Bacheloroppgave

Forfatter: Kim Sandvold

Dato: 20/05-2010

SAMMENDRAG

Denne rapporten viser hvilke utfordringer man står ovenfor og hvor nært man kan komme en iPhone applikasjon i form av en webside. Dette er på mange måter en ny måte å tenke på for utviklere funksjonelt sett fordi navigeringen er mer intuitiv og mer inn i tiden enn slik det tradisjonelt gjøres. I tillegg til dette belyses det måter man kan utnytte ekstern data som RSS som hovedkilde til det faktiske innholdet.

Prosjektet benytter ingen ny teknologi for å oppnå dette, men bruker veletablerte teknikker på en litt annerledes måte. Prosjektet har med dette oppnådd et fult fungerende system som har et brukervennlig grafisk grensesnitt tilpasset små mobile enheter som iPhone. Dette er også utviklet på en slik måte at data presenteres på effektiv måte. Dette er viktig for mobile enheter som iPhone, da man ikke alltid har tilgang på hurtige nettverk.

For fremtidige utviklere har også dette prosjektet en verdi fordi det drøftes hvilke utfordringer og problemer man står overfor i forhold til nettleseren Safari for iPhone sin ytelsesevne. Det finnes lite dokumentasjon og informasjon om Safari for iPhone og hva som skiller den fra tradisjonelle nettlesere, noe man er interessert i ved utvikling av denne type applikasjoner.

Innholdsfortegnelse

Forord

Fluefiske.net startet som en hobby for virksomhetens tekniske ansvarlig, Filip Blaauw, i 1999 og har siden den gang fått flere partnere. En av disse er Joakim Andreassen som tok seg av journalistiske oppgaver og ble virksomhetens redaktør. Senere ble Anne J Backsæther med sin redaksjonelle og journalistiske kompetanse, med som partner og ansvarlig redaktør slik at virksomheten i dag har vokst fra hobby til en bedrift med inntekter som er konkurransedyktige i miljøet. Fluefiske.net har ambisjoner om å utvide inntektsområde og målgruppe ved å gi ut et magasin fire ganger i året. De har også flere bidragsytere som hjelper med å holde virksomheten i gang. I takt med dette er de alltid på jakt etter å utvide sine webløsninger og derfor vil en løsning som denne ha en verdi for dem.

Jeg vil benytte sjansen til å takke alle som er involvert i Fluefiske.nets virksomhet som har bidratt til at systemet har blitt som det er. Denne oppgaven er blitt til ved hjelp av Fluefiske og utviklere som har delt sine erfaringer og meninger på fora som Stackoverflow.com og andre åpne ressurssider på internett som bidrar til god læring.

Jeg vil også benytte sjansen til å takke min veileder Monica Strand ved Høgskolen i Gjøvik som har kommet med konstruktiv kritikk gjennom denne prosessen og i tillegg vist engasjement for prosjektet.

Om undertegnedes forklaringer og formuleringer i perioder blir noe omstendelige, oppfordres leseren til å besøke *http://iphonify.emio.no/?client=1* for selv å teste funksjonalitet!

Dato ..... Underskrift .....

1. Innledning

1.1. Oppdragsgiver

Dette prosjektet er basert på et ønske fra ansvarlig redaktør i nettmagasinet Fluefiske.net, Anne Bachsæther, om å utvikle et tilleggssystem til bedriftens eksisterende webløsning. I dag har de en standard webbasert nettavis i tillegg til to ulike diskusjonsfora som benyttes aktivt av mennesker i miljøet både innenfor og utenfor Norges grenser. Fluefiske.net har også flere samarbeidspartnere med et stort engasjement og alle mente behovet for en utvidelse av websystemene var på sin plass.

Under samtaler med Fluefiske.net i forkant av oppgaven hadde oppdragsgiver, Fluefiske.net, mange ønsker om hva dette skulle gå ut på, noe de veldig gjerne kunne tenke seg var en løsning for mobiltelefoner. Hele staben til Fluefiske.net er i tillegg til fiskesporten over middels engasjerte i Apple sine produkter og naturligvis fikk jeg en følelse av at de ønsket noe som berører dette selskapet. De kom etter hvert med en forespørsel om jeg kunne lage en applikasjon til iPhone. Ettersom jeg som student ikke hadde noen opplæring eller kompetanse innenfor de programmeringsspråkene en slik applikasjon er laget med, kom jeg med noen innspill for å møte dem på halvveien.

På grunn av at de ønsket en løsning til iPhone og jeg har kompetanse i webprogrammering kom vi frem til at en webbasert løsning/versjon av deres eksisterende webside ville være et passende prosjekt som både har en verdi for oppdragsgiver og en relevant utfordring for meg som student.

Det å lage en tradisjonell webside til en mobiltelefon i seg selv er ingen stor nok utfordring for en oppgave som denne. Vi begynte derfor å diskutere hva dette prosjektet skulle innebære. Etter mange diskusjoner og innspill fra alle berørte parter kom vi frem til at en webbasert løsning til iPhone med tilsvarende funksjonalitet som en iPhone applikasjon har, ville vært et passende prosjekt. Dette skulle hovedsakelig bety at hovednavigasjonen ville ha en horisontal glidning og at alt av innhold hentes uten at man må tradisjonelt gjenoppfriske siden. Dette vil gjøre at løsningen vil ta i bruk standarder og teknikker som er relevante i forhold til dagens standarder. Flere nettsteder har i dag spesialtilpassede løsninger for mobile enheter, men ikke mange har

implementert en slik navigasjonsstruktur og derfor så oppdragsgiver dette som en litt unik løsning.

Fluefiske.net driver i dag en nettavis. Slik jeg ser det, vil det mest fornuftige til mobilløsningen være å ta vare på den informasjonen de publiserer på nettavisen og presentere dette innholdet på en iPhone. I dag benytter de seg av flere forskjellige publiseringsløsninger som Joomla(CMS), Invision Board(forum) og Simple Machines(forum). Ettersom de ikke hadde låst seg til én løsning hadde jeg lyst til å lage en løsning som er fleksibel i forhold til det å hente innholdsdata.

Fluefiske.net hadde mange ideer til innholdet i mobilløsningen. De ville ha ny informasjon som Google Maps integrert for å finne fiskevann og værvarsling der man er nå. Siden vi ble enige om å forholde oss til gjenbruk av allerede publisert informasjonen, og oppdragsgiver ikke til tilrettelagt for innholdsdata som nevnt over, vil ikke en slik funksjonalitet bli implementert i dette prosjektet.

1.2. Mål

Hovedmålet mitt som student ble å lage en webbasert løsning til Fluefiske.net som er i stand til å gjengi den informasjonen de publiserer på sine publiseringsplattformer. Denne gjengivelsen skal man kunne navigere i på tilsvarende måte som en iPhone applikasjon og informasjonen skal lastes dynamisk uten å gjenoppfriske siden på tradisjonell måte.

I et slik webløsning vil det være naturlig å ta vare på identifikatorer fra deres egen webside som logo, farger og andre elementer som identifiserer dem som et selvstendig foretak. Med en slik visuell funksjonalitet vil prosjektet delvis bestå av grafisk utforming, men det understrekes at selve programmeringen har hatt førsteprioritet.

En mobil enhet som iPhone har i forhold til en tradisjonell nettleser en begrenset plass og ytelse til å presentere et fullstendig innhold. Deres eksisterende webløsning inneholder massive informasjonsmengder i form av bildegrafikk, video og annonsebannere og dette ble vi enige om å utelukke. En mobil enhet som iPhone har ofte ikke tilgang på hurtige nettverk og derfor ble vi enige om å presentere det viktigste fra websiden for å redusere nedlastningsmengden av data.

Selv om en iPhone har teknologi og støtte for skalering av innholdet, noe som gjør det mulig å lese innholdet fra store websider, valgte vi å fokusere på å presentere det viktigste på riktig måte. I tillegg til dette skal ikke løsningen måtte laste inn så mye data som den ordinære websiden består av og det skal presenteres på en leselig måte for sluttbrukeren.

I denne sammenheng kan det virke naturlig og tilrettelegge løsningen for flere mobile enheter Dette ville forutsatt at jeg som utvikler hadde tilgang på et utvalg av mobile enheter, noe som ikke var tilfelle. Dette er noe av grunnen til av vi i fellesskap valgte å konsentrere oss om iPhone enheten - som i utgangspunktet var oppdragsgivers ønske.

Oppsummert blir prosjektets konkrete mål:

- Webbasert løsning til iPhone basert på innholdsdata fra den ordinære websiden.
- Løsningen skal ha tilsvarende funksjonalitet som navigasjon og et tilsvarende visuelt uttrykk.
- Løsningen skal være fleksibel og uavhengig av publiseringsverktøy, men med vekt på Joomla CMS(content management system) som er hovedplattformen til deres eksisterende løsning.

1.3. Arbeidsmetode

Dette arbeidet har gjennom hele prosessen bestått av jevnlige diskusjoner og beslutninger i samarbeid med oppdragsgiver og andre involverte parter, som også har bidratt med innspill underveis.

I utgangspunktet var planen å gjøre ferdig prosjektet i separate faser. Dette vil si å gjøre undersøkelser og forberede prosjektet i forkant av selve programmeringen. Sluttfasen skulle bestå av testing av systemet og deretter bruke siste delen på eventuelle feilsøkinger og vedlikehold til prosjektets slutt. Prosjektet har med dette benyttet seg av systemutviklingsmodellen fossefallsmetoden. Underveis i prosjektet har det forekommet tilbakesteg og endring av krav til prosjektet og dermed blandet inn iterative utviklingsmodeller i forhold til systemutviklingen(Sommerville, 2007).

En del av arbeidet har vært basert på undersøkelser om mobilenheten iPhone og dens nettleser, Safari. Dette har vært en nødvendighet før arbeidet med programmeringen og planleggingen kunne starte. Safari for iPhone er på mange måter annerledes enn alle de tradisjonelle nettleserne.

1.4. Avgrensning

Prosjektet har et potensielt vidt bruksområde. Med dette menes at løsningen kan fungere på veldig mange mobile enheter, men det er valgt å fokusere på iPhone spesielt. Ettersom oppdragsgivers opprinnelige ønske var en iPhone applikasjon vil funksjonalitet og utseende være rettet mot iPhone sitt visuelle uttykk. Dette prosjektet vil derfor være avgrenset denne mobile enheten.

1.5. Organisering av rapporten

Denne rapporten er organisert på den måten at generelle aspekter nevnes og drøftes i første rekke. Deretter drøftes mer konkrete sider som går mer direkte på selve produksjonen av løsningen som er laget.

Ordbruk i denne rapporten er lagt opp til at det skal være lett og forstå så langt det lar seg gjøre. Enkelte ganger oppstår det ord som ikke er selvforklarende og de defineres under.

I objektorientert programmering snakker man ofte om "metoder" som er den faglige terminologien(Lavin, 2006). I denne rapporten kaller jeg metoder for "funksjoner". Metoder brukes da i rapporten på samme måte som i dagligtalen - måten man gjør ting.

Forkortelsen RSS nevnes ofte ettersom dette er en stor del av oppgaven. RSS er en forkortelse for Rich Site Summary. Noen mener også at forkortelsen betyr Really Simple Syndication. Uansett hva man kaller det er RSS et format for web som ofte brukes av nettaviser og andre websider som oppdaterer innholdet sitt ofte. Mange nettlesere gir tilgang til å abonnere på nyheter automatisk slik at man umiddelbart får ferske nyheter (Lavin 2006). RSS kan også manipuleres i webprogrammering og dette er ideelt i dette prosjektet nettopp fordi vi skal gjenbruke allerede publisert data.

Programmeringskoden er konsekvent skrevet på engelsk. Man er tvunget til å skrive en del innebygde funksjoner og kodesegmenter på engelsk. Personlig ser jeg ingen grunn til å blande to forskjellige språk som norsk og engelsk i samme system. Syntaktisk vil ikke dette se bra ut og en dag skal kanskje systemet brukes av utviklere som ikke snakker norsk. Det taler imot å bruke det norske språk i programmeringskoden. Når dette er sagt er all kommentering skrevet på norsk for at jeg som student skal være helt sikker på at forklaringene skal være forståelig nok for sensor og andre som skal lese denne oppgaven.

Som utvikler arbeides det som nevnt med mange engelske begreper og dette blir en vane. Mange av disse ordene og uttrykkene har til tider vært vanskelig å finne logiske og korrekte norske oversettelser på. En beskrivelse av et sett med programmeringskode kalles i denne oppgave for ”skript”.

Programmeringskode som illustreres i rapporten kan ha forskjellig struktur fra den koden som er i selve løsningen. Den er endret slik at det skal se bedre ut i dette formatet og kommentarene er tilpasset rapportens tekst og tema. Fullstendig skript og kodesegmenter som det refereres til i denne rapporten legges med som vedlegg for.

Når det gjelder navnkonvensjoner og vaner angående programmeringen er det et eget kapittel for dette - "Prinsipper og teori angående programmeringen".

Litteraturhenvisninger brukes på vanlig måte i teksten; (etternavn, initial, år) og komplett litteraturliste finner man i slutten av rapporten. Alle henvisninger til stoff som er funnet på internett er satt opp som fotnoter.

1.6. Forberedende undersøkelser

Det er vanskelig å finne opp hjulet på nytt i disse dager, men man kan videreutvikle ideer og bruke teknikker og metoder som ikke brukes i tradisjonell webutvikling. Selv om denne løsningen ikke er en ny oppfinnelse er det en innovativ løsning, en nytenkende måte å strukturere webapplikasjoner på. For å finne ut om noen har laget en tilsvarende løsning som dette tidligere er det blitt gjort en rekke søk på internett. Dette førte frem til flere interessante treff, blant annet www.iphonify.net, som har mange likhetstrekk med dette prosjektet, men uten den type animert

navigasjon som jeg ønsket. Derfor hadde ikke iphonify.net en tilstrekkelig funksjonalitet, men allikevel ga den ideer til dette prosjektet i forhold til presentasjon av innhold.

www.webstuffshare.com[1] var en kilde til inspirasjon i forhold til navigert animasjon. Dette var ikke omfattende nok, men gav meg en ide om hvordan dette skulle kunne til se ut.

Når det gjelder publiseringsverktøyet Joomla finnes det et stort antall programtillegg man kan benytte seg av. På Joomlas ressursside finner man egne kjernemoduler som tilpasser websider til mobile enheter som iPhone[2]. Selv om denne siden inneholder programtillegg som til en viss grad kunne tilfredstilt målet i denne oppgaven, mangler dem de animerte visuelle uttrykkene en iPhone applikasjon har. Disse programtilleggene er på en annen måte ikke lik tanken om dette prosjektet fordi et av målene er å lage en applikasjon som er uavhengig av publiseringsverktøy.

Dette var noen av de beslektede prosjektene og systemene som var noe i nærheten av hva jeg var ute etter, men jeg kunne ikke finne en løsning som stod til alle mine krav. Det fantes mange mindre skript som viste hvordan man kan få til en horisontal animert navigasjon og RSS parsing, men ingen som har gjort noen prosjekter på størrelse med dette. Derfor er det grunn til å mene at dette prosjektet er i en viss grad nytenkende med tanke på sluttbrukerens opplevelse av løsningen som en webapplikasjon.

1.7. Valg av utviklerverktøy

Skal man holde på med programmering er det fordelaktig å benytte seg av de verktøy man liker og de man synes er oversiktlige. Det finnes mange forskjellige tekstbehandlingsprogrammer for programmering som egner seg godt, eller er ment for denne type jobber. I mitt tilfelle har jeg hele tiden mens jeg har drevet med webprogrammering benyttet meg av programmet Notepad ++ og derfor falt det meg naturlig å fortsette med dette da jeg skulle gå i gang med denne oppgaven.

Det er alltid hensiktsmessig å utvikle webbaserte løsninger lokalt på egen maskin fordi da er man uavhengig av internett og man kan holde systemet lukket for publikum inntil det er ferdigstilt. Ettersom en del av denne oppgaven baserer seg på serverbasert programmering, må man ha en

[1] http://webstuffshare.com/demo/iPhoneNav2/
[2] http://extensions.joomla.org/extensions/core-enhancements/mobile

type server installert som er tilpasset det bestemte behov. I mitt tilfelle var jeg avhengig av en Apache server med PHP installert. I tillegg måtte jeg ha en relasjonsdatabase, det mest vanlige i denne sammenheng er MySQL database. Oppgaven dreier seg ikke om hvordan man installerer og konfigurerer disse serverelementene, derfor valgte jeg en ferdig løsning der både Apace, PHP og MySQL er integrert i samme pakke. Denne heter WAMP Server og er Open Source[3].

Oppgaven handler også om å lage en passende layout tilpasset iPhone. Her benyttet jeg meg av Adobe sine grafiske verktøy. Hovedsakelig bruker jeg Fireworks[4] som er et grafisk program rettet mot webutvikling. Dette programmet tilfredsstiller alle krav til design til web. Personlig synes jeg Photoshop[5] er bedre på behandling av fotografier, men for objekt/vector grafikk mener jeg Fireworks eller Illustrator[6] egner seg best.

