høgskolen i oslo
avdeling for ingeniørutdanning

Studieprogram:
Postadresse: Postboks 4 St. Olavs plass, 0130 Oslo
Besøksadresse: Holbergs plass, Oslo

Telefon: 22 45 32 00
Telefaks: 22 45 32 05

PROSJEKT NR.
11-34

TILGJENGELIGHET
Åpen

PROSESSDOKUMENT

HOVEDPROSJEKTETS TITTEL	DATO
System for generering av webskjemaer for Mamut ASA	24.05.2011
	ANTALL SIDER: 22
PROSJEKTDELTAKERE: Kjetil Hårtveit, s155501	INTERN VEILEDER: Alfred Bratterud

OPPDRAGSGIVER	KONTAKTPERSON
Mamut ASA	Espen Johannessen

SAMMENDRAG
Systemet som er utviklet brukes til å generere webskjemaer. Hensikten med systemet er å øke datakvalitet ved å fjerne manuelle rutiner, redusere svinn av leads og øke salg gjennom bedre oppfølging av leads. Systemet er utviklet med Microsoft teknologier C#/ASP.NET med MSSQL og Windows Service.

3 STIKKORD
Mamut Webforms
Webskjemagenerering
.NET

Innholdsfortegnelse

1. Introduksjon

1.1 Om dokumentet

Dette dokumentet dekker hele systemutviklingsprosessen. Den gir ett innblikk i hvordan jeg opplevde alle fasene av prosjektet, samt innblikk i utfordringer jeg støtte på og hvordan de ble løst. Det forteller også om miljøet jeg jobbet under, utviklingsmetoden og verktøyene jeg brukte. Det forklarer også noe om kravspesifikasjonens rolle i prosjektet.

1.2 Takknemligheter

Før vi går i gang vil jeg takke veilederen min Alfred Bratterud for å gi gode tips og ha fortalt meg gode historier. Jeg vil også takke Espen Johannessen og alle andre hos Mamut for å være dem de er – og for å ha gitt meg oppdraget og hatt troen på at jeg kunne komme i mål med prosjektet.

2. Om systemet

Mamut ASA ønsket ett å få utviklet ett nytt system for å generere webskjemaer for innsamling av lead fra kampanjeaktiviteter. Det eksisterende CMS systemet og den nåværende rutinen for registrering av leads har svakheter som både har negativ virkning på det totale salget og tilgjengelige ressurser. Det nye systemet har som mål å strømlinjeforme og automatisere leadinnsamlingen så mye som mulig og dermed løse opp flokene som er tilstede.

Løsningen vil i stor grad være en Microsoft orientert løsning. Det vil bli utviklet i IDEen Visual Studio 2010 i rammeverket ASP.NET/ C# med en backend MSSQL database.

2.1 Om bedriften

Mamut[5] (OSE "MAMUT") ble etablert i 1994 og er en ledende leverandør av administrative programvareløsninger og Internettbaserte tjenester til små og mellomstore virksomheter i Europa. Mamut tilbyr løsninger for økonomistyring/regnskap, salgsstøtte, kunde- og kontaktoppfølging (CRM), innkjøp/logistikk, lønn/personal, prosjektstyring, e-handel, domener, e-post og webhotell. Mer enn 400.000 kunder i Europa forenkler hverdagen med løsninger fra Mamut.

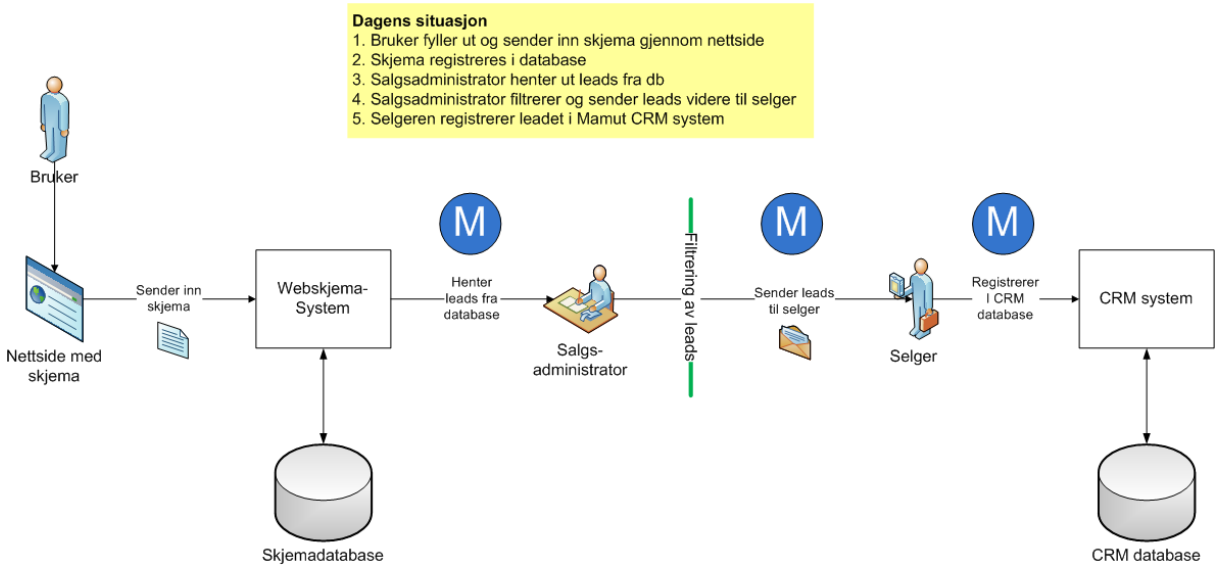
2.2 Opprinnelig situasjon vs. ønsket situasjon

