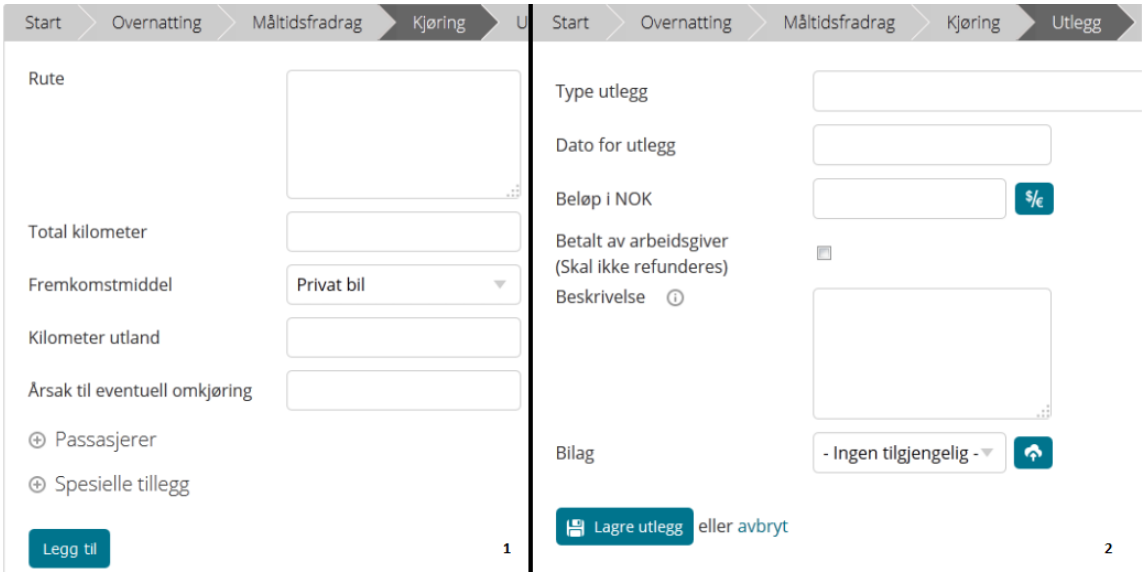
Kapittel 2

Analyse

2.1 Slik Fungerer HRessurs

Brukeren får tilgang til reise-modulen ved å logge seg inn på sin konto og deretter velge ≪mine reiser≫. Her får brukeren valget mellom å se på allerede registrerte reiser, eller legge til en ny. Registreringen foregår trinn for trinn, og det er enkelt for brukeren å holde oversikten over hvor han er. Det er både en meny som forteller hvor brukeren befinner seg i registreringsprosessen, samt en frem- og tilbakeknapp nederst på skjermen som viser antall trinn. Knappene i menyen øverst på skjermen er formet som piler som tydelig viser hvor man er samt at det gir inntrykk av at man befinner seg i en prosess som foregår stegvis. Det er ikke mulig å navigere seg frem før man har fylt ut nødvendige felt. Antall trinn avhenger av om man har benyttet eget kjøretøy, om man skal ha diettgodtgjørelse eller om man har vært i utlandet. Hukes det av for noen av disse dukker det opp et nytt trinn hvor man må fylle inn informasjon for dette valget. Hver side i applikasjonen er minimalistisk designmessig. Det er hvit bakgrunn og samme teksttype både i hoveddelen, menyen og knappene. Bakgrunnsfargen i menyen er grå, og blå i knappene på siden. Dette er det eneste som skiller ulike komponenter og funksjoner fra hverandre.

På første side må brukeren navngi reisen, fylle inn dato og klokkeslett for avreise og hjemkomst, hva som er formålet med reisen og reisested. Når man navngir hver enkelt reise er det også enklere og holde oversikten for brukeren når antall reiser registrert begynner å bli mange. Har brukeren huket av for diettgodtgjørelse dukker det opp to nye trinn hvor man skal legge inn overnattingssted, ankomst og avreise. På neste trinn kan brukeren legge til måltidsfradrag.



Hvis det har vært huket av for eget kjøretøy blir neste trinn registrering av reiserute. Ruten må oppgis så nøyaktig som mulig i et eget tekstfelt. For eksempel: Sarpsborg – Oslo via E6 til Tusenfryd, E18 til Oslo. I neste tekstfelt må brukeren oppgi nøyaktige antall kilometer. Brukeren kan ikke velge reiserute dynamisk, for eksempel ved å fylle ut fra- og til-felt som autofullføres og hvor avstanden regnes ut automatisk. Han kan heller ikke velge eventuelle via-punkter som legges inn automatisk. Utfra et brukerperspektiv vil dette kanskje virke altfor gammeldags og tungvint, samt gi inntrykk av lite automatisering. Neste steg blir å velge type kjøretøy fra en nedtrekksmeny. Deretter kan man eventuelt fylle ut informasjon om kjøring utland, årsak til omkjøring, medpassasjerer og andre spesielle tillegg. På tredje trinn kan brukeren fylle ut utlegg. Man må beskrive type utlegg, dato for utlegget og beløpet. Brukeren kan også laste opp eventuelle bilag. På nest siste trinn kan brukeren fylle inn ekstra informasjon, som om han har mottatt forskudd, og kostnadsbærere. På siste side får brukeren opp en detaljert oversikt over all informasjon som har vært fylt ut. Øverst på siden ligger en kort oversikt over registrert informasjon som startdato og varighet, eventuelle utlegg i kroner, og hvor mye som skal utbetales. Brukeren kan også se nåværende status på registreringen. Altså om den venter på innsendelse eller om den er godkjent/ ikke godkjent. Lenger ned på siden vises en digital versjon av selve reiseregningen med all nødvendig informasjon. Brukeren kan nå velge å signere og sende inn, eller gå tilbake og gjøre oppdateringer. **Litt mer om knapper/rekkefølge på inntasting av info osv, (til å begrunne valg av brukergrensesnitt senere).**