1.8. Litteratur

Noe som er spesielt med å bygge en slik webapplikasjon i forhold til tradisjonelle løsninger er at iPhone enheten er veldig ny på markedet og det er da naturlig at det finnes lite dokumentasjon og litteratur om iPhone OS og Safari for iPhone. Dette har ført til at enkelte problemer og utfordringer har krevd en mer empirisk tilnærmingsmåte i forhold til å bruke faglitteratur. Det mest relevante av faglitteratur er en utviklerguide for iPhone OS 3.0, skrevet av Richard Wagner dette året, 2010.

Heldigvis finnes det masse uoffisiell dokumentasjon og informasjon på internett fra utviklere som har gjort et godt forsøk på å finne ut av enkelte ting. Quirksmode.org[7] har en god del informasjon og websiden Stackoverflow.com[8] har 21605 registrerte stikkord i sin database med spørsmål fra andre utviklere, der majoriteten av disse er besvart. Så dette er gode ressurssider for å finne ut av utfordringer i utviklingen med iPhone løsninger som et alternativ til faglig litteratur.

[3] http://www.wampserver.com/en/
[4] http://www.adobe.com/products/fireworks/
[5] http://www.adobe.com/no/products/photoshop/photoshop/
[6] http://www.adobe.com/no/products/illustrator/
[7] http://www.quirksmode.org/
[8] http://stackoverflow.com/questions/tagged/iphone

1.9. Problemstilling

Problemstillingen til denne oppgaven er formulert slik; "*Hvordan utvikle en webside til iPhone med et tilsvarende visuelt uttrykk, samt bruke RSS som innholdsdata*".

2. Prinsipper og teori angående programmeringen

2.1. Tekniske aspekter ved iPhone

2.1.1. Utfordringer

Det å lage en webapplikasjon til en mobil enhet har i utgangspunktet samme utfordringer som en tradisjonell webapplikasjon. En nettleser på en mobil enhet leser på samme måte som andre nettlesere, klient kode som HTML, CSS og Javascript, derfor har man mulighet til å presentere websider på en tradisjonell måte. Alikevel finnes det forskjeller som gjør utviklingen av slike websider annerledes. Dette kan skape både fordeler og ulemper for en utvikler som er kjent med tradisjonelle metoder når det gjelder utviklingsprosesser. Jeg skal her belyse noen av de utfordringene man står overfor i dag når det gjelder denne overgangen til mobile enheter som iPhone.

Noen av utfordringene med å utvikle en webapplikasjon til en mobil enhet i dag har røtter i apparatets maskinvare. Tar man for seg iPhone er den i likhet med mange andre mobile enheter i dag utstyrt med en gyro teknologi som gjør at skjermbildets vinkel justerer seg i forhold til hvordan du holder den. Snur man telefonen fra horisontal til vertikal stilling følger skjermbildet med. Problemet med denne funksjonaliteten i forhold til webapplikasjoner er at skjermen på iPhone og de fleste andre telefoner ikke er kvadratisk. Altså justerer bredden seg hele tiden relativt i forhold til hvordan telefons vinkel i øyeblikket er. Websiden kan lett vise seg på en uønsket måte. Hvis man for eksempel bruker grafikk på websiden og denne allikevel er tilpasset det grafiske grensesnittet til telefonen, vil størrelsen endre seg i takt med justeringen maskinvaren gjør. Dette kan resultere i at grafikkens kvalitet blir redusert, noe man helst vil unngå.

Det er også viktig å tenke på brukervennlighet. Til forskjell fra tradisjonelle websider som man bruker en musepeker for å navigere er mobile enheter som iPhone ment å bruke fingeren på. En tradisjonell nettside kan ofte bli for tungvindt å håndtere med fingeren på en iPhone. iPhone er av mindre format enn tradisjonelle skjermer og fingeren er som regel større enn en musepeker. Dette taler for at en webside som er tilpasset iPhone er relevant og nødvendig for at brukervennlighet skal kunne oppnås.

Av mer konkrete tekniske utfordringer stiller HTML og CSS seg i lyset. iPhone bruker som standard nettleseren Safari. Dette er ikke en fullverdig versjon av den tradisjonelle utgaven. Dette er en begrenset versjon av den originale nettleseren, blant annet er mange CSS og Javascript funksjoner fjernet (Wagner, 2010). Dette er ting man må ta høyde for ved denne typen webutvikling.. Et resultat av dette er både begrensninger og utfordringer i utviklingsprosesser[9].

iPhonen sin maskinvare er også en utfordring å jobbe med. Den fysiske størrelsen i forhold til ordinære datamaskiner, er i takt med maskinvaren noe begrenset. Apple har ikke kommet med offisiell informasjon om dette, men uoffisielle kilder mener iPhone 3Gs har en CPU(Central Processing Unit) kapasitet på 600Mhz og internminne på 256 MB RAM noe som mange ganger mindre enn det en ordinær datamaskin i dag kan tilby. Dette bekrefter uoffisielt at ytelsen totalt sett til iPhone må tas i betraktning ved utvikling denne type websider[10].

2.1.3. Flash

I mange år har det vært populært å skape websider med animasjoner. Den mest populære programvaren for dette er Macromedia/Adobe Flash. I dag er det Adobe som har dette programmet, men de som først lanserte dette i 1996 var selskapet Macromedia som var innehaver frem til 2005 da Adobe kjøpte Macromedia[11].

iPhone har i dag(iPhone OS 3.0) ikke støtte for Flash. Det vil si at hvis man skal lage animasjoner og dynamiske visuelle effekter må man finne alternative metoder for å oppnå dette. Ettersom en

---

[9] http://developer.apple.com/safari/library/documentation/AppleApplications/Reference/SafariWebContent/Introduction/Introduction.html

[10] http://gizmodo.com/5286031/iphone-3gs-processor-and-ram-uncovered-600mhz-and-256mb

[11] http://en.wikipedia.org/wiki/Adobe_Flash

relativt stor andel av websider globalt sett inneholder helt eller delvis elementer av Flash, er dette en stor begrensning og ikke minst en veldig god grunn til å utvikle alternative websider.. Norske sider som innholder Flash teknologi har ifølge statistikken til Opera en prosentandel på 37,19 %.

Fig 1.a. - Statistikk over websider som inneholder Flashelementer[12].

Country	Total URLs From Country	# Usage Of Flash	% Usage Of Flash	Country	Total URLs From Country	# Usage Of Flash	% Usage Of Flash
United States	1,477,436	481,250	32.57%	Denmark	50,875	12,888	25.33%
Germany	407,638	101,914	25.00%	Australia	49,982	15,069	30.15%
Great Britain	244,554	74,037	30.27%	Switzerland	49,683	13,714	27.60%
France	139,400	57,968	41.58%	Russia	40,790	13,370	32.78%
Italy	137,070	55270	40.32%	Sweden	33,654	9,321	27.70%
Canada	133,506	41,316	30.95%	China	31,345	21,010	67.03%
Japan	124,976	39,674	31.75%	Czech Republic	26,728	11,520	43.10%
Netherlands	79,562	29,600	37.20%	Austria	24,563	6,783	27.61%
Spain	76,421	35,339	46.24%	Norway	21,185	7,878	37.19%
Poland	58,929	24,971	42.37%	Turkey	18,621	11,145	59.85%

Det er vanskelig å gjøre seg en oppfattning angående hvor mange av disse sidene som bruker flash elementer av helt nødvendige årsaker, som å presentere sitt primære innhold.. Det finnes "show-off" sider som består 100% av Flash og andre som har deler av innholdet, som menyer og andre sekundære elementer. Noe man bør tenke på er selskaper som betaler for å annonsere på websidene. En stor del av reklamebannere er Flash. Denne gruppe vil da ha et noe mindre marked på grunn av begrensningen i iPhone.

2.1.4. Java

Flash er ikke det eneste av velbrukte teknologier som mange websider består av. Webutviklere har i mange år vært kjent med Java og det er brukt over hele verden, men Apple har ikke latt iPhone OS 3.x få støtte for dette. Det sies at Steve Jobs, som er grunnleggeren av Apple[13], gikk ut i media og mente at ingen brukte Java lengre og derfor var det ikke vits å bruke tid og ressurser på å integrere dette i iPhone. Apple sine produkter som Mac OSX har forøvrig støtte for dette på høyde med Microsoft Windows.

[12] http://dev.opera.com/articles/view/mama-key-findings/#flash
[13] http://www.apple.com/pr/bios/jobs.html

I likhet med Flash begrenser også denne manglende støtten, utvikling av websider og man må finne alternative programmeringsspråk og metoder for å oppnå det man ønsker. Denne mangelen taler også for behovet for egne tilpassede websider til iPhone. Et diskusjonsemne i forhold til dette er sikkerhet til nettbank der de fleste banker bruker Java appleten bankID for sikker innlogging. Noen banker gir fremdeles tilgang til innlogging med vanlig HTML skjema ved bruk av iPhone, noe som kan oppfattes som et sikkerhetshull. Når dette er sagt handler ikke denne rapporten om dette temaet, men dette belyser behovet for Java støtte, som igjen argumenterer imot Jobs påstander om at ingen bruker Java i dag[14].

2.2. Kommentarer i programmeringen

Denne tekniske løsningen er laget med noen konkrete programmeringskonvensjoner. Disse listes opp nedenfor.

PHP / Javascript / CSS kommentarer:

```
/**
* BLOKK KOMMENTARER
*
* Disse kommentarene brukes til å beskrive større deler av en
* sammenheng som klasser og sider
*/

/* OVERSKRIFT KOMMENTARER - Kort besrivelse av f.eks. en funksjon */

// EN-LINJE-KOMMENTAR - Brukes for mindre bemerkninger (ikke CSS)
```

Ini kommentarer:

```
; Alle kommentarer i .ini filer
```

HTML kommentarer:

```
<!-- Generelle kommentarer i html kode -->
```

Inline script og style elementer kommentarer:

[14] http://pogue.blogs.nytimes.com/2007/01/13/ultimate-iphone-faqs-list-part-2/

```
<!-- javascript kodesegmenter eller CSS //-->
```

2.3. Navngivelser

Generelle orddelingskonvensjoner i programmeringen består av underline (_). Nedenfor listes noen forskjellige eksempler:

PHP og javascript:

```
javascript : variabel_navn = null;
javascript : funksjons_navn();

php : $variabel_navn = null;
php : funksjons_navn();
```

HTML :

```
<div id="id_navn"></div>

<div id="id::navn::verdi::target"></div>(spesielle tilfeller for
javascript/ajax)
```

CSS:

```
.klasse_navn{}
#id_navn{}
```

Filer:

```
program_fil.php
```

Den tekniske løsningens kodesegmenter følger disse reglene for å gjøre det mest mulig lesbart for eventuelle programmerere som skal gjøre endringer og videreutvikle i etterkant. Dette er kjente konvensjoner som er vanlige å bruke.

2.4. Valg av teknikker og språk

2.4.1. PHP

Et viktig aspekt med denne løsningen er å skape lesbar og fleksibel kode som har fokus på gjenbrukbarhet. En enkelt funksjon skal ikke forekomme med enn en gang i systemet. Derfor er det brukt en objektorientert tilnærming som skaper mer ryddighet og fleksibilitet på den delen som berører serverprogrammering. Noen mener objektorientert programmering har senere responstid enn vanlig prosedural programmering, men i denne skalaen og disse omstendighetene er det nærliggende å tro at hastigheten ikke vil være noen som helst hinder for de faktiske programprestasjoner. Selvsagt brukes objektorientert programmering for å vise kunnskaper og kompleksitet også.

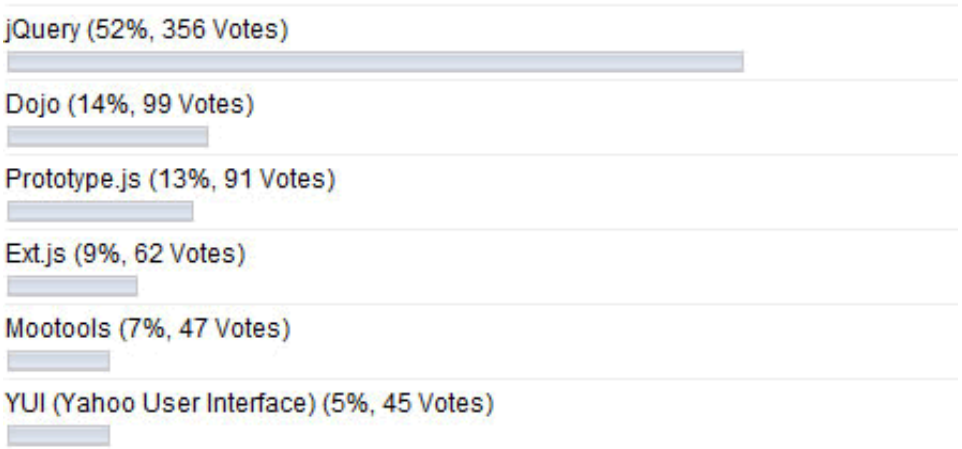
2.4.2. Javascript

Javascript er et sentralt programmeringsspråk på klientsiden. For å bestemme om spesielle teknikker eller biblioteker skal tas i er det utført noen undersøkelser. Målet er hurtighet og minst mulig kode og størst mulig oversikt for fremtidige teknikere.

I denne løsningen vil det bli brukt moderne teknikker og teknologier, og det vil legges vekt på så enkel som mulig tilgang til endringer i etterkant. Ettersom oppdragsgiver allerede har en teknisk ansvarlig som skal drifte dette senere vil dette være fordelaktig. Standard Javascript vil bety veldig mye kode og det er ikke ønsket i dette tilfellet. Derfor er det klart at et Javascript bibliotek vil ta seg av programmeringsmessige aspekter på klientsiden, fordi de er laget på en slik måte som gjør at Ajax forespørsler og behandlinger av hendelser(events) kan skrives med mindre kode. Dette er mer oversiktelig enn standard Javascript. Dette er også fordelaktig for meg som student som har lyst til å ta et dypdykk i et Javascript bibliotek.

Det å finne relevante tester og dokumentasjon på ytelsen av de forskjellige javascript bibliotekene krevde en del undersøkelser. Det var vanskelig å få et konkret svar på hvilke bibliotek som er mest effektivt. Undersøkelsene viste at jQuery er å anbefale av utviklere. Figuren under inneholder en representativ statistikk som viser at utviklere generelt sett favoriserer Javascript biblioteket jQuery.

Fig 2-a. - Javascript bibliotek stemmer[15].



Dette taler for at jQuery biblioteket egner seg til denne iPhone løsningen. Dette Javascript biblioteket er også fordelaktig fordi det er det man kaller "cross browser". Det vil si uavhengig av nettleser. Nå handler denne løsningen primært om nettleseren Safari, men i et tilfelle hvor iPhone skulle få en støtte for flere nettlesere, eller at løsningen vil fungere på andre telefoner med andre nettlesere, vil nettleserkompatibilitet være aktuelt[16]. Dette argumenterer også for at utviklere senere skal ha forutsetninger til å gjøre endringer og holde dette ved like.

Vi står også ovenfor noen utfordringer selv om jQuery er et såkalt "cross browser" bibliotek. Ettersom man ikke har en musepeker på iPhone har man ganske enkelt ikke mulighet til å bruke tradisjonelle hendelsesbehandlinger som "mouseover", mouseout" fordi en tradisjonell musepeker er en kontinuerlig enhet, mens fingeren er usammenhengende.

Eventer, eller hendelser i Javascript, er slått sammen på grunn av dette. Quirksmode.org har uoffisielt dokumentert noen av forskjellene på tradisjonelle metoder og iPhone sine egne. Fokuserer man på en lenke med ett trykk kalles eventen "hover". Denne er slått sammen med "mouseover", "mousemove", "mousedown", "mouseup" og dermer aktiveres eventen "click"[17]. Dette skaper utfordringer fordi man i tillegg til å oppnå en spesifikk funksjonalitet må tilpasse CSS koden til dette.

[15] http://roshanbh.com.np/2008/06/which-is-the-best-javascript-ajax-framework.html
[16] http://jquery.com/
[17] http://www.quirksmode.org/blog/archives/2008/08/iphone_events.html

2.4.3. CSS

Det visuelle uttrykket er en blanding av Javascript og CSS(cascading style sheet). Tradisjonelt brukes CSS til å forme utseende til en webapplikasjon. Dette organiserer vanlig HTML kode, og det samme gjelder for tradisjonelle webapplikasjoner som for iPhone websider. Allikevel finnes det noen begrensninger som må belyses.

Ettersom man ikke har en musepeker har Apple tatt bort eller slått sammen enkelte funksjoner for å finne en middelvei som nevnt i avnittet om Javascript ovenfor.

Apple har i forhold Safari for iPhone dokumentert lite for webutviklere, men Quirksmode.org har tatt tak i dette og kommet med opplysende informasjon med god verdi for utviklere. Dette gjelder først og fremst Javascript, men Javascript og CSS er på en måte nært beslektet. CSS funksjonene denne løsningen trengte har Apple selv dokumentert, og det er funksjonen; position:fixed;[18]. I denne løsningen var ideen å ha logo og toppseksjon alltid synlig som mange iPhone applikasjoner har, men på grunn av at Safari for iPhone funksjonelt sett kombinerer CSS posisjoneringsfunksjonene "fixed" og "absolute" er det ikke mulig ved hjelp av CSS å holde topplogo og bunnseksjon alltid synlig. Dette er det viktigste avviket i CSS som ikke berører Javascript.