Det gjeldende CMS systemet hos Mamut som brukes til å generere webskjemaer har ingen integrering med sporingseffekter som kan gi nyttig informasjon til markedsførerne. Dette innebærer at sporingskoden må manuelt flettes inn i hvert eneste skjema. Dette er en ugunstig manuell rutine som følger med seg samtlige ulemper:

1. man har ingen styring over hvilke skjemaer som samler inn kode
2. manuell innfletting krever programmering kompetanse
3. manuell innfletting åpner muligheten for brukerfeil som kan forårsake tap av sporingsdata og lavere datakvalitet
4. manuell rutine krever ressurser som kan spares

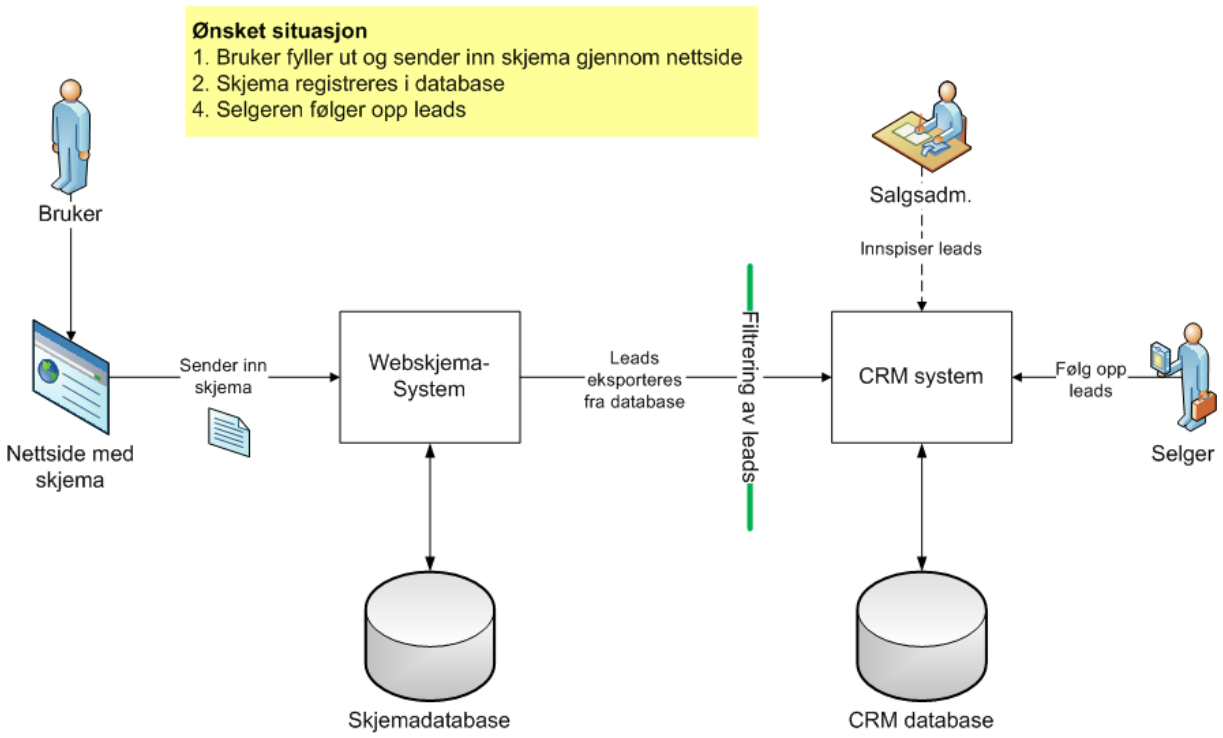
Det er også en besværlig prosess å opprette selve skjemaene. Det nye systemet vil ha som mål å løse problemer ved prosessen. Her har vi også mulighet til å legge til ekstra funksjonalitet som mangler i det eksisterende CMS systemet som kan bidra til bedre effektivitet.

Det største problemet derimot er at selgerne i dag er avhengig av e-postveksling for å registrere og følge leads. Leadsene går gjennom flere manuelle poster før de blir tatt opp og det er sannsynlig at de blir forkastet på veien. Dette systemet vil redusere de manuelle rutiner til ett minimum og sørge for høyere datakvalitet.



Opprinnelig situasjon

I diagrammet over kan vi se at salgsadministrator må hente ut leads fra databasen manuelt. Videre må vedkommende filtrere bort ugyldige leads og manuelt sende leadinfoen videre til en selger. Her skjer det nok en manuell rutine hvor selgeren må registrere leadet i CRM systemet.



Ønsket situasjon

I den ønskede situasjonen skal systemet som tidligere nevnt blant annet fjerne manuelle rutiner. Blant annet skal ikke salgsadministrator behøve å hente leads manuelt, men de skal automatisk bli eksportert fra systemet på en periodisk basis. Filtrering av leads kan da skje digitalt og leads skal importeres inn i CRM systemet utenom å gå igjennom selgere.
Merk at systemet mitt er begrenset til å eksportere leadsene fra databasen. Filtrering av leads og videre utover i kjeden utføres av andre systemer.

2.3 Mål med systemet

- Ressursbesparelser gjennom automatisering av rutineoppgaver.
- Øke datakvalitet ved å fjerne manuelle rutiner.
- Gir bedre informasjon i forhold til hvilke prosjekter som selger.
- Reduserer svinn av leads.
- Øke salget gjennom bedre oppfølging av leads.

