BACHELOROPPGAVE:

STYRESAKSDATABASE

FORFATTER(E):
Paul Magne Lunde
Rino Falstad
Simen Backstrøm

Dato:

Gjøvik 20.05.09

SAMMENDRAG AV BACHELOROPPGAVE

Tittel:	Styresaksdatabase(SSD)	Nr. :
		Dato : 20.05.09
Deltakere:	Paul Magne Lunde	
	Rino Falstad	
	Simen Backstrøm	
Veileder:	Tom Røise, Høgskolen i Gjøvik	
Oppdragsgiver:	Popkorn AS	
Kontaktperson:	Kai Arne Aspeli	
Stikkord (4 stk)	Styresaksdatabase, Sakshåndtering, ASP.NET, C#	

Antall sider: 66 + vedlegg	Antall bilag: 8	Tilgjengelighet: Åpen

Kort beskrivelse av bacheloroppgaven:

En styresaksdatabase har som hensikt å lette jobben ved registrering av saker, effektivisere sakshåndtering og å gjøre oppslag i gamle saker mindre tidkrevende. Popkorn har fått forespørsel om en slik løsning fra Løiten Almenning som til daglig mottar saker som skal registreres og behandles.

Løsningen er i første rekke utviklet med tanke på de krav som stilles for bruk hos Løiten, men vi har også lagt inn konfigureringsmuligheter som gjør at andre typer bedrifter kan benytte løsningen. Denne tilretteleggelsen er gjort ved hjelp av mulighet for å velge hvilke sakstyper det er ønskelig å registrere. I tillegg har vi utviklet løsningen ved hjelp av en modultankegang som gir åpning for spesialtilpassede utvidelser der dette er ønskelig. Sammenlagt blir dette et system som tilfredsstiller de krav Løiten har, og ved minimal konfigurasjon kan det brukes av andre typer bedrifter.

Hovedfunksjonene i løsningen er registrering og redigering av saker, opprettelse av styremøter og gjennomføring av disse. For å gjøre oppgaven med utsendelse av innkalling til møter og påminnelser enklere, er det mulighet for utseendelse av innkalling på e-post eller som brev, og påminnelser på e-post eller SMS.

Løsningen er utviklet i Microsoft Visual Studio 2008 ASP.NET med C# som programmeringsspråk. Årsaken til bruken av ASP.NET er krav om et webgrensesnitt.

FORORD

Prosjektoppgaven ble presentert av oppdragsgiver på Høgskolen i Gjøvik allerede oktober 2008 i sammenheng med emnet Objektorientert systemutvikling. To av gruppens nåværende medlemmer jobbet da med innledende kartlegging av prosjektet. Problemstillingen var tiltalende ettersom hele systemet skulle utvikles fra grunnen og som en webløsning. Samtidig var det å jobbe med en oppdragsgiver som driver med applikasjonsutvikling tiltalende for oss som programvareutviklere. Men det mest fascinerende med oppgaven var at løsningen skulle utvikles på .NET plattformen, som ingen av prosjektets medlemmer hadde erfaringer med fra før, men gjerne ville lære.

Ettersom den tidlige kartleggingen var av en overordnet og generell art, måtte vi foreta ytterlige kartlegging i starten av prosjektperioden for å få en mer prosjektspesifikk vinkling. Oppgaven har bydd på mange utfordringer som har krevd mye tid og arbeid. Vi har gjennom hele prosjektperioden jobbet jevnt og trutt med utviklingen, noe som har gitt oss et meget stort faglig utbytte.

Veileder for prosjektet har vært Tom Røise etter ønske fra gruppen. Vi vil takke for et godt samarbeid med meget god veiledning og gode råd.

Vi vil takke oppdragsgiver Popkorn og kontaktpersonene Kai Arne Aspeli og Stian Bakken spesielt. Samarbeidet har vært utmerket, støtten og oppfølgingen har vært upåklagelig.

Gjøvik 20.05.09

Paul Magne Lunde	*Rino Werner Falstad*	*Simen Tveit Backstrøm*
_____	_____	_____

INNHOLD

1 INNLEDNING

1.1 PROBLEMOMRÅDE

Et styresaksdatabasesystem er et system for å registrere, behandle og avslutte saker. Systemet skal kunne benyttes av bedrifter som ønsker elektronisk kontroll over saker og behandlingen av disse. Dette er et system som i stor grad vil kunne effektivisere behandling av saker hos bedrifter. For en mer omfattende oppgavebeskrivelse, se Forprosjektrapport i vedlegg F.

1.2 MÅLGRUPPE

Prosjektrapportens målgruppe er sensor, oppdragsgiver, studenter, veileder og andre ved Høgskolen i Gjøvik.

Målgruppen for løsningen som skal utvikles er små og mellomstore bedrifter i Norge, i første omgang for Løiten[1-1] og andre *allmenninger*. Popkorn[1-2], som blir eier av løsningen ser også muligheten for å bruke denne i sin egen bedrift.

Løiten Almenning er et foretak som er over 170 år gammelt. De eier hytter, jaktområder og lignende, men hovedbeskjeftigelsen er formidling av økonomiske midler til gårdene som ligger under allmenningen. Disse midlene må *hjemmelshaverne* søke om, det er i denne sammenhengen styresaksdatabasen kommer inn i bildet. Her er det et stort forbedringspotensial siden alle saker i dag behandles manuelt og arkiveres i papirformat. Løiten ser også på applikasjonen som en måte å redusere tidsbruket for beslutning av saker betraktelig.

Popkorn er en bedrift med kontorer i Hamar og på Lillehammer som utvikler skreddersydd programvare. Når Løiten så behovet for en effektivisering av saksbehandlingen sendte de en forespørsel til Popkorn. Som en følge av forespørsel angående studentoppgaver ved HiG ble oppgaven lagt frem for oss.

For lesing av rapporten er det forventet et kunnskapsnivå som ligger et trinn høyere enn gjennomsnittet ettersom det benyttes en del faguttrykk fra systemutvikling.

1.3 MÅL

Prosjektet har to hovedmål, men et tilleggsmål er muligheten for konfigurering mot andre typer organisasjoner og bedrifter.

Det første målet er å gi Løiten en elektronisk løsning for håndtering av saker og møter, ettersom gamle saker i dag hoper seg opp i permer med papirer. Praksisen er i dag at protokollen underskrives på neste styremøte. Med et system som gjør at man kan behandle saker og skrive ut en protokoll ved avslutningen av et møte, vil behandlingstiden av søknader kunne kortes ned betraktelig.

Målet for Popkorn sin del er å få et system som først og fremst tilfredsstiller bestillingen fra Løiten, men også å få et konfigurerbart styresakssystem for videresalg. Ved videresalg kan det også oppstå muligheter for utvikling av skreddersydde moduler for ulike bedrifter. Popkorn ser i tillegg en mulighet for at løsningen kan brukes for deres egne styremøter.

Resultatmålet for gruppen har vært å utvikle en feilfri og komplett applikasjon som tilfredsstiller alle krav fra Løiten og som kan brukes med minst mulig endringer. Vi har også hatt som mål å utvikle muligheten for konfigurering mot andre bedrifter slik at løsningen blir mye mer generell. I tillegg har en viss grad av tilrettelegging for at løsningen senere kan bli desentralisert vært en målsetting.

1.4 GRUPPAS BAKGRUNN OG KOMPETANSE

Gruppemedlemmene tar alle en Bachelor i Programvareutvikling og har mye av de samme interesser når det kommer til det faglige og tekniske. Den faglige bakgrunn er derfor innen datatekniske og systemutviklingsområder. Gruppemedlemmene hadde lite kjennskap til en allmennings gjøremål, domenekunnskap innen dette området kommer derfor fra møter med Popkorn og Løiten.

Gruppen har gjennom studiet fått erfaringer med flere programmeringsspråk og utviklingsmiljø, men Microsoft .NET og C# har vi ikke hatt undervisning i. Derfor har innlæringen av dette foregått på egenhånd med støtte fra oppdragsgiver. På server siden har vi vært nødt til å installere Microsoft Server 2003 og SQL Server 2008 på egenhånd ettersom vi ble nødt til å bytte ut serveren vi fikk låne av Popkorn. Dette var også en ny erfaring for gruppen.

For å få tak i data fra databasen som er av type MSSQL, må *T-SQL* læres inn fra grunn av ettersom ingen i gruppen har tidligere erfaring med dette heller.

1.5 TIDLIGERERE ARBEIDER

Noe av det tidligere arbeidet er tatt med i denne rapporten. Vi har brukt use case-diagrammet og alle detaljerte use case fra kravspesifikasjonen, med små endringer. Vi har også tatt med deler av supplementær spesifikasjon, domenemodell, logisk oversikt og anvendelse diagram. På sistnevnte er det gjort mye endringer.

1.6 UTVIKLINGSMODELL

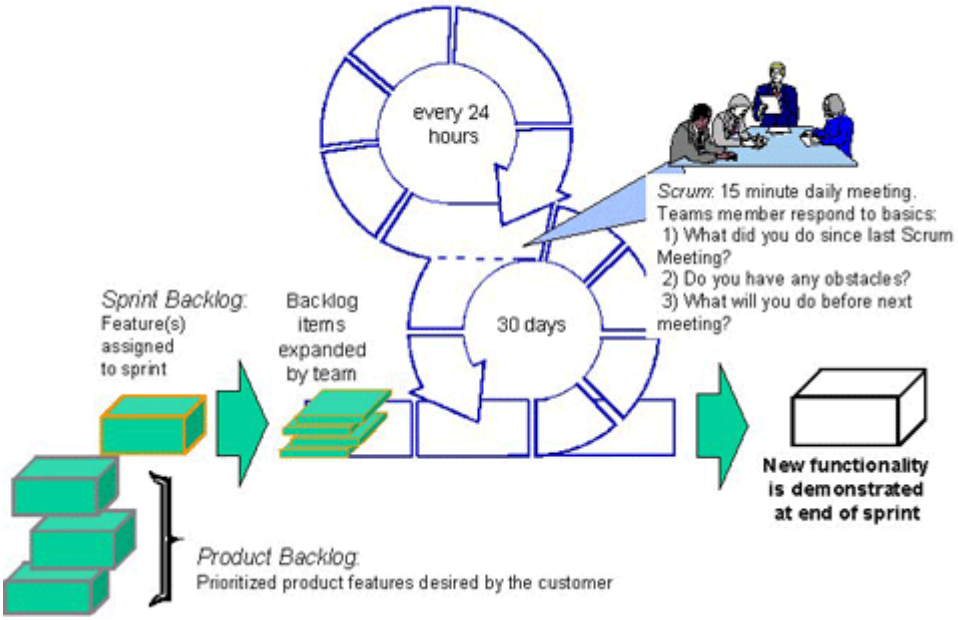
I utviklingsprosessen er det fornuftig å benytte en systemutviklingsmetodikk. Denne kan angi rekkefølge og strukturen, samt gi en angrepsvinkel på problemstillinger. Derfor er det viktig å finne en utviklingsmodell som passer til det spesifikke prosjektet.

Vi har diskuterte forskjellige systemutviklingsmodeller og kom frem til at en smidig modell som takler endringer bra er ideell. Vi vurderte to som aktuelle; XP og Scrum.

XP krever mye kontakt med oppdragsgiver, dette kan fort føre til problemer i forhold til ulikt tidsskjema hos oppdragsgiver og oss. Parprogrammering fungerer heller ikke optimalt med tre gruppemedlemmer, og ville gitt litt problematisk arbeidsfordeling og tregere arbeidstempo.

Etter utelukkingen av XP så vi at Scrum ville være den som passet prosjektet best. Den tar seg godt av styring av prosjektet når det kommer til struktur og tidsrammer. I tillegg oppfordrer Scrum til oppdeling av oppgaver i moduler som det jobbes med i en fastsatt tidsperiode (sprint). Oppgavene skal deles opp i arbeidsbolker på 4-16 timer, noe som gir et bra overblikk over gjenstående og utført arbeid i en sprint. Figur 1-1 viser hvordan Scrum skal organiseres med sprinter under utviklingen. Selv om figuren viser en 30-dagers sprint er det åpning for endring i tid, på grunn av tidsbegrensninger har vi derfor valgt å ha 14-dagers sprinter.

Ettersom Scrum gir stort spillerom for innspill fra andre utviklingsmodeller har vi valgte å benytte oss av use case hentet fra RUP, som er veldig god på dokumentasjon.



Figur 1-1 Scrum diagram[1-3]

1.7 RAMMER

ARBEIDSFORM

Gruppen har fått tildelt grupperom A018B sammen med en annen gruppe. Dette rommet har gruppemedlemmene stort sett benyttet som arbeidsplass under hele prosjektet. Rutiner og regler beskrives i Forprosjektrapporten i vedlegg F.

Ved oppstart av prosjektet ble det avtalt å ha møte med veileder hver tirsdag, med muligheter for avlysning. Det var ikke avtalt faste møter med oppdragsgiver, men meningen var å prøve å få til et møte etter hver sprint og også ved behov.

Gruppen har ført arbeidslogg for tidsforbruk hver uke (vedlegg C), i tillegg har møtereferater og annen dokumentasjon blitt utarbeidet fortløpende.

FREMDRIFTSPLAN

Gant-skjema med planlagt og faktisk forløp finnes i vedlegg B.

1.8 ROLLER

- Paul M. Lunde har vært prosjektleder og ansvarlig for sammensetting av dokumentasjon.
- Rino W. Falstad har vært ansvarlig for visuelle planer, skjemaer og dokumentasjonsdiagrammer.
- Simen Backstrøm har hatt ansvar for utstyr og oppretting/vedlikehold av nettsiden.
- Tom Røise, høgskolelektor HiG, har vært veileder for gruppen i prosjektperioden.
- Stian Bakken, fagsjef hos oppdragsgiver Popkorn, har vært støttespiller under innlæring av Visual Studio, .NET og C#.
- Kai Arne Aspeli, kontaktperson hos Popkorn.
- Johan Fisher, bestiller av løsningen.

1.9 RAPPORTEN

ORGANISERING AV RAPPORTEN

Høgskolen i Gjøvik gir et ferdig oppsett for hvordan en hovedprosjektrapport bør se ut. Denne malen med egne supplementer er brukt som utgangspunkt for oppbygningen av denne rapporten.

PRAKTISKE FORHOLD

LAYOUT

Rapporten er inndelt i kapitler med underkapitler, det er to nivåer av disse som er nummerert og vises i innholdsfortegnelsen. Skrift med *kursiv* vil være ord som finnes i terminologilisten (Vedlegg A), disse blir kun uthevet første gangen de forekommer. Skrifttypen ellers vil være standard for hele rapporten. Vi benytter oss av en noe modifisert versjon av numerisk referering med ”kapittelnummer” – ”referansenummer i kapittel”.

Sidene først i rapporten er i henhold til malen nummerert med romertall, mens det fra første kapittel er vanlig sidenummerering.

Bilder som er med i rapporten har en bildetekst med figurnummer og tittel. Nummereringen av disse er gjort under selve bildet med en beskrivelse. Figur nummereringen er på formen ”kapittelnummer” – ”figurnummer i kapittel”.

KAPITTELOPPSUMMERING

Rapporten er delt inn i 6 kapitler, i noen av disse er store og detaljerte skjema lagt ut i vedlegg. Det anbefales å lese disse når det refereres til dem. Alle vedlegg er tilgjengelig for lesere av rapporten ettersom de ikke inneholder noen form for konfidensielt materiale.

KAPITLER:

1. **Innledning**
   Inneholder overordnet beskrivelse av formålet med prosjektet, beskrivelse av arbeidsmetoder og en beskrivelse av rapporten generelt.
2. **Kravspesifikasjon**
   Inneholder alle krav til prosjektet før utviklingen startet.
3. **Design**
   Inneholder designet av løsningen, altså hvordan oppgaven vil bli løst.
4. **Implementering**
   Inneholder beskrivelse av de viktigste delene i løsningen i forhold til kravspesifikasjonen og design. Denne inneholder også en beskrivelse av verktøy og valg av disse.
5. **Testing**
   Inneholder beskrivelse av white- og blackbox testing og konklusjon.
6. **Avslutning**
   Inneholder drøftinger, evalueringer, forbedringspotensialer, videreutvikling og konklusjon.

VEDLEGG:

VEDLEGG A - Terminologiliste

VEDLEGG B - Framdriftsplan

VEDLEGG C - Timelogg, møtereferater

VEDLEGG D - Detaljerte funksjonelle krav

VEDLEGG E - Designskjemaer

VEDLEGG F – Forprosjekt

VEDLEGG G - CD-rommens innhold

VEDLEGG H – Kontrakter

2 KRAVSPESIFIKASJON

Her følger kravspesifikasjon og hva systemet skal inneholde. Dette omhandler ikke det som har blitt gjort, men kun det som skal bli gjort. Dette er altså krav før selve systemet ble utviklet.

2.1 BRUKERBESKRIVELSE

Løsningen skal bestå av en webapplikasjon for å behandle saker og styremøter. Kravene kommer fra en presentasjon og møter med oppdragsiver.

KRAV TIL SYSTEMET

Systemet skal kunne behandle saksbehandlers ønske om å registrere/redigere saker og møter, samt gjennomføre møter.

Ved saksregistrering skal eiendommer kunne knyttes opp mot saker og eventuelle vedlegg skal kunne legges til. Selve saken skal inneholde tittel, beskrivelse og et vedtaksforslag. Type sak velger man ut ifra konfigureringen som er gjort. I tillegg skal det sendes en SMS til søker, hvis han eller hun har mobilnummer registrert i databasen, med informasjon om at saken er registrert og hvilket saksnummer den har.

Ved møteregistrering skal saker som er løse, knyttes til møte og muligheter for å endre møtebesetning. Når møte lagres skal det automatisk etter et forhåndsinnstilt tidsintervall sendes en møteinnkalling på e-post til de av møtebesetningen som har dette. Møteinnkallinger består av en generert PDF med møtets detaljinformasjon. SMS-varsel sendes nærmere møtets dato. Tidsrommet for hvor lang tid i forveien e-post og SMS sendes, skal bestemmes via egne innstillinger. Man skal også ha mulighet til redigering av saker og møter.

Når man gjennomfører møter skal brukeren/styret kunne godkjenne/underkjenne saker. Etter endt møte skal man kunne generere møteprotokoll som PDF.

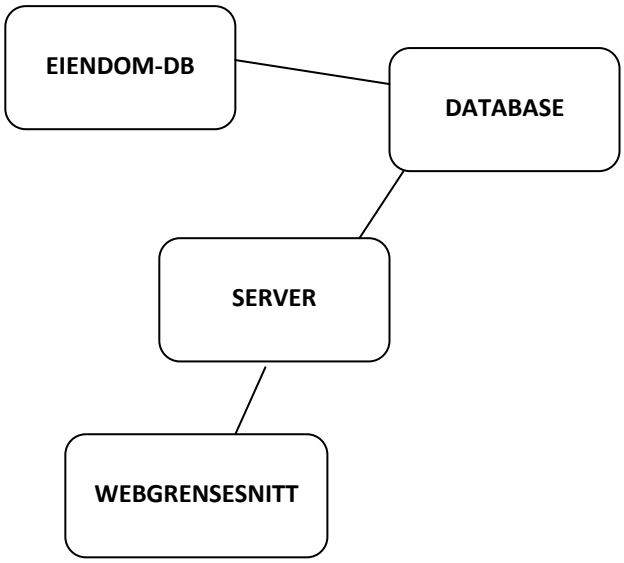
Persistente sider vil være viktig i tilfelle feiltrykk hvis man for eksempel ikke har lagret det man har gjort. Når nettleseren lukkes, må man også passe på at data lagres unna slik at de ikke mistes ved feillukking.

Det er ikke gitt krav fra Popkorn om noen form for *internasjonalisering*. Dermed vil systemet kun ha norsk språk i brukerinteraksjonen uten mulighet for endring av språk. Det er heller ikke noe krav om å kunne generere dokumentasjon ut fra kildekoden, men det er viktig med god kommentering.

OMGIVELSER

Hovedpoenget med applikasjonen er at brukere av systemet skal kunne korte ned behandlingstiden på saker, og kunne gå over til en elektronisk lagring av saker. Det at applikasjonen tar seg av elektronisk påminnelser og innkallinger via SMS/e-post vil også være en stor fordel. Systemet er i første omgang laget for allmenninger, men kan konfigureres til en hvilken som helst mellomstor eller liten bedrift.

Applikasjonen er ment å skulle brukes som en intranett-løsning i første omgang. Den skal ha webgrensesnitt som gjør den plattformuavhengig.



Figur 2-1 Systemets omgivelser

SYSTEMETS BRUKERE

Det er ikke fremmet noe krav om at løsningen skal kunne takle flere brukere samtidig, siden det kun er en saksbehandler på Løiten Almenning som skal bruke systemet. Ellers vil antall brukere avhenge av hva slags begrensninger databaseversjonen har.

Vi tar høyde for tre ulike typer brukere i vår løsning:

Superbruker (SU) – ved innlogging får brukeren tilgang til administrasjonsdelen i løsningen med tilgang til å legge til, endre og velge moduler og sakstyper, samt brukerhåndtering. Rettighetene omfatter også saks- og møtehåndtering, men det er ikke tiltenkt at dette skal benyttes av en SU.

Administrator (admin) – denne brukeren har rettigheter for håndtering av registrering og redigering løsningen. I tillegg må møteleder være pålogget som admin for å kunne gjennomføre et møte.

Standard (std) – denne brukertypen er tatt med på tanke på en mulig fremtidig desentralisert løsning. Rettighetene dekker kun innsyn ved møtegjennomføring og generelt innsyn i saker.

BEGRENSINGER

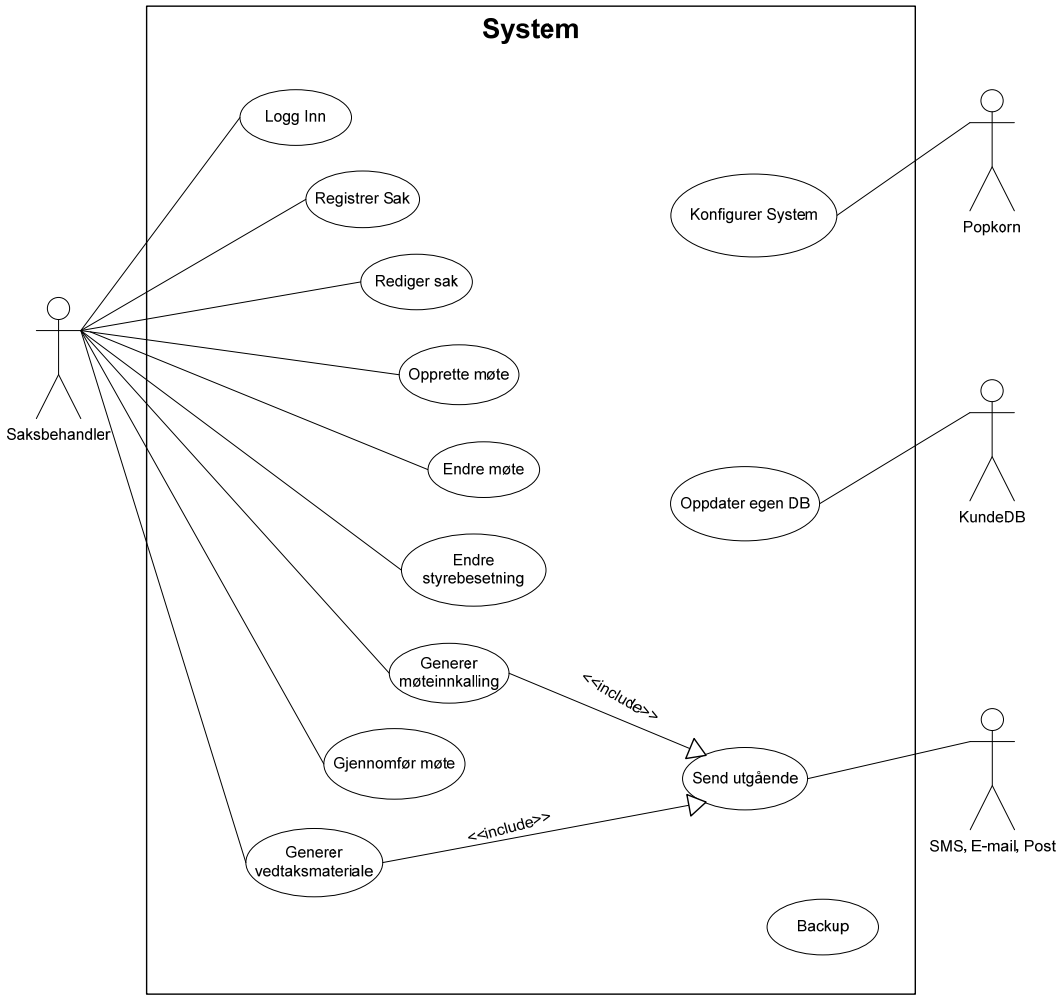
Stemmetelling på saker er ikke nødvendig i denne versjonen.

Applikasjonen skal heller ikke håndtere direkte kommunikasjon med en skanner. Skanning av vedlegg må gjøres manuelt, for eksempel at dokumentene lagres på serveren/nettverket og hentes opp igjen når man registrerer saken.

2.2 OVERORDNEDE FUNKSJONELLE KRAV

Denne delen vil vise en overordnet beskrivelse av de funksjonelle krav i form av use case.

USE CASE DIAGRAM



Figur 2-2 Use case diagram

KundeDB er beregnet for tilfeller der en bedrift har en eksisterende database som ønskes integrert i løsningen.