CSS har også en rolle når det gjelder optimalisering for hurtighet. Hvor mye dette i har å si i praksis kan diskuteres, men som et mål med å gjøre alt mest mulig effektivt er det tatt høyde for å optimalisere. Dette gjøres med ganske enkle konvensjoner og teknikker. Det brukes "short-hand-css"[19] der det kan brukes. Alle tilhørende spesifikasjoner til en enkelt klasse eller id skrives på en linje for å minimalisere størrelsen på filen.

Eksempel på "short-hand-css" mot ordinær css:

```
/* Ordinær CSS */
.klasse_navn{font-weight:bold;font-size:12px;line-height:140%font-
family:arial}
```

[18] http://developer.apple.com/safari/library/technotes/tn2010/tn2262.html#safari_on_ipad_readiness_checklist-4__modify_code_that_relies_on_css_fixed_positioning
[19] http://www.dustindiaz.com/css-shorthand/

```
/* Short-hand CSS */
.klasse_navn {font: bold 12px/140% arial}
```

2.4.4. HTML

Det å markere opp strukturen i et HTML dokument er på mange måter likt i forhold til Safari for iPhone som i tradisjonelle nettlesere. Om man skriver XHTML kode eller HTML5 kode er dette relativt likt. Men som nevnt tidligere er Safari for iPhone noe annerledes, mest på grunn av den gyrotekniske funksjonaliteten er det visse ting man må passe på. Tar man ikke høyde for dette kan problemer oppstås hvis man ikke markerer opp dokumentets metadata på rett måte.

2.5. Spesielle avvik til krav

2.5.1. Nettlesere

Nettleserkompatibilitet er tradisjonelt et vanlig krav i forhold til webutvikling. Det finnes flere alternative nettlesere til iPhone man kan benytte seg av, men valget står ikke på de mest vanlige og populære nettleserne som Opera, FireFox, Internet Explorer, Google Chrome, selv om at det sies at Opera skal komme med nettleser til iPhone[20]. Valget er ikke veldig utbredt selv om man har mulighet til å finne et alternativ til Safari for iPhone[21]. Ettersom ingen av disse nettleserne på noen måte er konkurrenter til Safari og Safari kommer med iPhone som standard kan man konkludere med at Safari er helt klart den mest brukte av disse.

Altså er ikke nettleserkompatibilitet et viktig fokus for denne løsningen. Når dette er sagt vil webutviklingen følge vanlige metoder som er kompatible med alle de mest vanlige nettleserne.

2.5.2. SEO

Søkemotoroptimalisering "SEO - Search engine optimization" er også et vanlig krav til websider i dag. Dette handler om å bygge opp websiden på en slik måte at søkemotorer kan registrere nettsiden så den bli søkbar på nettet(Pfaffenberger, B. Schafer, S. White, C. Karow, B. 2004). Man vil som oftest oppnå flest mulig treff ved søk på nettet. Mange selskaper tjener penger på

[20] http://www.rainbowskill.com/trans/no/internet-fundas/opera-mini-browser-is-coming-to-iphone.php
[21] http://browsers.about.com/od/iphonewebbrowsers/tp/iphone-web-browsers.htm

reklame og tar betalt for annonser i forhold til besøkstall på sine nettsider. Det er da naturlig å tro at en slik mobilløsning som denne ville hatt dette kravet, men det har det ikke.

Denne løsningen er et substitutt til oppdragsgivers ordinære nettsider og den er allerede optimalisert for søk på nettet. All informasjon du kan finne i denne iPhone-løsningen er allerede søkbart fra den opprinnelige websiden og det er ingen grunn til å duplisere dette. Et scenario som kunne fått uheldige konsekvenser hvis man søker etter en spesiell artikkel på nettet, og får treff som peker til iPhone-artikkelen, ville alle de som ikke bruker iPhone mistet mulighet til å se artikkelen. Dette er fordi løsningen er satt opp på en slik måte at websiden gjenkjenner hvilken nettleser og operativsystem som benyttes. Hvis man bruker en vanlig datamaskin blir man omdirigert tilbake til den ordinære websiden. Derfor må SEO unngås og det finnes ingen argumenter som sier at SEO skal være et krav i denne løsningen. Det ville skapt problemer.

2.6. RSS - Hvilke muligheter har man

All informasjon som fremkommer på denne løsningen kommer ikke direkte fra en database eller lokal lagring på noen som helst måte. Det kommer fra RSS strømmen til den gjeldende websiden. Hvilke muligheter man har med RSS må kartlegges. For å finne ut av dette listes det opp nedenfor hvilke elementer de mest populære publiseringsløsningene tilbyr som standard av RSS informasjon. Det er de tre elementene Title, Link og Description som er påkrevd (Crane, Pascarello, James, 2006), men publiseringsverktøy har ofte tilpasset dette delvis selv. For å finne ut hva de forskjellige systemene bruker av elementer er det foretatt empiriske undersøkelser for å finne ut om det faktisk er forskjeller på systemene.

Fig. 2.b. - Oversikt over normale RSS elementer i kjente systemer

Joomla	Drupal	EZ Publish	Wordpress	Felles elementer	Løsnigenes elementer
title link description guid author category pubDate	title link description comments dc:creator pubDate	title link description pubDate	title link description	title link description	title link description author* pubDate*

* = elementer som tas med hvis RSS strømmen kommer fra Joomla!

Ved bruk av disse standardelementene som ligger i RSS filene ser man at man ikke vil kunne få hele artiklene kun ut ifra RSS strømmer. På grunn av at løsningen skal være fleksibel i forhold til eksterne publiseringssystemer vil dette være mulighetene når det gjelder å hente informasjon.

Selv om man ikke kan dra all informasjon tilhørende en artikkel vil man få et overblikk over hva artikkelen inneholder. Dette kan også skape nysgjerrighet slik at man som oftest besøker den ordinære websiden i tillegg. Dette gagner både oppdragsgiver og annonsører fordi man også får besøk på den ordinære websiden. Dette er viktig for en nettavis som baserer seg på annonseinntekter.

3. Utførelse

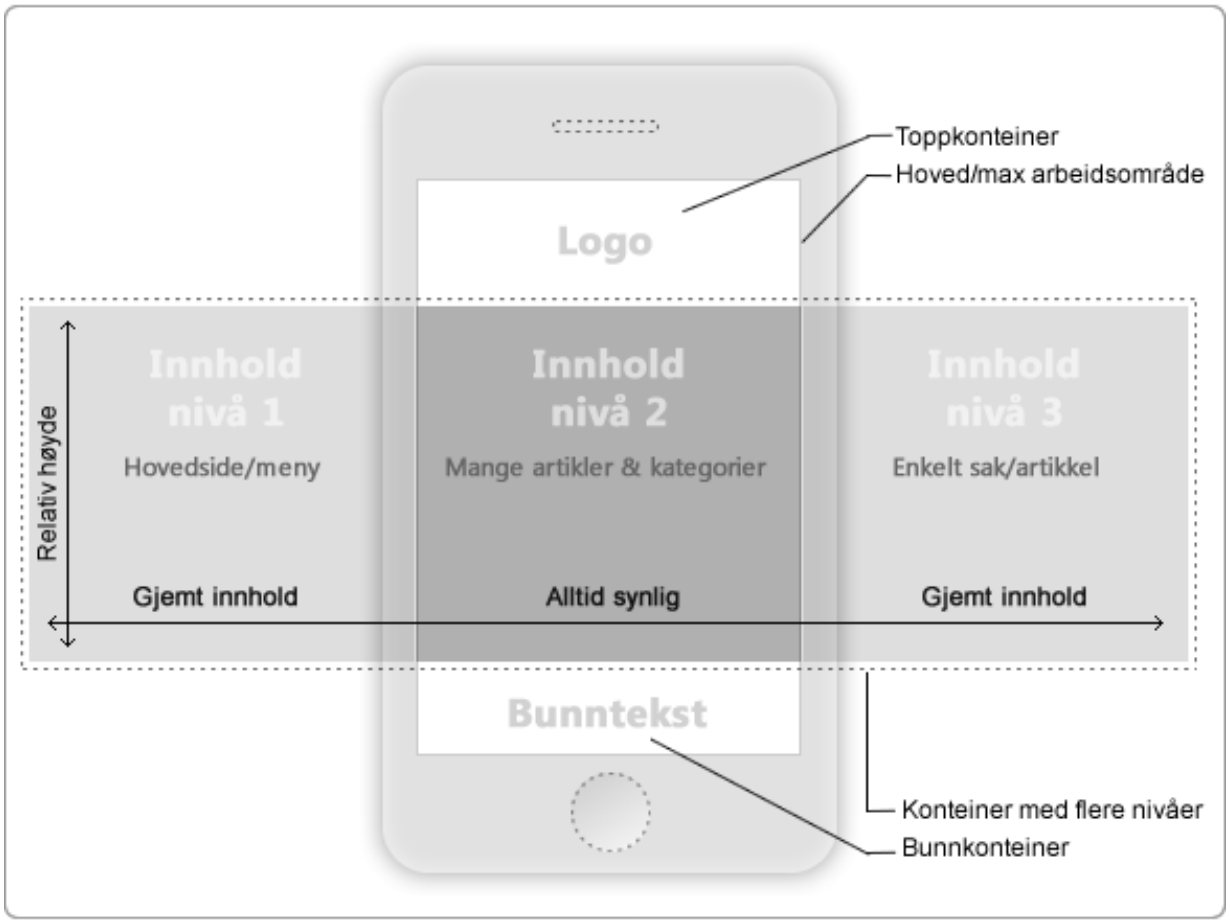
Utførelsen er av teknisk art og herunder tas det opp konkrete aspekter ved programmeringen og det som har med selve løsningen å gjøre.

3.1. Fremgangsmåte

For å skape denne løsningen så godt tilnærmet en iPhone applikasjon som mulig har det vært viktig å lage noen illustrasjoner som viser hvordan systemet kommer til å se ut. Dette hjelper med

planlegging av oppmerking av HTML og CSS for den visuelle delen. Illustrasjonen, Fig. 3.a. er relativt enkel, men utfordringen er hvordan dette i praksis vil fungere.

Fig. 3.a - Idé om hvordan løsningen skal fungere



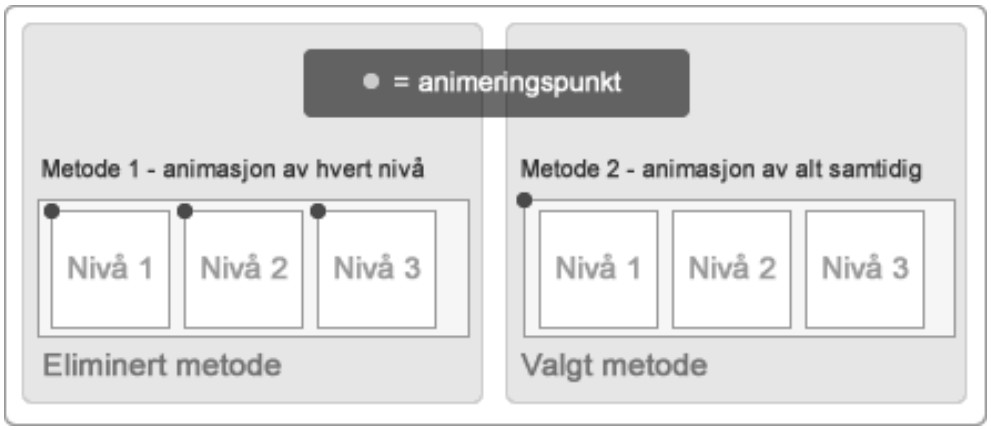
Denne konseptuelle illustrasjonen viser at websiden kommer til å deles opp i tre hoveddeler – Toppkonteiner, innholdskonteiner og bunnkonteiner. Topp- og bunnkonteiner er statiske elementer som ikke skal endre innhold. Innholds(hovedkonteineren) er elementet som all dynamisk funksjonalitet og innhold skal ha base. Denne konteineren skal animeres horisontalt ved navigering. Når animasjonen er fullført skal innhold lastes dynamisk ved hjelp av teknikken Ajax(Asynkron JavaScript og XML), som gjør det mulig å laste eksternt innhold uten å oppdatere websiden (Crane, D. Pascarello, E. James, D. 2006). Dette gir systemet tilgang til å laste innhold uten å gjenoppfriske websiden på tradisjonell måte. Dette er effektivt fordi man ikke laster hele siden på nytt, bare de elementer som faktisk er valgt av brukeren og dermed skal endre innhold.

Dette er fordelaktig ettersom målet er å skape en mest mulig effektiv og hurtig webside ved nedlastning av innhold.

3.2. Horisontal navigasjon

Den horisontale navigeringsmekanismen ble det brukt en del tid på å finne ut av hvordan det skulle oppnås. Hva som er mest hensiktsmessig og hva Safari for iPhone er i stand til å yte i forhold til animasjoner måtte undersøkes. Det ble utført en del tester med nettleseranimasjon på Safari for iPhone tidlig i utviklingsprosessen. Det viste seg at iPhone 3G har merkbart dårligere ytelse i forhold til den nyere versjonen iPhone 3Gs. Uansett er ikke ytelsen optimal på en iPhone. Tanken i begynnelsen var å ha flere forskjellige elementer som ble animert ved navigeringen slik at hvert innholdselement selvstendig ble animert. Dette skapte ytelsesproblemer. Med en struktur på tre nivåer ville det si tre elementer som animeres samtidig og dette bruker mye ressurser. Se Fig. 3.b, metode 1.

Fig. 3.b – Konseptuelle animeringsmetoder



Den ideelle metoden ble da å animere foreldreelementet i DOM-strukturen(dokument objekt modellen) innholdselementene ligger i. DOM strukturen er til for at man skal få tak i de elementene man vil på en logisk måte(Pfaffenberger, B. Schafer, S. White, C. Karow, B. 2004). Dette reduserer bruk av internminne og prosessorforbruk med 1/3 ved animeringen. Med denne metoden fungerte navigasjonen relativt godt.

Ettersom denne løsningen skulle bruke Javascript biblioteket jQuery var det mest hensiktsmessige å lage en såkalt ”plugin” til animeringen. Dette er et selvstendig skript, eller programtillegg som kan konfigureres ved kall på skriptet. Man kan på mange måter sammenligne dette med objektorientert struktur der man har en klasse med et tilhørende objekt. Nedenfor vises det hvordan dette er gjort. Skriptet er et utdrag fra det som brukes i løsningen. For å se det fullstendige scriptet som brukes i løsningen - se vedlegg A.

For å få dette skriptet til å fungere måtte skriptet kjøres umiddelbart når websiden er ferdig lastet. Dette gjøres med scriptet nedenfor:

```
jQuery(document).ready(function(){

    // plugin-kallet kommer her

});
```

En plugin må kalles på for at den skal fungerer. En plugin er på en måte som en vanlig funksjon og vil ikke ha noen funksjonell betydning så lenge man ikke kaller på den. Nedenfor vises plugin-kallet med konfigurasjonsvariabler. Konfigurasjonsvariablenes tilstedeværelse gjør at det er enklere for fremtidige teknikere å behandle funksjonaliteten, selv om dette ikke er påkrevd av systemet.

```
...
$('#container').appnav({
    loader_cont : '', // element(id) som tilhører loading indikatoren
    loader_gif : 'loader.gif', // filnavn på gif loader
    slider_speed : 300, // animasjonshastighet i millisekunder
    cont_width : 320 // vidde per innholdsnivå i piksler
});
...
```

Selve skriptet ligger i en egen fil. Det er her alle prosesser relatert til den horisontale animerte navigasjonen gjøres. Dette skriptet laster også inn RSS data med Ajax etter navigasjonen er fullført.