3. Om utviklingsprosessen

3.1 Utviklingsmetode

18

Begynnelsen av prosjektperioden var preget av fossefall metoden. Først ble det planlagt og undersøkt hva systemet skulle gjøre og hvordan det skulle bygges. Deretter ble det implementert gradvis, men det kom selvfølgelig nye forespørsler og som førte til at man gikk litt frem og tilbake mellom fossefallstegene. Dermed kan man si at i praksis var utviklingsmetoden noe mer iterativ enn planlagt.

Når det nærmet seg slutten av utviklingsperioden fant vi ut at det kunne vært nyttig med en prototype test av systemet. Systemutviklingen har altså også gjennomgått prototyping som en måte å teste og kvalitetssikre systemet ytterligere.

Prototypingen sørger for mange fordeler i systemutviklingen:

- Økt systemkvalitet
- Økt suksessjanse for systemet
- Økt produktivitet og effektivitet for brukerne av systemet
- Reduserte senere kostnader for utvidelse, debugging eller restrukturering

3.2 Miljø

Systemutviklingen ble utført i IDEen *Visual Studio 2010*[7]. Den støtter ett arkitekturisk mønster kalt *MVC*[1] (Model-view-controller) som jeg brukte som fundament i systemet. Mer konkret brukte jeg *MVC 2.0* som er en MVC programmeringsmodell som i tillegg inneholder nyttige oppdateringer for koding med ASP.NET. Versjonskontroll ble med *Subversion*[10] ble løpende brukt under utviklingen.

3.3 Teknologier

Microsoft Windows[8]


17

Windows ble først introdusert i 1985 som ett tilleggsprogram til MS-DOS som ett svar på den økende interessen i grafiske grensesnitt. I dag er Microsoft Windows verdens ledende operativsystem og det er estimert at de har 91% av klientene som bruker Internett.

ASP.NET 4.0 / C#

ASP.NET[1] er ett applikasjonsrammeverk utviklet av Microsoft og det brukes til å lage dynamiske nettsider, nettapplikasjoner og nettjenester. Det var først introdusert i 2002 med versjon 1.0 og er nå oppe i versjon 4.0.

C#[9] (C-sharp) er ett hardskrevet, objekt orientert programmeringsspråk. Det ble utviklet av Microsoft i 2001.

Microsoft SQL Server[12]
MSSQL er en relasjonell modell database utviklet av Microsoft for bruk med Windows kompatible applikasjoner.

Windows Service
En service er ett program som kjører i bakgrunnen og ikke har ett visuelt grensesnitt. I dette prosjektet brukes det til å foreta automatisk eksporteringer av leads.

3.4 Verktøy

Visual Studio 2010[7]
IDE av Microsoft for utvikling med .NET og andre Microsoft teknologier/produkter. Har støtte for versjonskontroll slik som *Subversion*. Hele Mamut Webforms systemet samt Windows Servicen ble utviklet i VS2010.

14

Microsoft Word
Tekstbehandlingsprogram brukt for å dokumentere systemet og utviklingsprosessen.

Microsoft Excel
Brukt til å loggføre timer og opprettholde oppgaveliste.

Microsoft PowerPoint
Brukt til å lage presentasjoner.

Microsoft Visio
Brukt til å lage grafiske modeller og skjermbilder.

15

Subversion[10]
Open-source versjons- og revisjonskontroll system laget av Apache, først lansert i oktober år 2000. Det brukes til å opprettholde gjeldende og historiske versjoner av filer.

Dropbox[11]
Dropbox ble brukt for å oppbevare alle dokumenter skrevet i sammenheng med systemutviklingen. Det ble også brukt som ett slags revisjonskontroll system ettersom det er mulig å rulle tilbake til tidligere versjoner av dokumentene. I tillegg overførte jeg all

kildekoden over dit på et regelmessig intervall for å skape redundans i tilfelle hardware kræsj på min side.

[16]

Gallio, NUnit og RhinoMocks
Alle de automative testene skrevet i Mamut Webforms er laget basert på testrammeverket NUnit og RhinoMocks. For å kjøre disse testene brukes Gallio Icarus applikasjonen.

FileZilla
Ble brukt for å håndtere filer på SFTP og FTP servere.

4. Prosessen

4.1 Begynnelsen – mislykket gruppearbeid

I utgangspunktet var jeg på gruppe med 3 andre studenter som jeg var godt kjent med etter de 3 årene på HiO. Vi begynte å søke etter hovedoppgave hos samtlige bedrifter men det var dårlig med respons. Det var også en gjennomgående uggen stemning ettersom frister ikke ble holdt og ikke alle var fornøyde med visse studenters innsats. Vi signerte til og med samarbeidsavtaler for å prøve å løse problemet, men det var ikke overbevisende nok. Ved midten av november 2010 bestemte jeg meg for å forlate gruppa og jobbe på egenhånd.

4.2 Finne ett hovedprosjekt

Jeg kom i kontakt med Mamut ASA om ett potensielt hovedprosjekt et par uker før juletider og heldigvis hadde dem ett prosjekt som virket perfekt for meg og min situasjon. De ønsket ett nytt friskt webskjema genereringssystem som var bedre til å samle og spore leads, som igjen blir brukt til å generere mer salg, enn det originale systemet og rutinene de bruker. Ettersom jeg alltid har vært fascinert av webkoding og programmering visste jeg at jeg måtte gripe sjansen. Jeg skrev prosjektskisse for prosjektet noen dager senere og prosjekt *Mamut Webforms* var født.