OVERORDNET USE CASE

Prioriterte use case beskrivelser:

Use Case:	**Registrer sak & Rediger sak**
Aktør:	Saksbehandler (administrator)
Mål:	Saken blir registrert riktig og lagret i databasen
Beskrivelse:	Saksbehandler oppretter en ny sak og legger inn saksmateriale og all informasjon relevant til saken. Her kan eiendommer og hjemmelshavere tilhørende saken knyttes og også vedlegger kan legges til. Systemet oppdaterer så databasen.

Use Case:	**Opprett møte & Endre møte**
Aktør:	Saksbehandler (administrator)
Mål:	Info blir utfylt, saker koblet opp mot møte, møtet blir opprettet og lagret i databasen
Beskrivelse:	Søknader som krever styrevedtak legges til møtet. Festesaker blir automatisk med på møtets saksliste, men kommer bare frem som vedtatte saker. Festesakene er ikke med på selve møtet, men er med i systemet for å protokollføres. Møtets dato og sted blir fastsatt. Møtet blir så lagret.

Use Case:	**Endre styrebesetning**
Aktør:	Saksbehandler (administrator)
Mål:	Styreinformasjon blir utfylt, medlemmer koblet til styret og det blir lagret i databasen
Beskrivelse:	Bruker legger inn informasjon om styreperiode og eventuelle merknader. Personer som skal sitte i styret blir så knyttet opp mot styret. Nye medlemmer som ikke finnes i databasen kan opprettes her. Styret blir så lagt til i databasen.

Use Case:	**Generer møteinnkalling**
Aktør:	System
Mål:	Møtets saker blir lagret i PDF.
Beskrivelse:	Hvis det finnes opprettet møte med saker, genereres møteinnkalling når datoen for møteinnkalling er gjeldende. Systemet henter møtets informasjon og alle saker knyttet til møte. Dette genereres det en PDF av.

Use Case:	**Gjennomfør møte**
Aktør:	Saksbehandler (administrator)
Mål:	Møtets saker blir gjennomgått, bestemt utfall og lagret i databasen
Beskrivelse:	Når møtets dato er dagens dato kan brukeren starte møtet. Saker gjennomgås i en liste. Beskrivelse og vedtak vises. Selve vedtaket kan kopieres fra forslaget, eller skrives nytt og man kan også legge inn egne notater ved saken. Sakens informasjon om eiendommer eller lignende vises frem sammen med vedlegger. Saken kan så godkjennes, underkjennes eller overføres til neste møte. Når møtet avsluttes kan man generere rapport i PDF.

Use Case:	**Generer vedtaksmateriale**
Aktør:	Saksbehandler (administrator)
Mål:	Rapport i PDF-form blir opprettet.
Beskrivelse:	Når møte avsluttes velger bruker å generere rapport. Systemet henter da alle saker som er vedtatt/underkjent, informasjon om møtet og lager en rapport av dette. Rapporten blir så konvertert til PDF.

Use case:	**Send utgående**
Aktør:	<Generer møteinnkalling>, <Generer vedtaksdata>
Mål:	Tar imot PDF-er/meldinger og videresender disse via e-post/SMS
Beskrivelse:	Systemet genererer PDF-dokumenter som sendes som styremøteinnkalling via e-postklient til respektive mottakere. Påminnelse om styremøte sendes til eksternt system via. e-postklienten. Vedtaksmelding sendes også via epost.

Use case:	**Oppdater egen DB**
Aktør:	KundeDB
Mål:	Oppdatere egen database mht. eiendom og hjemmelshaverdata.
Beskrivelse:	Systemets database oppdaterer seg mot organisasjonens database, slik at styresaker kan knyttes opp mot organisasjonens interne kundedataregister. Grunnen er at resten av systemet trenger oppdaterte eierdata. I systemets database er det eiendommer og hjemmelshavere som blir oppdatert.

2.3 DETALJERTE FUNKSJONELLE KRAV

DETALJERT USE CASE BESKRIVELSE

Vi har valgt å ta med to av use case beskrivelsene her. Disse to er ”Gjennomfør Møte” og ”Send Utgående”. Disse to er valgt fordi de er de mest essensielle delene ved systemet og også fordi de er de mest teknisk krevende delene. ”Gjennomfør møte”-caset inneholder hva standardbruker kan gjøre under et møte, mens send utgående bestemmer hva som kan skje når systemet skal sende ut e-poster og SMS.

Se vedlegg D for flere detaljerte beskrivelser av use case.

Use Case:	**Gjennomfør møte**
Aktør:	Standardbruker
Mål:	Holde oversikt over saker som blir behandlet i møte, ajourføre vedtak og sikre at interessenter blir informert i etterkant.
Beskrivelse:	Under hver styresaksmøteprotokoll, referat saksbehandlingsvedtak blir gjort i ”real-time”. Det vil være mulig å endre vedtaksforslag som er blitt gjort i forkant av møtet, eller godta forslag uten endringer. Det vil også være mulig å legge til kommentarer utover vedtaksteksten som vil følge saken, men vil kun vises i protokoll/møtereferat. Når møtet er over vil det være en enkel og brukervennlig måte å generere protokoll med alle vedtatte vedtak, samt eventuelle ekstra kommentarer. Her vil det være kontakt med use caset Generer vedtaksmateriale. Det vil også være kontakt med use caset Endre saksliste, siden det er gitt mulighet for å fjerne/legge til saker.
Type:	Viktig.
Pre betingelser:	Møte opprettet og startet.
Post betingelser:	Oppdaterer egen DB og tilkaller Generer vedtaksmateriale.
Spesielle krav:	

Detaljert hendelsesforløp:	
Aktør:	System:
1. Saksbehandler velger aktuelt møte.	
	2. Viser alle aktuelle saker som skal gjennomgås på valgt møte i en liste på venstre side i grensesnitt.
3. Saksbehandler klikker på sak som skal drøftes.	
	4. Viser tekst som er knyttet opp mot valgt sak. Det vil si vedtaksteksten og anbefalinger gjort av saksbehandler i forkant.
5. Når saken er ferdigbehandlet setter saksbehandler den til vedtatt eller ikke.	
	6. Lagrer vedtaket. Setter saken som åpen (siden det er mulig at ny informasjon skal legges til senere).
7. Vil gå til pkt. 2 i loop helt til det er tomt for	

saker eller møte er ferdig.	
7. Når møtet er ferdig, velges generer møtedokumenter.	
	8. Spør først om det er korrekt at møtet er slutt. Hvis det stemmer, kontaktes Generer vedtaksmateriale (som bl.a. genererer møteprotokoll og hvert enkelt vedtak skrives ut på papir i brevform).
Alternative scenarioer:	
Lovlige alternative scenarioer:	
3a. Mulig å overføre sak til neste møte 1. System fører ikke saken med på en eventuell rapport og saken blir løst fra møtet. 3b. Mulig å legge til ny sak. 1. Bruker må da innom ”Registrer Sak” og registrere og så tilbake å trykke ”oppdater” 5. Det vil være mulig å endre innholdet i vedtakstekst og legge til notater før saken settes til ferdigbehandlet. 1. System linker notat til valgt sak. 6. Default på systemet er at saken settes som åpen, men saksbehandler har mulighet til å overstyre og sette saken som avsluttet. 7. Hvis det er saker som ikke rakk å bli ferdigbehandlet, vil de bli satt som ubehandlet og gitt muligheter for å knyttes opp mot fremtidige møter. 8. Hvis knappen Avslutt møte ble trykket på ved et uhell, og det blir svart Nei på spørsmål om møtet er slutt, går system tilbake til der det var.	
Ulovlige alternative scenarioer:	
1. Hvis ingen sak er valgt, så vil ingen info om noen saker bli vist. 3a. Det er ikke lov å markere flere saker samtidig. Hvis forsøkt å markere flere saker ved å holde ctrl eller shift nede, vil siste sak som er klikket på være den eneste aktive. 3b. Databasefeil under opphenting, eller feil på data, systemet viser feilmelding og ber bruker prøve igjen / tilkalle systemadministrator. 6 & 8. Databasefeil under oppdatering, systemet gir melding om alvorlig systemfeil og at administrator må tilkalles.	

Use case:	**Send utgående**		
Aktør:	«Generer møteinnkalling», «Generer vedtaksdata»		
Interessenter:	Saksbehandler: Vil sende styremøteinkalling og sende vedtaksmelding til søker på respektive vedtak. Styremedlemmer: Vil motta styreinnkalling. Søkere: Vil motta vedtak på søknad.		
Mål:	Sende styremøteinkalling via epost, varsle om styremøte og søknadsprosess vha SMS, og poste materiale skriftlig via skriver/post.		
Beskrivelse:	Systemet genererer rapport/pdf-dokumenter som sendes som styremøteinkalling via epostklient til respektive mottakere. Påminnelse om styremøte sendes til eksternt system via. Epostklienten og sms. Vedtaksmedling sendes også via epost. Sms til søker sendes når sak er lagret.		
Trigger	Trigges automatisk dagen før styremøte for å sende varsling via epost til SMS-tjeneste		
Teknologi og datavarisjonliste:	SQL Express 2008 with Advanced Services: SQL Reporting Services (for genrering av rapporter) og .NET (for epostklient)		
Detaljert hendelsesforløp			
Aktør handling:	Systemrespons:	Epostklient:	Skriverklient:
1. Use caset starter med at aktør initieres som igjen inkluderer dette systemet.			
	2.a Send epost vedr. Styreinnkalling til mottakerliste		
	2.b Send epost vedr. Påminnelse om styremøte		
		3.a Send epost til mottaker	
		Gjenta punkt 3.a for hver mottaker i listen.	
	2.c Send dok. vedr. Styreinkalling til skriver for postgang.		
	2.d Send dok. Vedr Vedtak til skriver for postgang.		
			3.b Utskrift av dok.
			Gjenta punkt 3.b for hver

			mottaker i listen
5. Gir statusmelding til bruker.			
Alternative senario:			
Aktørhandling:	Systemrespons:	Epostklient:	Skriverklient:
Lovlige senario:			
1. Use caset starter med at aktør initieres som igjen inkluderer dette systemet.			
	2. Hvis ikke dokumenter er generert så startes rapportgenereringsfunksjonen (SQL RS)		
	Fortsett på punkt 2.a i hovedsenario.		
*Ulovlig senario **			
		*a. Alle mulige tider, klient feiler	
		1. Loggfør spesifikk feil	
		Gjenta punkt 1 for alle feil	
	2. Loggfør feil		
	3. Gi feilmelding til aktør		
4. Signaliser feil til aktør			

2.4 DETALJERT KRAVSPESIFIKASJON

SYSTEMETS EKSTERNE BRUKERGRENSESNITT

EKSTERNE MASKINGRENSESNITT

Systemet skal lages for skjermer fra 1280 x 800 og 14,1”-skjermer og større. Mindre størrelser vil få scrolling. Applikasjonen vil ha et felt på 1200 x 700 punkter. Grunnen til at det er ønskelig med såpass stort felt er å kunne få plass til alle elementer uten at det er nødvendig med scrolling.

EKSTERNE SYSTEMGRENSESNITT

Serveren som løsningen og databasen ligger må være en Microsoft Server, ettersom det skal benyttes en Microsoft SQL server og det utvikles i .NET 3.5.

2.5 SUPPLEMENTÆR SPESIFIKASJON

Her har vi valgt RUP-metoden *FURPS* for å definere supplementære kravspesifikasjoner. Denne metoden hadde vi grei kjennskap til fra før og vi så på denne som ideell for å få frem de resterende kravene.

FUNKSJONALITET

MODULER

Systemet skal være mest mulig generelt, det er derfor tatt avgjørelsen at det skal lages et eget konfigurasjonssystem som settes opp ved førstegangsbruk for hver bedrift.

Det skal ligge tilgjengelige moduler definert i en tabell i databasen. Når superbruker logger inn får denne muligheten til å velge hvilke moduler som skal benyttes. Det vil også ligge konfigurasjonsmuligheter for sakstyper i databasen. For Løitens’ del vil det blant annet være aktuelt med *festesak* og *bruksrettssak*. Disse passer ikke til alle bedrifter, og vil derfor være mulig å forandre.

SIKKERHET

Som nevnt i 2.1 er det definert 3 typer bruker (superbruker, administrator og standard). Alle typer av brukere må logge seg inn, og deres rettigheter avgjør hva de kan gjør. Dette ivaretar sikkerheten for at data i systemet ikke blir endret av ukyndige.

Det er tatt avgjørelsen at systemet utelukkende skal ligge i bedriftens intranett som standard. Dermed unngås mye av sikkerhetsproblematikken rundt beskyttelse av interne dokumenter. De fleste bedrifter vil gjerne ha en beskyttelse rundt sine styresaksdokumenter. For Løiten sin del er ikke dette noen høy prioritet. Det er ikke noe krav om ytterligere sikkerhet enn brukerinnlogging og intranett. Et aspekt som bør vurderes hvis applikasjon skal brukes av andre er håndtering av utseendelse av møteinnkalling via e-post, her bør det overveies kryptering.

BRUKERVENNLIGHET

MENNESKELIG FAKTORER

Det er ønskelig for systemet at det er oversiktelig og enkelt å bruke. Man kan oppnå dette ved å ha navigasjonen til hovedkategorier synlig og tilgjengelig hele tiden, som for eksempel bruk av fliker. I tillegg vil en snarvei fra forsiden til hver enkelt side være ønskelig.

YTELSE

Server skal være tilgjengelig hele tiden, med lavest mulig nedetid, dvs. at ved maks et møte i året så kan systemet være nede. Dette er ikke et høykritisk system, så det vil være et krav man kan leve med.

Responstid ved bruk av gui-elementer skal ikke være merkbart tregt for bruker, men dette avhenger av nettleseren som brukes. Ved krevende oppgaver skal bruker få informasjon om at løsningen arbeider.

I den spesifikke løsning for Løiten vil det være 5 eller mindre samtidige brukere av databasen, slik at det ikke settes store krav til effektivitet og samtidighetsproblematikk. Men det bør merkes at dette kan måtte tas hensyn til ved en senere anledning i forhold til bruk av systemet i andre firmaer med større brukermasse.

SYSTEMSTØTTE

KONFIGURERING

Som et tillegg har oppdragsgiver ytret ønske om at løsningen skal ha mulighet for en form for konfigurering slik at det er mulig å benytte løsningen i andre typer bedrifter enn Løiten Almenning. Popkorn har blant annet antydet at det kan være interessant å ta den bruk internt i deres egen bedrift.

I utgangspunktet skal det lages moduler med tanke på behovet Løiten Almenning har per i dag. ”Sak” for sakshåndtering og ”Møte” for møtehåndtering.

For å tilfredsstille et slikt krav må det for det første være mulig å velge borte noen sakstyper og legge til nye, alt ettersom i hvilken sammenheng løsningen skal brukes. For eksempel har en sakstype vedrørende eiendomsovertakelse liten verdi for Popkorn, mens det er mulig at det trengs andre sakstyper der som ikke er aktuelle for Løiten Almenning.

Det andre kravet for en konfigurerbar løsning er moduler – på samme måte som sakstyper varierer fra bedrift til bedrift kan behovet for nye moduler oppstå. Det kan tenkes at en bedrift kun vil benytte seg av muligheten for registrering av saker som ikke trenger videre behandling i et møte. Dermed trengs kun ”Sak”, og ”Møte” kan velges bort slik at den ikke vises. Selv om prosjektet har tittelen ”Styresaksdatabase” kan det tenkes at det er aktuelt å utvide med andre moduler som har med administrasjon å gjøre. Om det er ønskelig med en uavhengig modul for økonomi er det mulig å få en slik modul skreddersydd og integrert i den eksisterende løsningen. Disse må da utvikles først.

DESENTRALISERT LØSNING

Etter hvert kan det være aktuelt med en desentralisert løsning der møtedeltakere kan delta på et møte uten å være fysisk tilstede. En slik løsning krever en høy grad av sikkerhet, noe som vi ikke legger vekt på i løsningen ettersom den er tenkt brukt i et intranett. I tillegg krever en desentralisert løsning flere typer brukere med ulike rettigheter i motsetning til en intranettløsning.

2.6 BEGRENSINGER

SOFTWARE BEGRENSNINGER

MILJØ OG SPRÅK

Systemet kommer til å tilhøre Popkorn som foretar det meste av webutviklingen sin i ASP.NET og C# og vi vil derfor gjøre det samme. Dermed vil det være lettere for oppdragsgiver å videreutvikle applikasjonen. Ettersom vi har tilgang til Microsoft Visual Studio 2008 gjennom HiG, som oppdragsgiver er i ferd med å gå over til, skal vi bruke denne.

GRENSESNITT

Begrensning i form av grensesnitt vil kun være at systemet må kjøres gjennom en nettleser.

DATABASE

Databasen er begrenset til Microsoft SQL server. Det er benyttet Microsoft SQL server 2008 fordi dette kreves av Microsoft Visual Studio 2008 når *Reporting Service* skal benyttes.

OPERATIVSYSTEMER

Systemet blir plattformuavhengig ettersom dette skal være en webløsning. Men det kreves at serveren løsningen skal ligge på er en Microsoft Server.

KODESTANDARD

Vi skal skrive koden til applikasjonen med engelske navn på variabler, funksjonsnavn og lignende. Årsaken til dette er personlig preferanse. Kommenteringen skal skrives på norsk.

2.7 UTGIVELSER

Utgivelser av inkrementer skal være mulig etter hver 2. uke. Da skal vi ha fungerende kode som er mulig å vise og teste for oppdragsgiver. Møtene skal foregå ca annenhver uke. Det vil mot slutten av perioden også være mulig å legge ut applikasjonen på oppdragsgivers server. Da vil det bli mulig for Løiten Almenning og sensor å kunne teste hele applikasjonen.

3 DESIGN

3.1 SOFTWAREARKITEKTUR

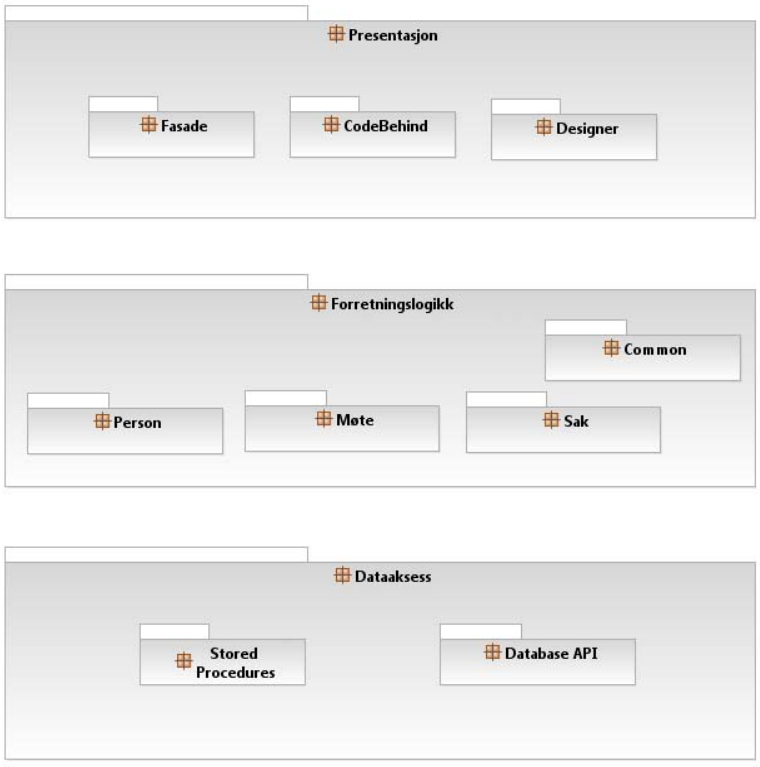
LAGDELING

For vår løsning har det fra første stund blitt planlagt en 3-lags arkitektur med et lag for presentasjon, et for forretningslogikk og et for dataaksess. I tillegg til våre tanker om en 3-lags arkitektur har valget blitt støttet av at Popkorn benytter seg av *n-tier*, en arkitektur som ASP.NET har lagt til rett for bruken av. Ettersom Popkorn benytter et slikt grunnoppsett har vi fått tilbud om å få et "skjellett" å ta utgangspunkt i. De tre lagene danner grunnstrukturen for løsningen vi har utviklet, i tillegg er det delt opp i flere deler innad i hvert lag (figur 3-1).

Presentasjonslaget – dette laget inneholder brukergrensesnittet, *eventhandling* og annen funksjonalitet som er direkte knyttet opp mot GUI-komponenter.

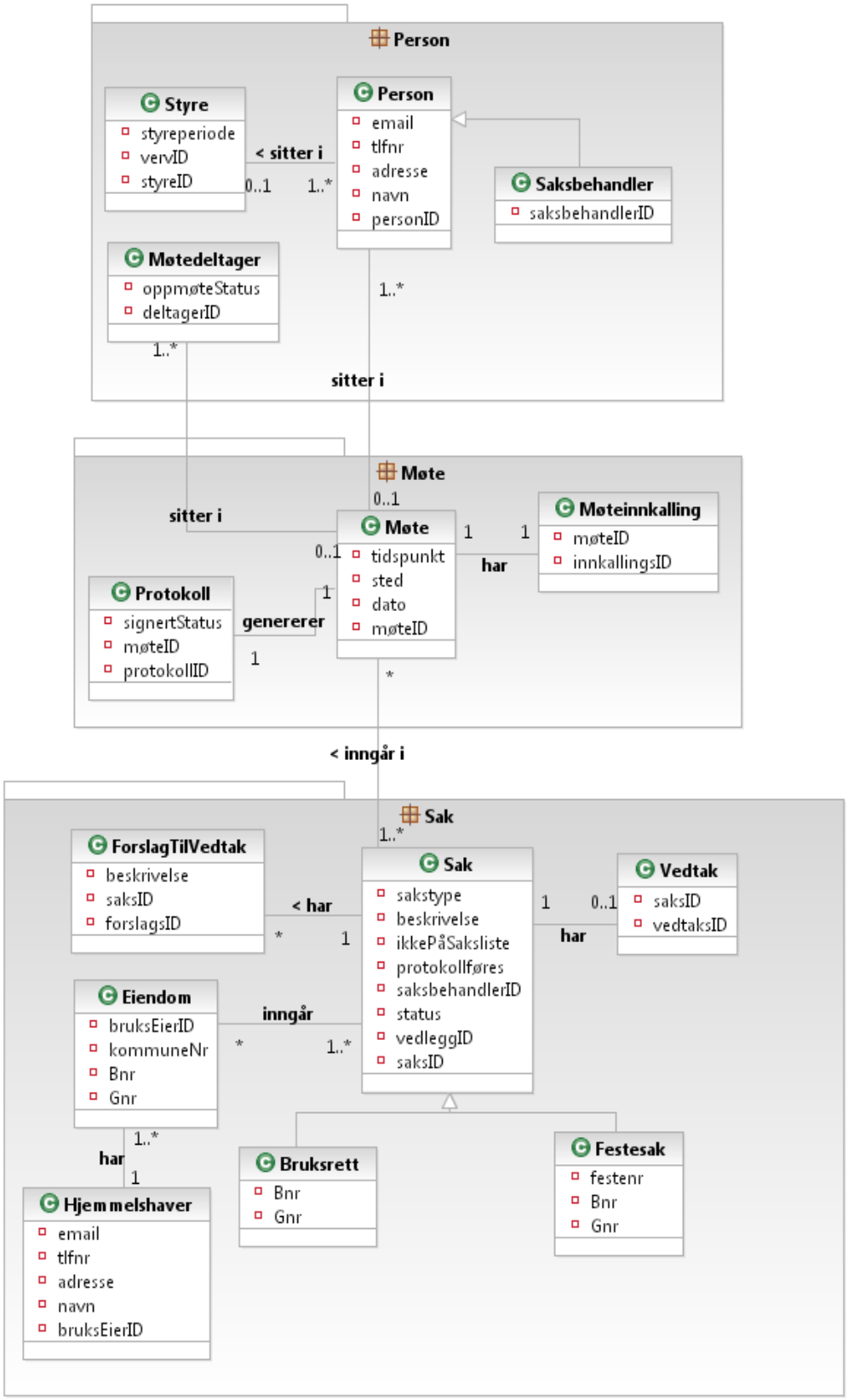
Forretningslogikklaget – alle funksjoner som ikke direkte berører grensesnittet skal ligge i dette laget. Dette laget gir kobling mellom presentasjon og dataaksess.

Dataaksesslaget – her ligger alt som har sammenheng med database, laget tar imot forespørsler om uthenting, lagring og sletting av innhold i databasen.



Figur 3-1 Lagdelingen i applikasjonen.

DOMENEMODELL



Figur 3-2 Domenemodell

Domenemodell består, som Figur 3-2 viser, i grove trekk av tre pakker:

Person – som inneholder informasjon om personer, samt om en person er saksbehandler eller sitter i styret. I samarbeid med Møte holder den også oversikt på hvem som er møtedeltagere.

Møte – lagrer informasjon om et møte og hvilke saker som er knyttet opp mot det. I tillegg har pakken funksjonalitet for å generere møteinnkallinger før et møte og protokoller når et møte er avsluttet.

Sak – har som hovedoppgave å lagre informasjon om saker som kommer inn for behandling. En sak kan være en av flere typer, men vi har valgt å vise Bruksrett og Festesak i dette tilfellet. Oftest har en sak en eller flere eiendommer knyttet til seg, hver eiendom har igjen en eller flere hjemmelshavere. Ved registrering av en sak legger saksbehandler inn et forslag til vedtak, når saken er oppe til behandling godkjennes enten dette forslaget eller et nytt vedtak utarbeides.

I lagdelingen kan domenemodellen plasseres i forretningslaget.