2.2 Universiell utforming

1.juli 2014 ble nye regler for universell utforming av IKT-løsninger, ≪Retningslinjer for tilgjengelig webinnhold (WCAG) 2.0≫. Regelen innebærer at alle nye IKT-løsninger i Norge, både innen privat og offentlig sektor, organisasjoner og lag, skal være universelt utformet. Kort sagt er dette en lovpålagt standard for utforming av IKT-løsninger. Eksisterende IKT-løsninger har frist til 21.januar 2021 på å rette seg etter kravene. Kravet er at nettløsninger må oppfylle 35 av de 61 kriteriene definert i WCAG 2.0 for å være godkjent. Kriteriene er delt inn i tre nivåer A, AA og AAA, der alle kravene i nivå A og AA (unntatt 1.2.3, 1.2.4 og 1.2.5) må være oppfylt.

Kravene innebærer at IKT-løsninger skal være tilgjengelige for alle, det tas hensyn både med tanke på brukervennlighet slik at alle skal ha forutsetninger for å bruke løsningen, også de som har nedsatt funksjonsevne i form av dårlig syn og hørsel, fargeblindhet osv. Kriteriene er delt inn etter 4 prinsipper. Prinsipp 1 – mulig å oppfatte, setter krav til bruken av kontraster, fargevalg, forstørrelsesmuligheter og bruk av tekst som alternativ til bilder, lyd og video. Prinsipp 2 – Mulig å betjene, setter krav til brukervennlighet i form av at brukeren skal fullt ut kunne bruke løsningen uavhengig av teknisk utstyr, blant annet slik at en løsning tenkt brukt med mus og tastatur, også skal være mulig å betjene kun via tastatur. Prinsipp 3 – Forståelig, innebærer at løsningene skal være forutsigbare, det skal være lett å forstå hvordan de skal brukes og informasjonen man finner skal være enkel å forstå. Krav til bruk av både enkelt språk og hjelpetekster bidrar til dette. Det siste prinsippet, prinsipp 4 – Robusthet, er rettet mer mot det tekniske. Her settes det noen retningslinjer i form av koding og tilgjengelighet. Innholdet må kunne tolkes på en god måte av forskjellig teknologi, nettsider må valideres, og koden må være riktig. I HTML, som vil bruke, blir som regel dette ivaretatt, men det er spesielt viktig å sikre god tilgjengelighet dersom man utvikler egne elemeter (widgets).

Vi har studert retningslinjene, for å vite hvilke hensyn vi må ta under utformingen av prosjektet. Vi har utelukket kriteriene som går på lyd og video, mer??, da disse ikke vil være aktuelle for vår del. En spesifisert liste over hva vi har utelukket, finnes i slutten av dette delkapittelet.

Ettersom dette er en løsning der brukeren skal angi informasjon, er alle underpunktene ≪Retningslinje 3.3 – Inndatahjelp≫ spesielt viktige. 3.3.1 – Identifikasjon av feil, og 3.3.2 – Forslag ved feil er viktige. Vi må her sørge for at dersom brukeren taster inn en ugyldig adresse, må det bli gitt tydelig tilbakemelding om dette. Her vil vi også bruke en autofullførfunskjon for å foreslå adresser under intasting, for å forhindre feil. 3.3.2 – Ledetekster eller instruksjoner innebærer at alle inndatafelt skal ha ledetekst eller instruksjoner som angir hvilke data som skal angis hvor. Vi vil her tydeliggjøre hvor start, stopp og viapunkter skal angis, at kjøretøy må velges. I denne sammenhengen vil vi også nevne viktigheten av

≪1.3.2 – Meningsfylt rekkefølge≫ med tanke på viapunktene som skal angis. Rekkefølgen viapunktene angis i, har betydning for hvilken kjørerute som beregnes. Dette skal vi gjøre ved å poengtere viktigheten av rekkefølgen i form av utfyllingsinstruksjonene , samt gi brukerne mulighet til å sortere listen ved å kunne flytte på tekstboksene til ønsket rekkefølge er oppnådd. Denne sorteringsmuligheten vil vi også tekstlig spesifisere. **mer om viktige kriterier** Disse rettninglinjene vil vi ikke gå nærmere inn på, da de ikke omfatter prosjektet vårt:

2.3 Autopass, rabatter

I Norge øker antall bomstasjoner stadig. Blant annet langs E6 mellom Gardermoen og Hamar har det på få år kommet fem bomstasjoner, og en sjette åpnes i juni 2015. Senest 3.februar 2015 var det nevnt i Halden Arbeiderblad muligheter for bompenger i Halden i forbindelse med bygging av to tuneller i fjellet under Fredriksten Festning. ≪– En bompengefinansiering gjør at man kan komme fortere i gang med den type prosjekter. Halden bør ta en runde rundt bompengefinansiering, mener han. Ordfører Thor Edquist tror ikke vi kommer utenom bompenger≫.