4.3 Forprosjekt

Etter juleferien møtte jeg Espen fra Mamut hvor vi diskuterte forprosjektet. Han viste meg hvordan dagens situasjon lå an med blant annet bruk av tegninger og modeller. I tillegg gjorde han det samme for den ideelle situasjonen, det vil si den ønskede effekten av prosjektet. Vi fikk også i større grad konkretisert målene av prosjektet. Til slutt fikk vi diskutert og illustrert i form av UML diagrammer den overordnede funksjonaliteten av systemet.

Forprosjektrapport

Hovedprosjekt i anvendt datateknologi ved Høgskolen i Oslo, våren 2011

STUDENTER	Kjetil Hårtveit, s155501
OPPDRAGSGIVER	Mamut ASA
PROSJEKT	Mamut ønsker å få utviklet ett nytt system for å generere webskjemaer for innsamling av lead fra kampanjeaktiviteter.
KONTAKTPERSON	**Espen Johannessen** *Marketing, Prosjektleder* Jobb: +47 23 20 35 28 E-post: espenjo@mamut.com
VEILEDER	Alfred Bratterud

Sammendrag

Mamut ASA ønsket ett å få utviklet ett nytt system for å generere webskjemaer for innsamling av lead fra kampanjeaktiviteter. Det eksisterende CRM systemet og den nåværende rutinen for registrering av leads har svakheter som både har negativ virkning på det totale salget og tilgjengelige ressurser. Det nye systemet har som mål å strømlinje forme og automatisere leadinnsamlingen så mye som mulig og dermed løse opp flokene som er tilstede.

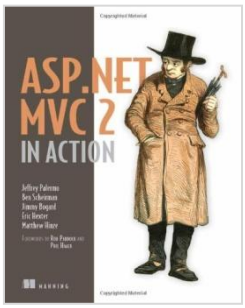
Uka etter hadde jeg skrevet ett utgangspunkt for rapporten og jeg hadde laget en HTML prototype av grensesnittet. Vi kontrollerte detaljer i rapporten, samt gikk igjennom prototypen. Vi kom frem til samtlige endringer som angikk både design, funksjonalitet og database struktur.
Til slutt pratet vi om veien videre og da kom vi frem til at jeg skulle begynne på funksjonalitetsspesifikasjonen (gjerne lage flere modeller/use caser/diagrammer) og oppdatere prototypen med de forandringene som dukket opp.

Etter omskrivninger av visse detaljer i forprosjektsrapporten var den klar for levering.

4.4 Designfasen

Etter forprosjektet ble mesterparten av tiden brukt til å jobbe videre med spesifiseringen av systemet, dets funksjonalitet og dets infrastruktur. Ettersom jeg hadde valgt å lage systemet i *MVC* rammeverk/mønsteret, og jeg hadde null erfaring med dette fra før, så var det ikke så lett å lage ett meningsfullt klassediagram eller komme med formeninger om infrastrukturen. Derfor var mesteparten av spesifiseringene fokusert på grensesnittet og funksjonaliteten som systemet krevde.

Det ble laget modeller for alt fra innholdsstruktur til flyt diagrammer av ”dagens situasjon” og ”ønsket situasjon”. Espen introduserte meg til en ny verden: Microsoft Visio. Aldri har modellering vært så moro før. [13]



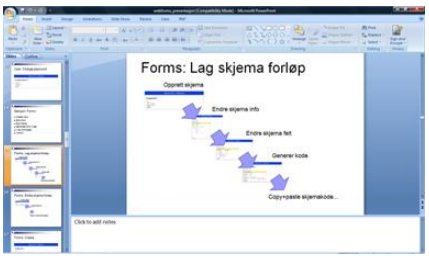
I denne perioden fikk jeg også boka ”ASP.NET MVC2 – In Action” i postkassa. Alltid koselig å lese programmeringsbøker før sengetid! Det ble mange koselige og lærerike netter med den.

Jeg begynte også så smått å opprette MVC rammeverket for prosjektet i Visual Studio slik at jeg kunne få mer praksis erfaring med rammeverket.

4.5 Implementeringsfasen

I starten av mars kunne jeg endelig begynne å kode litt. I starten var det naturligvis en ”nybegynner” opplevelse ettersom jeg måtte lære hvordan ASP.NET MVC2 fungerte. Ikke bare trenger man å vite hvordan man gjør ting, men det er jo viktig å få gode rutiner som samsvarer med ”best practice” prinsipper slik at man slipper å bruke tid på refaktorering av dette senere.

Ettersom jeg allerede hadde lagd HTML prototyper av siden i systemet var det lett å flytte disse over inn i det nye systemet. Dette ga jo en overfladisk følelse av oppnåelse, men det er mye igjen som gjenstår.
Enda hadde jeg ikke opprettet en database som kan lagre persistente data. Jeg hadde bare ett datalag som imiterte dette ved å returnere hardkodede verdier.



Relativt kort tid etter at jeg begynte med implementeringen hadde jeg ett møte med flere potensielle indirekte, om ikke direkte, brukere av systemet hos Mamut. Dette var blant annet personer som driver med webutvikling som vil generere webskjemaer, markedsførere som er interessert i sporing av leads og salgsavdelingen som følger opp disse. Som forberedelse til dette møte anbefalte Espen at jeg for eksempel lagde en powerpoint presentasjon for og lettere å beskrive systemet til deltakerne og det var det jeg gjorde. Her fikk vi mange innspill som kom godt med i den videre utviklingen av systemet.