3.2 KONFIGURERING

Som nevnt i supplementær spesifikasjonen er det ytret ønske fra oppdragsgiver om mulighet for konfigurering mot andre typer bedrifter enn Løiten Almenning. Denne konfigureringen har ingen større innvirkning på designdelen av løsningen ettersom det naturlig blir delt opp i egne deler som Sak og Møte. En mulighet for å gi et bedre overblikk over modulene i løsningen er å benytte *namespace* for hver modul. Bruk av namespace har ingen innvirkning på hvordan løsningen fungerer, men er til hjelp under utviklingen.

3.3 PERSISTENTE SIDER

For å gjøre menynavigering mulig uten ufrivillig tap av data må sidene i løsningen være persistente, det vil si holde på statusen i det øyeblikket en bruker velger en annen fane. Hvordan dette løses, blir vist nærmere under implementering, med vurdering av alternativer.

3.4 BRUKERGRENSESNITT

Viktigheten av et brukergrensesnitt som både er funksjonelt og visuelt bra, så vi på som høy. Derfor laget vi 4 GUI "*mockups*" som presenterte ulike uttrykk og navigasjonsmåter (figur 3-3). Disse ble presentert Løiten Almenning og Popkorn AS helt i begynnelsen av prosjektperioden.

Figur 3-3 GUI "mockups" presentert oppdragsgiver.

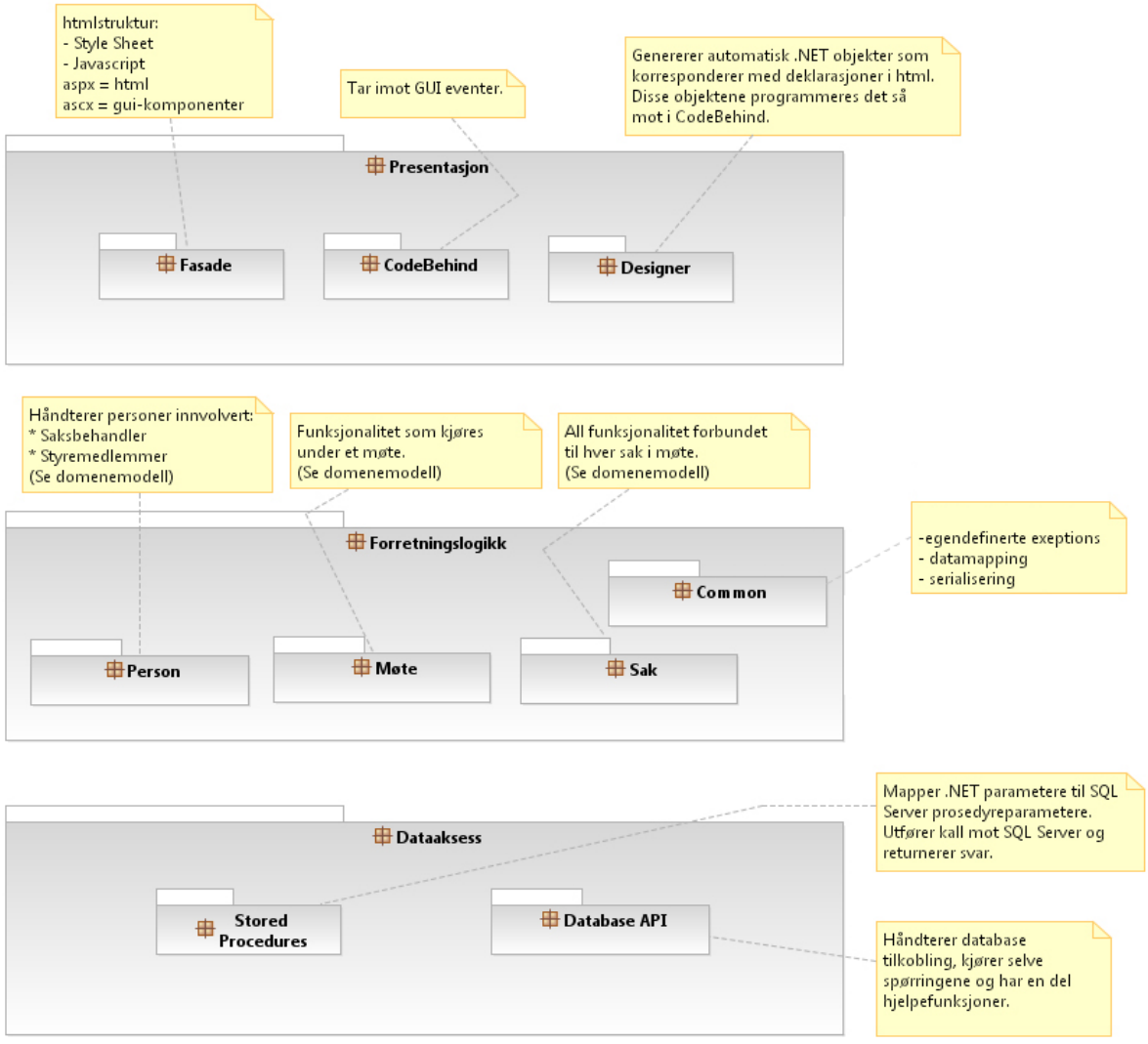
Etter litt diskusjon ble det enighet om å gå for forslag 2, hvor navigasjonen er styrt av fliker. I tillegg ble det foreslått å unngå bruk av scroll på sidene. Etter å ha undersøkt skjermstørrelsene som brukes av Løiten, som var 17" skjerm på bærbar og 21" på stasjonær, valgte vi å gå for 1200 x 700 punkter som ramme. All funksjonalitet skal ligge innenfor dette området. Størrelsen er i overkant stort, og vil ikke være å anbefale hvis dette hadde vært en vanlig webside. Selv om Løiten Almenning har stor nok oppløsning på skjermene sine til å vise applikasjonen uten at det scroller, er det ment at systemet også skal kunne brukes av andre bedrifter. Derfor er ikke tallet 1200 x 700 hentet helt ut av det blå. De færreste har mindre skjerm enn 1280 x 1024 (4:3), og der vil man unngå scrolling. Vi har også testet på en bærbar med 14.1" skjerm og 1280 x 800

(16:9) oppløsning, og det fungerte hvis nettleser ble satt til fullskjermvisning. Mindre enn dette vil ikke applikasjonen fungere bra på.

Vi har også litt forskjellige uttrykk på hoveddelen og konfigurasjonsdelen. Hoveddelen er den som brukes til vanlig av saksbehandlere, og konfigurasjonsdelen brukes av Popkorn AS ved første gangs oppsett. Det vil si at utseende og brukervennlighet er viktigere på hoveddelen.

Se vedlegg E for sidekart over hoveddel og konfigurasjonsdel.

3.5 LOGISK OVERSIKT



Figur 3-4 Logisk oversikt

Som nevnt har vi valgt å benytte en 3-lags deling i løsningen med presentasjon, forretningslogikk og dataaksess adskilt. Kontrollmekanismen bygger på call-return prinsippet, der presentasjonslaget sender forespørsler til forretningslogikklaget som igjen sender forespørsler til dataaksesslaget. Deretter returneres data på den samme måten. En slik lagdeling sikrer mer effektiv utnyttelse av ressurser, samtidig som det gir en bedre oversikt under utviklingen og vedlikehold i etterkant.

I første omgang er det ikke aktuelt å ta hensyn til problematikk rundt flere samtidige brukere. Om dette skal tas hensyn til ved en senere videreutvikling må det benyttes låsing eller lignende mekanismer.

ARKITEKTURMESSIG VIKTIGE DESIGN PAKKER

Presentasjonslaget – ASP.NET er lagt opp på en slik måte at presentasjonslaget deles opp i to deler, en del håndterer oppsett og utseende, mens den andre tar seg av funksjonalitet. Førstnevnte del består av aspx - filer som inneholder HTML og ASP kode som utgjør brukergrensesnittet. For å få det samme utseende på alle disse sidene har vi benyttet en såkalt *MasterPage* (mer informasjon i kapittel 4), en hovedside som fungerer som mal, samt inneholder navigasjonen. Alle undersider som knyttes opp mot MasterPage får det samme hovedutseende som denne, men det er mulighet for tilpassning av innholdet på hver enkelt side. Funksjonaliteten legges som standard ut i egne cs - filer som inneholder C# - kode. For hver enkelt aspx - side opprettes det en cs - fil og en designer.cs - fil, ved kompilering blir disse filene til en klassefil som hører til aspx - filen. Filen som heter designer.cs inneholder deklarasjonene av GUI-komponentene som ligger i den tilhørende aspx - filen. Selv funksjonaliteten programmeres ved hjelp av C# - kode i cs - filen. På samme måte som MasterPage anbefales å bruke når en løsning lages ved hjelp av ASP.NET har vi benyttet en såkalt BasePage som en superklasse for alle cs - filene. Denne oppretter *Singleton* - objekter av klasser i forretningslogikklaget som alle underklasser kan benytte. Selv om det i utgangspunktet ikke er nødvendig å sikre at det kun finnes et objekt av en klasse har vi valgt å gjøre det på denne måten for å spare minneplass. I tillegg inneholder BasePage fellesfunksjoner for navigasjonsoppsett som benyttes i alle underklassene. Dette laget inneholder 3 pakker:

Fasade – her ligger filene med selve grensesnittet med HTML/ASP kode, CSS stilsett og Javascript.

CodeBehind – inneholder filer med C# - kode som tar seg av eventer utløst av komponenter i grensesnittet, samt kommunikasjon med forretningslogikklaget.

Designer – her ligger filene som knytter sammen fasade med codebehind.

Forretningslogikklaget – dette laget er også delt opp i to deler. BLL delen inneholder funksjoner for håndtering av data til og fra database, samt noen nyttefunksjoner som brukes ellers i løsningen. Ved henting av data fra database benytter denne delen datamaps som ligger i Common delen av forretningslogikklaget. Datamaps som ligger i Common brukes for å konvertere data fra databasetyper til C# - typer og omvendt. Grunnen til at vi har valgt å lage dette selv istedenfor å bruke *NHibernate*[3-1] eller lignende verktøy, er at vi har prøvd å følge Popkorn sin praksis mest mulig, som bruker egendefinerte mappingklasser.

Laget inneholder følgende pakker:

Person – denne pakken er i praksis delt opp i deler som er lagt inn i møte og sak.

Møte og Sak – i disse pakkene foretas all møte- og sakshåndtering med databaseuthenting og lagring.

Common – inneholder alt av fellesfunksjoner som skal kunne brukes av alle lagene i løsningen. I tillegg ligger alt av datamapping i denne pakken.

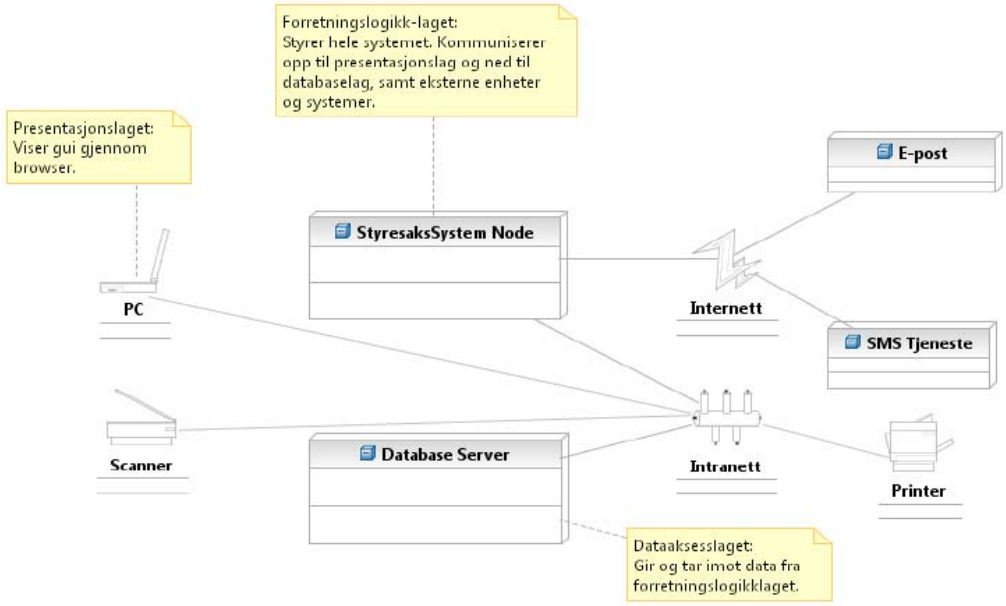
Setup – her ligger funksjonalitet som benyttes i forbindelse med brukerhåndtering, innstillinger og konfigurering.

Dataaksesslaget – laget inneholder klasser med funksjoner for å gjøre databasekall for henting, lagring og oppdatering. I tillegg ligger det en del hjelpfunksjoner i forbindelse med oppkobling. Laget har to pakker:

Stored Procedures – inneholder alle SQL-script som brukes for å lage Stored Procedures på SQL serveren.

DAL – har funksjonalitet for å koble til databasen, gjøre databasekall, samt en del hjelpefunksjoner i forhold til dette. Diagram av databasen ligger i vedlegg E under databasediagram

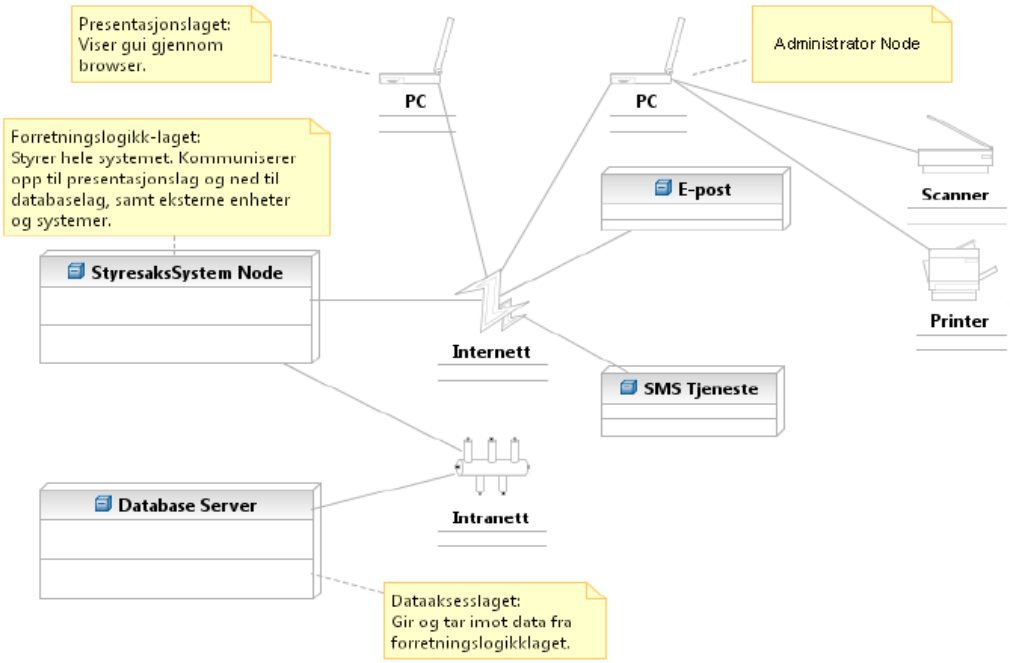
3.6 ANVENDELSE



Figur 3-5 Anvendelse ikke desentralisert

Ettersom løsningen skal utvikles slik at den vises i en nettleser betyr dette at den kommer til å kjøres på en server tilknyttet et intranett, vist som ”StyresaksSystem node” i figur 3-5. For å benytte løsningen benyttes en datamaskin som er tilkoblet det samme intranettet som serveren, med en eventuelt desentralisert løsning kan tilkoblingen skje via internett. Serveren har en tilkobling til internett for å kunne sende e-post og SMS ved møteinnkallinger og lignende. Via intranett er det mulighet for skanning av dokumenter og utskrift fra serveren. Databasen kan enten være på en frittstående server eller legges på den samme serveren som resten av løsningen.

En desentralisert løsning vil gi en litt annen figur. Som figur 3-6 viser så vil det være flere enheter tilkoblet utenfra gjennom internett til en server hvor applikasjon og database ligger. Vi ser for oss to typer ”noder” som har tilgang til applikasjonen: Administrator noden tilsvarer all brukerinteraksjon i intranett løsningen (figur 3-5), som vil si rettigheter til å opprette saker og møter samt avholde møter. Den andre noden tilsvarer en eller flere tilkoblinger til server gjennom internett med brukere som kun har lese rettigheter (standard bruker), og får presentert informasjon gjennom browser.



Figur 3-6 Anvendelse desentralisert

4 IMPLEMENTERING

I dette kapittelet går vi gjennom prosessen med implementeringen av løsningen. Det dekker mye av prosessen fra valg av verktøy og utviklingsmiljø til punkter som er spesifikke for applikasjonen. Vi har ikke mulighet til å vise og forklare alt vi har gjort, derfor har vi valgt ut noen høydepunkter som vi viser hvordan er løst.

4.1 UTVIKLINGSMILJØ

Oppdragsgiver benytter i hovedsak Microsoft teknologier i sin utvikling, med Microsoft Visual Studio som verktøy. Ettersom oppdragsgiver blir eier av løsningen vi utvikler er det naturlig at vi benytter oss av samme utviklingsmiljø. Vi har benyttet Microsoft Visual Studio 2008 som IDE og ASP.NET rammeverket ettersom løsningen skulle ha et webgrensesnitt.

ASP er basert på vanlig HTML, CSS, ASPX samt C#, der de førstnevnte benyttes for å lage grensesnittet, mens C# benyttes for å programmere funksjonalitet på serversiden. Vi har også benyttet oss noe av Javascript, da med rammeverket *JQuery* [4-1] ettersom Visual Studio til dels har lagt til rette for dette rammeverket. Denne tilretteleggelsen kommer frem med muligheten for å benytte Visual Studios innebygde IntelliSense på funksjoner som ligger i JQuery.

Ved oppstart av prosjektet ble vi enige om en kodestandard som alle gruppemedlemmer skulle bruke. Det fungerte bra fra starten, men etter hvert som prosjektet kom mer i gang kom våre egne særegne kodevaner frem ved navngiving av funksjoner, klasser og plassering av parenteser. Dette har dessverre ført til at ulike deler av løsningen har litt forskjellige utseende når det gjelder kode.

4.2 VALG AV VERKTØY

GENERELLE PROGRAMMER

Av ikke utviklingsspesifikke programmer har vi brukt følgende:

- Adobe Photoshop – Utvikling av grafiske komponenter.
- WYSIWYG Web Builder 5 – Utvikling av GUI ”mockups”.
- Microsoft Office Project – GANT skjema.
- Microsoft Office Visio – Diverse skjemaer.
- Microsoft Office Word – Bachelor rapport.
- LaTeX – Forprosjekt rapport.
- Notepad++ – LaTeX editor.
- IBM Rational Software Architect 6.0 – Kravspesifikasjon og design diagrammer.

Som opplisting viser så skrev vi forprosjektet i LaTeX og bacheloroppgaven i Word. Grunnen til denne overgangen er at vi ble sterkt anbefalt å bruke LaTeX i rapportskriving. Vi installerte nødvendig programvare og skrev forprosjektet ved hjelp av LaTeX, men grunnet formatering av maskinen hos alle gruppemedlemmene ble oppsettet av LaTeX fjernet. For å få mest mulig tid til rapportskrivingen valgte vi å bytte til Word når vi begynte på hovedrapporten.

VERSJONSSTYRING

Versjonsstyring er et hjelpemiddel man ikke klarer seg uten ved større prosjekter som strekker seg over tid, og hvor flere utviklere jobber på samme kildekode. Det gir to store fordeler: En versjonshistorie som muliggjør henting av en eldre versjon hvis noe skulle gå galt, og en integrasjon i kodebasen når det er flere kilder.

Vi har brukt TortoiseSVN [4-2] i vår versjonsstyring, som er det samme som Subversion (SVN), men med et grafisk grensesnitt. Grunnen til at vi valgte å bruke SVN er at IT-tjenesten på skolen tilbyr en slik konto på sine servere, programmet er gratis og vi har gode erfaring med SVN fra tidligere prosjekter.

SIKKERHETSKOPI

Som nevnt over så har vi bruk Subversion tjenesten som IT-avdelingen på skolen tilbyr. I tillegg har vi brukt et felles lagringsområde på skolen sine servere, hvor dokumenter og annet som tilhører prosjektet har blitt lagret. Ifølge IT-tjenesten tar de sikkerhetskopi av alt hver natt.

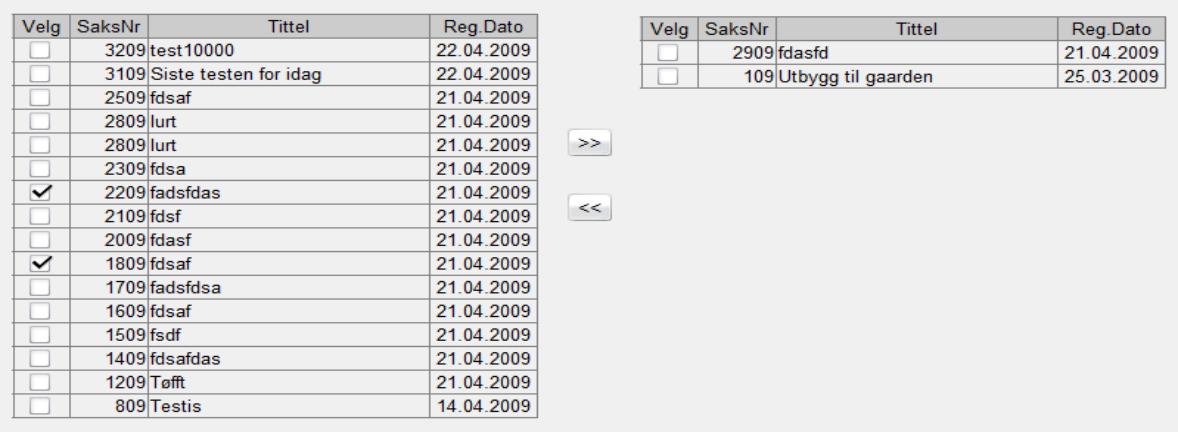
I tillegg har vi hatt som standard å oppdatere lokale filer på egne maskiner fra SVN hver dag. Dermed har vi til enhver tid hatt tre kopier av kodebasen i tillegg til serveren.

4.3 LÆRING

Ingen av gruppemedlemmene hadde noen erfaring med ASP.NET eller C# når vi startet på oppgaven. Vi hadde benyttet Visual Studio tidligere, men ikke 2008 versjonen som vi har benyttet under utviklingen.

Løsningen på vår uerfarenhet har vært å lære veldig mye samtidig som vi har utviklet, dette har gått forholdsvis smertefritt som følge av vår tidligere kodeerfaring med andre språk. Læringen har derfor i hovedsak bestått av innlæring av spesifikk syntaks for ASP og C#. Likevel har det blitt mange timer med prøving og feiling, samt internettsøking. Til vår fordel har vi hatt uvurderlig hjelp fra fagsjefen ved Popkorn, Stian Bakken, som vi har tydd til når vi har stått helt fast.

Som et eksempel på hvor mye tid og energi som har gått med på å lære ting som ser veldig enkelt ut for bruker av programmet, vil vi vise litt av hva som skjedde for å få frem en tabell med data hentet fra database.



Velg	SaksNr	Tittel	Reg.Dato
☐	3209	test10000	22.04.2009
☐	3109	Siste testen for idag	22.04.2009
☐	2509	fdsaf	21.04.2009
☐	2809	lurt	21.04.2009
☐	2809	lurt	21.04.2009
☐	2309	fdsa	21.04.2009
☑	2209	fadsfdas	21.04.2009
☐	2109	fdsf	21.04.2009
☐	2009	fdasf	21.04.2009
☑	1809	fdsaf	21.04.2009
☐	1709	fadsfdsa	21.04.2009
☐	1609	fdsaf	21.04.2009
☐	1509	fsdf	21.04.2009
☐	1409	fdsafdas	21.04.2009
☐	1209	Tøfft	21.04.2009
☐	809	Testis	14.04.2009

>>

<<

Velg	SaksNr	Tittel	Reg.Dato
☐	2909	fdasfd	21.04.2009
☐	109	Utbygg til gaarden	25.03.2009

Figur 4-1 Repeater tabell

Figur over viser hvordan det ble til slutt. Til venstre er en tabell med saker hentet fra database og til høyre en tabell som viser de sakene som er valgt.

Vi kunne gjort dette manuelt, å generere html kode i C# ved å kjøre en for-loop med alle sakene. Men det ligger innebygde verktøy i ASP.NET rammeverket som skal ta for seg slike problemstillinger, og som mest sannsynlig har en bedre algoritme for det enn vår. Det er i hovedsak tre forskjellige innebygde måter å vise datakilder; GridView, DataList og Repeater.

Etter litt undersøkelser virket GridView som en enkel måte å oppnå målet. En GridView ble definert i .aspx siden, og i Design View ble utseende bestemt. Her var det viktig at definerte kolonnenavn i GridView samsvarte med kolonnenavn fra database. I CodeBehind lagde man en metode som fikk data hentet fra en Stored Procedure i database.

```
private void CasesBindData() {
        DataTable_GetCasesMeeting cases = MeetingAccess.GetCases(1,
                ViewState["Search"].ToString(), ViewState["Expression"].ToString(),
                                                        ViewState["AscDesc"].ToString());
        gridCases.DataSource = cases;
        gridCases.DataBind();

}
```

I koden over kommer alle frie saker i databasen inn i `cases`. Da er det bare å sette `gridCases` (som er ID til GridView i .aspx side) sin datakilde lik sakene fra databasespørringen. Deretter er det å ”binde” data til GridView. Da blir det automatisk generert en ny rad (som html tabell) for hver sak. Veldig enkelt og greit, som vi fant ut av ganske fort. Bruk av GridView er verktøyet å bruke hvis man kun skal vise data. Men idet dette skal brukes videre blir det problemer, som vi fikk.