For å få en struktur på hvordan dette skjer må det forklares i detalj. Det som skjer når man trykker på et menyelement er at et Javascript aktiveres, altså skriptet kalles på og navigasjonen starter. Vanligvis i tradisjonell Javascript programmering ville man lagt til et attributt (event handler) som heter "onclick" på HTML menylenken for å kommunisere med Javascriptet. Dette er vist nedenfor.

```
<a href="#" onclick="javascript:appnav(); return false;"> lenkenavn </a>
```

Når man bruker jQuery kan man lett gjøre det samme på andre måter. Jeg har valgt å bruke et attributt som utløser scriptet. Et ankerelement(anchor, eng.) kan ha flere attributter med verdier. Mange av disse har en gitt hensikt til hvert sitt bruk, men enkelte attributter er mer eller mindre blitt borte i dagens bruk. For eksempel attributtet "rel" som enda er et godkjent attributt av W3C(world wide web consortium)[22], men har liten pragmatisk hensikt i dag. I utgangspunktet skal dette attributtet beskrive forholdet mellom dokumentet man er i og dokumentet som det pekes til. I dag vil de fleste nettlesere ikke bruke dette til noe bortsett i fra søkemotorer som kan dra nytte av informasjon i forhold til en lenke, men dette er ikke utbredt bland søkemotorene. Derfor kan man si at "rel" attributtet er ledig til. Med dette betyr det at "rel" attributtet kan brukes som en utløsende faktor ved bruk av Javascript og eventuelle parametre eller data kan legges i "href" attributtet som tradisjonelt brukes til å peke til den siden man skal til. "href" attributtet vil inneholde data og all data blir splittet opp med to stk kolon(::) som gjør at man kan kontrollere dataen som en tradisjonell spørrestreng(query string, eng.).

```
...
<a
    rel="nav"
    href="
        #level-1::
        fw::
        http://eksempel.com/feed=rss::
        mixed::
        2
">
...
```

[22] http://www.w3.org/TR/html401/struct/links.html

For å få javascriptet til å aktiveres på dette elementet når man trykker på lenken er spesielt. Javascriptet fungerer slik at hvis elementet man trykker på har en attributt som heter "rel" og som inneholder verdien "nav", som er en forhåndsbestemt verdi, så kjøres den valgte javascriptfunksjonen som inneholder skriptet som starter navigeringsfunksjonen. Dette er vist nedenfor.

```
...
$("rel[*=nav]").click(function(){

    // javascript kode...

});
...
```

Når man har trykket på elementet til navigeringen er det en hel del som må forberedes i skriptet før den horisontale navigeringen kan starte. Attributtet "href" inneholder all data som trengs for at navigasjonen skal fungere, men det må deles opp. Dette deles opp med to stk kolon og dette lar seg gjøre ved å bruke javascriptets innebygde funksjon "split()". Nå konverteres hele "href" verdien til en array av data.

```
...
href_array = str.split('::');
...
```

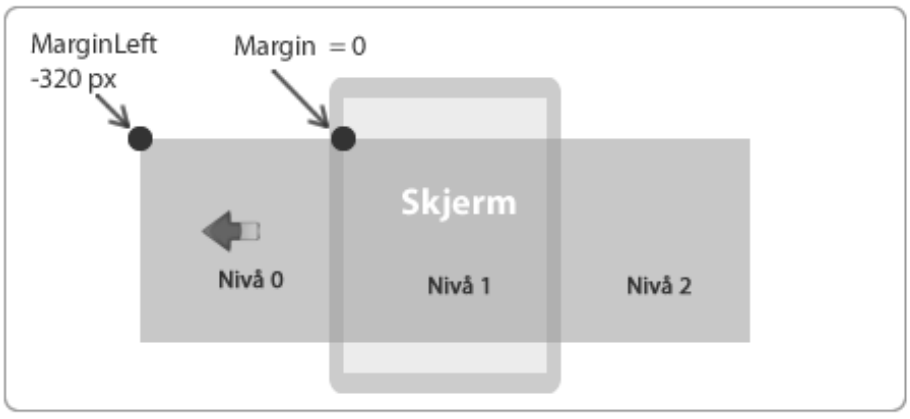
Første del av "href" verdien beskriver hvilket element man skal navigere til. Startelementet heter "level-0", derfor er det naturlig at man vil gå til "level-1. Andre del beskriver hvilken vei man vil at animasjonen skal gå. FW betyr forward og BW betyr backward. Tredje del innholder URL til RSS fil som skal vises og fjerde del innholder data om hvilken presentasjonstype man vil bruke. 1 = bare tekst, 2 = tekst og bilde 3 = bare bilde. Mer om disse presentasjonsmodellene kan de leses mer om i kapitelet "Innhenting av RSS data og presentasjon". På dette tidspunktet er det klart for at animasjonen kan starte.

Selve animasjonen er en blanding av Javascript og CSS. jQuery har en innebygd funksjon som heter "animate()"[23] som tar for seg ønsket animasjon. For å få et element til å animeres til en side brukes CSS funksjonen "margin" i form av javascript. For å finne ut hvor langt animasjonen skal gå har man en variabel med en standardverdi på 320 som er størrelsen på skjermen. Er man da på "level-0 og skal til "level-1" bruker man denne nivåbeskrivelsen til å finne ut animasjonslengden. Akkurat som nevnt ovenfor brukes javascriptfunksjonen "split()"[24] til å dele første array-elementet så man får en totalt numerisk nivåverdi.

```
...
slide_value = href_array[0].split('-'); // splitter eks.: level-1
...
```

Med dette ganges nivået med variabelen som har verdien for vidden på skjermen. Skal man da til nivå 1 blir regnestykket 1 * 320. Det vil si at man skal animere elementet til den har en verdi på -320 piksler. Er man da på nivå 1 og skal til nivå 2 blir regnestykket 2 * 320. Altså man kommer til å navigere til -640 piksler. Minus brukes for å kontrollerer animasjonsretningen. MarginLeft vil da alltid ble en minusverdi så lenge man ikke er på startsiden. Dette visualiseres i Fig. 3.c.

Fig. 3.c – Visualisering av navigeringsstruktur



[23] http://api.jquery.com/animate/
[24] http://www.w3schools.com/jsref/jsref_split.asp

For å animere dette må man finne elementet man skal bruke. Nedenfor vises dette og det som skjer er at man bestemmer at hovedelementets identifikator, id eller klasse, skal benytte seg av jQuery sin innebygde funksjon "animate()" og animere elementet til venstre.

```
...
$( ' hovedelementets id/klasse ' ).animate( {

    marginLeft: "-" + (slide_value * 320) + "px",

}, slider_speed /* animeringshastighet */ , function(){

    $(' body ').animate({

        scrollTop : 0

    }, slider_speed / 2, function(){

        // ajax forspørsel

    });

});
...
```

Når dette er gjort får skriptet en ny oppgave og det er og alltid animere seg til toppen av siden. Dette gjøres med samme metode som den horisontale animeringen. Når dette også er fullført gjenstår siste oppgave og det er å kjøre en Ajax forespørsel for å laste inn innholdet.

Det er et PHP skript som tar seg av prosesseringen av RSS data, men Ajax forespørselen henter den inn og presenterer det. jQuery har forenklet denne metoden i forhold til tradisjonell Javascript. jQuery har gjort dette godt forståelig og funksjonen som skal kalles på heter "ajax()"[25]. Inne i denne funksjonen bestemmer man hvilken måte man vil sende data på, URL som er filen man skal sende denne dataen til og variabler man vil sende med. Denne kalles "data". Ettersom i dette skriptet sender med metoden "GET" legger man til relevante data i en spørrestreng.

[25] http://api.jquery.com/jQuery.ajax/

```
...
$.ajax({

    type    : // sendingsmetoden GET,
    url     : // fil man skal sende data til,
    data    : // relevante variabler,

    success: function( response ){

        $(' /* element hvor man skal vise innholdet */ ').html( response );

    },

    error: function(response){

        alert( /* feilmelding */ );

    }

});
...
```

Dette illustrerer i hovedsak hvordan den horisontale navigeringen fungerer. Skriptet er fleksibelt og kan lett brukes i andre systemer. Skriptet er selvsagt laget for dette systemet, men et konstant fokus er at alt skal lett kunne konfigureres og endres i etterkant enten i dette systemet eller andre.

3.2.1. Javascript komprimering

Det skriptet som brukes i websiden i dag vises ikke med den struktur som vises ovenfor. For å effektivisere ytelsen på alle måter kan man komprimere javascriptkode. Dette kalles for minifisert-, eller minified kode som i realiteten er komprimert kode. En av de mest kjente metodene for å gjøre dette er Packer(Dean Edwards) komprimering[26]. Denne metoden fungerer ikke alltid med jQuery, men det finnes andre der ute som gjør en tilsvarende jobb. Dean Edward metoden komprimerer koden kraftig og dette kan i tilfeller med store skript resultere i at nettleseren dekomprimerer koden sakte og prestasjonen blir dårligere enn med ukomprimert

[26] http://jscompress.com/

skript (Wagner, 2010). Dette er noe av grunnen til at en annen kompressor er valgt i dette prosjektet. Dette prosjektet benytter seg av JS Minifier[27]. Målet er minst mulig filer, men med best mulig prestasjon og denne komprimeringsmetoden reduserer størrelsen betraktelig. I hovedkonfigurasjonen til systemet(config.php) kan man imidlertid endre en variabel for å velge om man vil bruke komprimert Javascript eller ikke. Med dette tiltaket har filstørrelse på skriptet som er illustrert ovenfor, blitt redusert fra 4450 byte til 2727. Selv om skriptet i utgangspunktet er relativt lite av størrelse, og ville sannsynligvis i praksis hatt mindre betydning, så illustrerer dette hvor hensiktsmessig en slik komprimering kan være er for å effektivisere prestasjonen.

Fig. 3.d. - Tabelloversikt over komprimert Javascript kode

Filnavn	**Ukomprimert (byte)**	**Komprimert (byte)**
jQuery-1.3.2.js	120619	57254
Ajax_tabs.js	2190	1623
Appnav.js	4450	2727
System.js	3420	2662

3.2.2. Designprinsipper

Selve strukturen er ikke noe nytt fordi Apple allerede har denne typen struktur på sine iPhone applikasjoner, men det er nytt i forhold til normale webapplikasjoner. Ved å strukturere websiden på denne måten oppnår man også oppfyllelse av enkelte designprinsipper.

Mange bruker i dag iPhone og når de betrakter denne webapplikasjonen vil de få et inntrykk av hva som kan komme til å skje når man for eksempel skal navigere. Det meste ligner på en iPhone applikasjon og prinsippet med gjenkjennelse av strukturelle elementer viser seg(Fig. 3.e.). Webapplikasjonens template/mal er også satt opp slik at man ikke skal være i tvil hva som er navigasjonsknapper og linker. Dette oppfyller også prinsippet om synlighet(Fig. 3.e.). Dette bidrar til at sluttbrukeren får en minimal læringskurve for å bruke systemet, og neste gang de skal besøke siden vet de hvordan det fungerer. Når man første gang ser det grafiske grensesnittet er

[27] http://fmarcia.info/jsmin/test.html

det nærliggende å tro at en person som allerede er bruker av en iPhone umiddelbart kunne forutse hva som kommer til å skje ved navigering(Benyon, Turner, 2005).

Fig 3.e. - Skjermbilde av det visuelle uttrykket - følger designprinspipper

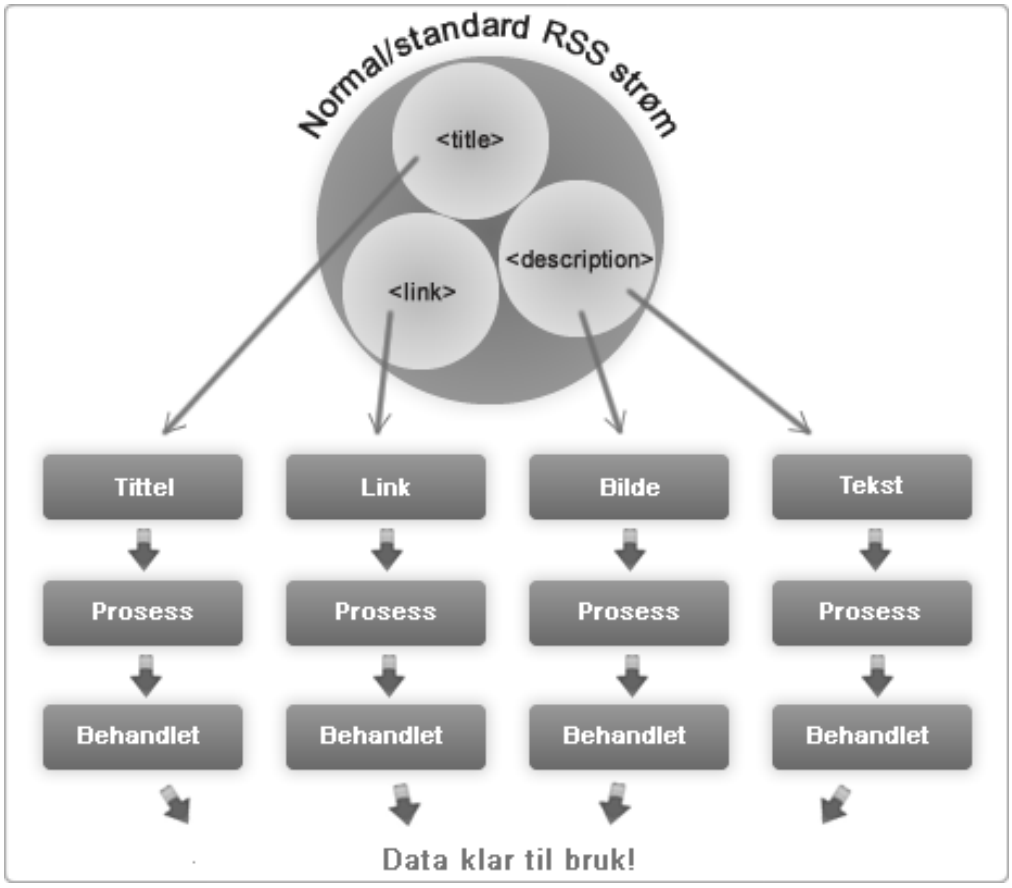
3.3. Innhenting av RSS data og presentasjon

Systemet baserer seg på RSS og inneholder derfor ikke selvstendig data. Dette er strømmer med data i form av XML som opprinnelig ligger på serveren til oppdraggiver(Pfaffenberger, B. Schafer, S. White, C. Karow, B. 2004). Vi bruker et fleksibelt PHP skript for å hente ut denne datastrømmen.

Problemet med RSS strømmer er at dataene i XML elementene, spesielt elementet ”description”, er av HTML struktur og innholdet er en skog av HTML elementer og data. Noe av målet med tanke på programmeringen er å effektivisere og kontrollere hvordan denne type data skal

presenteres for sluttbrukeren. Har man mulighet til å oppnå dette er det lettere å utvikle en presentasjonsmal med en struktur som er uavhengig av den originale strukturen. Med PHP har det latt seg gjøre å systematisere denne skogen av data. Fig. 3.f. illustrerer i form av et enkelt diagram oppgaven PHP skriptet har.

Fig 3.f. - Oversiktsdiagram



For å gå i dybden på hvordan dette faktisk gjøres vises det nedenfor et utdrag fra PHP skriptet som fanger opp RSS data og systematiserer det. Dette er bare utdrag fra den originale PHP klassen for å beskrive hva som skjer og hvorfor det er gjort på denne måten. Noe av koden er organisert litt annerledes enn i det fungerende skriptet og der er av estetiske grunner i forhold til forklaringen og formatet. For komplett PHP script se vedlegg B.

PHP klassen "load_content" har en konstruktørfunksjon som klargjør og initierer alle relevante variabler som skal brukes senere i skriptet. Som beskrevet i kapittelet ovenfor blir en rekke variabler sendt til dette scriptet i form av teknikken Ajax. Disse variablene ble sendt via GET metoden og konstruktøren fanger opp disse variablene ved bruk av samme type metode. Variabler som klargjøres er RSS strengen og presentasjonsmalen som skal brukes.

```
$_GET['variabel'];
```

En annen oppgave konstruktøren har er å klargjøre DOMDocument-objektet slik at man senere kan lese en RSS/XML fil. XML har DOM strukturen og derfor er dette riktig metode å bruke. Dette objektet brukes når RSS adressen skal lastes inn.

```
$this->xmlDoc = new DOMDocument();
$this->xmlDoc->load( $_GET[' /* RSS adressen */ '] );
```

Når alle variabler og objekter i konstruktøren er initiert kalles funksjoner som systematiserer og klargjør koden for presentasjon. PHP klassen har en hovedfunksjon, "output()", som fanger opp data fra RSS elementene. Denne funksjonen fanger som standard opp de tre påkrevde elementene i følge W3C - title, link, description[28]. Med disse elementene har man tilgang på det viktigste av data. En RSS strøm må ha en tittel, må kunne lenkes tilbake til den opprinnelige artikkelen og man må ha ingressen eller beskrivelsens av artikkelen. Dette går da gjennom en for-løkke som skriver ut et gitt antall elementer og sender dataen til andre funksjoner som tar seg av hvordan dette skal presenteres. Standard for systemet er et antall på 10 elementer. Dette er også ofte standardinnstillinger for RSS skript i publiseringsløsninger, men systemet kan konfigureres slik at man kan vise flere eller færre elementer.

```
for ($i = 0; $i < self::max_amount_of_articles; $i++){

        // Henter verdiene i elementene i rss filen
        $this->item_title = $element->item($i)->getElementsByTagName('title')
        ->item(0)->childNodes->item(0)->nodeValue;
```

[28] http://validator.w3.org/feed/docs/rss2.html

```
        $this->item_link = $element->item($i)->getElementsByTagName('link')
        ->item(0)->childNodes->item(0)->nodeValue;

        $this->item_desc = $element-
>item($i)>getElementsByTagName('description')
        ->item(0)->childNodes->item(0)->nodeValue;

        // Forskjellig presentasjonsoppsett
        if( $this->content_view == 1 ){

            $out .= $this->content_view_text();

        }else if( $this->content_view == 2 ){

            $out .= $this->content_view_mixed();

        }else if( $this->content_view == 3 ){

            $out .= $this->content_view_img();

        }
    }
```

Klassen setter også opp forskjellige presentasjonsmaler. Har en RSS strøm få bildeelementer kan administrator velge å sette opp visningen bare som tekst eller hvis det ofte er bildeelementer i RSS strømmen kan administrator velge bilder med tekst. Hvis man vil vise strømmen som et bildegalleri har man også muligheter for dette. Nedenfor vises de funksjoner som lager HTML oppmerking til de forskjellige presentasjonsmalene.

```
...
/* Mal for elementer med bare tekst */
public function content_view_text(){

    $html .= '<div class=\'items_text\'>';
    $html .= '<a rel=\'nav\' id=\''.$this->item_link .'\'
              href=\'#level-2::fw::'.$this->url.'\'>';
    $html .= '<div class=\'item_text_info\'>';
    $html .= '<div class=\'item_text_title\'>
```

```
            '.$this->prepare_title( $this->item_title , 25).'</div>';


    $html .= '</div>';
    $html .= '<div class=\'item_text_arrow\'>';
    $html .= '</div>';
    $html .= '<div class=\'clear\'></div>';
    $html .= '</a>';
    $html .= '</div>';
    return $html;
}

/* Mal for elementer med både bilde og text */
public function content_view_mixed(){

    $html .= '<div class=\'items_mixed\'>';
    $html .= '<a rel=\'nav\' id="' .$this->item_link .'"
              href=\'#level-2::fw::'.$this->url.'\'>';
    $html .= '<div class=\'item_mixed_image\'>';
    $html .= '<img src=\''.$this->item_img_path.'\' '.
             $this->image_size( $this->item_img_path ).' alt=\'bilde\' />';
    $html .= '</div>';
    $html .= '<div class=\'item_mixed_info\'>';
    $html .= '<div class=\'item_mixed_title\'>'.
             $this->prepare_title( $this->item_title, 20 ).'</div>';
    $html .= '<div class=\'item_mixed_desc\'>
             '.$this->prepare_text( $this->item_desc , 60).'</div>';
    $html .= '</div>';
    $html .= '<div class=\'item_mixed_arrow\'>';
    $html .= '</div>';
    $html .= '<div class=\'clear\'></div>';
    $html .= '</a>';
    $html .= '</div>';
    return $html;
}

public function content_view_img(){
```

```
    $html .= '<div class=\'items_img\'>';
    $html .= '<a rel=\'nav\' id="' .$this->item_link .'"
             href=\'#level-2::fw::'.$this->url.'\'>';
    $html .= '<div class=\'item_img_src\'>';
    $html .= '<img src=\''.$this->item_img_path.'\' '.
             $this->image_size( $this->item_img_path ).' alt=\'bilde\' />';
    $html .= '</div>';
    $html .= '<div class=\'clear\'></div>';
    $html .= '</a>';
    $html .= '</div>';
    return $html;
}
...
```

Ser man på disse funksjonene som lager HTML for presentasjonen ser man at noen ekstra funksjoner kalles. Dette er funksjoner som behandler rådata. Enkelte ganger vil man presentere datoformatet på en spesiell måte eller endre lengde på tittelen og lignende. En rekk funksjoner er laget med slike hensikter.