Ellers bestod det meste av implementeringen naturlig nok til å lage grensesnittet og implementere funksjonaliteten som var oppgitt i kravspesifikasjonen. Hovedfunksjonaliteten i systemet kan ramses ned til generering av webskjemakode, prosessering av innsendt webskjema data og manuell- og automatisk eksportering. Det interessante med implementering av automatisk eksportering er jo at det er nytteløst å kode dette på webserveren ettersom automatisk eksportering må kunne startes på hvilket som helst tidspunkt. Derfor satt jeg meg inn i Windows Service og lagde en service applikasjon som har som oppgave å utføre automatisk eksportering. Servicen oppdaterer seg selv basert på database informasjon og brukere av systemet kan da altså konfigurere servicen gjennom webgrensesnittet til systemet. Den kalkulerer også når automatisk eksportering skal foretas og den bruker såkalte ”timere” til å bestemme når den skal eksportere.

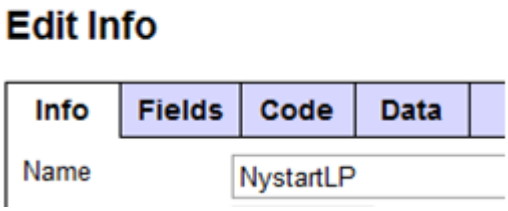
5. Det endelige systemet

Generelt sett er jeg fornøyd med sluttproduktet. All funksjonalitet som krevdes er implementert (noen ønskeliste ting mangler, men det kommer nok etter hvert) og det er kvalitetssikret til en såpass mengde at man kan puste litt rolig ut. Systemet er gått igjennom overhalinger både designmessig og funksjonsmessig gjennom prototyping og andre initiativer. Jeg tror jeg prøver å si at systemet begynner å føles komplett, noe som er mer enn å si at det simpelthen er ferdig. Noen skal jo faktisk bruke systemet man utvikler og da må det være kvalitetssikret for å kunne overleve.

"Welcome to Mamut Webforms. We hope you enjoy your stay."

5.1 I forhold til kravspesifikasjonen

Den øyeblikkelige forskjellen mellom sluttproduktet og prototypen fra kravspesifikasjonen er designet. Det ferdige produktet har fått en penere meny som er mer oversiktlig og gir raskere navigering rundt på nettstedet. Designoverhalingen har også ført til bedre brukervennlighet; for eksempel ved å endre bakgrunnsfargen på den aktive tabben så blir det lekende lett for brukeren å vite hvilken tabb vedkommende er på akkurat nå. Og la oss ikke lyve, vi liker det litt stilig uten at det blir for prippent!



Brukervennlig tabs

Funksjonelt sett så har kravspesifikasjonen blitt endret relativt hyppig i takt med utviklingen. Derfor kan man konkludere med at det ferdige systemet og kravspesifikasjonen er rimelig like. Viktige spesifikasjoner som har kommet frem underveis er blitt inkludert i kravspesifikasjonen og reflektert i systemet.

6. Utfordringer på veien

6.1 Revisjonskontroll feil

Det er veldig fort gjort å kaste bort timer på å fikse ett revisjonskontroll problem. Man endrer filstrukturen ved å flytte eller slette mapper og plutselig får man all verdens feilmeldinger når man prøver å comitte. Ved og istedenfor comitte alt før man endrer strukturen og comitter ofte kan man komme helskinnet fra det hele.

6.2 Hardkodede strenger

I .NET applikasjoner ønsker man alltid å kunne oppdage feil ved kompilering og dermed unngå overraskende runtime feil. Grunnleggende funksjonalitet i MVC2 rammeverket tar strenger som argumenter, for eksempel man har en metode (en såkalt *Action*) i en *Controller* som kalles *Index* og man ønsker å sende brukeren til denne Actionen. Da blir man nødt til å skrive navnet på metoden eksplisitt ”Index”. Om man endrer navnet på denne metoden i etterkant så vil omsendingen feile og feilen blir ikke oppdaget ved kompilering.

Løsningen var å bruke .NET sin *Expression*[2] klasse som argument istedenfor. Da kan man ta selve metodekallet som argument i *Expression* klassen og senere brukere refleksjon for å finne metodenavnet. Dette gir oss kompileringsfeilmelding om metoden endrer navn i fremtiden. Denne løsningen er brukt samtlige steder i systemet, ikke bare i *Controllers*.

En ulempe er at dette er naturligvis treigere enn og direkte å spesifisere metodenavnet, men tommelregelen vi følger er at bedre vedlikehold går foran fart – som regel.

6.3 Dynamic Query

Jeg støtte på ett problem når jeg brukte LINQ/SQL og skulle dynamisk sette en OrderBy verdi. Problemet er at det innebygde LINQ/SQL rammeverket ikke støtter dynamiske verdier men tar bare *Expression* uttrykk. Løsningen var ett bibliotek funnet på nettet kalt *Dynamic Query*[6] som støtter dynamiske spørringer med LINQ/SQL.

6.4 Logging av feil

En av kravene i systemet var å loggføre enkelte typer feil, slik som database (SQL) feil og e-mail sending (SMTP) feil. En løsning var å ramme inn hvert eneste SQL kall med try-catch blokker, men det hadde vært veldig upraktisk. Jeg valgte å utnytte *Application_Error* hendelsen som utløses hver gang ett unntak (exception) ikke blir behandlet. I denne metoden kunne jeg sjekke typen på unntaket og loggføre den om ønsket.

6.5 Testing av statiske klasser…

… fungerer veldig dårlig. Jeg måtte refaktorere en haug med statiske kall til å bruke objekter istedenfor klasser for å kunne mocke og teste skikkelig.