Fra og med 1.januar er det obligatorisk for biler med tillat totalvekt over 3,5t å ha autopass i Norge. Regelverket gjelder alle biler som tilhører ≪foretak, stat, fylkeskommune eller kommune, eller som på annen måte hovedsakelig benyttes i næringsvirksomhet≫. Straffen for å ikke følge disse reglene er bot på 8000kr, og denne satsen dobles dersom pålegget ikke etterkommes, og kjøretøyet stoppes uten brikke igjen innen 2år fra første straff. Til tross for påbudet om autopassbrikke i kjøretøy over 3,5t, vil vi når bompengekostnadene beregnes, ikke ta hensyn til eventuelle autopassavtaler de ansatte måtte ha. Dette er med bakgrunn i at priser, rabatter og betalingspraksis varierer veldig over hele Norge, slik vi i påfølgende avsnitt vil gå nærmere inn på.

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

Østfold Bompengeselskap, som drifter seks bomstasjoner i Østfold, tilbyr forskuddsbetaling. Avhengig av beløpet som forskuddsbetales, vil størrelsen på rabatten for passeringer ved bomstasjonene de drifter, variere. Ved påfyll av 805kr vil rabatten være 30%, 3450kr vil gi 40% rabatt og 5750kr vil gi 50% rabatt. De tilbyr kun rabatt til sine egne abonnenter, og alle andre passerende med autopass fra andre selskaper, vil måtte betale full pris. Et annet selskap her i Østfold, som har en helt annen praksis, er Svinesundsforbindelsen. De tilbyr kun etterskuddsbetaling, og gir 13 rabatt per passering for alle som har autopass. Tilsvarende rabatter for alle med autopass finnes blant annet ved Bomringen i Oslo, Kråkerøyforbindelsen, Bomringen i Kristiansand og flere andre selskaper. Nord-Jæren Bompengeselskap er et selskap som tilbyr både forskudd og etterskuddsbetaling for sine abonnenter, men her fordeler rabatten seg på 10% for privatkunder med etterskuddsbetaling, og 20% for alle med forskuddsbetaling, samt firmakunder med etterskuddsbetaling.

Det er ikke bare ulik praksis med autopassavtale som skiller bompengeselskapene. I Trondheim praktiseres rushtidspriser, som vil si at i tidsrommet 07-09 og 15-17 dobles taksten for å passere bomstasjonene i Trondheim sentrum. Disse tilhører prosjektet «Miljøpakke Trondheim». Resterende bomstasjoner i Trondheimområdet følger ikke rushtidpraksisen. Miljøpakken er et prosjekt som ble startet i 2009 for å begrense miljøproblemene i sammenheng med at Trondheimsområdet er et av områdene i Norge med størst befolkningsvekst. Det gjøres en rekke tiltak for å blant annet redusere klimautslipp ved hjelp av kortere bilkøer og mindre trafikkstøy, derav å prøve å redusere rushtidskøer. I Trondheim finnes det også tidsgrenser for om man skal betale for passeringene eller ikke. Bomstasjonene er grupperte i seks grupper, det Miljøpakken omtaler som «snitt». Hvert av snittene inneholder mellom en og ni bomstasjoner, og innenfor et tidsrom på en time, betaler man kun for en passering i hvert snitt. Denne praksisen finner man blant annet igjen i Haugalandspakken, som dekker bomstasjoner i Haugesund og på Karmøy, der man også kun betaler en passeringer innenfor en time.

Med så mange forskjellige rabattordninger og tidstyrte priser vil det være tilnærmet umulig å ta hensyn til hver enkelt bruker av HRessurs og en eventuell autopassavtale han eller hun måtte ha. Etterskuddsfakturering og utsendelse av oversikt over passeringer for de med forskuddsbetaling, skjer gjerne en tid etter passeringen fant sted. Hyppigheten for utsendelse varierer fra bompengeselskap til bompengeselskap, samt hvilke avtaler man har. Noen selskaper praktiserer månedlig etterskuddsfakturering, og andre sender en oversikt over passeringer når forhåndsbetalt beløp er brukt opp. Passeringene kan også være vanskelig å holde oversikt over, da disse ofte har lokale navn, som ikke sier den reisende så mye uten å være lokalkjent. Bedrifter har også ofte en månedlig frist for å registrere reiseregninger for måneden som har vært. Dette tilsier at man i de fleste tilfeller ikke har tid til å vente på faktura/oversikt over passeringer, og at det istedenfor må foretas en generell prisberegning ut ifra vanlige takster, uavhengig autopassavtaler.

2.4 Valg av ruteplanleggingstjeneste

Vi har undersøkt om det er mulig å benytte andre api'er enn vegvesenets ruteplantjeneste, som vi fikk presentert av Infotjenester. Finnes det andre systemer som inneholder de ulike funksjonalitetene vi ser etter? Vi tok for oss andre nettbaserte tjenester til å beregne kjøreruter, for å se hvilken tilleggsinformasjon de kunne tilby utover å beregne kjøreruta. Vi testet NAF Ruteplanlegger og veibeskrivelsetjenesten hos 1881, Google Maps, Gule sider og Kvasir. Felles for alle er at de oppgir kjørelengde for strekningen, samt at de har en mulighet til å legge til via-punkter ved å angi stedsnavn. Google Maps og 1881 tilbyr i tillegg en grafisk løsning for å kunne dra kjøreruten for å endre veivalg. Bompengekostnader er det færre som viser. Kvasir og Gule Sider benytter samme tjeneste, og inkluderer ingen informasjon om bomstasjoner langs ruta. NAF og Google oppgir at det er bomvei på del av strekningen, uten å spesifisere nøyaktig hvor på strekningen denne befinner seg, ei heller prisen. 1881 oppgir prisen for vanlig bil og lastebil, i tillegg navn på bomstasjon og hvor den ligger (grafisk og med koordinater).