Figur 4-1 viser bare en av flere tilfeller hvor vi hadde behov for å manipulere tabellen med data etter at den var generert. I dette tilfellet skal hver rad som er markert på venstre side fjernes når bruker klikker på >> og radene skal dukke opp på høyre side. Dette støttet GridView dårlig, som førte til mye googling, prøving og feiling.

Vi prøvde alle mulige ”hacks” på GridView siden den allerede var implementert. Deretter prøvde vi DataList, som hadde de samme begrensningene som GridView. Vi fikk til å lage en annen

datakilde enn databasespørring, slik at vi kunne legge inn rader. Men dette ble ikke ”husket”. Hvis man la inn en ny rad, erstattet den rad eller radene som allerede eksisterte.

Løsningen var Repeater. Den krever mer koding for å virke, men til gjengjeld har man full kontroll. Under vises koden med forklaring:

```
protected override void OnInit(EventArgs e) {
        base.OnInit(e);
        repChoosenCases.ItemDataBound += new
RepeaterItemEventHandler(repCases_ItemDataBound);

}
```

I OnInit metoden defineres en metode for `repChoosenCases` (som er ID til en Repeater) sin databinding.

Metoden for databinding vises under. Her setter man selv inn hva som skal være hvor.

```
void repChoosenCases_ItemDataBound(object sender, RepeaterItemEventArgs e) {
        if (e.Item.ItemType == ListItemType.Item || e.Item.ItemType ==
                                                        ListItemType.AlternatingItem) {
                RptChoosenCases cc = (RptChoosenCases)e.Item.DataItem;
                Literal litId = (Literal)e.Item.FindControl("litIdChoosen");
                Literal litSaksnr = (Literal)e.Item.FindControl("litSaksnrChoosen");
                Literal litTittel = (Literal)e.Item.FindControl("litTittelChoosen");
                Literal litRegdato = (Literal)e.Item.FindControl("litRegDatoChoosen");
                litId.Text = cc.Id.ToString();
                litSaksnr.Text = cc.Saksnr.ToString();
                litTittel.Text = CommonTools.FixedLengthString(cc.Tittel,
                                                        casesTitleLength);
                litRegdato.Text = cc.RegDato.ToString();
        }

}
```

Så når metoden under kjøres, blir metoden over kjørt for hver rad som kommer med `RepeaterItemEventArgs e`.

```
private void ChoosenCasesBindData() {
        repChoosenCases.DataSource = casesList;
        repChoosenCases.DataBind();

}
```

Datakilden er `casesList`, som er en liste med objekter som holder på alle data i en rad. Problemet er at denne lista må holde på sine objekter ved postback, slik at man kan legge til og fjerne objekter etter behov (som vil si at det legges til eller fjernes rader i tabellen i .aspx siden). Løsningen ble å lagre dette unna i ViewState. Koden under viser hvordan dette gjøres.

Dette eksemplet viser at det har vært en lang vei og gå for å vise noe så enkelt som en tabell.

```
#region ViewState variabler
private List<RptChoosenCases> casesList = new List<RptChoosenCases>();private
List<RptChoosenStaff> staffList = new List<RptChoosenStaff>();
#endregion

#region ViewState
private void LoadLocalViewState(object viewState) {
	Hashtable localViewState = (Hashtable)viewState;
	casesList = (List<RptChoosenCases>)localViewState["casesList"];
}

private object SaveLocalViewState() {
	Hashtable localViewState = new Hashtable();
	localViewState["casesList"] = casesList;
	return localViewState;
}

protected override void LoadViewState(object savedState) {
	if (savedState != null) {
		object[] viewState = (object[])savedState;
		base.LoadViewState(viewState[0]);
		LoadLocalViewState(viewState[1]);
	}
}

protected override object SaveViewState() {
	object[] viewState = new object[2];
	viewState[0] = base.SaveViewState();
	viewState[1] = SaveLocalViewState();
	return viewState;
}

#endregion
```

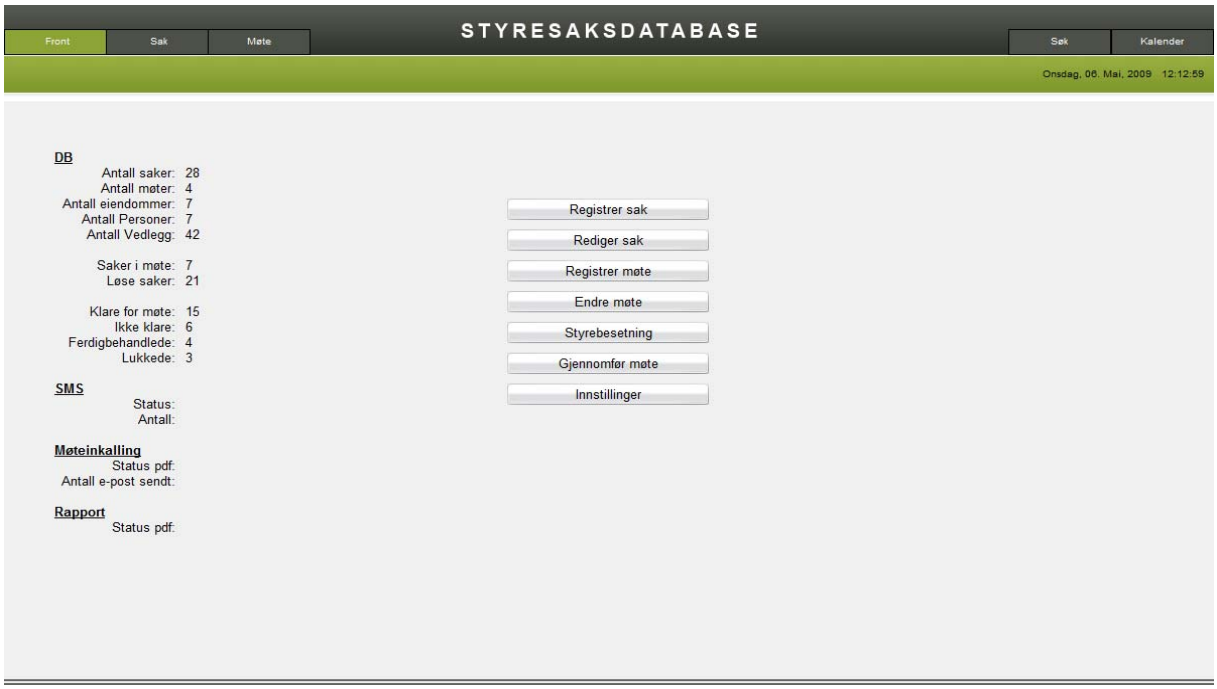
4.4 APPLIKASJONEN

Her vises applikasjonen, hvordan den ser ut etter implementering, litt veien frem og hvordan vi har løst noen av grunnsteinene i systemet.

BRUKERGRENSESNITT

HOVEDDEL

Figur 4-2 viser et skjermbilde av forsiden av løsningen. Denne siden viser en oversikt på venstre side over innhold i database, snarveier i midten og fliker øverst.



Figur 4-2 Forside på applikasjon

Oppsettet vi har brukt i koden er bruk av masterpage og contentpage, som er ASP.NET sitt svar på templates i *PHP*. Det vil si at man oppretter en master fil (.Master som filendelse) og definerer områder. I vår applikasjon er topp delen, fra det grønne feltet og opp, definert i master fil. Under er det definert et område (`asp:ContentPlaceHolder`) hvor alle subsider som hentes blir plassert.

Koden hvor subsidene legges i masterpage ser slik ut:

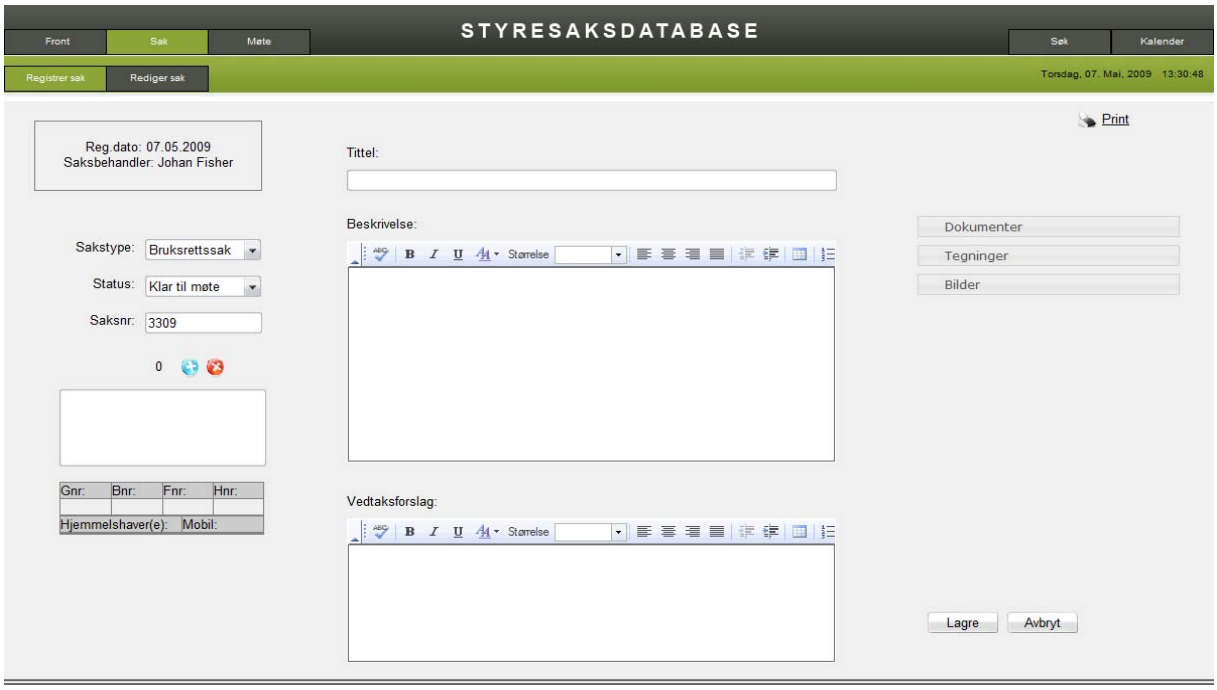
```
<asp:ContentPlaceHolder ID="ContentPlaceHolderMain" runat="server">
</asp:ContentPlaceHolder>
```

I subsidene må alt innhold som skal vises ligge innenfor et ”content” område, som koden under viser:

```
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolderMain" runat="server">
<!-- Innhold på side -->
</asp:Content>
```

Sak

Hovedkategorien sak har to underkategorier; Registrer og Rediger sak.



Figur 4-3 Registrer sak

Registrer sak (figur 4-3) brukes når nye saker skal registreres i systemet. På venstre side velger man sakstype, som i dette tilfellet kan være Bruksrettssak, Festesak, Referatsak eller Generell sak. Dette er oppsettet som passer Løiten Almenning, men hvis en annen bedrift skal bruke applikasjonen skal det være mulig å konfigurere for andre sakstyper. Videre er det mulig å sette status på saken; Klar til møte, Ikke klar til møte, Ferdigbehandlet eller Avsluttet. Bakgrunnen for denne muligheten er for eksempel at en sak som registreres kanskje mangler vedlegg eller lignende, dermed kan den ikke settes som klar til møte. Ettersom det skal være mulig å legge inn eldre saker som allerede er ferdig behandlet er det nødt til å være mulighet for å sette en sak direkte som ferdigbehandlet eller avsluttet. Saksnummer blir automatisk tildelt, men med mulighet for redigering. Nederst på venstre side kan man knytte eiendom(er) opp mot saken. Ved å klikke på plusstegnet dukker et flytende vindu (figur 4-4) opp som viser alle registrerte eiendommer med mulighet for søking.

Figur 4-4 Flytende vindu hvor registrerte eiendommer i databasen vises.

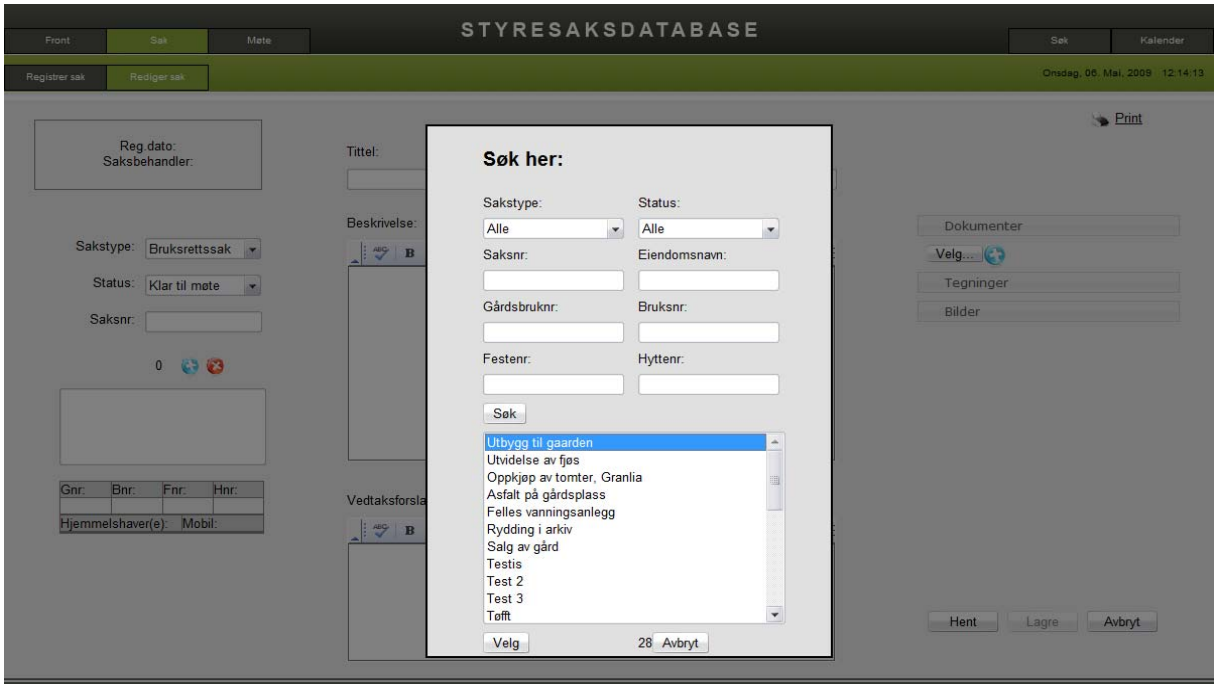
Ved å klikke på en eiendom legges den inn med tilhørende data.

Det er også lagt inn en SMS funksjon. Idet en sak er blitt registrert vil hjemmelshaver(e) som har mobil nummer registrert bli varslet om at saken deres er registrert og med hvilket saksnummer. Vi har i tillegg lagt inn mulighet for å sende en egendefinert SMS.

Midtdelen (se figur 4-3) har et tittelfelt for saken øverst, beskrivelse av saken og forslag til vedtak. Beskrivelse og vedtaks felt har en rik tekst editor, som gjør det mulig å skrive formatert tekst.

Til høyre er område for vedlegg. Der kan man laste opp bilder, PDF eller lignende under kategoriene dokumenter, tegninger og bilder.

Den andre subkategorien innen sak er Rediger sak. Den har helt likt grensesnitt som Registrer sak, men med forskjellen at bruker er tvunget til å velge en sak idet man kommer inn på siden. Figur 4-5 viser at et flytende vindu kommer opp med saker som er registrert i databasen. Vinduet gir mulighet til å søke på utallige måter. Resten av siden er satt inaktiv ettersom det ikke skal være mulig å redigere før en sak er valgt. Når sak er valgt blir alle data hentet fra database og legges inn i tilhørende felter.



Figur 4-5 Rediger sak.

Møte

Under hovedkategorien møte ligger subsidene Opprett møte, Endre møte, Styrebesetning og Gjennomfør møte. Opprett møte har samlet all informasjon om selve møtet på venstre side og saker som kan knyttes opp mot og som er knyttet til møte på høyre del (se figur 4-6).

Figur 4-6 Opprett møte.

På venstre side vises info om dato og innlogget saksbehandler. Under vises møtebesetningen, som er valgt styre i perioden (hentes fra database). Vi har lagt inn mulighet til å endre styrebesetning for spesifikke møter, siden det kan være situasjoner hvor ikke alle kan møte (for eksempel ved sykdom). Hvis man trykker på ”Endre” knappen under møtebesetningen, dukker det opp et flytende vindu (figur 4-7), hvor det vises valgt møtebesetningen for perioden øverst og alle personer som ligger i databasen under.

Nederst på venstre side settes møtedetaljer. Hvis det velges en dato som har vært, kommer en feilmelding. Når det er valgt en lovlig dato, settes dato for når møteinnkalling skal sendes på e-post og SMS automatisk inn. Verdiene for dette blir satt i Innstillinger som det er en snarvei til på forsiden. I dette eksemplet er det satt opp at møteinnkalling sendes en måned i forveien og SMS varsel en uke før møte skal avholdes. I tillegg settes sted og klokkeslett for møtet, på klokkeslett er det lagt inn en sjekk for at tiden for avslutning av møte er etter start av møte.

Figur 4-7 Vindu for å endre møtebesetning til et spesifikt møte.

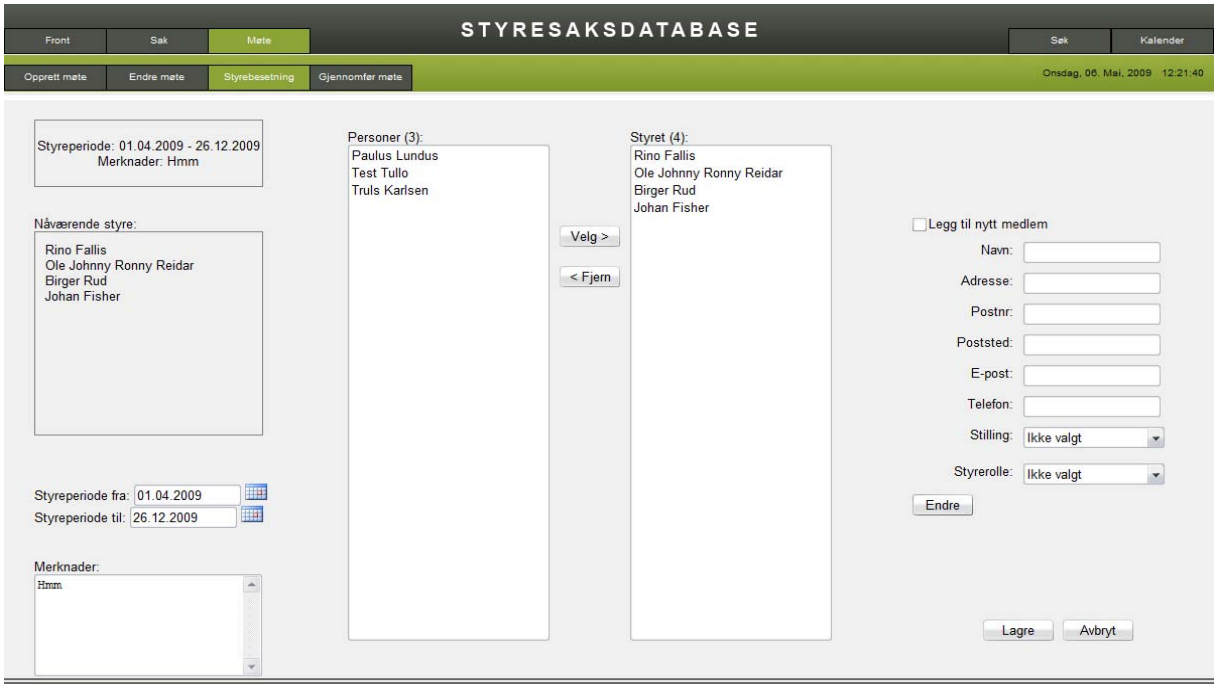
Til høyre for møtedetaljer er saksområdet, i midten vises alle saker som ikke er knyttet opp mot noe møte og som er satt som klar til møte. Det er mulighet for å søke med tekst, samt en mulighet for å sortere ved å klikke på overskriftene i tabellen. Hvis man klikker på SaksNr, blir det sortert etter saksnummer økende hvis den var synkende sortert fra før, eller omvendt. For å velge saker som skal knyttes til møte haker man de av og trykker på knappen merket >>, og de overføres de til høyre side som valgte saker. Det er mulig å frigi valgte saker med samme fremgangsmåte og deretter trykke på << knappen. For å registrere møtet er det bare å klikke på opprett knappen nederst til høyre.

Endre møte er nesten identisk til Opprett møte, men med den forskjellen at man må velge møte som skal endres først (figur 4-8), på samme måte som Rediger sak er lagt opp.

Figur 4-8 Endre møte.

Vinduet som viser opprettede møter har en søkefunksjon og sorteringsmuligheter på tittel og sted.

Styrebesetning fliken (figur 4-9) viser gjeldende styre, med mulighet til å gjøre forandringer og legge til helt nye personer (som ikke eksisterer i databasen).

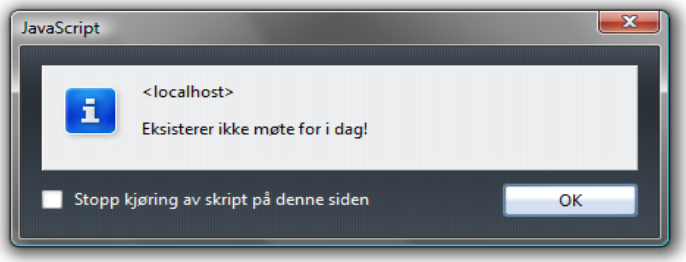


Figur 4-9 Styrebesetning.

Til venstre vises styreperioden som fra – til dato øverst, en opplisting av styret, velge ny dato og mulighet for å legge til merknad. I midten er det to felter, hvor feltet til venstre viser alle personer som er i databasen og som ikke er styremedlem, mens til høyre vises de som er i styret. Det er mulig å forandre på styret med ”Velg” eller ”Fjern” knappen i midten.

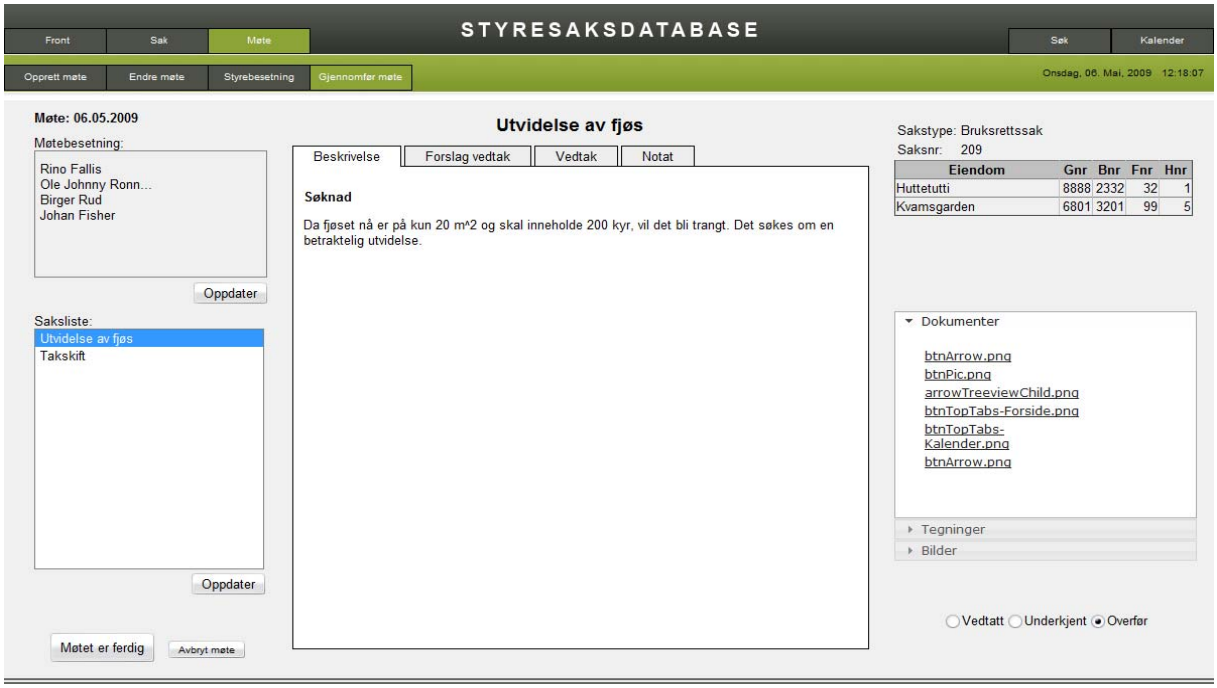
Helt til høyre er det personopplysningsfelter, som fylles med data når man klikker på en person, slik at informasjon kan endres. I tillegg kan man legge til en helt ny person i databasen ved å huke av ”Legg til nytt medlem”.

Den siste fliken under hovedkategorien Møte er Gjennomfør møte. Når bruker klikker på Gjennomfør møte fliken, sjekker systemet opp mot databasen om det eksisterer møter for dagens dato. Hvis ikke kommer et varsel om at det ikke eksisterer møte (figur 4-10), og bruker blir sendt til forsiden.



Figur 4-10 Varsel om at det ikke eksisterer møte.

Hvis det eksisterer flere møter på samme dato, som kan skje en sjelden gang, vises et flytende vindu lik figur 4-8 (Endre møte) hvor bruker kan velge hvilket møte som skal gjennomføres. Figur 4-11 viser hvordan et møte kan vises.