6.6 Testing av innebygde MVC klasser

Enkelte klasser slik som *System.Web.Mvc.UrlHelper*[2] krever mange underliggende variabler og instansierte objekter satt for å kunne utføre mange av dets metoder. Dette gjør at det blir ett herk å mocke og teste disse klassene. Ved å bruke [4]Decorator mønsteret kan man gjøre livet litt enklere. Man dekorerer problemklassene med en overliggende klasse og den overliggende klassen (dekoratøren) definerer alle metodene som originalt blir brukt på den underliggende

klassen. Dekoratøren kan da simpelthen definere metodene som *virtual* og det blir en bris å teste dem.

6.7 Enkoding trøbbel

Under live prototype testingen så vi at tegnene æ, ø og å ble gjort om til spørsmålstegn ved prosessering av skjemadata. Dette er en uakseptabel situasjon når mange av kundene er norske. Det var ett klassisk tilfelle av ”hvordan debugger jeg dette?” ettersom skjemaet sender dataen til webserveren og som med det ikke er bare-bare å slå på debuggern i Visual Studio og se hva som kommer inn. Men dog det så jo ut som en elementær enkoding sak. Jeg klarte å reprodusere problemet ved å bruke samme enkoding som Mamut brukte på sine landingssider (ISO-8859-1) og jammen var det årsaken til problemet. Dessverre blir ikke kodesettet sendt til serveren med post data, dermed blir det noe mer problematisk ettersom man ikke kan dekode dataen basert på kodesettet som ble brukt for innsending.

Løsningen var halvveis ”hacky” og halvveis skalerbar. Systemet leter etter en feil i strengen (ukjent tegn) for å så tolke skjemaet som ISO-8859-1 hvis ett ukjent tegn blir funnet. Om dette ikke er tilfelle blir vanlig enkoding brukt (vanligvis UTF-8).

6.8 Timer-out!

Det ser ut som at klassen *System.Timers.Timer*[2] som jeg brukte til automatisk eksporterings nedtelling har samtlige problemer ved seg. Blant annet sluker den alle feil som blir kastet i sine *Elapsed* hendelser[19]. Dette tilsier at feilene ikke propagerer opp til applikasjonen og feilbehandling på nivå over er nytteløst. Ett annet problem er av skikkelig hodebry sorten, nedtellingen nekter å starte på automatiske eksporterings nedtelleren av en eller annen grunn.

Jeg tror det beste er å slutte å bruke *Timers.Timer* og heller begynner å bruke *System.Threading.Timer* klassen.

7. Tidsforbruk og registrering av timer

7.1 Timeregistreringssystem

Under hele prosjektperioden førte jeg timeliste for oppgaver gjort i sammenheng med hovedprosjektet. Timeføringen ble gjort i MS Excel og der brukte jeg ett hjemmelagd oppsett bestående av to ark: oppgaveliste og timeliste. Oppsettet var veldig inspirert av relasjonsdatabaser; det brukte IDer for å binde oppgaver sammen med timeregistreringene slik at det er mulig å kalkulere hvor mange timer hver enkelt oppgave tok. Det støttet også flere nivåer med oppgaver; det vil si at oppgaver kunne være understått andre oppgaver, for eksempel er det en oppgave kalt *Brukerdokumentasjon* som er understått *Prosjektrapport*.

Arbeidsliste

ID	ParentID	Hva	Kortversjo	Frist	Arbeid gjort	Kommentar
1	5	Statusrapport		29. okt. 2010	100 %	
2		Finne prosjekt		19. nov. 2010	100 %	
3	5	Prosjektskisse		3. des. 2010	100 %	
4	5	Forprosjekt		28. jan. 2011	100 %	
7		Prosjektmandat		26. jun. 2011	100 %	
11	7	Lag modell for dagens situasjon + ønsket situasjon			100 %	
14	20	Oppdater prototype (etter møte på 26.01.2011)			100 %	

Utdrag fra oppgavelisten

Som man ser er hver oppgave identifisert ved kolonnen *ID* som er tvillingen med *Primary key* kolonnen (hver rad må være unik) i en relasjonsdatabase. *Parent ID* kolonnen definerer hvilken oppgave den er understått, hvis tilfelle. Feltet *Hva* definerer hva oppgaven går ut på, mens de andre feltene bidrar til å definere oppgaven ytterligere.

Timeliste

Oppgave ID	Hva	Kortversjon	Antall timer	Dato	Kommentar
2	Kontaktet oppdragsgiver og blitt enig om prosjekt		5	2010-12-05	
3	Jobbet med prosjektskisse		2	2010-12-08	
9	Møte med oppdragsgiver		1,5	2011-01-21	
4	Laget utkast til forprosjektrapport		3	2011-01-21	
7	Jobbet med prosjektmandat		3	2011-01-24	
8	Møte med veileder		1,5	2011-01-25	
9	Møte med oppdragsgiver		1,5	2011-01-26	
4	Jobbet med forprosjekt		1	2011-01-26	
14	Oppdaterte prototypen		3	2011-02-01	
8	Møte med veileder		1	2011-02-01	
11	Laget modell for dagens og ønsket situasjon		2	2011-02-02	
10	Laget use case diagram for funksjonaliteten i systemet		0,5	2011-02-02	