For å finne ut om 1881 eller Ruteplantjenesten egnet seg best, tok vi en nærmere sjekk på hvilken informasjon de inneholdt. Vi vet at det finnes flere forskjellige rabattordninger og tilleggspriser rundt omkring i Norge, slik som rushtidspriser, rabatt for gitte antall passeringer innen et tidsrom osv. For å finne ut om noen av løsningene inneholdt informasjon om disse spesialprisene har vi testet ruteberegning gjennom de bomstasjonene vi på forhånd vet har slike ordninger. Haugesund har xxxxxx og ved noen bomstasjoner i Trondheim sentrum har de doble satser i rushtiden. Mengderabatter med tidsbegrensing ble ikke tatt hensyn til på hverken 1881 eller ruteplantjenesten, men 1881 viser informasjon om rushtidspriser. Ved beregning av kostnadene, benyttes vanlig takst uten rushtidtillegg. I veibeskrivelsen, står det beskrevet at det mellom 07:00 og 09:00, samt 15:00-17:00 er dobbel takst.

1881 tilbyr en bedre kartløsning som er mer brukervennlig. Kartløsningen hos ruteplantjenesten er uoversiktlig, og kan virke gammeldags. I ruteplantjenesten må adresser angis korrekt. En tastefeil midt i adressen vil ikke gi resultater. Ved å utelate de siste bokstavene i et stedsnavn, vil den i noen tilfeller gjette riktig, men dersom flere steder begynner med samme navn, vil den velge tilfeldig. Et søk på ≪Storgata≫ blir ≪Storgata, Moss≫. Brukeren har ingen mulighet til å påvirke dette, utenom å rette opp i ettertid når han eller hun ser at byen er feil. 1881 tilbyr autofullførfunksjon, så et søk på ≪Storgata≫ gir ut en liste over de forskjellige ≪Storgata≫-er som finnes. Man vil i tillegg raskt oppdage en eventuell skrivefeil i adressen, da adressen man ønsker å skrive inn, ikke lenger vil dukke opp blant forslagene.

1881 sitt system for veibeskrivelser innehar alle nødvendige data vi trenger. Vi konkluderte derfor med at 1881 egner seg best, med tanke på at det inneholder litt mer detaljinformasjon om noen av bomstasjonene, det er mer brukervennlig å søke, kartløsningen er bedre, og det er mulighet for å grafisk endre kjøreruta, dersom man ser at ruta ikke er den man har kjørt. Det som skiller 1881 negativt fra ruteplantjenesten er at kjøreruter vil bli beregnet gjennom Sverige dersom dette er korteste vei. Dette er uhensiktsmessig da dette vil komplisere beregning av priser, da det etter xxxxx-regler er andre takster for utenlandskjøring. 1881 opplyste (kilde) at de hadde et API, som etter en gratis testperiode, ville koste penger. Ved nærmere undersøkelse viste det seg at dette kun var et API for person- og bedriftssøk, ikke veibeskrivelser, og uten noe kartbasert innhold, slik at denne ikke ville dekke de behovene prosjektet har. Ved nærmere undersøkelser av api-et hos ruteplantjenesten, viste det seg at de har tilleggsinformasjon om navn på bomstasjoner,som kan

fås frem ved å legge til ”&Navn”i URL’en som blir sendt inn til Ruteplantjenesten sitt API. JSON filen vi får returnert vil inneholde ett objekt som heter nvdb:Bomstasjon, med detaljinformasjon om bomstasjoner. Hvis det er 5 bomstasjoner på den angitte ruten vil JSON filen som returneres inneholde 5 bomstasjonobjekter, hvert enkelt objet inneholder navn, takst litenbil, takst storbil. Så for å kunne liste ut alle bomstasjonene er det bare å søke i JSON objektet etter navnet nnvdb:Bomstasjonså får du listet alle bomstasjoner på ruten samt passeringskostnader ved disse.

For å oppfylle alle ønskene/kravene til prosjektet har vi har vært avhengig av å bruke Google API til flere deler av vår webapplikasjon. Enn så lenge vi vil benytte Googles autocompletefunksjon, for å gjøre adressesøk mer brukervennlig. Google sin autocompleete er lagt til i tekstfeltene der adresser skal angis, fungerer slik at brukeren får opp forslag til start og stopp og vialokasjoner når man fyller inn tekstfeltene. Deretter vil Google hente koordinatene til disse stedene. I hovedsak vil resten av applikasjonen basere seg på APIet til ruteplantjenesten.

Ruteplantjenesten foretar søk baser på koordinater, og benytter den europeiske standarden ERST89 som måles i meter. Google derimot er basert på den amerikanske WGS84 som bruker grader. Man må også ta hensyn til ≪sone≫, for eksempel regnes norsk sone som 33. Vi trengte derfor en konverter som konverterer mellom koordinatene WGS84 og ERST89. Denne konverteren får inn koordinatene fra Google, konverter dem til europeisk standard, og sender de nye dataene til Ruteplantjenesten i en JSON-string.

Det hadde selvsagt vært enklere og mer oversiktlig å kun benytte seg av et api. Dessverre finnes det ingen system tilgjengelig som både kan tilby autofullført søk, gi god informasjon om avstander, detaljinformasjon bomstasjoner (med priser) og kjøreretning.