En av de viktigste behandlingsfunksjonene er funksjonen som fjerner alle HTML elementer fra råkoden. Dette gjøres relativt enkelt ved å bruke en innebygd funksjon i PHP som heter "strip_tags()"[29]. Denne funksjonen brukes hovedsakelig med data som kommer fra "description" elementet. I tillegg til å fjerne all HTML kode vil man også kontrollere lengden på teksten. Denne funksjonen gjør begge deler og man kontrollerer dette ved å sende med ønskede parametre.

```
private function prepare_text($src, $length){

    $fixed_text = strip_tags($src);
    $dots = ( strlen( $fixed_text ) > $length) ? '...': null;
    if($length == 0){
        return $fixed_text;
    }else{
        return substr($fixed_text, 0, $length).$dots;
```

[29] http://php.net/manual/en/function.strip-tags.php

```
    }
}
```

En annen behandlingsfunksjon som er viktig idet man vil rydde opp i denne "skogen av data" er å dra ut et bildeelement. Ofte har man i ingressen på en artikkel et bilde. Dette er blitt vanlig i dag. For å finne dette bilde og fjerne alt annet benyttes et regulært uttrykk som har den egenskap at den leter etter et <img /> element i råkoden og finner innholdet i attributtet "src" for å få hele filbanen til dette bildet.

```
private function prepare_image( $src ){

    $pattern = '/<img[^>]+src[\\s=\'"]+([^"\'>\\s]+)/is';

    if( preg_match( $pattern, $src, $match ) ){
        return $match[1];
    }else{
        return self::default_image;
    }
}
```

Med disse to behandlingsfunksjonene er det oppnådd en separasjon av artikkelens ingress. Bildet er skilt fra teksten og teksten er skilt fra alle andre elementer. Dette er nå klart for å brukes i presentasjonsmalene. Dette gjør at det er mulighet for å kontrollere innholdet når man lager HTML for malene.

Som nevnt ovenfor har systemet i dag tre forskjellige presentasjonsmaler. Dette gjelder for nivå 1 i systemet. Dette er nivået hvor alle artikler fra en spesifikk kategori eller menyelement vises. Det neste nivået er der hvor man presenterer kun én artikkel - en valgt artikkel ut ifra utvalget i nivå 1.

Det er enkelte ganger ønsket å bruke utvidede elementer for å gjøre artikkelen mer fyldig når man skal vise den spesifikke artikkelen. Presentasjonen av dette har en test som legger til to ekstra elementer hvis RSS dataen kommer fra publiseringsplattformen Joomla. Joomla merker alltid sine filer med sitt navn og dette drar man nytte av i dette scriptet. Det som skjer er at man leser RSS filen og hvis det finnes et metadataelement som heter "generator" og verdien i dette

elementet starter med "Joomla" så tilgjengeliggjør man RSS elementene "author" og "pubDate". Med dette kan forfatter av artikkelen og dato da artikkelen ble publisert brukes i presentasjonen. Nedenfor vises et utdrag av funksjonen som lager HTML oppmerking av den spesifikke artikkelen.

```
public function single_content(){

    $element = $this->xmlDoc->getElementsByTagName('item');
    for($i = 0; $i < 10 ; $i++){

      $item_title = $element->item($i)->getElementsByTagName('title')
      ->item(0)->childNodes->item(0)->nodeValue;

    $item_link = $element->item($i)->getElementsByTagName('link')
    ->item(0)->childNodes->item(0)->nodeValue;

      $item_desc = $element->item($i)->getElementsByTagName('description')
    ->item(0)->childNodes->item(0)->nodeValue;

    $item_img_path = $this->prepare_image( $item_desc );

    /* Hvis pekeren til artikkelen stemmer med den som ble sendt
    med i ajax forespørselen så kjøres denne (# fungerer ikke så vi tar bort
    denne)*/
    $sendt_url = explode('#', $item_link );

    /* Hvis mottatt lenke er lik linke i RSS filen så skal
    dette elementet vises */
    if( $sendt_url[0] == $this->art_url ){

        $html = ''; // output variabel

        /* Leser selve RSS filen for å finne identifikatorer */
        $handle = fopen( $this->url, "rb" );
        $contents = stream_get_contents( $handle );

        // Hvis RSS er joomla - tar vi med dato for forfatter
        if( strpos( $contents, '<generator>Joomla' ) !== false ){
```

```
    $item_pub = $element->item($i)->getElementsByTagName('pubDate')
    ->item(0)->childNodes->item(0)->nodeValue;

    $item_author = $element->item($i)->getElementsByTagName('author')
    ->item(0)->childNodes->item(0)->nodeValue;

    $date_and_author = '<div class=\'items_single_pub\'>
    '.$this->init->language('OPEN', 'GEN_WRITTEN_BY').',
    '.$item_author.' '.$this->prepare_date( $item_pub ).'</div>';

// hvis ikke - ikke ta med noen ekstra felt
}else{

    $date_and_author = '';

}

fclose($handle);

$html .= '<div class=\'items_single\'>';
$html .= '<div class=\'items_single_title\'>'.$item_title.'</div>';
$html .= $date_and_author;

if( $item_img_path != self::default_image ){

    $html .= '<img
        src=\''.$item_img_path.'\'
        width="280px" alt=\'bilde\' />';

}

$html .= '<p>'. strip_tags( $item_desc ).'</p>';
$html .= '<div class=\'clear\'></div>';
$html .= '<div class=\'items_single_link\'>';
$html .= '<span class="item_links item_links_left">
         <a href="'.$item_link.'" target="_blank">
         '.$this->init->language('OPEN',GEN_VIEW_COMPLETE_ART').'
         </a></span>';
$html .= '<span class="item_links item_links_right">
         <a rel="fb_share" href="'.$item_link.'::'.$item_title.'">
```

```
                '.$this->init->language('OPEN', GEN_FB_SHARE').'</a>
                </span>';

        $html .= '<div class=\'clear\'></div>';
        $html .= '</div>';
        $html .= '<blockquote>
                "'.$this->init->language('OPEN', GEN_ART_INFO').'"
                </blockquote>';
        $html .= '</div>';
        }
    }
    return $html;
}
```

Helhetlig gjør denne klassen det mulig å kontrollere strømmen av data RSS filene tilbyr. Skriptet er laget på en måte som gjør at det uvesentlig hvilken type RSS strøm man bruker. Uansett om man bruker RSS strømmer fra publiseringssystemer som Joomla, Wordpress, EZ Publish eller andre systemer vil man alltid kunne kontrollere og presentere innholdet på valgt måte.

3.4. Språkfiler

En webløsning som tar sikte på å være fleksibel og gjenbrukbar bør ha en mulighet for å bruke flere enn ett språk. Hvis dette systemet i fremtiden skal brukes av andre enn oppdragsgiver så vil det å kunne endre språk være fordelaktig.

Språkfiler kan lages på mange forskjellige måter og mange bruker XML formatet til dette. Det brukes også i mange tilfeller PHP filer med definerte konstanter. Dette er gode metoder å benytte seg av, men det finnes også andre alternativer.

Det er både fordeler og bakdeler med språkfiltypene XML og PHP. En PHP fil med konstanter er fordelaktig på den måten at du ikke trenger et komplisert script for å lese konstantene. Konstantene er globale variabler(Huge E Williams & David Lane, 2004) og kan derfor implementeres i og utenfor funksjoner og klasser. Dette taler for at det er relativt enkelt å bruke dette ved utvikling av skript, men det finnes også bakdeler med konstanter. Konstanter krever et

snev av programmeringskompetanse selv om det ikke er veldig vanskelig å endre disse konstantene. Dette er nok også den måten å lage språkfiler på som en enklest å bruke i systemet når det skapes, men fokuset her er teknikere som skal drifte det i etterkant.

Eksempel på PHP språk fil

```
<?php
define('WRITTEN_BY', 'Skrevet av');
?>
```

Alternativt til PHP konstanter er XML formatet. Dette formatet kan være fordelaktig å bruke på den måten at hvis man forstår DOM(dokument objekt modellen) strukturen er det relativt lett å lage slike filer. Dette krever da kompetanse om DOM strukturen og man må ta hensyn til elementenes struktur og derfor er det ikke sikkert at dette valget vil være fordelaktig fordi denne løsningen, da den skal være så enkel og forstå som mulig for senere utviklere. Derfor vurderes andre metoder for å lage språkfiler.

Eksempel på XML språkfil:

```
<?xml verion="1.0" ?>
<lang>
    <written_by>Skrevet av</written_by>
</lang>
```

Et annet alternativ er å bruke .ini filer som språkfiler. Ved bruk av denne type filer kan man med minimal programmeringskompetanse endre eksisterende språkfiler og lage nye. Syntaksen kan være enkel og minimal. Dette taler for at språkfilene bør bruke dette formatet ettersom det vil være enkelt for teknikere som senere skal behandle disse.

Eksempel på INI språkfil:

```
[LANG]
WRITTEN_BY = Skrevet av
```

INI-filer brukes av noen av de store webbaserte systemene som tilbyr språkpakker. Joomla for eksempel bruker dette som språkfilformat og det er grunn til å tro at de fleste som har kunnskaper

nok til å drifte en stort og komplekst system som dette også er i stand til å endre eller lage språkfiler av denne typen. Derfor falt valget på dette formatet.

Systemet har i dag to integrerte språkfiler, og det å lage nye er enkelt. Fremgangsmåte for å lage en slik fil er å navngi filen med et ISO standardisert prefiks etterfulgt av .ini suffikset som f.eks. pl.ini, se.ini, dk.ini etc.. Filen skal legges i mappen som heter "languages" og systemet fanger opp denne filen automatisk slik at man kan logge seg inn for få muligheten til å velge denne.

3.5. Maler / templates

Systemet har også støtte for maler, eller templates som det heter. Ofte bruker man templatesystemet Smarty eller andre lignende, men denne løsningen har et eget system. Dette er en del av systemarkitekturen som gjør det enkelt å endre designet for sluttbrukeren hvis dette er ønsket(Horgen, 2007). Malene er satt opp som vanlig HTML kodet filer med kall på PHP funksjoner. I tillegg til vanlig HTML oppmerking kalles PHP funksjonene med én parameter. Dette er funksjonsnavnet. Nedenfor vises noen eksempler som brukes i maler til denne løsningen.

```
<?php echo $content->load('doctype'); ?>
<html>
    <head>
        <?php echo $content->load('system_name'); ?>
        <?php echo $content->load('charset'); ?>
        <?php echo $content->load('viewport'); ?>
    </head>
    <body>
        ...
```

Dette gjør det enkelt for den som skal administrere systemet i etterkant. Man slipper å ha kunnskaper om hvordan syntaksen i HTML metadata er og man kan lett bygge HTML rundt disse funksjonskallene. For å forstå hvordan disse kallene fungere vises nedenfor et utdrag av klassen som viser hvordan funksjonene fungerer. Det er allerede opprettet et objekt av denne klassen i index filen til systemet og den er alltid tilgjengelig.

```
/* Klassen utvides med "config" slik at man har tilgang på
```

```
konfigurasjonsvariabler */
class tpl_elements extends config{

    /* Distributørfunksjon - henter de ønskede metoder/funksjoner  */
    public function load( $element_raw ){

        $element_no_blankspace =
        str_replace(' ', '', $element_raw); // hvis det forekommer mellomrom
        $element_ready =
        strtolower($element_no_blankspace);
        return $this->$element_ready(); // kaller på den rette funksjonen
    }

    /* Funksjon for dokumenttypedefinisjonen */
    public function doctype(){

        $dtd = '<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile
        1.1//EN"'."\n";
        $dtd .= '"http://www.openmobilealliance.org/tech/DTD/xhtml-
        mobile11.dtd">'."\n";
        return $dtd;
    }

    /* Funksjon for charset - eller symbolsettet som skal brukes */
    public function charset(){

        $iso = '<meta http-equiv="content-Type"
        content="text/html; charset='.parent::meta_charset.'" />'."\n";
        return $iso;
    }
    // flere funksjon er under her ...
}
```

Den eneste kunnskapen man behøver angående PHP for å lage maler/templates til dette systemet er en oversikt over hvilke funksjoner som er tilgjengelig. Nedenfor listes opp funksjoner som er tilgjengelige for utvikling av nye maler.

```
<?php echo $content->load('doctype'); ?>        // dokumenttypen
<?php echo $content->load('charset'); ?>        // meta
<?php echo $content->load('robots'); ?>         // meta
<?php echo $content->load('viewport'); ?>       // meta
<?php echo $content->load('distribution'); ?>   // meta
<?php echo $content->load('language'); ?>       // meta
<?php echo $content->load('system_name'); ?>    // websidens tittel
<?php echo $content->load('system_js'); ?>      // systemavhengig javascript
<?php echo $content->load('system_css'); ?>     // bane til malens css fil
<?php echo $content->load('main_menu'); ?>      // Hovedmenyen som liste
<?php echo $content->load('sub_menu'); ?>       // Undermenyen som liste
<?php echo $content->load('stats'); ?>          // Google analytics
<?php echo $content->load('footer'); ?>         // rettigjetsinformasjon ++
```

Dette er alle de statiske funksjonene man trenger for å lage en mal/template. Alt annet av visuelt innhold lastes inn dynamisk med Ajax og der trenger man bare å definere elementer som skal ta imot disse dataene.

3.6. Administrasjon av løsningen

Det er også tatt i betraktning at oppdragsgiver og deres ansatte skal kunne administrere innholdet på iPhone løsningen. Med tanke på at systemets fremtid skal kunne være til nytte for flere bedrifter enn oppdragsgiver, er det opprettet en egen webside for løsningen. Dette er basen for systemet og alle eventuelle bedrifter vil kunne få en nøkkelindikator som er unik for den enkelte bedrift i form av en unik URL. Man får en innloggingsside der man taster inn e-post og et passord og fra her kan man administrer menyelementer og RSS adresser.