Utdrag fra timelisten

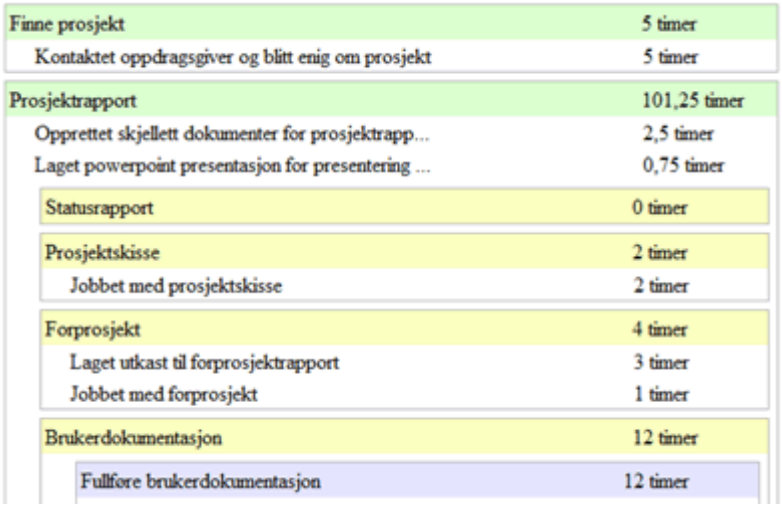
Her ser vi at hver timeinnføring ikke har en egen *ID*, men er bundet til en *Oppgave ID* - altså til en oppgave. *Antall timer* kolonnen definerer antall timer som er registrert for arbeidssesjonen. *Dato* forteller når sesjonen ble utført.

7.2 Timefordeling

Med dette hjemmelagde excel systemet som utgangspunkt var det mulig for meg og dynamisk å utregne hvor mange timer hver oppgave tok, samt hver underoppgave. En ferdigbehandlet oversikt over timebruken for prosjektperioden kan sees i tabellen under:

Oppgave	**Timer**
Finne prosjekt	5
Prosjektrapport	101,25
Prosjektmandat	6,5
Møte med veileder	5,5
Møte med oppdragsgiver	6,75
Implementasjon	347
Annet	2,5
Total	**474,5**

Merk at denne oversikten bare inkluderer toppnivå oppgavene og ikke underoppgavene. For å se en grafisk behandlet versjon kan denne finnes som vedlegg til prosjektrapporten. Hele excel dokumentet kan finnes i program cden som følger rapporten.



Sniktitt på grafisk behandlet timeliste.

8. Kravspesifikasjonen

8.1 Dens rolle

Kravspesifikasjonen har hatt en mangfoldig rolle for systemet. Den har vært til god veiledning for GUIet av systemet og den har også lagt fundament for innholdsstruktur og database struktur. Ellers har den også vært nyttig for å kommunisere systemet med oppdragsgiver og veileder og har derfor hatt en direkte effekt på utviklingen av systemet.

Den er også viktig for fremtidige utviklere/vedlikeholdere av systemet ettersom den inneholder spesifikasjoner, defineringer og forklaringer på løsninger som er gjort i systemet. Systemet er altså tett definert av kravspesifikasjonen og har oversikt over essensielle detaljer om hvordan det fungerer.

8.2 Endringer

Her følger en revisjonsliste for kravspesifikasjonen.

Rev.	Dato	Endringer
1	25.01.2011	• Skjellett med innholdsfortegnelse
2	02.02.2011	• Lagt til database tabell oversikt og diagram • Lagt inn skjermbilder av prototypen med forklaring av grensesnittet
3	17.02.2011	• Oppdatert databasefelt og database diagram • Lagt til innholdsstruktur og diagram ○ Har segmentert innholdet i seksjoner • Lagt til hovedmeny og samtlige andre sider • Organisert sidene i funkspekken etter seksjoner slik at det blir ryddigere
4	26.02.2011	• Oppdatert database felt
5	27.02.2011	• Endret database felt og tabeller til å bruke skrivemåten Pascal • La til ”Country” felt på ”Forms: List all” siden
6	03.03.2011	• La til to felt i ”Export settings” siden
7	05.03.2011	• Endret database feltet ReplyEmailType (varchar) til ReplyEmailUseHtml (tinyint) • Lagt til Name databasefelt i Fields tabellen • Laget en FieldsFormdata tabell med FK til Formdata og fjernet de ”hard-kodete” feltene Company, Name, Email osv som var i Formdata tabellen • Lagt til DateLastEdit i Forms tabellen
8	10.03.2011	• Lagt til flere database tabeller og forhold • Oppdatert database tabeller og felter med ytterligere definisjonsdetaljer • Lagt til Logging seksjon • Oppdatert innholdsstruktur • Oppdatert prototype og forklaringer
9	12.03.2011	• Omstrukturert rapporten • Lagt til innhold i introduksjonsdelen som forklarer systemet og formålet • Oppdatert funksjonalitetsaktiviteter

		• Lagt til detaljer om datavekslingen mellom system og webskjema • Lagt til ressursoversikt • Oppdatert og lagt til use case diagram • Lagt til funksjonalitetsdel i funkspekken (funksjonalitetsdel fra tidligere versjon er splittet opp til andre deler)
10	16.03.2011	• Lagt til spesifiseringsutgangspunkt for generering av CSV filer • Lagt til spesifikasjoner for generering av skjema • Lagt til spesifikasjoner for prosessering av skjema • Lagt til spesifikasjoner for loggføring • Lagt til spesifikasjoner for feilbehandling
11	18.03.2011	• Lagt til Name felt i CustomUrlParametersFormdata • Utført oppdateringer etter kommentarer fra oppdragsgiver • Lagt til spesifikasjoner om automatisk eksportering • Gjort endringer i loggføringspesifikasjonen • Oppdatert prototype
12	25.03.2011	• Oppdatert ressurser • Oppdatert automatisk eksportering spesifikasjoner • Ytterligere tweaks i spesifikasjoner • Oppdatert prototype
13	29.03.2011	• Oppdatert prototype • Oppdatert databasefelt • Lagt til spesifikasjon og forklaring for cookie login • Lagt til spesifikasjon og forklaring for generering av klientside validering kode på serversiden
14	10.04.2011	• Oppdater database felt • Oppdatert prototype
15	11.05.2011	• Oppdatert database felt • Lagt til spesifikasjon for automatisk eksportering email
16	20.05.2011	• Endret automatisk eksportering spesifikasjoner • Endret generering av CSV fil spesifikasjoner • Fylt ut mer i introduksjonen • Fylt ut om MVC • Lagt til tittelside