Vi har hatt et ønske om å vise ruten på kart, slik at brukeren kan kontrollere at den beregnede ruta stemmer. Vi har forsøkt å kombinere kjøreruteberegning hos Ruteplantjenesten, og vise denne på Google-kart. Et problem som dukker opp når vi kombinerer disse, er at kjøreretning på kartet kan vises forskjellig fra system til system. Det vil si at informasjonen om avstand og bompenger vi får fra Visveg vil stemme med det brukeren skriver inn, men ikke nødvendigvis med kjøreretningen som vises på Google-kartet, selv om begge er satt til å beregne korteste avstand. Google beregner korteste vei uavhengig av landegrenser, mens Ruteplantjenesten beregner korteste vei i Norge. For eksempel kan en bruker ha kjørt fra Oslo til Kirkenes gjennom Norge, via E6, mens Google vil vise korteste distanse, altså gjennom Sverige og Finland. Google tilbyr ingen mulighet for restriksjoner i forhold til landegrenser. På den andre siden støtter ikke Ruteplantjenesten kjøreretninger utenom Norge. Google støtter heller ikke andre karttjenester kombinert med deres autocomplete-funksjon.
Vi har diskutert om vi virkelig har behov for en karttjeneste i vår app. Vi skal ikke levere et system som støtter veibeskrivelser, men et system som skal behandle reiser etter at de har funnet sted. Kartet er ment som et hjelpemiddel for å bekrefte reiseruten som ble valgt da brukeren gjennomførte sin reise. Kan denne informasjonen vises på andre måter? I stedet for å hente informasjon om valgt reiserute fra Google, kan vi heller vise den i tekstformat fra Ruteplantjenesten. For eksempel: Bruker A kjørte fra Oslo til Trondheim, E6 til Hamar, om Østerdalen, og videre på E6 frem til destinasjonen. Jeg vil argumentere for at denne informasjonen best vises punktvis i tekstformat fremfor en stiplet linje på et kart.

2.5 Hvorfor .net?

Vi har valgt å utvikle vår applikasjon innenfor Microsoft sitt .Net rammeverk. Vi har undersøkt mulighetene for å benytte oss av andre teknologier, språk og rammeverk som kanskje ville ha ført til at vi satt igjen med en mer åpen og frittstående applikasjon, men da måtte oppdragsgiver i så fall tilpasse den til sitt allerede eksisterende system. Vårt ønske er at oppdragsgiver skal kunne fase inn vår applikasjon inn i personalsystemet uten for mange tilpasninger. Infotjenester benytter i dag .Net i hele sin virksomhet. Personalsystemet HRessurs er som nevnt delt inn i flere kategorier, men felles for hele systemet er teknologien som ligger bak. HRessurs som webapplikasjon er laget i ASP .Net. Siden vårt system på sikt skal erstatte dagens ordning for registrering av reise og utlegg er det naturlig at vi velger samme plattform som HRessurs allerede er basert på.

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

2.6 ASP.net vs ren HTML

Vi har fra uke 7 begynt å flytte noe av logikken over på server. Det er for at systemet skal være mer vedlikeholdbart. For eksempel i forhold til gjenbruk slik at man slipper å lage samme komponent flere ganger. Men også for å unngå ≪cross-domain≫, altså at brukerens nettleser som ligger på vår server oppretter kontakt med en annen server for å innhente informasjon. I stedet skal vår server innhente informasjonen (feks fra Ruteplantjenesten), og så vise dataene til brukeren i nettleseren. Brukeren skal kun være på vår server hele tiden. Andre fordeler er at man løser ≪interoperability problems≫, altså at vi muliggjør at forskjellige systemer på forskjellige plattformer kan kommunisere med hverandre. En mulighet vi nå jobber med for å løse dette er å sette opp en web-service som oppretter kontakt med vegvesenets server. Det vil si at nettleseren sender en ajax-request til vår web-service som så sender en forespørsel til Ruteplantjenesten. Når denne svarer får webservicen et svar tilbake (i vårt tilfelle JSON), som så sender denne tilbake til frontend Javascript som viser dataene til brukeren.

Vi har både sett på hvordan man kan gjøre dette i Microsoft sitt ASP .Net rammeverk som gjør jobben med å snakke med en web-service mindre krevende, men også hvordan dette kan la seg gjøre fra en ren htmlside. Microsoft sitt ASP .Net er ment å gjøre jobben med å lage en dynamisk webside enklere. Det vil si at det å vedlikeholde websiden (innholdet) skal være mer overkommelig. For eksempel hvis man vil inkludere en ny side (web forms/aspx) eller redigere innholdet i en meny. Men samtidig vil arbeidet (i teorien) med å ≪snakke≫ med en web-service vært enklere via web forms enn i ren html. Microsoft introduserte muligheten for at backendkoden ligger i aspx.cs-filer (.cs hvis man bruker C#), mens statisk kode ligger i aspx-filene. Det vil si at all kode som skal behandle funksjoner basert på brukervalg på siden behandles av filene med aspx.cs-endelse. Alle behandlede og kompilerte filer i ASP .Net lagres i ulike kataloger som er tilgjengelig for alle sider på nettstedet. Det inkluderer også web-services. Alle filer (inkludert WSDL- filer som bestemmer hvordan kommunikasjonen skal foregå) som refererer til en web service ligger i en egen katalog slik at denne kan nås enkelt i koden fra alle sider, altså web forms som Microsoft kaller det.