Figur 4-11 Gjennomfør møte.

På venstre side vises møtebesetningen, saksliste og knapp for å avbryte eller avslutte møte. Under møtebesetning og saksliste, er det en oppdater knapp. Det vil si at hvis for eksempel et styremedlem er borte, kan man gå inn på Endre møte fliken, gjøre endringer på aktuelt møte, gå tilbake til Gjennomfør møte og trykke på oppdater knappen. Samme fremgangsmåte hvis det er ønskelig å legge til eller fjerne saker fra saksliste. Her er det selvsagt viktig at status på side holdes, slik at det som er gjort fram til man ønsker endringer husker. Dette ivaretas ved hjelp av funksjonaliteten for persistente sider vi har lagt inn.

Som skjermdumpen på figur 4-11 viser, så er en sak markert (Utvidelse av fjøs). Idet en sak velges gjøres en spørring til database på saksnummer, og alle data om sak legges inn på siden.

I midten er det et informasjonsfelt med fire fliker. Første flik viser beskrivelsen som saksbehandler har lagt inn på saken, den er låst ettersom det ikke er nødvendig å forandre den på møtet (og hvis det blir behov kan det gjøres på Rediger sak). Forslag vedtak fliken er satt opp på samme måte, ved at forslaget vises i en låst form.

På Vedtak fliken blir vedtaksforslaget kopiert inn, der er det lagt inn en rik tekst editor slik at vedtak kan forandres og eller utvides. Samme løsning er benyttet for Notat fliken. I utgangspunktet finnes det ingen tekst på denne, ettersom hensikten med notat er å gi mulighet for registrering av ekstraordinære opplysninger som inhabilitet eller lignende.

Øverst på høyre side vises diverse informasjon om saken, som sakstype, saksnummer og hvilke eiendommer som er knyttet til saken med tilhørende gårds-, bruks-, feste- og hyttenummer. Under er vedleggene knyttet til saken lagt opp i en ”trekkspill” visning. Nederst kan man vedta, underkjenne eller overføre saken. Hvis man velger overfør, vil ikke saken bli tatt med i det aktuelle møte sin møteprotokoll, og det vil være mulig å knytte saken opp mot et annet møte.

Når møte er ferdig klikker man på ”Møtet er ferdig” knappen nederst til venstre. Da dukker et flytende vindu opp som viser en status på sakene som har blitt behandlet.

Hvis det er saker som er satt som ”overfør” vises en advarsel for å være sikker på at det ikke har skjedd en feil. Det er da mulig å klikke avbryt og endre status på saken.

Hvis man klikker OK vil møtet lagres i database som avsluttet uten at det blir generert noen protokoll. For å få en protokoll må det klikkes på ”Generer protokoll” knappen. I det man gjør det startes Reporting Service. Siden dette kan ta litt tid, avhengig av server, kommer en melding om at det arbeides.

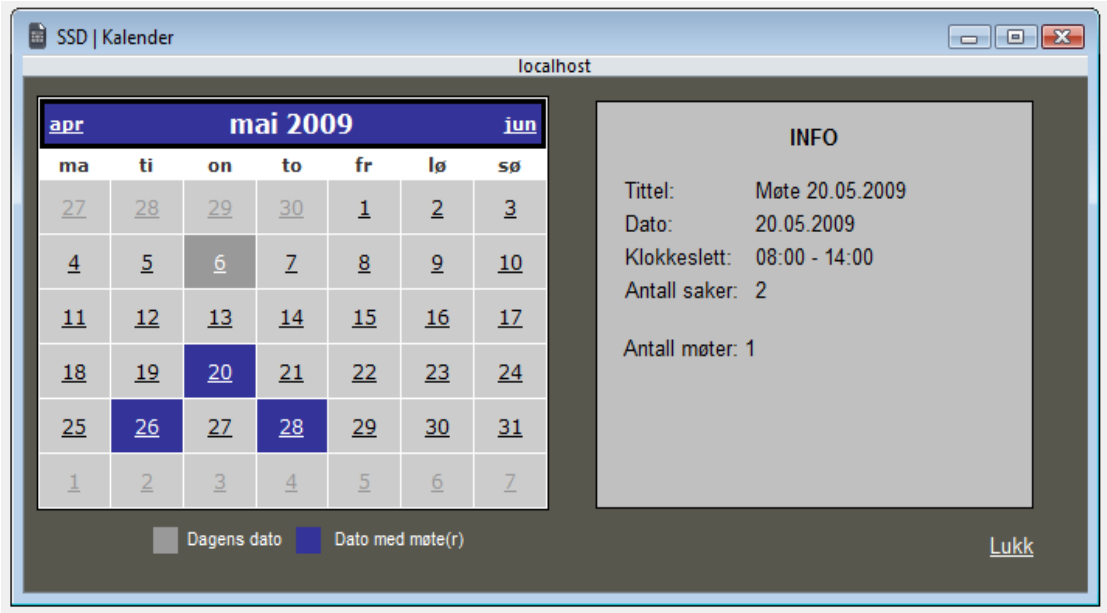
Når protokoll er ferdig generert blir den automatisk lagret på server. I tillegg vises en link hvor det er mulig å se på protokollen i nettleser eller adobe reader (figur 4-12).

Figur 4-12 Møte avsluttes vindu med status på møtet og link til å åpne ferdig generert protokoll.

Som figur 4-12 også viser, så er nå Avbryt knappen grå, som vil si at man ikke kan gå tilbake til møtet og gjøre forandringer. Årsaken til denne løsningen er ivaretakelse av integriteten av data som er lagret i databasen.

Kalender

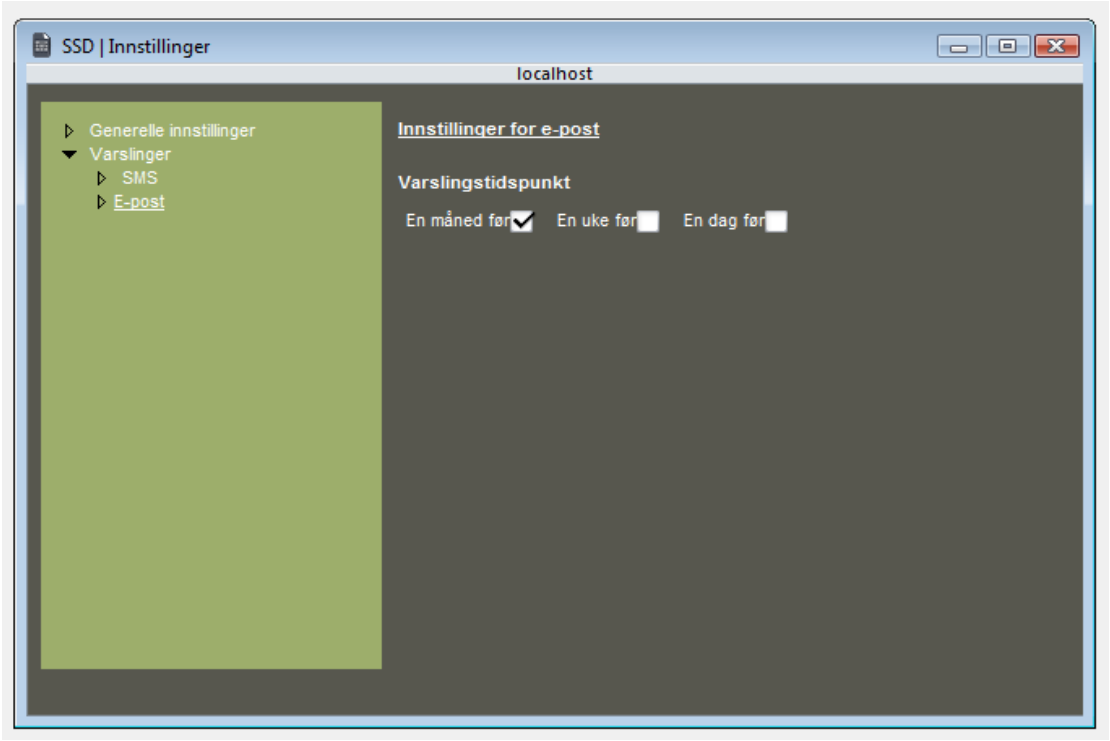
Ved å klikke på Kalender øverst til høyre åpnes en kalender i eget vindu (figur 4-13).Denne har som hensikt å vise møter som eksisterer, når de skal avholdes og litt informasjon som hvor mange saker.



Figur 4-13 Kalender som viser møter som er planlagt med litt informasjon om hver enkelt.

Innstillinger

På forsiden er det en knapp som heter Innstillinger, som åpnes et eget vindu. Med mulighet for å sette når møteinnkalling skal sendes på et møte, både e-post og SMS (figur 4-14).



Figur 4-14 Innstillinger for applikasjonen.

KONFIGURASJON

Konfigureringen er løst ved hjelp av en tankegang som er basert på hovedmoduler med undermoduler som inneholder funksjonalitet. For eksempel er Sak en hovedmodul med undermodulene Registrer og Rediger sak. På denne måten er det mulig å legge til både en ny hovedmodul og en ny undermodul ved behov.

For å håndtere konfigureringen har vi laget et eget oppsett for en superbruker av løsningen. På denne siden er det mulighet for å velge hvilke moduler som skal være synlige for en bruker uten superbruker rettigheter. I tillegg er det lagt inn en Modulhåndtering der det er mulig å endre på modulene som allerede er lagt inn eller legg til helt nye moduler. For å holde orden på hvilke moduler som er valgt for visning lagres en id i en tabell i databasen. Ved innlogging som administrator blir data om valgte moduler lest inn, informasjon om modulene hentes inn og linker til modulene vises i menyen.

Som kravspesifikasjonen forteller er det også behov for å legge til nye sakstyper for ulike bedrifter, samtidig som det må være mulighet for å velge bort de som ikke trengs. Dette er løst

på en måte som minner om moduloppsettet. En må ha superbrukerrettigheter for å få tilgang ettersom det ligger i samme modul. På siden finnes det et alternativ for Sakstypehåndtering med mulighet for å redigere og legge til sakstyper. Hver sakstype har en eller flere attributter tilknyttet, som tekstbokser med tilhørende navn og lignende. Hvilke sakstyper som er valgt lagres i databasen på samme måte som modulene. Tanken er å lese inn disse når det skal legges til en ny sak for å vise hvilke alternativer brukeren har. Dette har vi dessverre ikke fått tid til å implementere og må derfor tas videre av oppdragsgiver.

BRUKERHÅNDTERING

Løsningen har støtte for tre type brukere, men det er kun mulig å legge til administratorer og standard brukere via grensesnittet vi har laget. Det er tiltenkt kun en superbruker for løsningen, denne brukeren benyttes av Popkorn og ligger fast i databasen.

Løsningen er nå lagt opp slik at det kun er mulig for en superbruker å legge til nye brukere i systemet. Dette kan være noe upraktisk og bør endres så en administrator har tilgang til dette. Det er mulighet for å legge til, endre og slett brukere i løsningen.

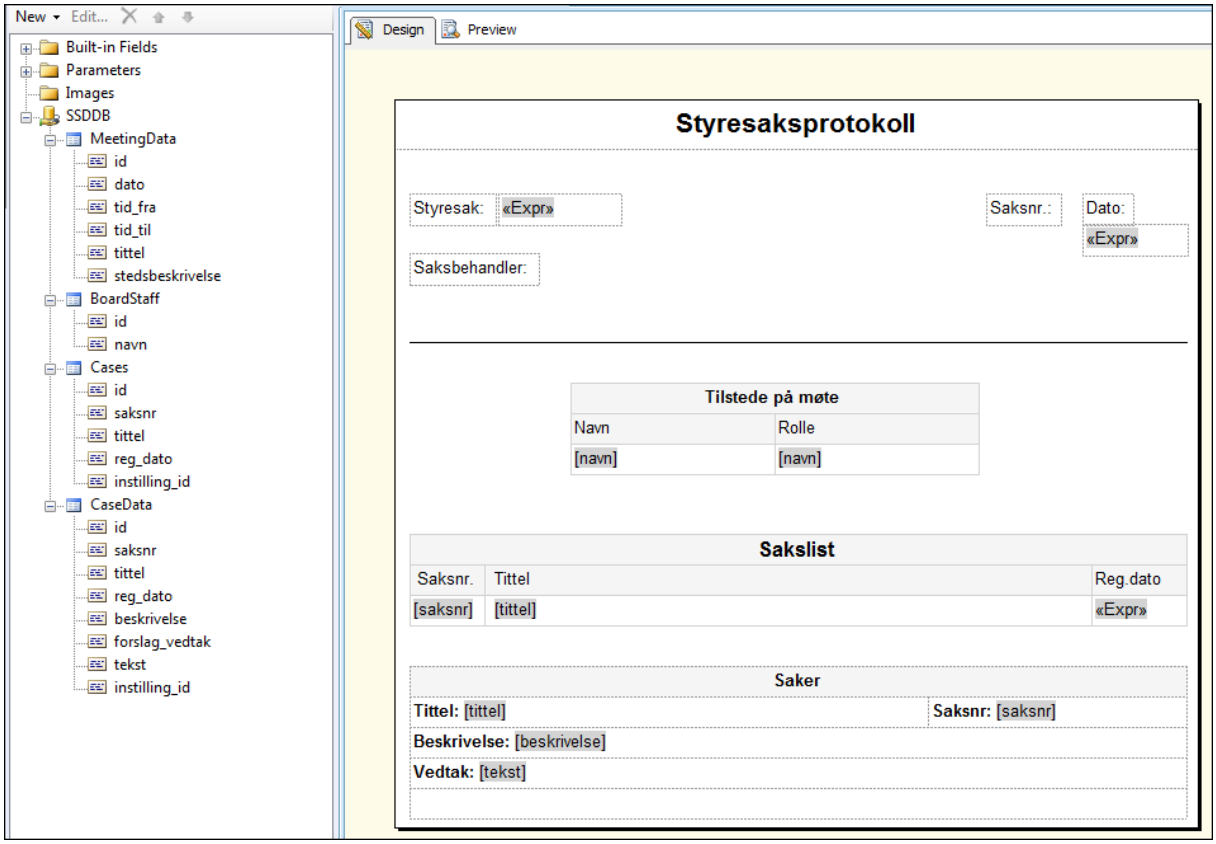
Hvordan vi har løst brukerhåndteringen har noen svakheter og har stort forbedringspotensial som vi diskuterer senere i rapporten.

REPORTING SERVICE

Et kjerneelement i applikasjonen er å få generert møteinnkalling og møteprotokoll. I designfasen av prosjektet gjorde vi en del undersøkelser på problemstillingen, som er følgende: Det skal hentes inn data fra database samt vedlegg som er lagret på server, og dette skal konverteres til et PDF dokument. Vi endte opp med en enkel og gratis løsning. Microsoft tilbyr 2 gratis versjoner av SQL Server2008, som heter SQL Server 2008 Express og SQL Server 2008 Express with Advanced Services. Den siste har innebygd Reporting Service, en integrert rapport lager som kan eksportere til bl.a. PDF.

Reporting Service bruker Web Service til å kommunisere mellom applikasjon og SQL Server. I VS 2008 gjøres det ved å legge til en Web Service i prosjektet (laget) som behøver tjenesten, i vårt tilfelle ble det Web laget som er presentasjonslaget.

Når det er opprettet en forbindelse, er neste steg å designe rapporten. I VS 2008 må man opprette et Report Service prosjekt som ligger under Business Intelligence Projects kategorien. Innenfor det opprettede prosjektet velges databasen som data skal hentes fra, og opprette en eller flere rapporter og designe disse.



Figur 4-15 Design av rapport.

Figur 4-15 viser hvordan malen på protokollen som genereres etter et møte ser ut. Felter med [ ] eller «» inneholder data fra database.

Koden under kjøres når det ønskes protokoll fra et gjennomført møte. Etter å ha opprettet forbindelse til Reporting Service, deklareres nødvendige variabler, som at det skal være en PDF, filnavnet og hvilken protokoll mal som skal brukes. I tillegg sendes møte ID med, og ut i fra den henter rapporten selv alle nødvendige data fra databasen som trengs for å generere en møteprotokoll.

```
/// <summary>
/// Genererer protokoll ved hjelp av Reporting Service.
/// </summary>
protected void GenerateProtocol() {
        // Oppretter forbindelse til Reporting Service
        rs = new RS2005.ReportingService2005();
        reExec = new RE2005.ReportExecutionService();
        reExec.Credentials = new System.Net.NetworkCredential("brukernavn",
                                                              "passord");
        rs.Credentials = new System.Net.NetworkCredential("brukernavn", "passord");
        // Deklarering og initiering
        string historyID = null;
        string deviceInfo = null;
        string format = "PDF";
        Byte[] results;
        string encoding = String.Empty;
        string mimetype = String.Empty;
        string extension = String.Empty;
        RE2005.Warning[] warnings = null;
        string[] streamIDs = null;
        string fileName = @"c:\ProtokollMote" + ViewState["meetingTitle"].ToString() +
                                                                          ".pdf";
```

```
        string _reportName = @"/SSD.Reports/Meetingprotocol";
        string _historyID = null;
        bool _forRendering = false;
        RS2005.ParameterValue[] _values = null;
        RS2005.DataSourceCredentials[] _crendentials = null;
        RS2005.ReportParameter[] _parameters = null;



        // Prøver å generere rapport
        try {
                _parameters = rs.GetReportParameters(_reportName, _historyID,
                                                _forRendering, _values, _crendentials);
                RE2005.ExecutionInfo ei = reExec.LoadReport(_reportName, historyID);
                RE2005.ParameterValue[] parameters = new RE2005.ParameterValue[1];

                if (_parameters.Length > 0) {
                        parameters[0] = new RE2005.ParameterValue();
                        parameters[0].Label = "";
                        parameters[0].Name = "id";
                        parameters[0].Value = ViewState["meetingID"].ToString();
                }

                reExec.SetExecutionParameters(parameters, "no");

                // results inneholder rapporten
                results = reExec.Render(format, deviceInfo, out extension, out encoding,
                                                out mimetype, out warnings, out streamIDs);
                // Skriver rapporten til fil på server
                using (FileStream stream = File.OpenWrite(fileName)) {
                        stream.Write(results, 0, results.Length);
                }

                // Lagrer unna rapport slik at den også kan åpnes hos klient hvis ønskelig.
                ViewState["results"] = results;

        } catch (Exception ex) {

        }

}
```

I tillegg har vi gitt brukeren et valg om å kunne få åpnet protokollen på egen maskin. Hvis det blir gjort, kjøres koden under.

```
/// <summary>
/// Åpner generert protokoll.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
public void btnOpenPDF_Click(object sender , EventArgs e) {
        Byte[] results = (Byte[])ViewState["results"];
        Response.ClearContent();
        Response.AddHeader("Content-Disposition", "attachment; filename=Report.pdf");
        Response.AddHeader("Content-Length", results.Length.ToString());
        Response.ContentType = "Application/pdf";
        Response.BinaryWrite(results);
        Response.End();

}
```

E-POST UTSENDELSE

Møteinnkallinger blir i hovedsak sendt som e-post. Dette har vi dessverre ikke fått til å fungere helt 100%. Problemene har bestått av to ting; det første er at vi ikke har hatt en egen e-post konto til serveren (vår egen e-post konto fungerte ikke), og for det andre opprettingen av en kø funksjonalitet. Hvis et møte som opprettes skal avholdes om 6 måneder, skal ikke møteinnkallingen sendes før om 5 måneder. Det vil si at disse møtene og deres møteinnkallinger først skal sendes frem i tid.

Selv om vi ikke har fått dette til å fungere har vi gjort en del undersøkelser rundt problemene. Og koden som vi kan bruke er som følger:

```
private void SendEmail() {
        RS2005.ReportingService2005 rs = new RS2005.ReportingService2005();
        rs.Credentials = new System.Net.NetworkCredential("brukernavn", "passord");

        string report = "/SSD.Reports/MeetingCalll";
        string desc = "Send email";
        string eventType = "TimedSubscription";
        string scheduleXml = @"<ScheduleDefinition><StartDateTime>2008-04-21T11:30:00-
                                    11:35</StartDateTime></ScheduleDefinition>";

        RS2005.ParameterValue[] extensionParams = new RS2005.ParameterValue[8];

        extensionParams[0] = new RS2005.ParameterValue();
        extensionParams[0].Name = "TO";
        extensionParams[0].Value = "meg@test.no";

        extensionParams[1] = new RS2005.ParameterValue();
        extensionParams[1].Name = "ReplyTo";
        extensionParams[1].Value = "ssdserver@hig.no";

        extensionParams[2] = new RS2005.ParameterValue();
        extensionParams[2].Name = "IncludeReport";
        extensionParams[2].Value = "True";

        extensionParams[3] = new RS2005.ParameterValue();
        extensionParams[3].Name = "RenderFormat";
        extensionParams[3].Value = "MHTML";

        extensionParams[4] = new RS2005.ParameterValue();
        extensionParams[4].Name = "Subject";
        extensionParams[4].Value = "Møteinnkalling";

        extensionParams[5] = new RS2005.ParameterValue();
        extensionParams[5].Name = "Comment";
        extensionParams[5].Value = "Her er møteinnkallingen med sakspapirene…";

        extensionParams[6] = new RS2005.ParameterValue();
        extensionParams[6].Name = "IncludeLink";
        extensionParams[6].Value = "True";

        extensionParams[7] = new RS2005.ParameterValue();
        extensionParams[7].Name = "Priority";
        extensionParams[7].Value = "NORMAL";

        RS2005.ParameterValue parameter = new RS2005.ParameterValue();
        parameter.Name = "id";
        parameter.Value = "12";

        RS2005.ParameterValue[] parameters = new RS2005.ParameterValue[1];
        parameters[0] = parameter;

        string matchData = scheduleXml;
        RS2005.ExtensionSettings extSettings = new RS2005.ExtensionSettings();
        extSettings.ParameterValues = extensionParams;
        extSettings.Extension = "Report Server Email";

        try {
```

```
                rs.CreateSubscription(report, extSettings, desc, eventType, matchData,
                                                                        parameters);
        } catch (SoapException e) {

        }

}
```

SMS UTSENDELSE

Systemet gjør bruk av SMS som et verktøy til varsling og beskjeder. Når en sak er blitt registrert i systemet, sendes det en SMS til hjemmelshaver. Dette er en automatisert funksjon, men det er også lagt opp til at saksbehandler kan sende en egendefinert melding. Det er også en automatisert SMS funksjon som sender varsel til møtedeltagere om kommende møte en bestemt tid i forveien.

Etter litt undersøkelser og i samråd med Popkorn gikk vi for en løsning som involverer en tredjepart, som ble PSWinCom [4-3]. Det er et norsk firma som tilbyr en SMS-tjeneste man kan abonnere på.

Etter at Popkorn hadde kjøpt tjenesten for oss, fikk vi tilsendt brukernavn og passord for tjenesten. Deretter var det bare å gå inn på hjemmesiden til PSWinCom og laste ned en pakke som var tilrettelagt for .NET rammeverket, samt noen kodeeksempler.

Tredjepartskomponenter som er tilrettelagt for .NET er veldig enkelt å implementere. Hovedkjernen er en .dll fil som legges i bin mappe til prosjektet som skal bruke det. I dette tilfellet heter den PSWinCom.Gateway.Client.dll. For å legge til komponenten åpnes prosjektet i Visual Studio, høyreklikk på ”References” i Solution Explorer og velge ”Add Reference” som åpner et vindu. Deretter velges fliken som heter ”Browse” og man finner igjen .dll filen i bin mappen. Når dette er gjort er en linking til SMS tjenesten klar.

I klassefilen (codebehind) til siden som skal sende SMS gir man først beskjed om at man skal bruke tjenesten:

```
using PSWinCom.Gateway.Client;
```

Deretter definerer og initierer man en SMSClient variabel:

```
private SMSClient smsSender = null;
```

Selve koden som sender sms beskjed ser ut slik:

```
// Data for autentisering av konto
smsSender = new SMSClient();
smsSender.Username = "brukernavn";
smsSender.Password = "passord";
smsSender.PrimaryGateway = "http://sms3.pswin.com/sms";

// Ny melding blir satt opp med opplysninger om mobil nr og
// meldingsinnhold
Message msg = new Message();
msg.ReceiverNumber = "47999999";
msg.Text = "Hei. Dette er en melding fra Loiten Almenning. Saken din er registrert i
                                    systemet og vil bli behandlet så fort som mulig.";
```

```
msg.RequestReceipt = false;
msg.SenderNumber = "Loiten Almenning";

// Sender beskjeden
smsSender.Messages.Clear();
smsSender.Messages.Add(1, msg);

smsSender.SendMessages();
```

Ved å tilkalle en slik metode blir SMS-meldinger sendt.