3.6.1. Administrasjonspanel

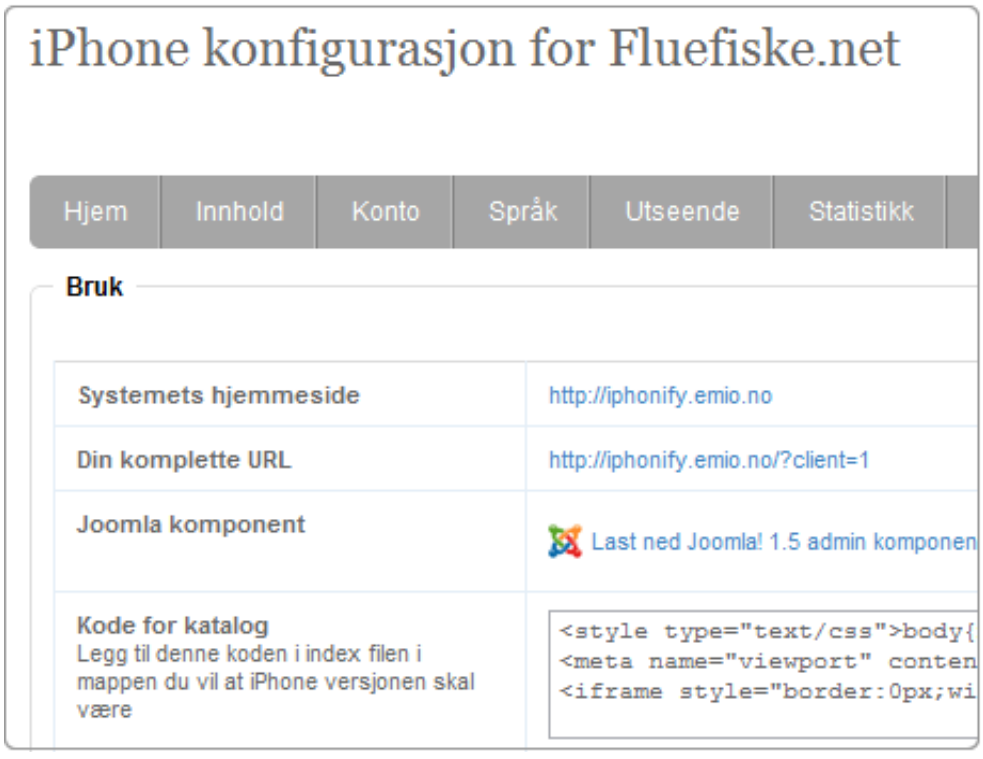
Administrasjonspanelet var ikke en opprinnelig del av prosjektet. Da prosjektet ble planlagt var hele fokuset på den delen som sluttbrukeren kan se, den tilpassede løsning for iPhone. Til å begynne med lå alle menyer og konfigurasjoner i XML filer. Med tanke på at oppdragsgivers ansatte skulle drifte dette systemet i etterkant kom vi frem til at det var litt for komplisert og endre disse filene for å endre innhold. Dette ville også forutsatt at den teknikeren som skulle

gjøre dette hadde kompetanse om XML og DOM strukturen. Det ble derfor i tillegg til det opprinnelige prosjektet lagt planer for et brukervennlig administrasjonssystem.

På dette tidspunktet ble systemet avhengig av en database, noe den opprinnelige planen heller ikke hadde. Da systemet er blitt tilpasset slik at det er støtte for flere brukere holder det ikke lengre å bruke XML filer som lagringsplass. Det er da naturlig å benytte seg av en database fordi klienttilknyttet data er relasjonsdata, og derfor er det naturlig å bruke en relasjonsdatabase som tar vare på informasjonen. Dataene som ligger i databasen er primært tilegnet administrasjonssystemet og ikke hvordan det presenteres for en besøkende bruker.

Det ble planlagt et grafisk grensesnitt til administrasjonssystemet. Grensesnittet skulle være enkelt og lett forståelig. Resultatet ble den strukturen som Fig 3.g. viser.

Fig. 3.g. - Delvis skjermbilde av administrasjonssystemet



Det finnes en bakdel med dette administrasjonssystemet og den bakdelen hviler på oppdragsgiver. Dette vil forutsette at oppdragsgiver må ha en egen innlogging til iPhone

løsningen. Dette betyr flere passord og brukernavn og huske på, men når det er sagt bruker oppdragsgiver i dag tre forskjellige systemer med tre forskjellige innlogginger, så de er vant med slike rutiner.

Fordelen er da at systemet passer for flere brukere/bedrifter og at systemet er totalt uavhengig av deres eksisterende løsninger. Med en serverkrasj på bedriftssiden vil ikke iPhone løsningen berøres. Den vil selvsagt ikke fungerer som normalt ettersom RSS strømmene kommer fra deres opprinnelige nettside, men systemets tilstand bevarer sin status.

3.6.2. Joomla administrator komponent

På den måten systemet er satt opp går man til løsningens webside og logger seg inn for å administrere innholdet. Etter som oppdragsgiver bruker publiseringsløsningen Joomla kan man installere en proprietær administrasjonskomponent. Komponenter i joomla er deler av systemarkitekturen som har sin unike funksjon for at det skal være oversiktelig for å videreutvikle systemet[30]. Denne komponenten fungerer som en standard Joomla komponent og man installerer den på vanlig måte.

Komponentens oppbygning er laget på en slik måte at innholdet ligger som i en ”iframe” som peker til løsningens webside. Ettersom denne komponenten er en tilleggspakke som lar deg administrere systemet fra Joomla er det ingen nødvendighet, bare et alternativ til å gå til løsningens webside for å gjøre akkurat det samme. Det gjør også systemet mer brukervennlig for oppdragsgiver, fordi det bidrar til å samle flere systemer på ett sted.

4. Resultat

I dette prosjektet har det oppstått enkelte problemer og hindringer i forhold til utførelsen. Nedenfor belyses noen av de viktigere elementene som har blitt berørt av problemene og hvordan dette underveis har blitt løst på forskjellige måter.

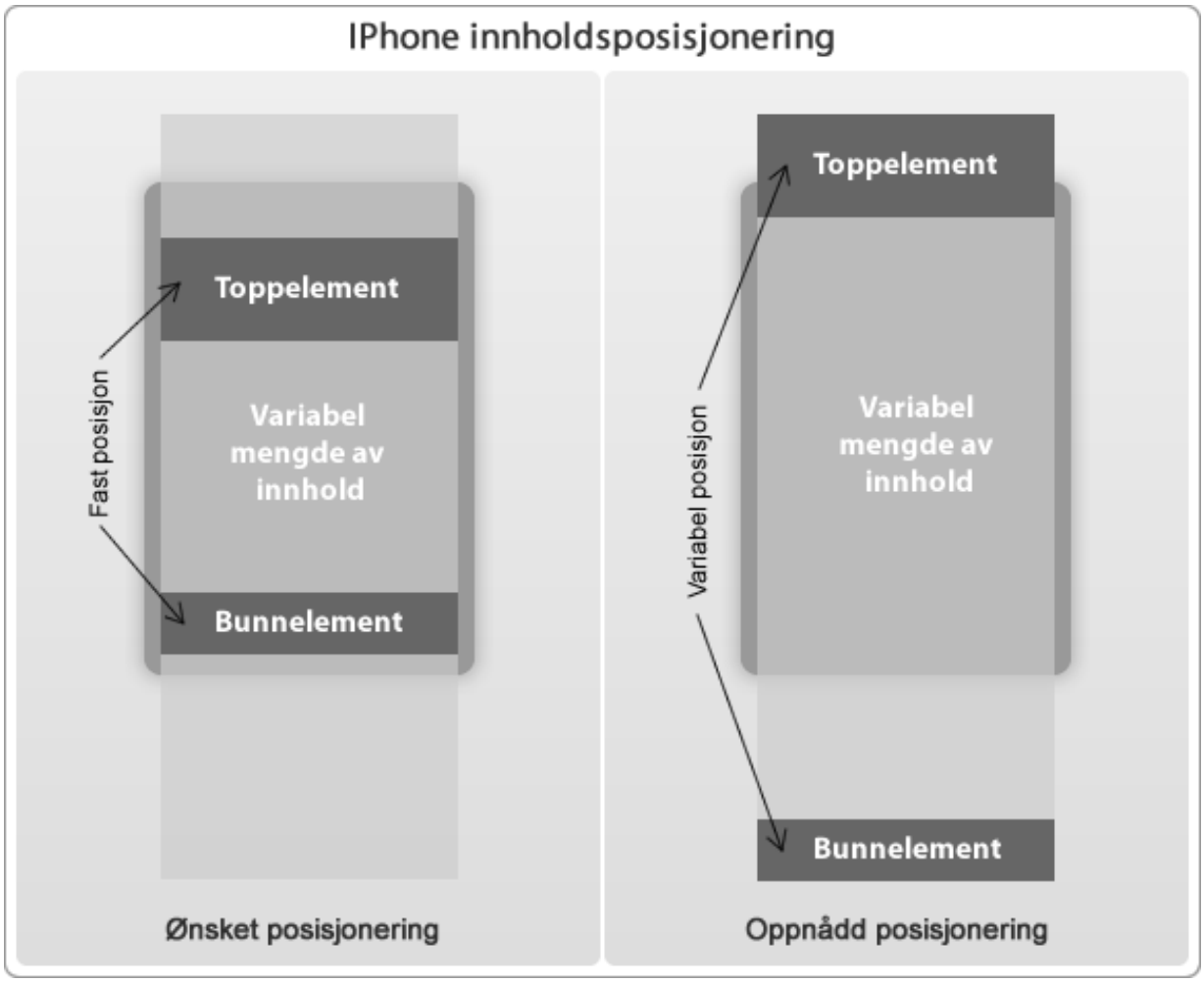
[30] http://docs.joomla.org/developing_a_model-view-controller_component_-_part_1

4.1. Det visuelle uttrykket

Dette prosjektet er ikke hovedsakelig en designoppgave, men et prosjekt som skal forholde seg til et bestemt visuelt uttrykk. Hvordan dette oppleves for sluttbrukeren er viktig. Målet var å skape en navigasjon med funksjonalitet som tilsvarer funksjonaliteten til en iPhone. Dette er på mange måter oppnådd, men noen aspekter har vært til hinder for at denne funksjonaliteten ikke er blitt helt lik. Dette dreier seg om nettleseren Safari for iPhone sin manglende ytelse. Som nevnt tidligere i oppgaven er det ytelsesforskjeller på iPhone versjonene som gjør at sluttbrukeren kan oppleve systemet som upresist og med treghet. Når man også i praksis fokuserer på et navigeringselement(holder fingeren på elementet) merker man et semitransparent område som overkjører "hover" funksjonen i CSS spesifikasjonene. Et annet hinder for at websiden ikke ble helt lik funksjonaliteten til en iPhoneapplikasjon er på grunn avmangler i CSS posisjoneringen. Her menes "position:fixed" som normalt ville gjort at toppelement og bunnelement ikke ville blitt berørt av skrolling vertikalt. Bortsett ifra dette er funksjonaliteten oppnådd i forhold til det opprinnelige målet.

Den horisontale animeringen av navigasjonen fungerer som den skal og man får en følelse av at løsningen har røtter i en iPhone applikasjon. Før prosjektet var i gang var det forventet at Safari for iPhone skulle ha full støtte for CSS spesifikasjoner, men dette problemet ble ikke oppfattet før undersøkelser ble foretatt. Litteratur ble lest og testinger ble utført. Disse problemene er imidlertid løst med alternative metoder. Et scenario der man har en lang liste med menyelementer og man skroller nedover for å så navigere til den spesifikke artikkelen vil man i utgangspunktet ikke automatisk se artikkelen fra toppen, fordi systemet animeres horisontalt, og alle innholdselementer animeres samtidig. Alternativt har systemet en 3-stegs prosess når det gjelder navigeringen. Dette er nevn i kapittelet om "horisontale navigeringsstrukturen" og steg 2 i denne prosessen er å animere websiden til toppen. Dermed vil man kunne lese den spesifikke artikkelen fra toppen selv om CSS posisjoneringen "fixed" ikke er støttet. Når dette er sagt hadde posisjoneringen "fixed" bare vært hensiktsmessig for å vise toppelementene hele tiden og ikke hatt en rolle i å få det faktiske innholdet til toppen som steg 2 i navigasjonsprosessen. Dette steget har løst problemet når det gjelder hvor innholdet befinner seg, men bare delvis hvordan toppelementene presenteres. Fig. 3.f. viser forskjellen på ønsket posisjonering i forhold til hva som faktisk ble oppnådd.

FIG. 3.f. - Illustrasjon om ønsket posisjonering



4.2. Effektivisering av nedlastningstid

En viktig side i forhold til å lage en webside for mobile enheter mot en tradisjonell webside er nedlastningsytelse. I dag er det blitt vanlig å ha relativt hurtige linjer i husstanden fra bredbåndsnettet. Dette gjør at websider lastes ned med en hastighet som i mange tilfeller gjør at det er mindre vesentlig hvor mye data websiden inneholder. Websiden kan betraktes omtrent med en gang. Men fokuset på mengden av data som skal lastes ned viser seg mer vesentlig når man benytter seg av f.eks. 3G nettet eller EDGE som mobile enheter ofte bruker der de ikke har tilgang på private nett. Spesielt i dette prosjektet, hvor oppdragsgiver drifter en stor nettavis med større mengder grafiske annonser, er dette et viktig fokus.

Tidligere i rapporten er det nevnt at det har vært viktig å skille bilder fra tekst og tekst fra bilder for å få bedre kontroll på hvordan innholdet skal presenteres. Et underliggende mål har også vært å redusere mengden med data som skal lastes ned. Dette er blitt oppnådd på grunn av denne separasjonen av tekst og bilde. iPhone versjonen inneholder heller ingen annonser, noe som bidrar til mer effektiv nedlastningstid.

4.3. Administrering av systemet

Tidlig i prosessen var det ikke et administrasjonssystem til denne løsningen et mål. Tanken var å legge menyelementer og data i XML filer, men etter hvert viste behovet seg for en enklere metode å gjøre det på. Derfor ble det planlagt et administrasjonssystem. Man kan diskutere om dette skulle vært fokusert på fra første stund, men på grunn av at det var vanskelig å forutse arbeidsmengden i forhold til det primære målet, ble dette ikke fokusert på før senere i arbeidsprosessen. Dette ble da en større del av prosjektet og målet med administrasjonssystemet ble oppnådd og det i større grad enn forventet. Til og med er det laget en enkel Joomla komponent som gagner oppdragsgiver positivt. Denne utvidelsen av prosjektet har ikke vært til hinder for det opprinnelige målet.

Dette administrasjonssystemet har også bidratt til større kompleksitet i forhold til det tekniske. I utgangspunktet var det ikke planlagt å bruke en database, men behovet viste seg idet administrasjonssystemet skulle planlegges. Større kompleksitet i forhold til programmering har da bidratt til økt læring og gitt relevant erfaring.

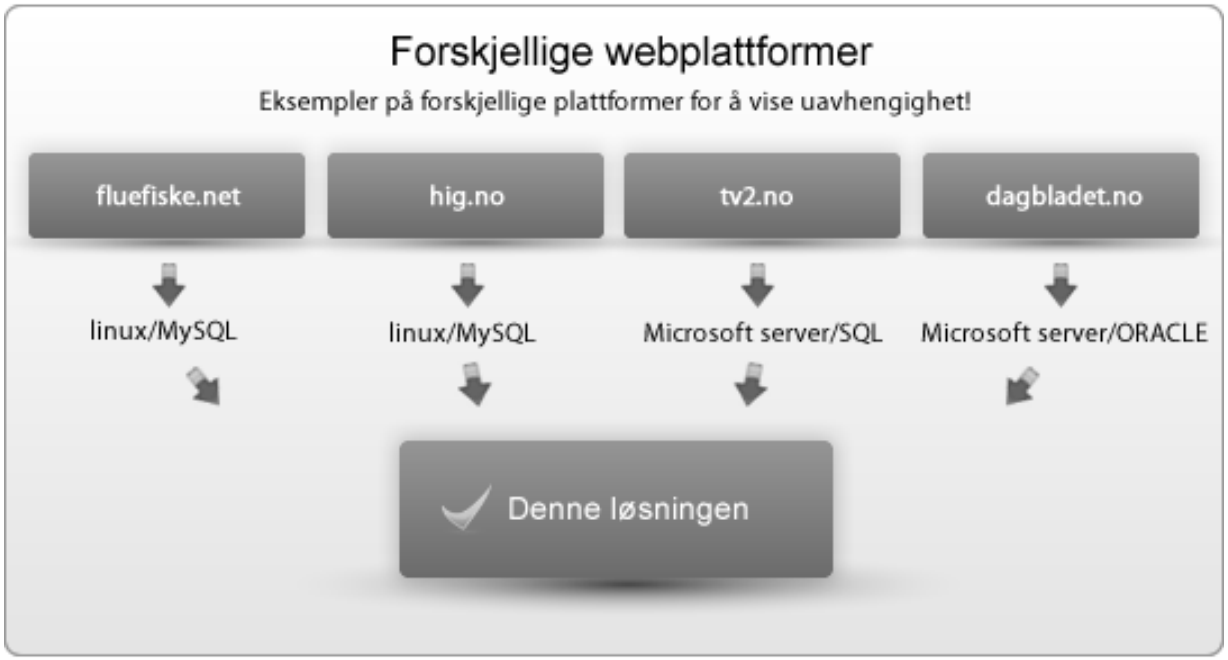
4.4. Bruksområde og kompatibilitet

I utgangspunktet var dette prosjektet ment spesielt for oppdragsgiver. Det er oppdragsgiver som kommer til å dra nytte av det, men et av målene gjennom prosessene har vært å skape dette slik at det også kan ha en nytteverdi for andre også.

Systemet ligger på en uavhengig server og derfor er det uvesentlig hvordan server man selv har til sine systemer. Har man for eksempel selv en Microsoft server med ASP serverscripting eller en Linux server med PHP installert vil ikke dette ha noe å si for iPhone-løsningen. Systemet har også mulighet for flere brukere slik at man kan opprette en konto og administrer sin egen

mobilløsning. Det eneste som kreves er at den som benytter seg av denne løsningen har mulighet for RSS strømmer som er hovedkilden til innholdet.