Men Infotjenester er i ferd med å lansere HRessurs i ny drakt. Det innebærer ikke bare rent utseendemessig, men også bak sløret. Tilbakemeldinger fra flere av utviklerne ved bedriften er at aspx-systemet som ligger bak dagens HRessurs begynner å bli uoversiktlig og vanskelig å vedlikeholde. Mest sannsynlig vil neste generasjon HRessurs bestå av rene HTML-filer frontend, mens selve funksjonaliteten fortsatt befinne seg i .Net baserte web-services. Vårt dilemma blir da om vi skal fortsette å utvikle vår reiseapp i det dynamiske aspx-biblioteket som teknisk sett skal gjøre jobben med denne typen webapp mer overkommelig, samt at dagens ordning er laget i aspx, eller om vi skal utvikle for fremtidens HRessurs og dermed tilpasse oss Infotjenesters behov i større grad.

Her må det komme mer om hvorfor ren HTML er bedre (også for oss) enn aspx. Alle kilder tilsier noe annet. Venter på mer info fra Petter om dette.

Kapittel 3

Design/utforming/planlegging

3.1 Struktur/innhold/utforming

HER KOMMER DET MER / IKKE FULLFØRT KAP

Hvordan vi vil dele opp prosjektet (Eks HTML/JS/CSS – WEB SERVICE (C#) – INTERFACE – API. . . . Bør vel også være litt om hvorfor vi deler opp (vedlikehold, med tanke på utbytting av API. . .).

At vi ikke skal bruke asp.net og hvorfor?

3.2 Leveranse og Mal

Beskrivelse av hva de forskjellige leveransene skal inneholde.

HER KOMMER DET MER / IKKE FULLFØRT

Kapittel 4

Implementasjon/produksjon/gjennomføring

4.1 Alt i ett løsning

Vårt første versjon baserte seg kun på HTML og javascript. Det bruker også 2 sepparerte API systemer. for den grafiske biten samt kordinater bruker vi Google Map API, og for all nyttig kjøredata håndteres det av Ruteplan tjenesten som er eid av Statens Vegvesen, vi tok kontakt med vegvesenet og fikk opprettet en access til dere lukket API og har fri tilgang til det iløpet av hele Bachelor perioden.

Om løsningen er varig i etterkant med tanke på om vegvesenet lar private bedrifter bruke løsningen dere er ikke undersøkt. men vi jobber med saken. Når man taster in start stop addresse har vi brukt google sitt API til og auto fullføre de forskjellige søkene man sender inn i form av adresser. Når auto fullføring av søk er ferdig og man trykker beregn vil en funksjon kjøre som konverterer adressen til søket om til kordinater. Deretter konverteres kordinatene om til ett format som Ruteplantjenesten forstår, I det kordinatene er ferdig konvertert blir de byggdt inn i en URL string som vi sender til Ruteplan sitt REST API, hvor vi får returnert et JSON objekt med forskjellige type informasjon.

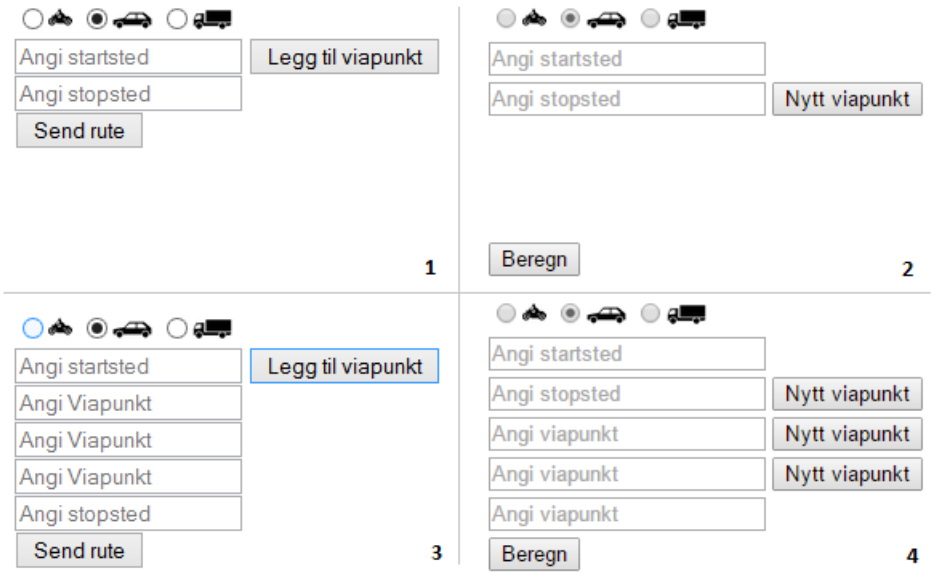
I det JSON objektet henter vi så ut total kjørelenge, beregnet tid samt navn på bomstasjoner og takstert for passeringer. Det blir så igjen returnert til HTML siden som skiserer ruten grafisk på ett kart og returnerer de spesifikke dataene. **Må utdypes**

4.1.1 Utforming av viapunktliste

I vårt brukergrensesnitt, har vi muligheter til å angi kjøretøy, start og slutt for reisen, samt viapunkter underveis i reisen. For å angi start, slutt og viapunkter bruker vi HTML-tekstfelt, og for å angi de aktuelle adressene, bruker vi google autocomplete som et ekstra hjelpemiddel. Vi bruker radiobuttons for å angi kjøretøy, og vi har valgt personbil som defaultverdi, da dette er mest vanlig (omformuler). For å angi viapunkter begynte vi med flere skjulte tekstfelt, som vi endret til synlig, når brukeren trykket knappen ≪Nytt viapunkt≫. Dette er en lite hensiktsmessig løsning, da tekstfeltene opptar plassen de er på, selv om de ikke er synlige. Dette førte til et stort tomrom mellom tekstfeltene for å angi start og stopp, samtidig som at antall viapunkter ikke var fleksibelt. Det antall forutbestemte tekstfelt, gav antall mulige viapunkter. For å løse dette, har vi lagt tekstfeltene inn i en ordnet liste. Denne er i utgangspunktet tom, og hver gang knappen ≪Legg til nytt viapunkt≫ trykkes, legges det til et listeelement med tekstfelt. Dette gjør at antall viapunkter ikke

er forutbestemt av antall forhåndsdefinerte tekstfelt, samtidig som viapunktene ikke opptar plass i brukergrensesnittet før de legges til. Dette er illustrert i skjermbildene i figuren under.