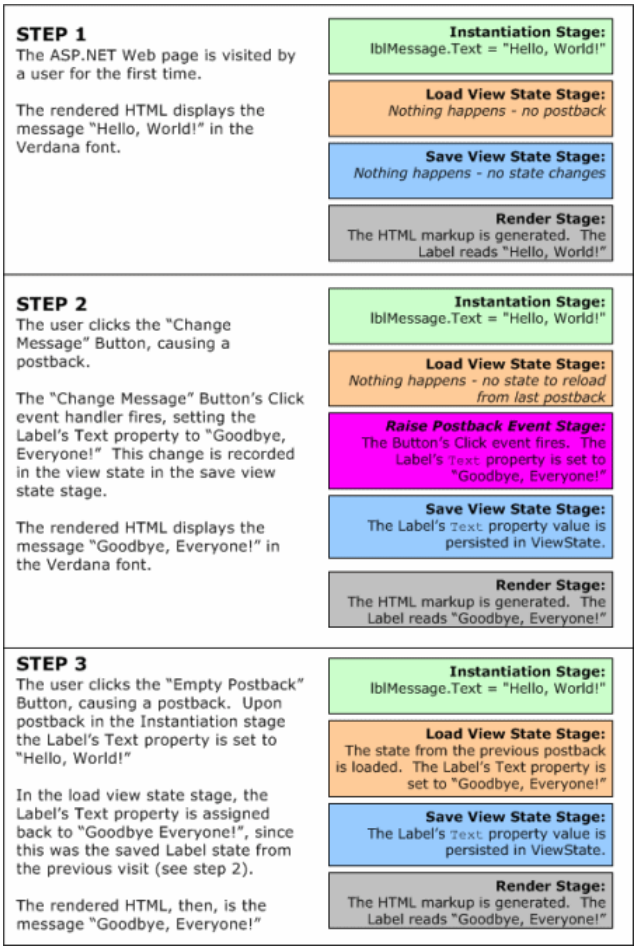
PERSISTENTE SIDER

Det er tidligere nevnt at sidene skal huske sin egen status slik at bruker kan ”vandre” mellom flikene uten at noe arbeid forsvinner. Hvis dette hadde vært en desktop-applikasjon, så hadde et slikt krav ikke vært noe problem, men i et web-grensesnitt er dette forholdsvis komplisert. De fleste vet at når man går til en annen side, nullstilles siden man forlater. Dette er uheldig i denne applikasjonen siden vi har lagt opp til at det man av og til må endre eller sjekke noe på en annen side. I tillegg er dette en sterkt ønsket funksjonalitet både fra Løiten og Popkorn.

Vi gjorde en del undersøkelser rundt problemet, både på Internett og ved samtale med Stian Bakken. Han hadde hatt lignende problemstillinger og hadde funnet en løsning i artikkelen ”Persisting the state of a web page” av Martin Jericho [4-4]. For en dypere forklaring på hvordan koden virker, vil vi anbefale å lese artikkelen.

Før vi ukritisk brukte denne løsningen ville vi prøve ut en idé vi hadde. Vi opprettet et Singelton objekt for hver side som behøvde å bevare status. I Page_Load metoden til hver side hentes alle data fra objektet inn. Når man forlater siden gjør man det motsatte i Page_Unload metoden, ved at alle data på siden legges inn i objektet. Dette fungerte veldig bra. Vi valgte likevel å gå for løsningen til Stian Bakken, grunnen var i hovedsak at vår fremgangsmåte ville bli en tung måte å vedlikeholde en side på siden det var så sterk kobling mellom innhold på side og det å ta vare på alle data.

Løsningen som Stian Bakken hadde funnet, laget av Martin Jericho, benytter seg av ASP.NET sin innebygde *ViewState* funksjonalitet. ViewState sikrer at verdier som er lagt inn på en side holder seg ved PostBack. Hver gang det trykkes på en knapp eller lignende som trenger å bruke C# kode på server, skjer det en PostBack, som vil si en reloading av siden. Figur 4-16 viser en illustrasjon på hva som skjer i ViewState når man endrer verdi i en label. I tillegg til at ViewState brukes for å holde orden på brukerinput, kan den også brukes til å holde på egendefinerte variabler. Dette er en forenklet forklaring av funksjonen til klassen PersistentStatePage.



Figur 4-16 ViewState syklus [4-5].

I praksis er det ganske enkelt å bruke. Vi opprettet klassen PersistentStatePage i Web laget, og alle sider som skal ha mulighet til å huske sin status arver fra denne klassen.

```
public partial class doMeeting : PersistentStatePage
```

Over vises klassedeklarasjonen til doMeeting, som er Gjennomfør møte. Hvis det er ønskelig å bevare status på Gjennomfør møte, som er i alle situasjoner bortsett fra når det klikker Avbryt, brukes `RedirectSavingPageState("side som skal vises");` istedenfor `Response.Redirect("side som skal vises");` som er den vanlige måten å gå til ny side. Ved å gjøre dette bevares alt. Når man ikke vil ta vare på status, eller nullstille status, bruker man vanlig `Response.Redirect("side som skal vises");`.

Dette gjorde problemet med bevaring av status mye mindre enn antatt, men dette førte til et annet problem. Fram til implementering av persistente sider, foregikk all sidenavigasjon via masterpage som inneholder alle linker. Dette måtte omgjøres slik at redirigering kunne foregå fra subsidene og dermed få bevart status hvis ønskelig.

Problemet med å ha flikene på masterpage og funksjonaliteten på subsidene var å unngå sterk kobling, siden ideelt sett skal ikke masterpage vite om sidene som bruker den. Løsningen ble å definere en EventHandler på masterpage, som alle subsidene kan abonnere på. Dermed tilbyr bare masterpage en event, uten å vite om hvem som abonnerer.

På masterpage sin codebehind er det følgende kode som får alt til å virke:

```
public event CommandEventHandler eventLinkButton;

protected void LinkButtons_Click(object sender, EventArgs e) {
	LinkButton linkButton = (LinkButton)sender;
	if (eventLinkButton != null)
		eventLinkButton(this, new CommandEventArgs(linkButton.CommandName,
                                                   linkButton.CommandArgument));

}
```

Første linje definerer eventen. Funksjonen LinkButtons_Click er den som kjøres når en flik blir klikket på. Koden sjekker om eventLinkButton er null eller ikke. Hvis den er null, vil det si at ingen abonnerer på eventen, og det skal ikke legges inn noen data. Hvis det er abonnert på eventen legges navn og hvilken side som ønskes inn (CommandName og CommandArgument).

På subsidene, som abonnerer på event er følgende kode:

```
protected override void OnInit(EventArgs e) {
	base.OnInit(e);
	Master.eventLinkButton += new CommandEventHandler(Master_eventLinkButton);

}
```

OnInit er en innebygd metode i ASP.NET som kjøres ved initiering av side, og som vi overkjører ved å legge til en abonnering av event fra masterpage.

Når det da klikkes på en flik og en event blir kjørt, blir følgende kode i subsider kjørt:

```
protected void Master_eventLinkButton(object sender, CommandEventArgs e) {
	RedirectSavingPageState(e.CommandArgument.ToString());

}
```

`CommandEventArgs e` inneholder siden som bruker ønsker, og da er det bare å legge det inn i PersistentStatePage sin `RedirectSavingPageState` og status ivaretas.

I tillegg må CommandEventArgs bli satt og farge må skiftes på den fliken som tilhører side bruker er på. Det er løst ved hjelp av funksjoner i BasePage, en egen klasse som alle sidene arver fra. Når man oppretter en ny side i ASP.NET opprettes en .aspx fil som er klient siden, og en .aspx.cs fil som er klassen på server til siden. Alle disse .cs filene arver fra `Page` objektet til ASP.NET. Det vi har gjort er å sette BasePage til å arve fra `Page` og PersistentStatePage fra BasePage. Dermed kan vi legge inn diverse globale metoder i BasePage som alle sidene har tilgang til, som å sette CommandArgument og skifte farge på fliker. Under viser en slik metode, som setter CommandArgument på flikene til subsiden:

```
public void SetSubTabs(int tid) {
	for (int i = 0; i < selectedSub.Count; i++) {
		if (selectedSub[i].Topid == tid) {
			LinkButton linkbutton =
                                (LinkButton)Master.FindControl("LinkButtonSub"+i);
			linkbutton.Text = selectedSub[i].Name;
			linkbutton.CommandName = selectedSub[i].Name;
			linkbutton.CommandArgument = selectedSub[i].Lok;
		}
	}

}
```

RIK TEKST EDITOR

Siden saksbehandler skal skrive inn beskrivelse av hver sak samt vedtak, ble det ytret ønske om en rik tekst editor i applikasjonen. Vi gjorde en del undersøkelser både med søking på internett, men også ved å spørre noen ressurspersoner på området, som er Øyvind Kolloen, Mathias Bjerke og Stian Bakken. Vi endte opp med 3 editorer å velge mellom – Cute Editor[4-6], TinyMCE Editor [4-7] og FCK Editor[4-8].

Den første vi prøvde var Cute Editor. Grunnen til at vi valgte den er at det er en ASP.NET editor og Popkorn hadde ved tidligere anledninger brukt denne med hell. Editoren var ikke gratis, men vi kunne bruke en trial versjon den første måneden og etter det var oppdragsgiver villig til å kjøpe lisens. Dessverre ble vi ikke 100% fornøyd med editoren i vår applikasjon. Den sinket visningen av sider som inneholdt editoren for mye og den fikk en forskyvningsfeil ved postback. Det vil si at den økte i lengde og spratt tilbake til innstilt lengde for hver postback. På toppen oppførte den seg ikke helt bra i Internet Explorer som sannsynligvis kommer til å bli brukt mest for å vise løsningen.

Dermed stod vi igjen med TinyMCE og FCK editor. TinyMCE hadde vi fått anbefalt på det varmeste som den aller beste blant editorer, og den var gratis. Men den ligger under LGPL Open Source lisens, som vil si at bruk av editor skal gjøres i open source ånd. Dette betyr igjen at ved kommersiell bruk må videre avtale inngås. Dette passet dårlig siden planen er at Popkorn skal videreutvikle applikasjonen slik at de kan selge den til andre bedrifter enn Løiten.

FCK editor fikk vi også varmt anbefalt, som også var gratis, og er under CDL lisens som gir full frihet for bruk i også kommersiell sammenheng. Vi byttet ut Cute Editor med denne med veldig gode resultater. Det ble mye lettere å laste sider med editoren på, den er lett å konfigurere og har ingen forskyvningsfeil. Det aller best av er at den også har en .NET versjon som gjør implementeringen like lett som med SMS tjenesten. Ved å legge FredCK.FCKeditorV2.dll filen i bin mappen og en referanse i prosjektet. I .aspx filen måtte vi legge inn en assembly referanse øverst:

```
<%@ Register Assembly="FredCK.FCKeditorV2" Namespace="FredCK.FCKeditorV2"
TagPrefix="FCKeditorV2" %>
```

Deretter er det bare å legge følgende kode der man ville ha en editor:

```
<FCKeditorV2:FCKeditor ID="fckResolution" runat="server" Height="478px" Width="548px"
StartupFocus="True" DefaultLanguage="no" ToolbarSet="Basic"
ToolbarStartExpanded="True"></FCKeditorV2:FCKeditor>
```

Det ble nevnt at editoren var lett å konfigurere. Figur 4-17 vises alle funksjonene editoren kan ha.

Figur 4-17 FCK editor med alle funksjoner.

Dette er mye mer enn hva vi trengte og dessuten tar det verdifull plass. Heldigvis er det en config javascript fil hvor man kan legge til og fjerne funksjonalitet. I tillegg kunne man velge mellom ulike ”skin”. Figur 4-18 vises editoren etter at vi har fjernet mange funksjoner og satt den til MS Office ”skin”:

Figur 4-18 FCK editor med våre utvalgte funksjoner og office ”skin”.

4.5 DISTRIBUSJON

Ettersom løsningen kjøres i nettleser er det ingen krav til distribusjon til brukere, det eneste som kreves er at løsningen med tilhørende database ligger på en Microsoft Server. Ettersom vi har utviklet i Visual Studio 2008 er denne serveren nødt til å være MS Server 2008.

5 TESTING

Vi vurderer to ting som viktig for applikasjonen – robusthet og brukervennlighet. For å sikre dette må det utføres brukertester, i vårt tilfelle har vi har benyttet White-box og Black-box testing. I tillegg til resultatene fra disse testene har det kommet frem forslag til forbedringer ved fremvisning av applikasjonen til oppdragsgiver.

5.1 WHITE-BOX TESTING

White-box testing er transparent testing, som vil si at den eller de som tester har kjennskap til den interne strukturen i applikasjonen. I dette tilfellet vil det si oss i gruppa ettersom vi som utviklere av systemet har naturlig nok kunnskap om hva som skjer innad i applikasjonen.

Testingen har foregått kontinuerlig ved implementering av funksjonalitet. Denne testingen skjer blant annet ved kjøring av debug modus. Ved slutten av hver sprint, og før fremvisning til Popkorn, har vi prøvd å stressteste systemet mer ved å teste funksjoner som man ikke har laget selv, for å se om en annen vei til målet fører til feil.

Denne typen testing har vist seg verdifull ved flere anledninger under utviklingen. Ved et tilfelle der data for en sak skulle hentes fra database så alt ut til å være i orden. Men ved en nærmere sjekk ved hjelp av debug modus oppdaget vi at over 7000 kopier av samme sak ble sendt fra databasen.

5.2 BLACK-BOX TESTING

Black-box testing blir utført av personer som ikke har kjennskap til hva som skjer bak fasaden.

Vi hadde en uhøytidelig test der tre bekjente ved HiG som ikke kjente til applikasjonen fungerte som testpersoner. Ettersom ingen av disse kjenner til hva en saksbehandler gjør ble ikke testen helt reelle, men testen ga likevel en god indikasjon på mangler og forbedringsområder. Foregangsløpet i testen vises under.

- Registrer sak – Alle gikk inn på riktig side og skjønte hva de skulle gjøre ganske fort. Ettersom alle manglet domenekunnskap var det ikke logisk å knytte en eiendom mot en sak, i tillegg manglet det en beskrivende tittel i feltet for eiendommer. Litt usikkerhet rundt vedlegg, men alle skjønte bruken av linkene på høyre side for opplasting fort. Det ble etterlyst en melding om vellykket lagring fra en av testpersonene.

- Rediger sak – Det oppsto litt forvirring rundt bruken av det flytende panelet som vises med en gang ettersom det ikke vises noen saker med en gang. Søkemulighetene ved testgjennomføring er sakstype, status, saksnr, eiendomsnavn, gårdsbruksnr, bruksnr, festenr og hyttenr. Testen vist at søkefelt for tittel er nødvendig. Når testen ble

gjennomført havnet brukeren tilbake på samme side etter lagring, dette ble påpekt som ulogisk.

- Opprett møte – Testerne skjønte det meste, men knytting av frie saker til et møte var ikke helt intuitivt. Årsaken til dette er oppsettet med en tabell midt på siden med ”checkbox” for hver sak. Testerne huket av de sakene de ville ha med og klikket direkte på opprett møte. Veien om >> knappen for å overføre sakene til valgte saker var det ingen som skjønte uten hjelp. Vi ser nå at måten vi har løst dette på er unødvendig komplisert, veien som testerne forsøkte hadde fungert på en utmerket måte.

- Endre møte – her hadde testerne samme ankepunkt på det flytende vinduet som ved redigering av sak. Bortsett fra dette fungerte alt etter planen.

- Gjennomføre møte – Når testpersonene skjønte at man ikke kunne avholde et møte hvis det ikke eksisterte ett i dag, gikk det ganske greit. To av testerne klikket rundt på midtdelen av siden hvor det er fliker uten å skjønne hvorfor disse ikke inneholdt noe data. Etter en stund måtte vi si at en sak måtte markeres på venstre side før data blir lagt inn. På grunn av manglende domenekunnskaper var det ikke veldig intuitivt å sette hver sak som vedtatt eller underkjent.

5.3 KONKLUSJON

Vi ser at det burde vært utført flere tester tidligere i utviklingsfasen for å kunne utbedre det som ble avdekket i testen. Mye av årsaken til at dette ikke ble gjort er at store deler av løsningen måtte implementeres før det kunne gjennomføres en test.

Tidlig i prosjektperioden ble det gjort en løs avtale om testing av løsningen hos Løiten så fort hovedelementene var på plass. En test med en saksbehandler hadde vært veldig nyttig for å avdekke mangler som ikke skyldtes manglende domenekunnskap. Dessverre ble aldri en slik test gjennomført på grunn av problemer med koordinering av tidspunkt.

Mot slutten av prosjektperioden har vi fått hjelp fra Popkorn til å legge løsningen ut på nett. Mye av årsaken til at vi ønsket dette var for å gjøre det lett for Løiten å prøve ut, og dermed komme med tilbakemelding. Når vi skriver denne rapporten har vi dessverre ikke fått noen tilbakemelding som det er verdt å nevne.

6 AVSLUTNING

6.1 DISKUSJON AV RESULTAT

Nå som utviklingsfasen er ferdig og resultatet har blitt gransket og testet i sin helhet ser vi mye bra. Vi har nådd hovedmålet, som var å få til en fungerende versjon hvor viktig funksjonalitet er implementert. Løsningen vår kan gå gjennom hele løpet fra å registrere saker, opprette møter og tilslutt å gjennomføre møtene.

Det som er litt paradoksalt er at vi er mest fornøyd med de mer ”kjedelige” aspektene av løsningen. Man skulle kanskje tro at det å få til persistente sider eller SMS funksjonalitet ville være en bra følelse, men siden dette endte opp med løsninger hentet utenfra og vår oppgave ble å få det inn i løsningen, forsvant noe av kompleksiteten. Det skal nevnes at det ble en del omstrukturering, til det bedre, mellom masterpage og dens subsider ved implementering av persistente sider (se kapittel 4.5).

Det som var en stor utfordring og tilfredsstillende å få til, var rapportgenerering ved hjelp av Reporting Services. Det meste av informasjon om emnet på internett omhandlet SQL Server 2005. Vi måtte bruke SQL Server 2008 (siden vi utviklet i Visual Studio 2008) og der var det en del forskjeller som vi måtte finne ut av.

Vi er mest fornøyd med helheten av prosjektet. Fra organiseringen av selve løsningen, hvor det er klart skilte lag, til kodespesifikke ”gullkorn”. En ting som ikke har blitt nevnt så mye i rapporten er bruken av Stored Procedures. Fra før hadde vi kun erfaring med å gjøre databasespørringer gjennom en SQL streng. Dermed var det en liten overgang til å kun bruke Stored Procedures, noe som vi i ettertid er veldig fornøyd med å ha brukt.

Det må nevnes at selv om vi er veldig fornøyd med resultatet, så nådde vi ikke helt målet om en komplett løsning. Løsningen er blitt mer en prototype enn en leveringsklar applikasjon, så det gjenstår ennå litt jobb før den kan tas i bruk av Løiten.

Tross dette er vi fornøyd med det vi har utviklet med tanke på at vår tidligere erfaring med .NET var ikke eksisterende, samtidig som vi har hatt begrenset med tid til både å lære og utvikle.

6.2 FORBEDRINGSPOTENSIALE

Her vil vi gå gjennom de punktene som vi mener bør gjøres for å få løsningen 100% ferdig.

Brukergrensesnitt

Et forbedringspotensiale som er en gjenganger på flere plasser i løsningen er graden av intuitive løsninger i brukergrensesnittet. Løsningen slik den er i dag inneholder en del logiske brister i form av mindre gode navigasjonsløsninger og rekkefølgen av handlinger ved benyttelse av en funksjon.

I tillegg har vi gjennom hele utviklingsfasen vært hjemsøkt av nettleserproblematikk, som oppstår som følge av litt forskjellige standarder i forskjellige nettlesere. Dette påvirker i første omgang utseende i form av plassering, størrelse og form. Vi har i hovedsak sjekket utseende i Opera ettersom denne har den mest diplomatiske standarden av de tre mest populære (Internet Explorer, Opera, Mozilla Firefox). Det kan argumenteres for at utseende burde vært mer tilpasset Explorer ettersom dette er nettleseren som er aller mest i bruk, men vi mener at en bra webapplikasjon skal og bør fungere tilfredsstillende i alle av de mest brukte nettlesere.

Plassering av kode

Når vi beskriver lagdelingen i kapittel 3 sier vi at all funksjonalitet som ikke er tilknyttet grensesnittet skal ligge i forretningslogikklaget. Det har likevel havnet en del kode i CodeBehind siden veldig mye går på å hente ut og/eller legge inn data til en side. Et eksempel er all kode som trengs for å opprette en Repeater og dens innhold (se kapittel 4.3).

Innstillinger

Muligheten for innstillinger av intervaller av e-post og SMS utsendelse har ikke blitt implementert per i dag. Den informasjonen som ligger i databasen har blitt lagt direkte inn uten bruk av grensesnittet. Grensesnittet for funksjonaliteten er klar, men selve funksjonen er ikke implementert, denne må ferdigstilles før løsningen tas i bruk. Selv om det er en liten del, er den viktig i forhold til utsendelse av møteinnkallinger og møtepåminnelser.

Søk

Søkefunksjonen for oppslag har vi, i likhet med Innstillinger, kun implementert grensesnittet. Søking er en praktisk detalj som bør implementeres, men er ikke viktig for noen annen del av løsningen. Ettersom vi har søkefunksjonalitet på flere plasser ved registrering av saker og møter har ikke søking i oppslagssammenheng fått høy prioritet.

En annen løsning på søk/oppslag funksjonalit er å fjerne Søk som en egen flik, og heller legge dette inn på forsidene til Sak og Møte. Per i dag finnes ingen av disse forsidene, men de kan med fordel lages for å ha muligheten for oppslag. Dette ville gitt en ny og viktig dimensjon til applikasjonen.

Maler

Som regel finnes det en eller flere typer saker som har det samme oppsettet hver gang, derfor bør det være mulighet for å opprette bedriftsspesifikke maler som er redigerbare. Dette gjelder også for oppsett av innkallinger og protokoller.

Feilhåndtering

Etter at vi har sluttet å programmere oppdaget vi at løsningen ikke har noen form for feilhåndtering ved databasekall eller feillogging. Dette er en veldig beklagelig forglemmelse som er et must å få inn i løsningen før den tas i bruk.

Utsendelse av innkalling

Siden vi ikke har fått tid til å implementere e-post utsendelse, er ikke funksjonalitet for møteinnkalling helt ferdig. Rapporten er designet og klar for bruk, men den er ikke i bruk i løsningen vi leverer.

Ny brukere

Brukerhåndtering er i vår løsning forbeholdt en superbruker, dette er absolutt ingen optimal måte for noen parter. En bedrift som benytter løsningen må kunne legge til nye brukere uten å måtte kontakte Popkorn som har superbruker rettigheter. Derfor bør det legges inn mulighet for at en administrator har mulighet for å ta seg av brukerhåndteringen etter første gangs oppsett som utføres av Popkorn.

6.3 VIDERE ARBEID

Her vil vi ta for oss punkter som ikke er nødvendig for applikasjonen å ha nå, men som vil gi nye muligheter i bruk.

Desentralisering

Vi har flere ganger nevnt muligheten for en desentralisert løsning i fremtiden, i forhold til brukertyper er det også lagt til rett for dette. Om det skal realiseres derimot må det opprettes en form for visningssider for møte og saker. Disse siden må utvikles med tanke på en standard bruker som ikke har rettigheter til annet en å avgi stemme under møtegjennomføring. I tillegg må sikkerheten i løsningen bedres betraktelig ettersom vi har utviklet den med tanke på bruk i et intranett.

Moduler

Ettersom vi har lagt til rette for utvidelse av løsningen ved hjelp av nye moduler er dette en mulighet for å kunne få et mer allsidig administrasjonsverktøy.

Statistikk og historikk

I forbindelse med stemmegivning kan det være ønskelig med en oversikt over avgitte stemmer, samt en historikk over hvem som har stemt på en gitt sak.

Oppfølging av saker

Ved behandling av noen saker kan det være aktuelt med en eller annen form for etterbehandling/oppfølging etter vedtak for å sjekke faktiske forhold. I denne forbindelsen mangler løsningen i dag et grensesnitt for registrering av klarerte forhold.

Flere brukere

Ettersom løsningen i først omgang har blitt utviklet med tanke på bruk i Løiten har vi ikke lagt inn funksjonalitet for håndtering av flere samtidige brukere. For å kunne bruke løsningen i andre og større bedrifter kan det være aktuelt å legge inn en form for låsing ved registrering av saker og møter. Denne låsingen kan for eksempel foregå ved låsing av lagring til databasen, slik at brukerne må vente på tur. Om dette skal fungere er det i tillegg nødt til å legge inn en ekstra sjekk for å sikre at et saksnummer ikke brukes to ganger, eller det blir et hopp i saksantallet.

Integrering av dokumentskanning

Det kan være ønskelig med en større grad av automatisering ved skanning av dokumenter som skal være vedlegg til en sak. Men dette kan regnes som en tilleggsfunksjonalitet som ikke er viktig i forhold til annen funksjonalitet. Det kan derimot være et velkomment tillegg i forhold til effektivisering av saksregistreringen.

6.4 EVALUERING AV ARBEIDET

ORGANISERING

Forprosjektrapporten i vedlegg F beskriver hvordan vi har valgt å organisere prosjektløpet. Vi har klart å holde oss til den planlagte organiseringen med roller, ansvarsområder og planlagt arbeidsmengde (se timelogg i vedlegg C). Det har gått smertefritt i forhold til diskusjoner og uenigheter, som sjelden har oppstått, i gruppen.

Bruken av Scrum som utviklingsmodell i prosjektperioden har fungert veldig bra, med 14-dagers sprinter. Men ved oppstarten var vi nødt til å endre noe på lengden på en sprint på grunn av problemer med møteavvikling (se vedlegg B). Denne endringen påvirket nødvendigvis påfølgende sprint, men etter disse innledende endringene har vi fulgt fastsatt lengde. Scrum er i utgangspunktet veldig bastant når det kommer til tid, men vi valgte å avvike fra modellen ettersom vi manglet noe informasjon for å fortsette.

ARBEIDSFORDELING

Vi valgte å dele opp oppgaven i tre hoveddeler, Møte, Sak og Konfigurering, og fordele disse mellom gruppemedlemmene. Dette har gitt en bra og forholdsvis jevn arbeidsfordeling på alle i gruppen. Denne måten å fordele oppgavene på har fungert veldig bra, ettersom alle har hatt hovedansvaret for en del. Dette har sikret at vi til enhver tid har visst hvor langt hver del av løsningen har kommet i utviklingen.