FIG. 3.g. – Illustrasjon om plattformuavhengighet



Dette taler for at uansett hvilken plattform man selv bruker og om man drifter en nettavis, en kommunal nettside eller et nettsted for en høgskole vil denne løsningen være til nytte. Hvilken plattform man selv bruker vil være irrelevant. Derfor er bruksområdet potensielt større enn oppdragsgiver sine nettsider, selv om at det er disse som er prioritert i dette prosjektet.

5. Avslutning

Dette prosjektet har på noen måter vært annerledes enn hvis det hele hadde handlet om veletablerte teknikker og metoder. Det har tidvis vært lite dokumentasjon, noe som har økt behovet for undersøkelser og testing av enkelte teknikker. Dette er på grunn av at iPhone er en relativ ny enhet på markedet. En stor del av det opprinnelige målet er oppnådd selv om noe måtte elimineres på grunn av manglende støtt i nettleseren Safari for iPhone.

Totalt har prosjektet gitt økte kunnskaper og erfaring med alle de teknikker og teknologiene som er benyttet. Prosjektet har også gitt innsikt i hvordan en eventuell webløsning i fremtiden kan se ut og fungere.

Prosjektets gjennomføring har vist fordeler og ulemper ved webløsninger på mobile enheter, kontra tradisjonelle nettlesere, men i all hovedsak vist én aktuell måte å utvikle en slik type webapplikasjon. Prosjektet har også vist hvordan man kan utnytte RSS som et innholdsgrunnlag på en måte som gjør presentasjonen effektiv og med et målrettet visuelt uttrykk.

6. Litteraturliste

1. Wagner, Richard. *Safari and WebKit Development for iPhone OS 3.0*.Indianapolis, US. Wiley Publishing (2010).
2. Lavin, Peter. *Object-Oriented PHP*. No Starch Press Inc. San Francisco US. (2006).
3. Crane, D, Pascarello, E, James, D. *Ajax in Action*. Manning Publications Co. Greenwich, CT, US.(2006).
4. Pfaffenberger, B, Schafer, M., S, White, C, Karow, B. *HTML, XHTML, and CSS Bible*. Wiley Publishing, Inc. Indianapolis US.(2004).
5. Sommerville, Ian. *Software Engineering*. Addison Wesley. England, UK. (2007).
6. Benyon, David, Turner, Phil. *Designing interactive Systems,People, activities, contexts, Technologies*. Addison Wesley. England, UK. (2005).
7. Horgen, A., Svend. *Webprogrammering i PHP*. (2 utg). Tisip.Norge (2005).

7. Vedlegg - innholdsfortegnelse

Vedlegg A

jQuery Application Plugin

```
/**
* JQUERY IPHONE APPLIKATION NAVIGATION PLUGIN
*
* Forfatter       : Kim Sandvold
* Skrevet         : 12/03-2010
* Oppdatert       : 18/03-2010
*
* Hensikten med denne plugin'en er et skipt som dynamisk kan simulere
navigasjonen
* til en iphone/ipod applikasjon uten for mye konfigurasjon. Plugin'en er en
del av
* en bacheloroppgave i medieteknologi
*/



jQuery.fn.appnav = function(settings){

      /* Standard konfigurasjonsvariabler hvis ikke de blir overkjørt ved kall
på pluginen */
      config = $.extend({

            loader_cont : '',                   // element som skal vise
forespørsels indikatoren
            loader_gif  : 'loader.gif',         // indikator fil
            slider_speed : 1000,                // animasjonshastighet i
millisekunder
            cont_width : 480                    // standard kontainer vidde

      }, settings);

      // initierer variablen til senere bruk
      var rss_content_type = null;
      var rss_url;
      var art_url= '';

      /**
      * Når man klikker på en anchor element som har en "rel" attributt som
har verdien "nav_home" går man til start
      */
      $('a[rel*=nav_home]').live('click', function(){
            $( '#container-inner').animate( {
                  marginLeft: "0px"
            });
            event.preventDefault();
      });

      /**
      * Når man klikker på en anchor element som har en "rel" attributt som
har verdien "nav" kjøres denne
      */
      $('a[rel*=nav]').live('click', function(){
```

```
            /* Dynamiske variabler lages for hver man klikker på et
navigasjonselement.     */

            var art_url = $(this).attr('id');

            // Behandling av href verdien  ( - hash)
            str = $(this).attr('href').substring(1);


            // lager en array av href verdien og bruker 2 stk kolon som
splitter
            href_array = str.split('::');

            if(href_array[2] != null){
                  rss_url = href_array[2];
            }

            rss_content_type = href_array[3];

            menu_number = href_array[5];

            client_id = href_array[4];

            // hvilket nivå som er gjeldende målet
            slide_level = href_array[0].split('-');


            // Lager et tall for hvor langt slideren skal gå
            slide_value = parseFloat( config.cont_width ) * slide_level[1];

            $( '#container-inner').animate( {

                  marginLeft: "-"+slide_value+"px",

            }, config.slider_speed, function(){ // I det animasjonen er
fullbyrdet starter en annen animasjon som alltid tar deg til toppen

                  $('html, body').animate( {scrollTop : 0},
config.slider_speed, function(){

                        if( href_array[1] == 'fw' ){

                              request_data(
                                    href_array[2], // fil prefix

      'page='+rss_content_type+'&url='+rss_url+'&ct='+rss_content_type+'&href=
'+art_url,
                                    art_url,
                                    href_array[6]
                              );
                        }

                        request_sub_menues(client_id, menu_number);
                  });
                  return false;
            });
            event.preventDefault();
```

```
	});

	/**
	* Laster inn data for undermenyene
	*/
	$('a[rel*=sub_menu]').live('click',function(){

		$('.cats ul li a').removeClass('active');
		$(this).addClass('active');

		rss_url = $(this).attr('href').substring(1).split('::');

		request_data(
			href_array[2], // fil prefix

	'page=null&url='+rss_url[0]+'&ct='+rss_url[1]+'&href='+art_url,
			art_url,
			rss_url[2]
		);
		return false;
	});

	request_sub_menues = function(c_id, menu_id){

		$.ajax({
			type : 'GET',
			url : 'content/rss_content_sub.php?',
			data : 'client_id='+c_id+'&menu_id='+menu_id,

			cache : false,
			success: function(sub_response){
				$('.cats ul').html(sub_response);
				$('.cats ul li a:first').addClass('active');
			},
			error: function(sub_response){

				alert('Noe er galt..'+ console.log(sub_response) );
			}
		});
	}

	/**
	* Funksjon som tar seg av alle forespørsler av data(ajax) til ved
navigeringen
	*/
	request_data = function( file, qstr, href , element_name){

		var indicator_cont = ( href == '' ) ? '1': '2';

		// Lager et element som viser animert indikator mens innholdet
laster
		$('#response'+indicator_cont).append('<div
id=\'indicator\'></div>');
		$('#indicator').css({
			backgroundImage: 'url('+config.loader_gif+')',
			backgroundRepeat: 'no-repeat',
			backgroundPosition:	'center center'
```

```
	});
	$.ajax({
		type : 'GET',
		url : 'content/rss_content.php?',
		data : qstr,
		cache : false,
		success: function(response){

			if( href == ''){
				$('#response'+indicator_cont).html(response);
			}else{

				$('#response'+indicator_cont).html(response);
			}
			$('.page_desc').html(element_name);
			$('#'+config.loader_cont).hide();
		},
		error: function(response){
			alert('Noe er  * galt..'+ console.log(response) );
		}
	});
}
                                }
```

Vedlegg B

PHP RSS klasse

```
<?php
include('../config.php');
include('../init.php');

// lager et objekt av klassen "load_content" som tar i mot superglobale
variabler fra ajax forespørselen
$content = new load_content();

// skriver ut ferdigstilt HTML for presentasjon
echo ($_GET['href'] != '') ? $content->single_content(): $content->output();

class load_content{

      /**
      * Klasse som henter data fra RSS og lager passende html ut av det for å
presentere
      * det som web-innhold. Fordelen med denne klassen er at den er fleksibel
og kan brukes i alle
      * xml/rss parsingene løsningen foretar.
      *
      * Forfatter       : Kim Sandvold
      * Skrevet         : 26.01.2010
      * Oppdatert       : 17.05.2010
      * Systemkrav      : lese-rettigheter for eksterne filer i apache
konfigurasjonen
      *                         GD library
      *                         PHP 5 >
      */

      // konstante variabler

      const max_heading_length = 21;
      const max_thumb_width = 62;
      const max_amount_of_articles = 10;
      const date_format = 'l d/m-Y';

      /**
      * Konstruktor som initiers når man kaller på klassen. Felles funksjoner
samles her
      */
      public function __construct(){

            // objekt av initieringsscriptet - på grunn av språkfilene
            $this->init = new init;

            // initierer XML DOM parser'en så vi kan lese rss/xml filen
            $this->xmlDoc = new DOMDocument();
            $this->xmlDoc->load( $_GET['url'] );/* tar imot parametre fra ajax
forspørselen */

            // tar imot url fra ajax forespørsel
            $this->art_url = $_GET['href'];

            $this->url = $_REQUEST['url'];
```

```
            // tar imot innholdstypen (ct = content type. 1 2 3 -> typer- 0 ->
ingen -> default nr 2)
            if($_GET['ct'] == 0){
                  $this->content_view = 2;
            }else{
                  $this->content_view = $_GET['ct'];
             }
      }

      /**
      * Funksjon som formatterer datoen på en gitt måte.
      */
      private function prepare_date($date_source)     {

            $date_raw = strtotime($date_source);
            $date = date(self::date_format, $date_raw);
            return $date;
      }

      /*
      * Funksjon som har den funksjonen å renske html koden for html-
elementer. Fordelen med dette
      * er at da kan vi presenterer kun teksten i igressen uten den originale
formateringen.
      */
      private function prepare_text($src, $length){

            $fixed_text = strip_tags($src);
            $dots = ( strlen( $fixed_text ) > $length) ? '...': null;
            if($length == 0){
                  return $fixed_text;
            }else{
                  return substr($fixed_text, 0, $length).$dots;
            }
      }

      /**
      * Fordi tittelen på en artikkel kan være lang, og at et grensesnitt på
en mobiltelefon er
      * begrenset i størrelse kan det være hensiktsmessig å begrense antall
symboler i tittelen
      */
      private function prepare_title( $src, $length ){

            $title = (strlen($src) > $length) ? substr($src, 0, $length).'..':
$src;
            return $title;
      }

      /**
      * Denne metoden leter etter et "img" element i kilden/html fra
ingressen. Finner metoden et slikt element
      * tar den vare på kun "img" elementet. Dette gir tilgang til å skille
teksten fra bildet.
      *
      * Vesentlig for   : strukturering av html
```

```
	*/
	private function prepare_image( $src ){

		// bruker et regulært uttrykk for å dra ut bildeadressen(url) i
img tag'et - hvis det finnes noen
		$pattern = '/<img[^>]+src[\\s=\'"]+([^"\'>\\s]+)/is';

		// Tester om kildekoden finner et img tag. Hvis, returner komplett
filbane, eller bruk et standard bilde
		if( preg_match( $pattern, $src, $match ) ){
			return $match[1];
		}else{
			return $this->init-
>template_path().'/images/default_thumb.jpg';
		}
	}

	/**
	* Ettersom bilder i ingressen på en artikkel kan ha forskjellige
størrelser må vi ha en metode
	* som finner en riktig størrelse. Det metoden gjør er å finne ut om
vidden er mindre enn høyden,
	* og hvis den er det så er det vidden som får en gitt størrelse. Er det
omvendt er det høyden
	* som får en definert høyde. Dette sikrer at bildet(presentert som en
thumbnail) aldri får glipper,
	* eller tomrom verken vertikalt eller horisontalt.
	*
	* Vesentlig for   : presentasjonen
	*/
	private function image_size($src){

		@list($width, $height, $type, $attr) = getimagesize($src); // @
fjerner evt error meldinger

		/**
		* En påstand om "true" eller "false" trenger ingen "if else" test.
Et syntaktisk alternativ er
		* den ternære kondisjonelle operatoren som ikke nødvendigvis er
mer effektiv, men kan være
		* plassbesparende ved komprimering av kode.
		*/
		if($width != null){
			$size_attr = ($width < $height) ?
'width=\''.self::max_thumb_width.'\'': 'height=\''.self::max_thumb_width.'\'';
		}else{
			$size_attr = null;
		}
		return $size_attr;
	}

	/**
	* Funksjon som genererer html til websiden
	*/
	public function output(){

		$element = $this->xmlDoc->getElementsByTagName('item');
```

```
            $out = '';
            // for-løkkke som kjører igjennom alle elementer i rss/xml filen
som heter "item"
            for ($i = 0; $i < self::max_amount_of_articles; $i++){

                  // Henter verdiene i elementene i rss filen
                  $this->item_title       = $element->item($i)-
>getElementsByTagName('title')->item(0)->childNodes->item(0)->nodeValue;
                  $this->item_link  = $element->item($i)-
>getElementsByTagName('link')->item(0)->childNodes->item(0)->nodeValue;
                  $this->item_desc  = $element->item($i)-
>getElementsByTagName('description')->item(0)->childNodes->item(0)->nodeValue;

                  // Den fulle adressen til det eventuelle bildet i ingressen
                  $this->item_img_path = $this->prepare_image($this-
>item_desc);

                  // Forskjellig presentasjonsoppsett
                  if( $this->content_view == 1 ){
                        $out .= $this->content_view_text();
                  }else if( $this->content_view == 2 ){
                        $out .= $this->content_view_mixed();
                  }else if( $this->content_view == 3 ){
                        $out .= $this->content_view_img();

                  }
            }
            $out .= '<div class="clear"></div>';

            return $out; // returnerer komplett html som presenteres
      }

      /**
      * Funksjon som lager mal for elementer bare med text
      */
      public function content_view_text(){

            // Output er variabelen som til slutt inneholder all html som skal
presenteres
            $html = '';
            $html .= '<div class=\'items_text\'>';
            $html .= '<a rel=\'nav\' id=\''.$this->item_link .'\'
href=\'#level-2::fw::'.$this->url.'\'>';

            $html .= '<div class=\'item_text_info\'>';
            $html .= '<div class=\'item_text_title\'>'.$this->prepare_title(
$this->item_title , 25).'</div>';
            $html .= '</div>';

            $html .= '<div class=\'item_text_arrow\'>';
            $html .= '</div>';

            $html .= '<div class=\'clear\'></div>';

            $html .= '</a>';
            $html .= '</div>';
```

```
            return $html;
      }

      /**
      * Funksjon som lager mal for elementer med både bilde og text
      */
      public function content_view_mixed(){
            // html er variabelen som til slutt inneholder all html som skal
presenteres
            $html = '';
            $html .= '<div class=\'items_mixed\'>';
            $html .= '<a rel=\'nav\' id="' .$this->item_link .'"
href=\'#level-2::fw::'.$this->url.'\'>';

            $html .= '<div class=\'item_mixed_image\'>';
            $html .= '<img src=\''.$this->item_img_path.'\' '.$this-
>image_size( $this->item_img_path ).' alt=\'bilde\' />';
            $html .= '</div>';

            $html .= '<div class=\'item_mixed_info\'>';
            $html .= '<div class=\'item_mixed_title\'>'.$this->prepare_title(
$this->item_title, 20 ).'</div>';
            $html .= '<div class=\'item_mixed_desc\'>'.$this->prepare_text(
$this->item_desc , 58).'</div>';
            $html .= '</div>';

            $html .= '<div class=\'item_mixed_arrow\'>';
            $html .= '</div>';

            $html .= '<div class=\'clear\'></div>';

            $html .= '</a>';
            $html .= '</div>';
            return $html;
      }

      /**
      * Funksjon som lager en mal for bildegalleri
      */
      public function content_view_img(){

            $html = '';
            $html .= '<div class=\'items_img\'>';
            $html .= '<a rel=\'nav\' id="' .$this->item_link .'"
href=\'#level-2::fw::'.$this->url.'\'>';
            $html .= '<div class=\'item_img_src\'>';
            $html .= '<img src=\''.$this->item_img_path.'\' '.$this-
>image_size( $this->item_img_path ).' alt=\'bilde\' />';
            $html .= '</div>';
            $html .= '<div class=\'clear\'></div>';
            $html .= '</a>';
            $html .= '</div>';
            return $html;
      }

      /**
      * Funksjon som skriver ut bare den valgte artikkelen og viser den med
```

```
      * tittel, link til orgiginale artikkelen og ingressen/beskrivelsen.
      * Tar med ekstra data hvis RSS strømmen kommer fra Joomla!
      */
      public function single_content(){

            $element = $this->xmlDoc->getElementsByTagName('item');

            /* for-løkkke som kjører igjennom alle elementer i rss/xml filen
som heter "item" */
            for($i = 0; $i < 10 ; $i++){

                  $item_title       = $element->item($i)-
>getElementsByTagName('title')->item(0)->childNodes->item(0)->nodeValue;
                  $item_link        = $element->item($i)-
>getElementsByTagName('link')->item(0)->childNodes->item(0)->nodeValue;
                  $item_desc        = $element->item($i)-
>getElementsByTagName('description')->item(0)->childNodes->item(0)->nodeValue;

                  $item_img_path    = $this->prepare_image( $item_desc );

                  /* Hvis pekeren til artikkelen stemmer med den som ble sendt
med i ajax forespørselen så kjøres denne */
                  $sendt_url = explode('#', $item_link );

                  /* Hvis mottatt lenke er lik linke i RSS filen så skal dette
elementet vises */
                  if( $sendt_url[0] == $this->art_url ){

                        $html = ''; // output variabel

                        /* Leser selve RSS filen for å finne identifikatorer
*/
                        $handle = fopen( $this->url, "rb" );
                        $contents = stream_get_contents( $handle );

                        // Hvis joomla - tar vi med dato for forfatter
                        if( strpos( $contents, '<generator>Joomla' /* dette
går igjen i feeds fra joomla */ ) !== false ){

                              $item_pub = $element->item($i)-
>getElementsByTagName('pubDate')->item(0)->childNodes->item(0)->nodeValue;
                              $item_author = $element->item($i)-
>getElementsByTagName('author')->item(0)->childNodes->item(0)->nodeValue;
                              $date_and_author = '<div
class=\'items_single_pub\'>'.$this->init->language('OPEN',
'GEN_WRITTEN_BY').', '.$item_author.' '.$this->prepare_date( $item_pub
).'</div>';

                        // hvis ikke - ikke ta med noen ekstra felt
                        }else{
                              $date_and_author = '';
                        }
                        fclose($handle);

                        $html .= '<div class=\'items_single\'>';
                        $html .= '<div
class=\'items_single_title\'>'.$item_title.'</div>';
```

```
                        $html .= $date_and_author;
                        $html .= '<img src=\''.$item_img_path.'\'
width="280px" alt=\'bilde\' />';
                        $html .= '<p>'. strip_tags( $item_desc ).'</p>';
                        $html .= '<div class=\'clear\'></div>';
                        $html .= '<div class=\'items_single_link\'>';
                        $html .= '<span class="item_links item_links_left"><a
href="'.$item_link.'" target="_blank">'.$this->init->language('OPEN',
'GEN_VIEW_COMPLETE_ART').'</a></span>';
                        $html .= '<span class="item_links item_links_right"><a
rel="fb_share" href="'.$item_link.'::'.$item_title.'">'.utf8_encode($this-
>init->language('OPEN', 'GEN_FB_SHARE')).'</a></span>';
                        $html .= '<div class=\'clear\'></div>';
                        $html .= '</div>';
                        $html .= '<blockquote>"'.utf8_encode($this->init-
>language('OPEN', 'GEN_ART_INFO')).'"</blockquote>'; // artikkel informasjon
                        $html .= '</div>';
                    }
            }
            return $html;
      }
}
                                        ?>
```