1 – viapunktene ligger i liste, ingen viapunkter er lagt til. 2 – viapunktene ligger som usynlige tekstfelt, legg merke til avstanden ned til knappen «Beregn». 3 – Her er det lagt til viapunkter i lista, og det er fremdeles plass til flere. 4 – Her er det også lagt til viapunkter, og de utfyller den «ledige plassen» i bilde 2. Her er det ikke plass til flere viapunkter.



4.1.2 HTML - Web Service Versjon 1

I vår løsning, skal brukegrensesnittet være knyttet mot en web service på en server (hvilken?), for å ha så lite logikk som mulig på klientsiden. Vi skal ha klientsiden, med html, javascript og css, og web servicen med c#. Denne første løsningen baserer seg i utgangspunkt på testdata. Vi skal sende inn start, stopp, via og kjøretøy til web service, slik den ferdige løsningen skal, web servicen skal så returnere dummydata for bomstasjoner, priser og avstand. Web service skal senere kobles mot ruteplantjenesten, slik at dummydata kan endres til virkelige verdier beregnet på grunnlag av angitte steder og fremkomstmiddel. Etter å ha søkt rundt på nett for å finne ut hvordan dette best mulig kan gjøres, så det ut til at den enkleste og mest utbredte måten å gjøre dette på var ved bruk av Microsoft sin useService. Eksemplene på nett var mange, og de var veldig relevant med tanke på løsningen vi hadde tenkt. Vi prøvde mange eksempler, både ved å klippe ut og lime inn kode, og ved å laste ned hele løsningene som skulle fungere. Ingen klarte derimot å opprette forbindelse til web servicen, og ingen gav feilmelding på hvorfor. Dette ble testet i firefox og chrome. Etter å ha studert Microsoft sin dokumentasjon av useService for å finne ut av hvorfor dette ikke fungerte, viste det seg at denne funksjonen var faset ut siden Internet Explorer 10, som ble utgitt høsten 2012. Da dette er nokså nylig, med tanke på at en del fremdeles kjører eldre versjoner en stund før de tvinges til å oppgradere, fantes det også en del nyere eksempler der useService ble brukt. Det skal også nevnes at ved å teste løsningen i Internet Explorer 11, fikk vi feilmelding. Ved nærmere undersøkelse, viste det seg også at hovedtyngden av eksempler var datert t.o.m 2013. Publiserings-dato, og å teste i alle browsere er noe vi vil ha fokus på å sjekke opp i, dersom vi møter tilsvarende

problemstillinger underveis.

Da vi ikke skal bruke aspx, som beskrevet i kap xx.. ble løsningen først å bruke xmlHttpRequest. Her sender vi inn parameterne i verdipar (formelt navn?). Viapunktene samler vi i en tekststreng, adskilt med komma. På denne måten kan vi sende x-antall viapunkter inn som kun et parameter, slik at vi ikke må ha ulike metoder for hvert antall viapunkter. I webservicen splitter vi denne tekststrengen på komma, slik at vi får viapunktene fordelt i et array, slik at viapunktene kan behandles hver for seg under kalkuleringen av ruta.

Vi har en klasse, CalculatedRoute, der et objekt av klassen, inneholder all informasjon om en kjørerute. Dette objektet inneholder atributtene start, stopp, vehicle (valgt kjøretøy), viapunkter, totalpris for liten og stor bil og barriers, som inneholder navn på alle bomstasjonene, samt priser for stor og liten bil. Dette objektet genereres når brukeren velger ≪Send rute≫, og returneres etter kalkulering av ruten, tilbake til klientsiden, der vi parser (pakker ut) objektene, og fremstiller relevant informasjon for brukeren. Denne første løsning inneholder bare testdata, slik at bomstasjonene får navn fra Bom0 til Bom4, og prisene for hver av bomstasjonene generes tilfeldig med randomfunksjon. Denne beregningen skal utvides til å foregå med grunnlag i data fra ruteplantjenesten.

Eksempel på objektstruktur Da web service som nevnt, ikke returnerer flerdimensjonale array, har vi valgt å returnere JSON. Json har den fordelen at selv om selve elementet er fledimensjonalt, returneres det som en endimensjonal tekststreng, som vi igjen kan parse i javascript for å hente ut objektene. Web servicen har XML som standardformat for data som returneres. Dette ville ikke la seg endre, og meningene på utallige diskusjonsforum var delte på om det i det hele tatt var mulig å få web service til å returnere JSON eller ei. Løsningen ble å returnerer JSON pakket inn i XML-tagger. Disse fjerner vi i javascript, før vi parser jsonobjektet.