SUBJEKTIV OPPLEVELSE AV OPPGAVEN

Under følger en kort beskrivelse av opplevelsen av oppgaven fra hvert gruppemedlem.

Paul (prosjektleder) – Jeg har opplevd prosjekt som en veldig bra erfaring i form av kontakt med både ”kunde” og ”ledelse” i form av henholdsvis Løiten og Popkorn. Muligheten til å forsøke seg på utviklingen av en reel løsning har vært veldig lærerikt og morsomt. Ettersom utviklingen har foregått med ASP.NET har jeg og vi vært nødt til å settes oss inn i mye nytt som kommer godt med når vi kommer ut i arbeidslivet. På toppen av det hele har gruppen fungert som en drøm, med god innsats og godt humør under hele løpet. For min del har bacheloroppgaven vært en veldig bra og viktig erfaring på alle måter.

Simen – Jeg har opplevd prosjektet som særdeles positivt for min videre karriere. Erfaringer med .NET var noe vi ikke har fått kompetanse på ifra studieforløpet og jeg ser på dette som en stor fordel å ha med seg videre. Selve arbeidet med prosjektet har fungert utmerket, spesielt med tanke på det å jobbe i gruppe. Samarbeidet og humøret har vært upåklagelig og problemer har blitt løst med mye muntlige diskusjoner og ofte i full felleskap. Jeg føler vi virkelig har fått

uttrettet noe med prosjektet og laget en løsning som forhåpentligvis vil kunne brukes. Vi har lagt ned utrolig mye tid og kommet fram til en løsning vi kan gå god for.

Rino – Min opplevelse av prosjektet er kun positivt! Det har gitt et betydelig bidrag til kunnskapsnivået, siden vi ikke har hatt så mye Microsoft teknologi på skolen. Har også vært lærerikt å jobbe med et reelt prosjekt ute i "den virkelige verden". I tillegg har det vært veldig tilfredsstillende å starte med en idé som har endt opp i fungerende programvare. Vil også bemerke at vi har fungert veldig godt som gruppe. Alle har vært med å holde humøret oppe, selv når det har vært frustrerende perioder. Nå som vi nærmer oss slutten på prosjektet (og skoletiden her på HiG), er det nesten litt vemodig.

6.5 ERFARINGER

Prosjektet har medført mange erfaringer for alle i gruppa, og alle disse kan ikke nevnes her. Men vil peke på et tilfelle som hører til implementeringsdelen av prosjektet.

Som tidligere nevnt hadde ingen i gruppe noen erfaring med ASP.NET utvikling, men vi har alle en del erfaring med webutvikling i PHP, HTML og Javascript. Ingen av oss synes noe om nettsider som lastes på nytt hver gang en bruker klikker på en link. Derfor prøvde vi i starten å unngå dette ved og benyttet oss av javascript rammeverket JQuery og en del AJAX funksjonalitet. Men vi oppdaget fort at ASP.NET ikke er lagt opp på en slik måte at dette fungerer. Årsaken til dette er postback som finnes i ASP.NET. Som vil si at hver gang for eksempel en knapp trykkes blir hele siden lastet på nytt og alt som blir lagt inn ved hjelp av klientside script forsvinner. Dette står i sterk kontrast til hva vi er vant med fra PHP.

Vi fant en vei rundt dette ved å la serverdelen av applikasjonen få vite om de forandringene som er gjort på klientsiden. Ved å lagre unna dataene i et ASP element som heter asp:HiddenField og litt triksing, er det mulig å få ønsket vårt om å unngå postback oppfylt.

Vi klarte å få til sider som var mye "WEB 2.0", men valgte likevel å gå bort fra en slik løsning og heller implementere alt på måten ASP.NET legger opp til. Grunnen er at vi ville gått glipp av mange kraftige verktøy som ligger i å bruke C# maksimalt på serversiden, og koden ville blitt veldig vanskelig å vedlikeholde/videreutvikle.

6.6 KONKLUSJON

Det å gjennomføre et reelt prosjekt for en reel kunde har vært en veldig lærerik og nyttig erfaring. Vi har fått prøvd oss gjennom et, nesten, fullstendig utviklingsløp med kravspesifisering, design og implementering. Det eneste trinnet vi ikke har fått gjennomført er realisering gjennom levering til kunde.

I løpet av prosjektperioden har vi dratt nytte av mye tidligere tilegnet kunnskap innen systemutvikling og programmering, samt noe prosjektstyring. Vi har fått mye lærdom rundt kombinering av disse grenene innen informasjonsteknologi.

Vi har også vært heldig med tanke på at vi har hatt utfordringen med å sette oss inn i, og lære et helt nytt og ukjent utviklingsmiljø og programmeringsspråk. Kunnskapen om .NET og C# er høyaktuell ved overgangen til arbeidsmarkedet i dag, og har gitt oss en bredere horisont i forhold til webutvikling.

Den endelige konklusjonen må bli at tiden vi har brukt på bacheloroppgaven har vært en veldig lærerik og ikke minst morsom periode, og har vært en veldig bra avslutning på studiene ved HiG.

7 REFERANSER

[1-1]: Løiten - http://www.loitenalmenning.no/

[1-2]: Popkorn - http://www.popkorn.no/

[1-3]:] Scrum-diagram - http://www.controlchaos.com/about/

[3-1]: NHibernate - https://www.hibernate.org/343.html

[4-1]: JQuery - http://jquery.com/

[4-2]: TortoiseSVN - http://tortoisesvn.tigris.org/

[4-3]: PSWinCom - http://www.pswin.com/DesktopDefault.aspx?lang=no

[4-4]: ”Persisting the state of a web page” av Martin Jericho - http://www.codeproject.com/KB/applications/persistentstatepage.aspx

[4-5]: ViewState syklus - http://msdn.microsoft.com/en-us/library/ms972976.aspx

[4-6]: Cute Editor - http://cutesoft.net/

[4-7]: TinyMCE - http://tinymce.moxiecode.com/index.php

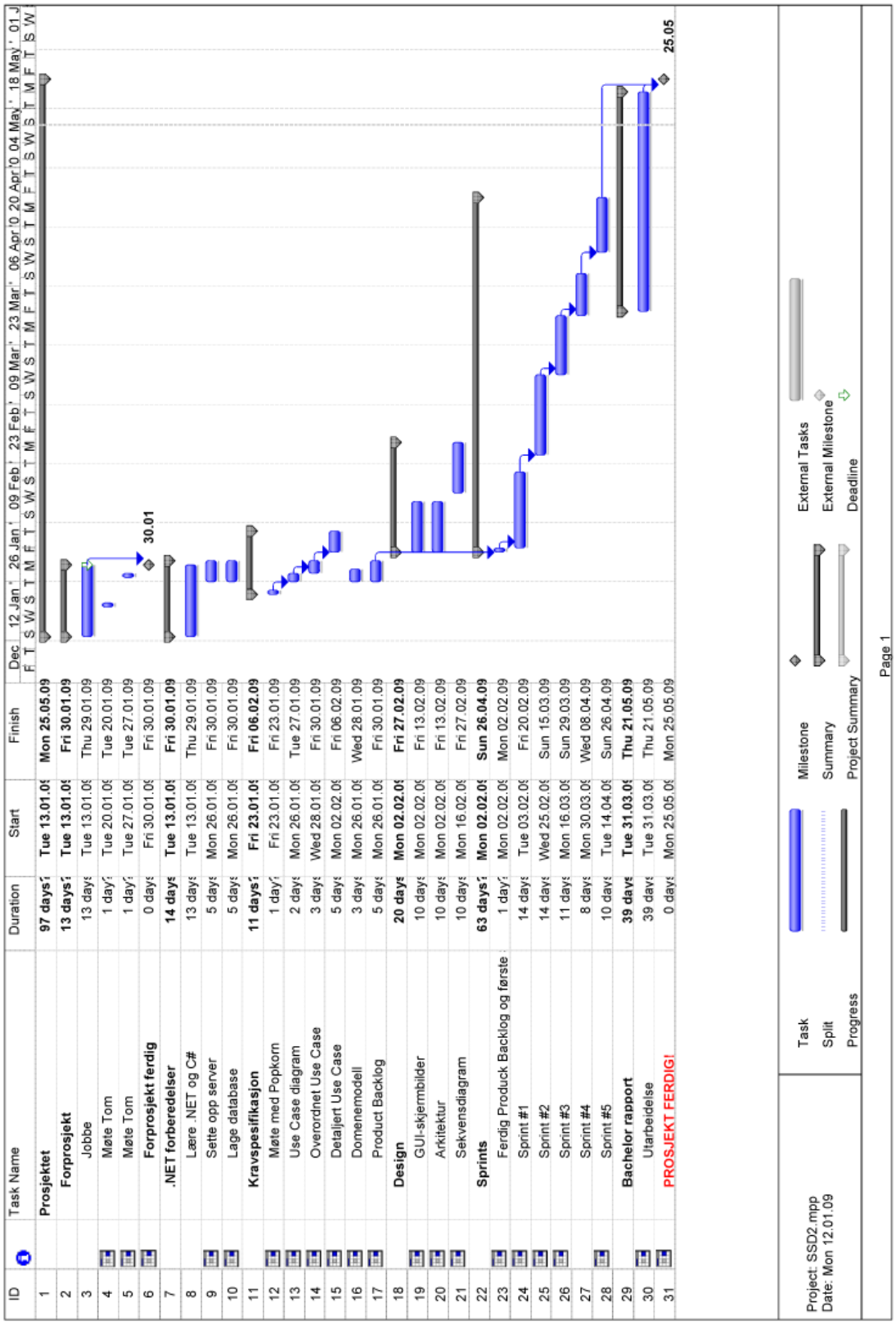
[4-8]: FCK editor - http://www.fckeditor.net/

VEDLEGG A – TERMINOLOGILISTE

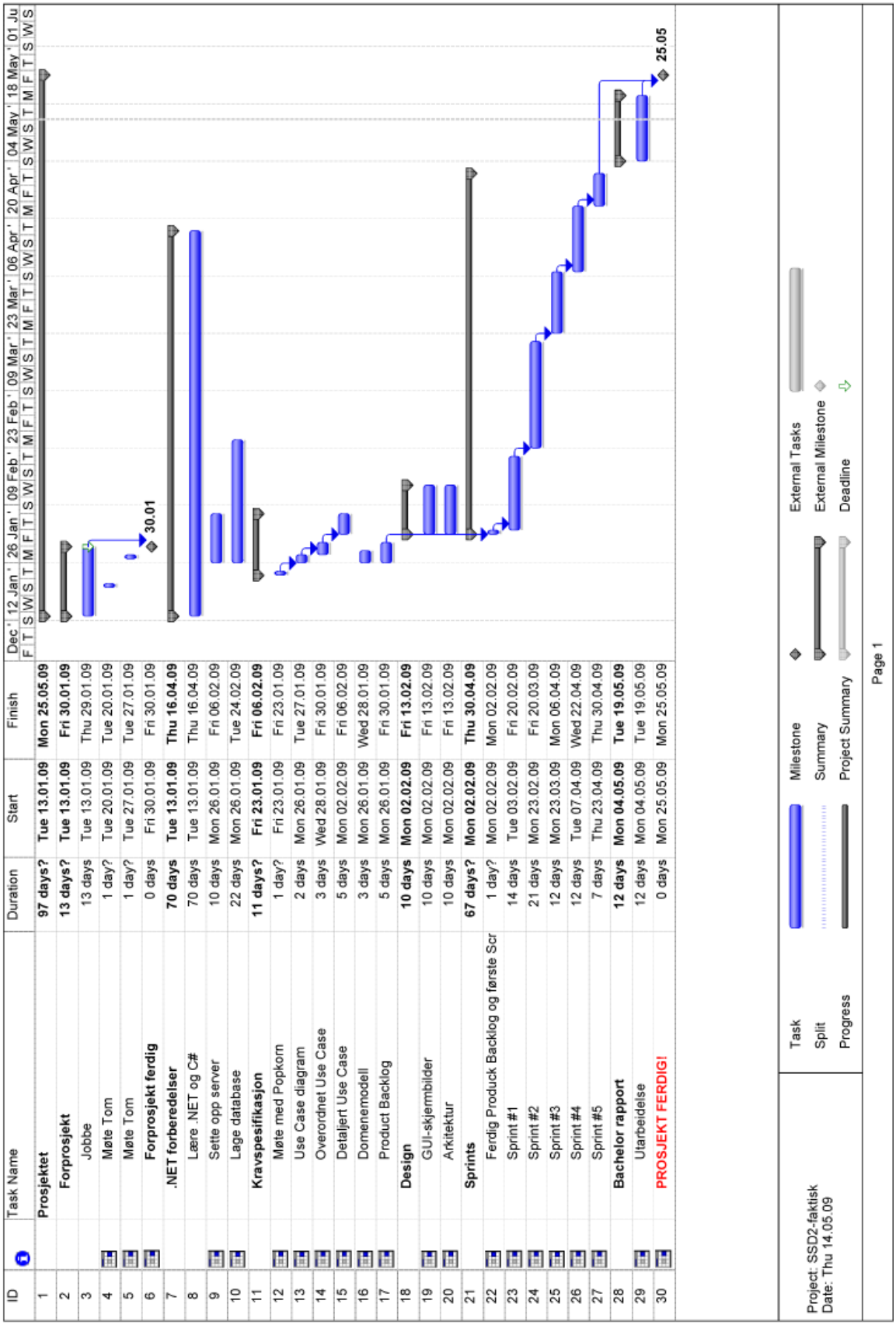
Allmenning(er)	Område som en bestemt gruppe har bruksrett til.
Hjemmelshaver	Er en som står som eier av en eiendom/gård. Det kan også være flere hjemmelshavere på en eiendom.
Persistent side	Statusen på siden holder seg selv om bruker skulle navigere seg bort.
FURPS	Akronym for en modell som klassifiserer programvare kvalitet som bl.a. er brukt i RUP. Functionality, Usability, Reliability, Performance, Supportability.
Festesak	Hvor det foreligger et leieforhold mellom grunneier og leietaker.
Bruksrettssak	Rett knyttet mellom person og eiendom.
T-SQL	Transact-SQL – Microsoft sitt proprietære SQL som de har sammen med Sybase.
Internasjonalisering	Muligheter for et internasjonalt marked, ved at programvaren kan tilpasses til lokale forhold i et annet land enn opphavslandet.
N-tier	En klient-server arkitektur hvor presentasjon, forretningslogikk og database er separert. N står for antall som i praksis vil si minst en 3-lags arkitektur.
Eventhandling	Hva som skjer når det skjer en interaksjon, for eksempel hvis en knapp trykkes.
Namespace	En måte å gi en unik identitet. Namespace i .NET kan sammenlignes med pakker i Java.
Mockups	Forslag på grafiske brukergrensesnitt uten funksjonalitet. Består ofte som et bilde.
Reporting Service	MS SQL Server gir muligheter for rapportgenerering i Reporting Service, hvor kommunikasjonen mellom applikasjon og SQL Server foregår v.h.a. Web Service.
Singleton	Design pattern fra ”The Gang of Four”, som garanterer et det er kun et objekt av en klasse.
NHibernate	Rammeverk for å konvertere en objektorientert modell til relasjonsdatabase modell, og omvendt.
Stored Procedures	Lagret prosedyre i databaselaget.
JQuery	Javascript rammeverk som forenkler en del funksjonalitet og i tillegg sikrer at script virker i alle browsere.
MasterPage	ASP.NET sitt svar på templates i PHP. Det vil si en hovedside som bestemmer felles utseende til en webapplikasjon.
Contentpage	Innholdssidene som legges inn i MasterPage ved navigering.
ViewState	Administrering av sidestatus. I ASP.NET ligger ViewState skjult på sidene. Det inneholder og ivaretar status.
PHP	Serverside språk og rammeverk som ikke er lisensbelagt. Motsvar til C# og .NET 3.5.

VEDLEGG B - FRAMDRIFTSPLAN

PLANLAGT GANTSKJEMA



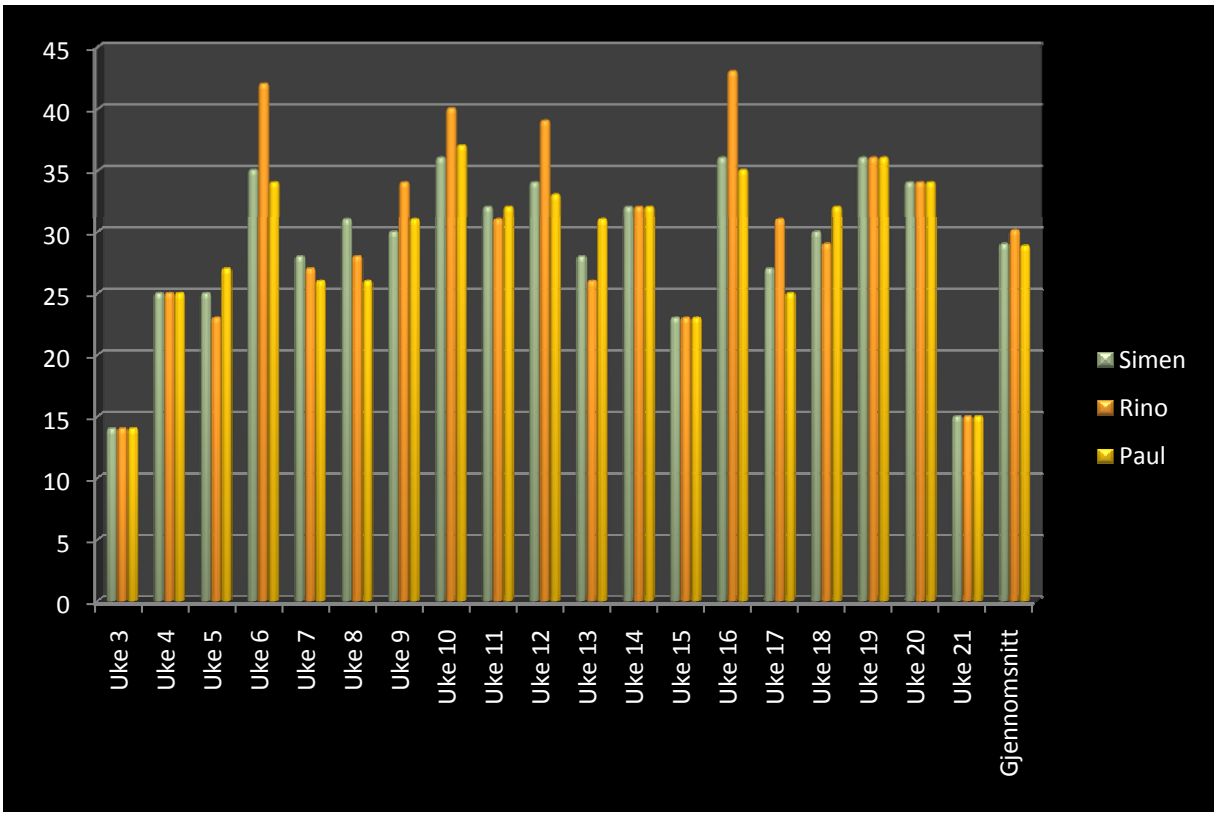
FAKTISK GANTSKJEMA



ID	Task Name	Duration	Start	Finish
1	**Prosjektet**	**97 days?**	**Tue 13.01.09**	**Mon 25.05.09**
2	**Forprosjekt**	**13 days?**	**Tue 13.01.09**	**Fri 30.01.09**
3	Jobbe	13 days	Tue 13.01.09	Thu 29.01.09
4	Møte Tom	1 day?	Tue 20.01.09	Tue 20.01.09
5	Møte Tom	1 day?	Tue 27.01.09	Tue 27.01.09
6	**Forprosjekt ferdig**	0 days	Fri 30.01.09	Fri 30.01.09
7	**.NET forberedelser**	**70 days**	**Tue 13.01.09**	**Thu 16.04.09**
8	Lære .NET og C#	70 days	Tue 13.01.09	Thu 16.04.09
9	Sette opp server	10 days	Mon 26.01.09	Fri 06.02.09
10	Lage database	22 days	Mon 26.01.09	Tue 24.02.09
11	**Kravspesifikasjon**	**11 days?**	**Fri 23.01.09**	**Fri 06.02.09**
12	Møte med Popkorn	1 day?	Fri 23.01.09	Fri 23.01.09
13	Use Case diagram	2 days	Mon 26.01.09	Tue 27.01.09
14	Overordnet Use Case	3 days	Wed 28.01.09	Fri 30.01.09
15	Detaljert Use Case	5 days	Mon 02.02.09	Fri 06.02.09
16	Domenemodell	3 days	Mon 26.01.09	Wed 28.01.09
17	Product Backlog	5 days	Mon 26.01.09	Fri 30.01.09
18	**Design**	**10 days**	**Mon 02.02.09**	**Fri 13.02.09**
19	GUI-skjermbilder	10 days	Mon 02.02.09	Fri 13.02.09
20	Arkitektur	10 days	Mon 02.02.09	Fri 13.02.09
21	**Sprints**	**67 days?**	**Mon 02.02.09**	**Thu 30.04.09**
22	Ferdig Produck Backlog og første Scr	1 day?	Mon 02.02.09	Mon 02.02.09
23	Sprint #1	14 days	Tue 03.02.09	Fri 20.02.09
24	Sprint #2	21 days	Mon 23.02.09	Fri 20.03.09
25	Sprint #3	12 days	Mon 23.03.09	Mon 06.04.09
26	Sprint #4	12 days	Tue 07.04.09	Wed 22.04.09
27	Sprint #5	7 days	Thu 23.04.09	Thu 30.04.09
28	**Bachelor rapport**	**12 days**	**Mon 04.05.09**	**Tue 19.05.09**
29	Utarbeidelse	12 days	Mon 04.05.09	Tue 19.05.09
30	PROSJEKT FERDIG!	0 days	Mon 25.05.09	Mon 25.05.09

VEDLEGG C -TIMELOGG, MØTEREFERATER

TIMELOGG



Vi har valgt å vise hvor mange timer hvert av gruppemedlemmene har brukt på oppgaven per uke. De siste søylene viser gjennomsnittlig arbeidsmengde som har vært tett på 30 timer hver uke.

MØTEREFERATER

MØTE 08.10.2008

Dette møtet fant sted før selve bacheloroppgaven startet, men er et særdeles viktig møte. Gruppens medlemmer var også ikke helt den samme.

Opprettet:
08.10.2008.

Prosjektmøte #1 - 2008: Styresakssystem for Popkorn A / S, og ekstern kunde Løiten Almenning.

Tilstede: HiG v/ Øyvind, Rino, Simen og Chris Stian, Popkorn v/ Kai Arne Aspeli og Løiten Almenning v/ Johan Fischer.

Tid: 08.10.2008, 0900 - 1100 (slutt ca. 1110). **Sted:** Hamar, Popkorn A/S.

Sammendrag:
Løiten almenning gikk gjennom hvordan saksprosessen er i dag og hvordan de ser for seg at dette kan bli løst av en elektronisk løsning, og diskusjon rundt funksjonalitet og krav. Popkorn la samtidig frem innskytelser ved de enkelte punktene og ting som kom opp under diskusjon, samt viste frem en skisse av tabellmodeller for sak, møte og eiendomsoppsett.

NB! Nytt møte avtalt ca kl. 0900-ish, onsdag 22.10.2008, i forbindelse med mappe 4, Kai Arne kommer da alene på besøk til HiG, oppmøte kantina. (Husk å reservere møterom / grupperom!).

Referatutkast: Gangen i en styresak:
Saksbehandler database i tillegg

- Saksbehandler legger søknad digitalt med engang, selv om h*n kanskje ikke skal i selve møtet.

Styresak kan stå alene eller knyttes til én eller flere eiendom'er (f.eks. vannledning til flere eiendom'er).

Skrive innkallinger til møte og linke ulike saker til gjeldende møte (med en kort beskrivelse) og forslag til vedtak. MAL!!

Feste saker er aldri med på møter, men vil gjerne være med i systemet / programvaren, fordi den skal protokollføres.

- velge sakstype (så festesak blir automatisk lagt inn i kallingen og møtets saksliste, men kommer opp som vedtatte saker i protokollen).

De to eneste typene som både er på innkalling, møtets saksliste og da selvfølgelig i protokollen, vil da være av typen bruksrettssaker og annet (dvs, alt annet enn festesaker).

Rapportfunksjon som sender automatisk .pdf som mail (til de som er innkalt til møte??), og

utskrift klar til postlegging for de som ikke har mail.

Konfigureringsfunksjonalitet som avgjør om det vil være én pdf for hele møteinnkallingen med bilder og saker og søknader og hele pakka, eller mulighet for å separere opp i enkelte filer.

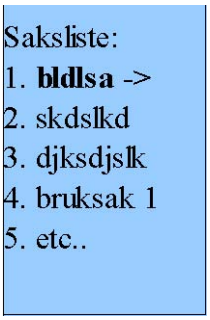
I protokollen kun vedtakstekst på saken, ikke hvor mye de evt. får i støtte / utbetalt beløp, da dette ikke blir vedtatt / utregnet før i etterkant som en egen utestående prosess. Bemerk! Det et alternativt ønske fra kunde (Løiten Almenning) om å kunne føre på dette i etterkant på bruksnr'et til søkeren.

Protokoll / referat / saksbehandlingsvedtak foretas live under styremøte, det ønskes da derfor:

- Mulighet for å endre vedtaksforslag (alle bruksrettsaker har et ferdig forslag i innkallingen og sakslista) og / eller godta gitt forslag uten endringer.