Vedlegg C

Statusrapport 1 - 28/02-2010

Tilstand og fremdrift

Prosjektet har fra oppstarten utviklet seg ide til et hovedrammeverk til systemet. Jeg har gjennom denne prosessen fått prøvd ut en del forskjellige metoder og teknikker – programmeringsmessig - slik at jeg har fått oversikt over hvilke muligheter jeg har til å gjennomføre prosjektet på best mulig og mest effektiv måte. Mye tid har gått med til å finne ut hvordan systemets struktur skal være. Dette er med tanke på senere vedlikehold av systemet for bedriftens tekniske personell.

All samarbeid med oppdragsgiver når det gjelder hovedstruktur på det webbaserte systemet er fullført. Det foreligger enighet mellom oss om hvordan resultatet av hovedstrukturen skal bli. Videre samarbeid med oppdragsgiver vil hovedsakelig gjelde fortløpende korrigeringer og ideer for videreutvikling. Når dette er sagt har de ikke tilgang til å komme med radikale ideer til omstrukturering og funksjonalitet, men mindre endringer ettersom de ofte oppdaterer og oppgraderer sine egne systemer. Dette vil ikke påvirke mitt prosjekt negativt eller på annen måte hindre meg i å fullføre mine mål.

Prosjektets/tankens utvikling

Ettersom fluefiske.net sine publiseringssystemer ofte endres og oppgraderes er jeg blitt enda mer bevisst på å gjøre mobilløsningen totalt uavhengig av deres ordinære systemer. Jeg har sett at det virkelig er nødvendig å levere et produkt som ikke vil påvirke disse hyppige endringene deres, slik at de skal kunne dra nytte av det i lengre tid.

Risikovurderinger

Et spørsmål om prosjektet er i risikosonen for å ikke bli fullført ville svaret vært; nei det er det ikke. Prosjektet er i rute og kanskje nærmere målet enn planlagt i første omgang. Prosjektet kan kanskje bli litt mer omfattende enn planlagt, i den forstand at jeg har mulighet for å utvide systemet med nye ønsker fra oppdragsgiver hvis jeg klarer å holde meg foran prosjektplanen. Selvsagt har jeg fokus på det konkrete prosjektet og vil ikke utvide dette med mindre det gagner den opprinnelige planen. Samarbeid med oppdragsgiver Når det gjelder kommunikasjon mellom meg og oppdragsgiver er det veldig bra. Jeg har jevnlig kontakt med tre personer som har tilknytning i bedriften både elektronisk(skype) og fysiske møter for å dele ideer og meninger.

Prioriteringer

De neste ukene vil det prioriteres å gjøre helt ferdig navigasjonsstrukturen. Førsteutkastet er ferdig, men må justeres og forbedres. Også testing av systemets ressurs forbruk i forhold til safari for iphone 3g og 3gs vil det brukes mye tid på. Etter dette er fullført kommer jeg til å konsentrere meg om hvordan systemet skal administreres av Fluefiske i etterkant. Dette har jeg mine egne tanker om, men gjenstår og komme til fullstendig enighet med Fluefiske.

Tidsbruk

Frem til nå har jeg brukt ca 113 timer på prosjektet (ifølge min excel logg). Dette er både på egenhånd og møter og annen kommunikasjon med oppdragsgiver. Min vurdering Etter å ha jobbet med dette prosjektet en stund har jeg blitt mer bevisst på hvordan dette kan gjennomføres på best mulig måte. Systemet kommer til å bli ganske omfattende, mer omfattende enn planlagt. Dette er først og fremst fordi jeg nå ser nytten av det for oppdragsgivers del og kan se for meg hvordan de vil kunne bruke systemet i etterkant av dette prosjektet. For det andre har jeg fått en

indikasjon på hvor langt jeg kan strekke meg selv og hva jeg selv er i stand til å gjøre i forhold til xml, html, css, php og javascript/ajax språkene. Selv har jeg stor tro på at dette prosjektet kommer til å bli et fullstendig og tilfredsstillende system for alle parter.

Vedlegg D

Statusrapport 2 - 26/03-2010

Tilstand og fremdrift

Prosjektet har siden forrige statusrapport vokst på forskjellige måter. Etter veiledning med Monica Strand fikk jeg en del nye tanker som jeg har vurdert og som har hjulpet meg i å ta et standpunkt om hvor omfattende sluttproduktet skal være. Utforsking av tekniske utfordringer har jeg brukt med tid på enn å skrive rapport. Jeg finner det mest hensiktsmessig å ha klare standpunkt for den tekniske delen før jeg skriver masse om teori - selv om jeg har skrevet ned alt jeg har gjort så jeg kan vurdere og argumentere senere for mine standpunkt.

Prosjektets utvikling

Jeg har under hele prosessen hatt fokus på at publiseringsløsningen Joomla ville være den primære plattformen ettersom Fluefiske.net har dette som deres hovedløsning, men ettersom jeg vektlegger fleksibilitet og kompatibilitet til flere systemer har dette fokuset endret seg. Til å begynne med hadde jeg tenkt å i tillegg til å bygge et administrator komponent til Joomla for å administrere mobilløsningen. Dette vil ikke være ideelt hvis et bytte av publiseringsverktøy vil finne sted i fremtiden. Jeg har da valgt å finne andre måter som vil sikre kompatibiliteten. Dette vil kreve egen innlogging, noe som ikke var en del av den opprinnelige planen der innloggingen til Joomla ville være tilstrekkelig.

Jeg tenker at denne endringen i systemet forbedrer systemet i seg selv og i tillegg får jeg utfordret meg selv enda mer etter som en ekstern brukerinnlogging vil bety at det mest hensiktsmessige vil være å bruke en database. Dette hadde jeg ikke i utgangspunktet planer om ettersom den opprinnelig planen var å bruke innloggingen til joomla. Databasen vil kun være en del av innloggingen da og ikke ha noe med innholdet for mobilløsningen. Risikovurdering På dette tidspunkt foreligger det ingen risiko for at systemet vil være fungerende og ferdig innen gitt frist, men ettersom jeg har endret litt i planene for systemet må jeg bruke litt mer tid på den tekniske delen enn planlagt. Dette vil si litt utsettelse av deler av rapporten, men ingenting av dette er et hinder for å bli ferdig.

Samarbeid med oppdragsgiver

Samarbeid med oppdragsgiver har fra sist statusrapport vært mindre omfattende. På grunn av mye jobb fra begge parter har kommunikasjonen holdt seg for det meste via Skype og telefon. Dette har ingen negative følger ettersom all planlegging og samarbeid med oppdragsgiver i utgangspunktet er fullført. De tekniske endringen jeg selv har tatt et standpunkt til vil i liten grad berøre hvordan systemet faktisk vil fungere for Fluefiske.net bortsett ifra ekstern innlogging i stedet for at det blir en del av publiseringsløsningen Joomla.

Prioriteringer

Tiden fremover i april vil bestå av programmering og rapportskriving. Etter enda flere tekniske undersøkelser og testinger enn sist er systemet i ferd å endre seg. Når jeg selv føler at systemet er fungerende og godt vil jeg avtale en ny veiledningstime hos Monica Strand, men venter nok til uken etter påsken. Min Vurdering Jeg har selv fremdeles veldig troen på dette prosjektet. Det kommer til å bli et nyttig og spennende resultat av dette. Med tanke på at en database vil bli en del av prosjektet øker dette vanskelighetsgraden på prosjektet og jeg får selv brukt meg selv enda mer.

Vedlegg E

Statusrapport 3 - 29/04-2010

Tilstand og fremdrift

Prosjektet fra forrige statusrapport har utviklet seg mye. Systemet jeg lager for oppdragsgiver er ikke helt ferdig, men klar for testperioden 1. mai - til innleveringsdatoen 20. mai. Denne perioden skal for min del brukes til å teste systemet på forskjellige versjoner av iPhone og andre mobile enheter får kartlegge hvordan systemet fungerer på de forskjelle enhetene.

Prosjektets rapport er også siden sist statusrapport vokst veldig. Rapporten har rundet 22 sider og det kommer til å bli flere. Etter veiledning 30/04-2010 håper jeg på gode råd videre i skriveprosessen.

Prosjektets utvikling

Prosjektet har utviklet seg kraftig gjennom hele prosessen. Nye problemer relatert til Safari for iPhone har tvunget meg til å endre enkelte valg. Når dette er sagt har denne utviklingen bare vært positiv i forhold til systemet.

Risikovurdering

Risikovurderingen i dette prosjektet går på hvorvidt systemets tilstand er i forhold til tidsplanen og innleveringsfristen samt og de opprinnelige tankene og ideene lar seg gjøre på en enhet som iPhone. I dag ser det ikke ut til at det foreligger noen risiko for at disse momentene ikke vil fullføres. I testperioden til systemet vil jeg få mer informasjon om hva som fungerer og hva som evt. ikke fungerer.

Det har imidlertid oppstått noen problemer i forhold til lesing av RSS strømmer som jeg har prøvd å løst i litt over en uke, men har nå delvis klart å løse dette. Dette kommer jeg til å ha fokus på under testperioden.

Samarbeid med oppdragsgiver

I denne fasen har samarbeid med oppdragsgivers primærkontakt vært mindre, men desto større med teknisk ansvarlig og redaktør. Dette har vært fordelaktig fordi man alltid har godt av å høre

hva andre innad i organisasjonen mener og tror. Selvsagt er det naturlig at teknisk ansvarlig i organisasjonen vet hva som til en hver tid skjer og skal skje.

Prioriteringer

Prioriteringene på dette tidspunkt er å skrive ferdig rapporten og få tilbakemeldinger og gjøre analyse fra de forskjellige mobile enhetene systemet skal testes på.

Vedlegg F

Prosjektdagbok

Dato	Aktivitet	Timer totalt(ca)
11.01.2010	Møte med oppdragsgiver - Planlegging av prosjekt	5
13.01.2010	Planlegging av prosjekt	6
14.01.2010	Testing av iPhone enheten	7
18.01.2010	Møte med oppdragsgiver - idemyldring	3
19.01.2010	Lese litteratur og koding - javascript VS Safari for iPhone	8
23.01.2010	Programmering - testing	9
27.01.2010	Programmering - javascript testing	8
01.02.2010	Programmering - javascript testing	7
02.02.2010	Programmering - testing - ajax/javascript navigering	6
05.02.2010	html/css/grafisk - template for oppdragsgiver	8
08.02.2010	html/css/grafisk - template for oppdragsgiver	9
09.02.2010	html/css/javascript - navigasjonsstruktur	6
12.02.2010	javascript/php - rammeverk	4
16.02.2010	PHP programmering og møte med oppdragsgiver	7
19.02.2010	PHP/javascript - testing	6
22.02.2010	PHP - filtrering av RSS innhold	6
23.02.2010	PHP - filtrering av RSS innhold	8
01.03.2010	PHP - filtrering av RSS innhold	6
02.03.2010	GUI for administrasjonssystem	10
05.03.2010	PHP/Ajax - admin funksjonalitet	11
08.03.2010	PHP/Ajax - admin funksjonalitet	12
09.03.2010	PHP/Ajax - admin funksjonalitet	9
15.03.2010	PHP/Ajax - admin funksjonalitet	9
16.03.2010	PHP/Ajax - admin funksjonalitet	9
22.03.2010	PHP/Ajax - admin funksjonalitet	7
23.03.2010	PHP/Ajax - admin funksjonalitet	8
26.03.2010	PHP/Ajax - admin funksjonalitet	7
01.04.2010	Rapportskriving - klargjøre disposisjon	8
02.04.2010	Rapportskriving - ferdig disposisjon og skriving	9
03.04.2010	Rapportskriving - finne kilder og henvisniner	1
04.04.2010	Rapportskriving	1
05.04.2010	Rapportskriving	10
06.04.2010	Rapportskriving	10
07.04.2010	Rapportskriving	2
08.04.2010	Rapportskriving	6
09.04.2010	Rapportskriving	6
12.04.2010	Rapportskriving /språkfilsystem - planlegging og	9

	undersøkelser	
13.04.2010	Rapportskriving	10
14.04.2010	Rapportskriving	6
15.04.2010	Rapportskriving	8
16.04.2010	Rapportskriving	8
17.04.2010	Rapportskriving	6
18.04.2010	Rapportskriving / programering	4
19.04.2010	Rapportskriving / programering	12
20.04.2010	Rapportskriving / programering	12
21.04.2010	Rapportskriving / programering	10
22.04.2010	Rapportskriving	8
23.04.2010	Rapportskriving	4
24.04.2010	Rapportskriving / programering	10
25.04.2010	Rapportskriving / programering	9
26.04.2010	Rapportskriving / veiledning med Monica Strand	4
27.04.2010	Rapportskriving / programering	11
28.04.2010	Rapportskriving	4
29.04.2010	Rapportskriving	8
30.04.2010	Rapportskriving	8
01.05.2010	Rapportskriving / HTML - laget ny default template i tillegg til fluefiske	8
02.05.2010	Rapportskriving / kontroll - kontroll av kilder. Skriving	7
03.05.2010	Rapportskriving / testperiode - retting av evt negative tilbakemeldinger	10
04.05.2010	Rapportskriving / testperiode - retting av evt negative tilbakemeldinger	10
05.05.2010	Rapportskriving / testperiode/ veiledning - sendt veileder førsteutkastet av rapport	11
06.05.2010	Rapportskriving / testperiode - retting av evt negative tilbakemeldinger	7
07.05.2010	Rapportskriving / testperiode - retting av evt negative tilbakemeldinger	10
08.05.2010	Rapportskriving / testperiode - retting av evt negative tilbakemeldinger	6
09.05.2010	Rapportskriving / testperiode - retting av evt negative tilbakemeldinger	10
10.05.2010	Rapportskriving / testperiode - retting av evt negative tilbakemeldinger	13
11.05.2010	Rapportskriving / testperiode - retting av evt negative tilbakemeldinger	7
12.05.2010	Rapportskriving / testperiode - retting av evt negative tilbakemeldinger	8
13.05.2010	Rapportskriving / testperiode - retting av evt negative tilbakemeldinger	10

14.05.2010	Rapportskriving / testperiode - retting av evt negative tilbakemeldinger	11
15.05.2010	Rapportskriving / testperiode - retting av evt negative tilbakemeldinger	10
16.05.2010	Rapportskriving / kontroll av kode	10
17.05.2010	Rapportskriving / kontroll og korrektur av gramatikk	6
18.05.2010	Rapportskriving / kontroll og korrektur av gramatikk	12
19.05.2010	Rapportskriving og korrektur /ferdigstilling av rapport	12
	Total tid:	**583**