4.1.3 HTML - Web Service Versjon 2

Etter et sprintmøte med Petter, fikk vi bekreftet at det skulle la seg gjøre å få endret slik at web service returnerer et JSON-objekt. Dette ville ikke la seg gjøre med xmlHttpRequest, selv om alt annet skulle tilsi at den skulle returnere JSON. Et nytt forsøk ville da være å endre xmlHttpRequest til Ajax-request, i håp om at dette ville fungere. For å redusere kode, blant annet på viapunkter, vil det være hensiktsmessig å sende inn et objekt som parameter, istedenfor verdipar.

versjon 2 under utvikling (til ferdig implementert prosjekt i alle delene vi skal ha med.) – Skrives underveis etter hvert som vi jobber videre.

4.1.4 programmer vi har brukt:

- Visual Studio 2013
- Visuam Paradigm Free UML design tool
- Surveymonkey.com

4.2 Innhenting av informasjon

Den informasjonen vi har trengt for å kunne fullføre arbeidet har blitt innhentet hovedsakelig fra nett. Glenn organiserte også et møte med en av hans forelesere Per Bisseberg ved HIOF for å få flere synspunkt på vårt arbeid, samt mer informasjon om hvordan ASP.Net applikasjoner med web service virker.

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

4.3 SCRUM - utførelse

Vårt produkt er utviklet på en standard som er satt av Infotjenester for utvikling av produkter, vi har benyttet en flerlags model innen programmering som generelt sett betyr at vi skal ha det mest mulig vedlikeholdbart samt at det skal være enkelt og bytte ut deler Vi utviklet en versjon 1 som vi benyttet til å få en bedre oversikt over hva eventuelle brukere føler de trenger i en slik modul, denne versjonen benyttet vi til å utarbeide en spørreundersøkelse som vi delte ut til et bredt spekter av personer, ut ifra alder og IT kunnskaper. Dette ga oss mye feedback på hvordan vi kan få produktet til et nytt nivå innen brukervennlighet. Dette tok vi med oss videre når vi ferdigstilte produktet og disse endringene ble gjort (. . .) (. . .) må utfylles når vi er så langt.

skal skrives om, Litt om foreløpig scrumresultater. litt om Backlog /userstories/milestone og at vi definerer på git. Gjerne screenshot fra første milestone. Begrunnelse for hvorfor vi ikke har hatt klart deifnerte sprinter fra begynnelsen (pga å lære oss nye ting, analyse muligehter, finne ut om prosjektet var gjennomførbart osv)

Kapittel 5

Testing/evaluering

5.1 Testing

Vi tester underveis, beskriver hvilke endringer vi finner ut bør gjøres. Lar andre teste – gjerne svare på et spørreskjema med hva som er bra/bør endres. Oppdragsgiver kommer med tilbakemeldinger. Begynne på spørreskjema (til etter testing av ferdig løsning og evt spørreskjema til å avklare nå hva som er foretrukket av manuell intasting/autocomplete, kartbaser/tekstbasert oppsummering av reisen osv), samt en analyse av resultane fra undersøkelsen, og hvilke endringer vi gjør ut i fra resultatene

5.2 Spørreundersøkelse

I forbindelse med første utkast av programmet vårt valgte vi å gjennomføre en spørreundersøkelse på en stor variasjon av personer med forskjellig alder og IT kunnskaper, dette for å forsikre oss om at vi er på rett vei når det kommer til brukervennlighet og funksjonalitet. Her vil vi la personer med erfaring fra føring av reiseregning teste programmet i tillegg til personer med veldig liten erfaring fra bruk av pc. Målet er å gjenspeile en så stor brukergruppe som mulig, og ettersom variasjonene skal være store, er det i tillegg viktig å dokumentere i undersøkelsen, hvilken brukergruppe den som svarer, tilhører.

1. Navn
2. Aldersgruppe
3. PC erfaring/kunnskap
4. Syns du produktet var enkelt og bruke? / brukervennlig
5. Er det noe du syns mangler?
6. Hva var bra?
7. Er det noen funksjoner du syns burde være med?
8. Når du taster inn Fra og Til, så kommer det opp eksempler på steder, syns du dette bør være med?

9. Når ruten er tastet inn foretrekker du en grafisk fremstilling av ruten, med bruk av kart, eller foretrekker du en skriftlig rutebetegnelse?

DISSE SKAL UTBEDRES, KUN UTKAST

Kapittel 6

Diskusjon

Skrives når vi nærmer oss ferdig.

Her skal det dokumenteres at vi har lært noe undersis, ikke bare levert et produkt til oppdragsgiver. I hvilken grad ble målene nådd? Ble leveransene fullført? Hvordan metoden fungerte?

Bør nevne om flerdelt prosjekt, at vi på skolen kun har hatt programmering på et nivå. Lært å splitte opp. Diskutere for om løsningen oppfyller alle deler, om målet med enklere reiseregistreing er nådd. Fikk vi levert alt som er beskrevet under leveranser? Hva var med i hver av leveransene iforhold til hva som skulle være med? Hvordan fungerte metoden (Scrum?), hadde vi noe nyttig av det? Ble det noen endringer underveis pga scrum? Hva fungerte bra/ikke bra? Hva ville vi gjort annerledes? Hvilke problemer oppsto?

Kapittel 7

Konklusjon

Sammendrag av diskusjonskapittelt. Legge vekt på det viktigste vi fant ut/lærte. Godt diskusjonskapittel = konklusjonen kan holde med en side. Legg vekt på tydelig språk. Sensor leser sannsynligvis først sammendrag, så konklusjon. Hvordan ble produktet iforhold til våre/oppdragsgivers forventning? Gjenta de viktigste punktene fra diskusjonskapittelet. Hva bør bli gjort videre ved en evt videreføring av prosjektet? Evt gi råd for de som skal jobbe videre med det.

Bibliografi