9. Konklusjon

Systemutviklingen har vært veldig vellykket selv om jeg jobbet alene gjennom hele perioden. Det aller meste av funksjonaliteten sitter og vi har allerede begynt å teste systemet i ett produksjonsmiljø. Dette viser at systemet er stabilt og snart modent nok til å brukes i en ekte setting. Jeg er fornøyd med å ha kommet frem til ett resultat som innfridde i hvert fall minimumsmålene, og forhåpentligvis er kvaliteten der til at systemet overlever.

Prosessen har vært tidskrevende, men veldig interessant og lærerik. Spesielt har jeg lært mye om C# språket og MVC(2), alt fra å bli kjent med rammeverk klasser til oppdagelse av nye muligheter med språket og ikke minst anvendelse av disse.

Under følger en noe sporadisk liste over hva jeg har lært:

- Å lage en MVC(2) strukturert applikasjon i C# / .NET.
- Å bruke generic Func delegates for å kalle medlemmer (members) av objekter. Dette gjør at man kan for eksempel sende inn ett lambda uttrykk som argument og få ut resultatet av kallet uten fare for runtime error (feilen blir funnet i kompileringen). Før brukte jeg refleksjon med strenger av medlemmene for å oppnå det samme, men det har åpenbare svakheter.
- Å bruke attributter.
- Å lage og bruke utvidelsesmetoder.
- Å bruke og utvide det innebygde feilmeldingssystemet for MVC (ModelState).
- Å unngå store forsinkelser grunnet trivielle feil i revisjonskontrollsystemet. Ved å være forsiktig under flytting og sletting av mapper, samt committe ofte, så slipper man å bruke mye tid på å fikse revisjonssystemet til å ikke lage feilmeldinger.
- Å bruke linq uttrykk for å minimalisere kode og øke leseligheten.
- Å publisere en lokal Microsoft SQL server database struktur til en ekstern database.
- Å lage, installere, starte, stoppe, uinstallere og ikke minst debugge en windows service.
- Å sette opp eksportering av rapporter med intervall.
- Lage enhetstester for en MVC strukturert applikasjon.
- Skrive enhetstester med NUnit og RhinoMocks.

10. Referanseliste

10.1 Referanser

1. *http://www.asp.net/mvc* - MVC - Introduksjon til MVC
2. *http://msdn.microsoft.com/* - Microsoft Developer Network - C#/.NET kodehjelp og –referanse
3. *http://stackoverflow.com/* - Stackoverflow.com - Kodehjelp
4. *Design Patterns - Elements of Reusable Object-Oriented Software*, s. 175-184
5. *http://www.mamut.com* – Mamut ASA
6. *http://weblogs.asp.net/scottgu/archive/2008/01/07/dynamic-linq-part-1-using-the-linq-dynamic-query-library.aspx* - Dynamic Query
7. *http://msdn.microsoft.com/en-us/vstudio/aa718325* - Visual Studio
8. *http://en.wikipedia.org/wiki/Microsoft_Windows* - Microsoft Windows
9. *http://en.wikipedia.org/wiki/C_Sharp_%28programming_language%29* - C#
10. *http://en.wikipedia.org/wiki/Apache_Subversion* - Subversion
11. *http://en.wikipedia.org/wiki/Dropbox_%28service%29* – Dropbox
12. *http://en.wikipedia.org/wiki/Microsoft_SQL_Server* - Microsoft SQL Server
13. *http://www.bumsflied.co.cc/Save-ASPNET-MVC-2-in-Action-best-buy* - Bilde av "ASP.NET MVC2 – In Action" bokomslag
14. *http://www.dsoftwares.com* - Visual Studio logo
15. *http://www.gaj-it.com/19333/microsoft-office-available-free-online/* - MS Office logo
16. *https://chrome.google.com/webstore/detail/cleemiokmnpncbdoepicpphinodgekfi?hl=no* – Dropbox logo
17. *http://trendsupdates.com/microsoft-may-help-obama-administration/* - Microsoft logo
18. *http://travelinformationindia.blogspot.com/2010/11/kerala-watrerfalls.html* - Bilde av foss
19. *http://connect.microsoft.com/VisualStudio/feedback/details/286105/system-timers-timer-swallows-exceptions* - System.Timers.Timer klassen "svelger" alle kastede unntak

10.2 Litteratur

- Palermo, Scheirman, Bogard, Hexter, Hinze (2010). *ASP.NET MVC2 - In Action*
- Crane, Pascarello, Darren James (2006). *Ajax - In Action*
- Gamma, Helm, Johnson, Vlissides (2009). *Design Patterns - Elements of Reusable Object-Oriented Software*
- Ann-Mari Torvatn (2007). *Dokumentasjonsstandard - Hovedprosjekter i data/IT*

11. Stikkordliste