- Mulig ønske om merknadsfelt / evt. kommentarfelt hvor man kan legge til kommentarer utover selve vedtaksteksten som følger saken. Denne kommentardelen vil imidlertid kun vises i protokollen / møtereferatet. (eks. Navn navnesen var inhabil i saken og stemte ikke; 4 for og 1 i mot etc...).

Når møtet er over, ønskes det en trykknappsak som: generer protokollen med alle vedtatte vedtak og eventuelle ekstra kommentarer og annet av funksjonalitet som er ønskelig. (kan komme mer senere).



Eksempel på skisse av skjermbilde som brukes under styremøtets effektuering:

Valgt sak / sak under behandling: Saksliste:

Møte: ..dsads Sak: 1. bldlsa

1. **bldlsa ->**
2. skdslkd

Gitt søknad og ekstra papirer etc? Eller annet

3. djksdjslk

materiale relevant til saken

4. bruksak 1
5. etc..

Forslag til vedtak: dsjldajdasl

Vedta forslag Endre

NB! Bruksnr for eiendom linkes til vedtak

+ festesaker i protokollen

Hente dbinfo og knytte opp mot, fra skogdatabase (statisk, ikke dynamisk i runtime, muligens midt på natta annenhver uke / måned etc...), dvs lage eget API mot styresaksdatabasen slik at den lettere kan generelt porteres til andre typer firmaer, som også enten har eller ikke har ekstern kundedatabase fra før.

Bakgrunn for systemet (problemet):

- Mange vedtak og saker (600 bruksrettigheter i omløp, 1000 hyttetomter m.m) som gjør at det er vanskelig å holde kontroll på hva som er vedtatt, ikke vedtatt og som avventer behandling.

I tillegg til protokoll skal det være utskrift av enkeltsak (altså hver bruksretts nåværende utøver som har fått en sak behandlet med vedtak), skal få en utskrift av saken, og det vedtatte vedtaket med underskrift av almenningsbestyrer. Dette må kunne genereres automatisk på en eller annen måte.
M.a.o. kopiere saksbeskrivels + nr, + gnr + bruksnr + navn + vedtakstekst i brevform.

VEDLEGG D – DETALJERT KRAVSPESIFIKASJON

DETALJERT USE CASE BESKRIVELSE

Use Case:	**Registrer sak**
Aktør:	Saksbehandler
Mål:	Registrere ny sak og legge til i systemet.
Beskrivelse:	Saksbehandler oppretter en ny sak og legger inn saksmateriale og all informasjon relevant til saken. Systemet oppdaterer så databasen.
Type:	Viktig.
Pre betingelser:	Saksbehandler er innlogget og databasen er operasjonell.
Post betingelser:	Saken ble vellykket lagt inn og all informasjon knyttet opp mot database.
Spesielle krav:	Forutsetter at saksmateriale er gjort om til elektronisk format i forkant av saksopprettelsen.

Detaljert hendelsesforløp:	
Aktør:	System:
1. Saksbehandler klikker på "registrer sak" i gui'en.	
	2. Systemet viser registreringsskjema og relevante brukeropsjoner til skjerm.
3. Saksbehandler taster inn saksnr, og identifikasjonsnummer på den / det saken gjelder. (Bruksnr, fødsels- og personnummer o.l.).	
	4. Systemet henter ut registrert materiale basert på identifikasjonsnummeret, og oppdaterer dette automatisk på skjermbildet.
5. Saksbehandler velger sakstype.	
	6. Systemet oppdaterer registreringsskjemaets nedre del i forhold til valgt sakstype.
7. Saksbehandler legger inn sakstypeinformasjon (saksbeskrivelse, forslag til vedtak osv).	
8. Saksbehandler trykker på legg til saksvedlegg.	
	9. Systemet åpner opp en fildialog.
10. Saksbehandler velger fil fra disk / minnepinne / cd / osv og trykker på ok.	
	11. Systemet henter ut filadressen og oppdaterer skjermbildet med filen.
12. Steg 8 – 11, repeteres til alle filer er lagt til. Saksbehandler trykker da på "Registrer".	
	13. Systemet oppdaterer databasen med den nye saken sine data, henter vedlagte filadresse(r) fra angitt posisjon, overfører den / de til server og legger inn link til filen(e) i databasen. Viser vedlagt fil som én linje på skjerm.
	14. Systemet kommer med melding til skjerm om at saken ble registrert.

Alternative scenarioer:
Lovlige alternative scenarioer:
3. Systemet kan ikke finne noe på inntastet identifikasjonsnummer, eller det er tastet inn på feil måte (eks. bokstaver istedenfor tall o.l.), systemet viser så feilmelding relativ til hva som ble gjort feil, og ber brukeren taste inn på nytt.

10. Saksbehandler vil avbryte filhentingen, og trykker på avbryt, systemet viser forrige skjermbildet, uten å gjøre noen endringer. 12a. Saksbehandler ønsker ikke legge opp saken allikevel, og trykker på avbryt. Systemet avbryter så "registrer sak", og rydder vekk evt. registrert informasjon, og sender bruker tilbake til forrige skjermbilde uten å gjøre noen endringer. 12b. Saksbehandler ønsker å fjerne en vedlagt fil, og trykker på fjernknappen ved siden av filen. Systemet fjerner filinformasjonen og oppdaterer skjermbildet.
Ulovlige alternative scenarioer:
4. Databasefeil under opphenting, eller feil på data, systemet viser feilmelding og ber bruker prøve igjen / tilkalle systemadministrator. 9. Kan ikke lese fil, systemet gir så errormelding til skjerm. 13. Databasefeil under oppdatering, systemet gir melding om alvorlig systemfeil og at administrator må tilkalles.

Use Case:	**Opprette møte**
Aktør:	Saksbehandler
Mål:	Opprette et møte, med møtebesetning og styresaker
Beskrivelse:	Søknader som krever styrevedtak legges til møtet. Festesaker blir automatisk med på møtets saksliste, men kommer bare frem som vedtatte saker. Festesakene er ikke med på selve møtet, men er med i systemet for å protokollføres. Møtets dato og sted blir fastsatt.
Type:	Viktig
Pre betingelser:	Søknader som skal til styre må være registrert.
Post betingelser:	Blir generert en møteinnkalling til det opprettede møte.
Spesielle krav:	Må ha bruksrettsaker eller andre saker enn festesaker som ikke tidligere har vært styrevedtatt

Detaljert hendelsesforløp:	
Aktør:	System:
1. Oppretter et nytt møte med tidspunkt.	
	2. Sjekker gyldig dato, lagrer møtet.
3. Legg til møtedeltager.	
	4. Lagrer deltager på aktuelt møte.
5. Steg 3 – 4, repeteres til alle deltagere er lagt til.	
6. Legg sak til møte.	
	7. Linker sak til aktuelt møte.
7. Steg 6 – 7, repeteres til alle ønskede saker er lagt til. Saksbehandler trykker da på "Opprett møte".	
	8. Systemet lagrer alle registreringer og kommer med melding til skjerm at møtet ble vellykket opprettet.

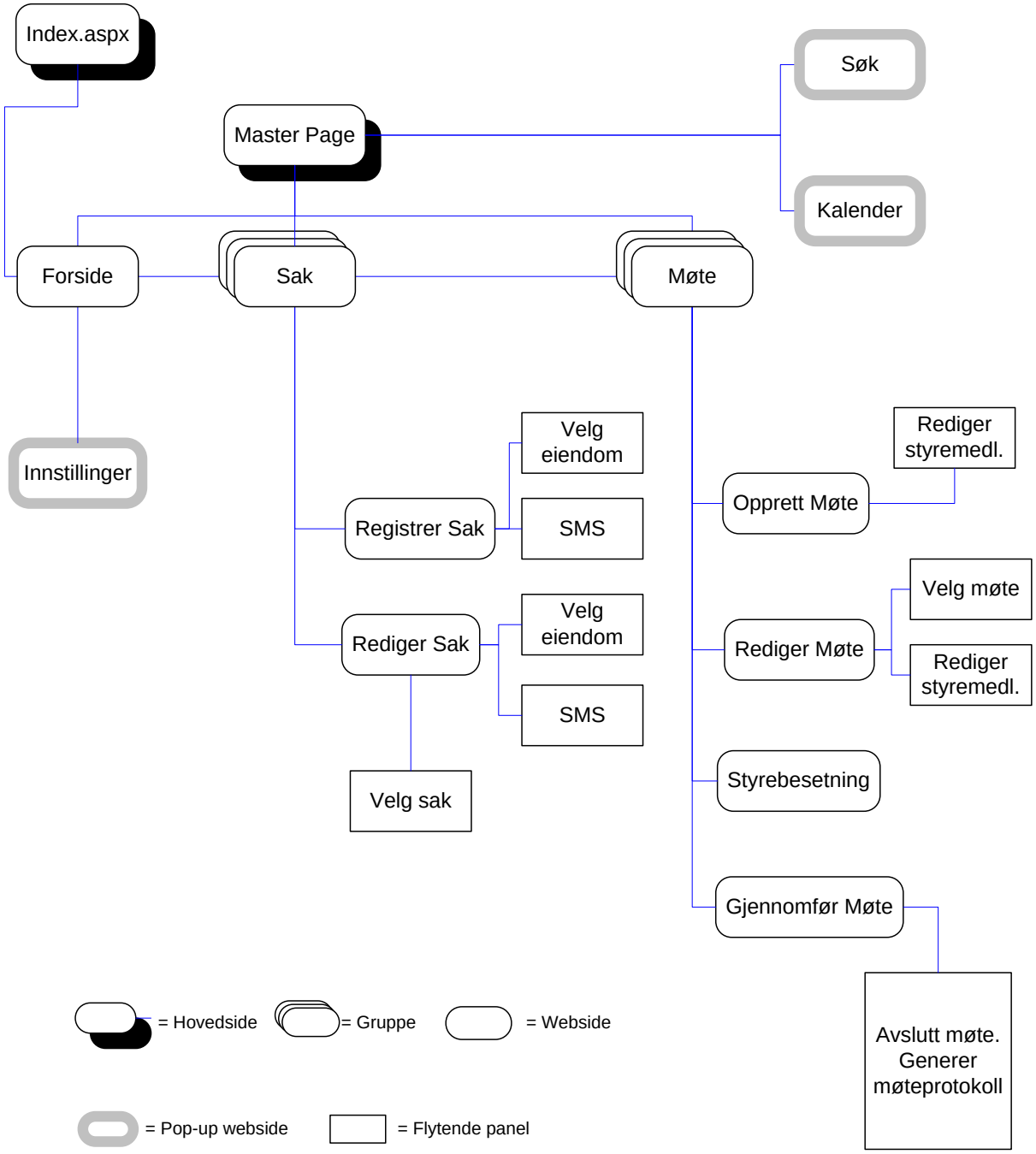
Alternative scenarioer:
Lovlige alternative scenarioer:
7. Saksbehandler ønsker ikke opprette møtet allikevel, og trykker på avbryt. Systemet avbryter så "opprette møte", og rydder vekk evt. registrert informasjon, og sender bruker tilbake til forrige skjermbilde uten å gjøre noen endringer.
Ulovlige alternative scenarioer:
3. Hvis aktøren kobler på personer som ikke sitter i styret, kommer det en feilmelding opp og personen blir ikke lagt til. 6. Hvis noen saker ikke klarer å få tilgang til alle dokumenter av en eller annen grunn, kommer en melding opp om hvilken søknad det gjelder. Styresaken må da legges opp på nytt.

Use Case:	**Oppdater egen DB**	
Aktør:	KundeDB	
Mål:	Oppdatere egen database mht. kundedata.	
Beskrivelse:	Systemets database oppdaterer seg mot KundeDB slik at styresaker kan knyttes opp mot organisasjonens interne kundedataregister. Grunnen er at resten av systemet trenger oppdaterte kundedata.	
Type:	Viktig.	
Pre betingelser:	Egen db er klar for oppdatering.	
Post betingelser:		
Spesielle krav:		
Detaljert hendelsesforløp:		
Aktør:	System:	KundeDB system respons:
1. Use caset starter med at KundeDB automatisk trigger oppdateringsAPI periodisk.		
	2. Oppretter kobling til KundeDB	
		3. Bekreft tilkobling.
	4. Spørring til KundeDB via. stored prosedures i respektive DB.	
		5. Returner felt.
	6 a. Eksisterer felt, oppdater felt i KundeDB.	
	6 b. Eksisterer ikke felt, opprett felt i KundeDB.	
	Gjenta steg 3 – 4 for alle felter i Skogdatabasen.	
	7. Commit data.*	
	8. Lukker kobling til Skogdatabasen.	
		9. Bekreft frakobling.
	10. Oppdater egen db systemlogg.	
Alternative scenarioer:		
Lovlige alternative scenarioer:		
Ulovlige alternative scenarioer:		
Illegal alternative flow a*		
	*a. Alle mulige tider, System feiler:	
	1. Ikke commitede data forkastes-KundeDB rulles tilbake til forrige tilstand	
	2. Frakoble databasen.	
		3. Bekreft frakobling.
	4. Oppdater egen db systemlogg.	
	5. Gi feilmelding.	
6. Signaliser feil til stakeholder.		
7. «Oppdater egen DB».		
Repeter punkt 7 iht. konfigurasjonsinst. og/eller til timeout.		
Illegal alternative flow b*		
		*b. Alle mulige tider, system feiler:

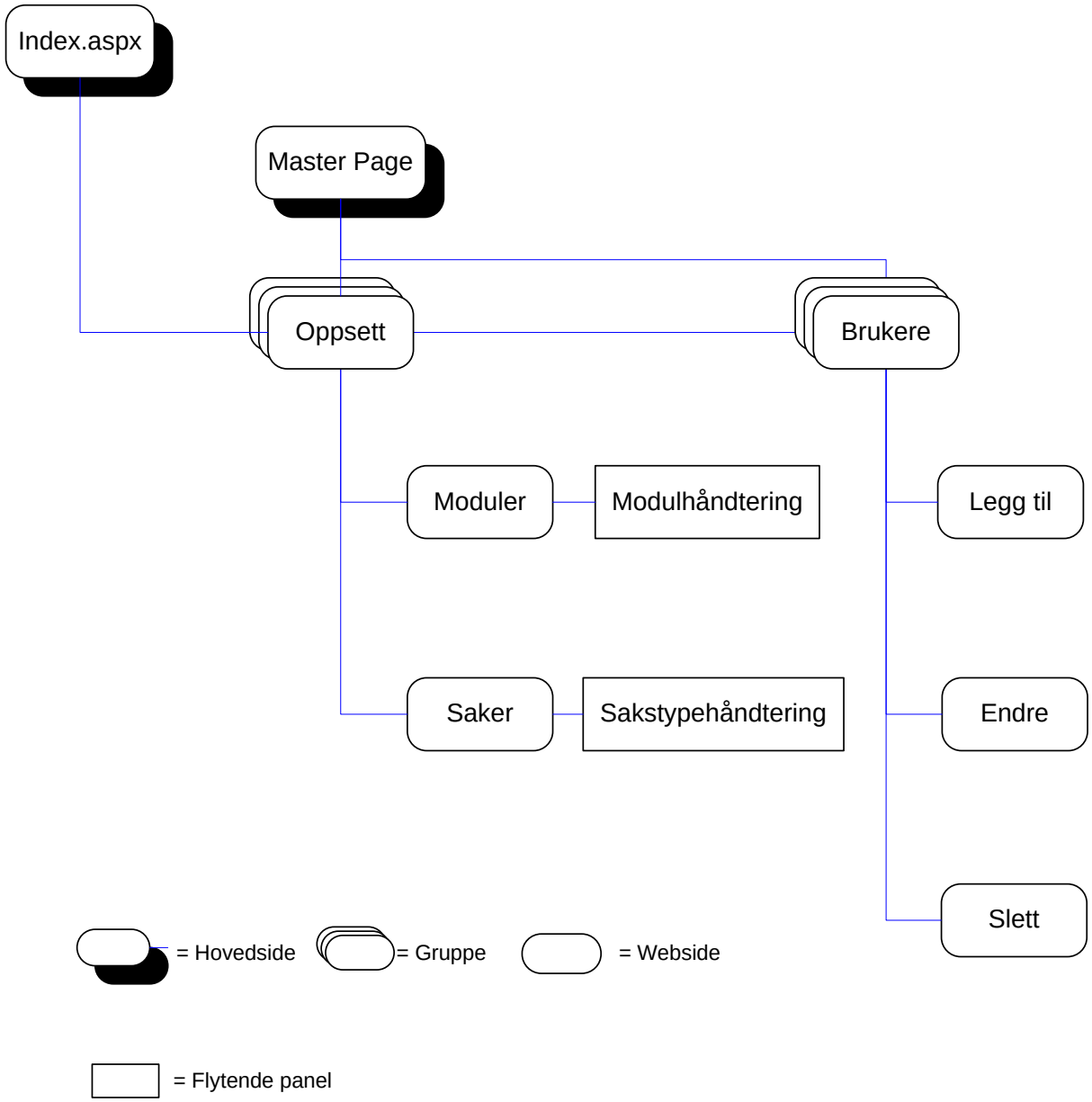
		1. Gi feilmelding.
	2. Frakoble databasen.	
	3. Oppdater egen db systemlogg.	
	4. Oppdater egen db systemlogg.	
	5. Tilkobligsforespørsel sendes til databasen.	
	Repeter punkt 5 iht. konfigurasjonsinst. og/eller til timeout.	
	6. Oppdater egen db systemlogg.	
	7. Gi feilmelding.	
8. Signaliser feil til stakeholder.		

VEDLEGG E – DESIGNSKJEMAER

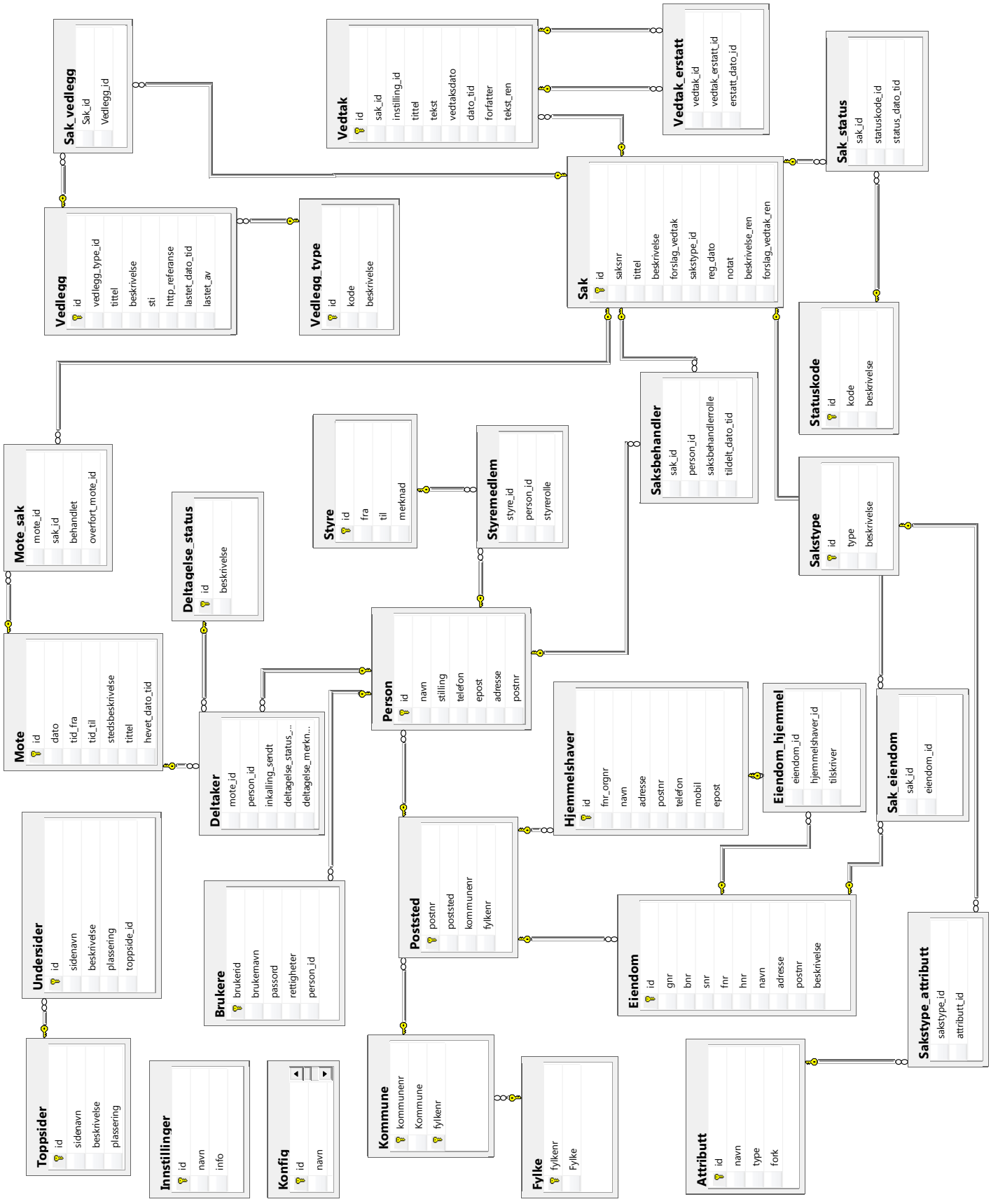
SIDEKART – HOVEDDEL



SIDEKART – KONFIGURASJONSDEL



DATABASEDIAGRAM



Forprosjekt

Bacheloroppgave 2009
Styresaksdatabase

Høgskolen i Gjøvik

Simen Tveit Backstrøm
Rino Werner Falstad
Paul Magne Lunde

Innhold

1 Mål og rammer

1.1 Innledning

Popkorn er en bedrift med tilholdssted på Hamar som har spesialisert seg på levering av skreddersydde IT-løsninger. På forespørsel fra Løiten Almenning, et selskap som forvalter jord- og skogbruksområder på Løten, har Popkorn gitt i oppgave å lage et system for organisering av styresaker.

1.2 Effektmål

Når Løiten Almenning får på plass en slik løsning betyr det at de får enklere tilgang til saker, samt betraktelig forkortet behandlingstid av søknader.

For Popkorn som eier av løsningen innebærer en fullt fungerende løsning med mulighet for konfigurering økonomisk utbytte gjennom videresalg til andre bedrifter. På sikt ser de også for seg å benytte løsningen internt for å håndtere sine egne saker.

1.3 Resultatmål

Målet gruppen har satt seg er å utvikle en fungerende løsning som tilfredsstiller alle krav som Løiten Almenning har stilt. Etterhvert som delleveransene av løsning er på plass skal de testes av brukerene hos Løiten Almenning og vurderes av Popkorn. I tillegg til at løsningen skal tilfredsstille minimumskravene, skal muligheten for konfigurering (se punkt 2.1) mot bruk i andre bedrifter enn Løiten Almenning utvikles.

Oppdragsgiver har nevnt at de ønsker en åpning for en desentralisert nettløsning i fremtiden. En protoype eller kartlegging av en slik løsning er et tillegg som vi vil ta med om tiden strekker til, men vil ikke bli prioritert fremfor andre deler av oppgaven.

1.4 Læringsmål

I løpet av prosjektperioden kommer vi til å prøve ut det vi har lært tidligere i studiet, dette innebærer systemutvikling som i stor grad kommer til å ha benytte. Sammen med dette kommer programmeringskunnskapene vi har fra flere tidligere emner til å være uvurderlige.

Ettersom systemet skal utvikles i .NET må vi sette oss inn i og tilegne oss

kunnskap om bruken av dette utviklingsmiljøet, programmeringsspråket C# og teknologien rundt ASP.NET. Dette er et helt nytt område som vi ikke har fått noen innføring i tidligere i studieløpet.

1.5 Rammer

Utover de tidsfristene som er satt av høgskolen har ikke oppdragsgiver ytret noe ønske om andre tider eller flere delleveranser. Vi kommer likevel til å gi oppdragsgiver en delleveranse hver 14. dag.

Vi har kun fått føringer på utviklingsmiljø fra oppdragsgiver, utover dette har vi ingen definerte rammer for arbeidsprosessen.

2 Omfang

2.1 Oppgavebeskrivelse

Oppgaven er å lage en styrsaksdatabase som skal benyttes for håndtering, planlegging og gjennomføring av styremøter, samt tilhørende saks- og vedtaksbehandling av styresaker. I tillegg til kravene og forventningene fra Løiten Almenning ser Popkorn for seg muligheten til et konfigurerbart system som kan benyttes i ulike bedrifter. Tilpassning til andre bedrifter skal gjøres ved å legge til eller fjerne moduler i løsningen.
Hovedpunkt i oppgaven i prioritert rekkefølge:

- Utvikle en fullt fungerende løsning som tilfredsstiller kravene fra Løiten Almenning.
- Gjøre løsningen modulbasert og egnet for konfiguering mot andre bedrifter.
- Kartlegge/lage prototype for en desentralisert versjon av løsningen.

De to første punktene skal gjennomføres i løpet av prosjektperioden og ende ut i fungerende programvare. Vi har som mål å få med det siste punktet i tillegg, men det kommer ikke til å bli prioritert fremfor andre deler av oppgaven.

Løsningen skal ha mulighet for å opprette møter utover i året, disse møtene skal vises i en kalender. Hvert møte skal ha en modul for saksbehandler hvor han/hun kan legge til informasjon og forslag til vedtak. Når sakslisten er klar skal det sendes ut innkallinger til styremedlemmer. Protokollen for møtet skal tas vare på og saker som ikke blir behandlet på et møte skal flyttes til neste. Grensesnittet på løsningen skal være web-basert. For håndtering av dokumentgenerering må løsning inneholde generering av PDF og/eller OpenXML dokumenter. I tillegg skal det være mulig med scanning og opplasting av vedlagte saksdokumenter. Innkallinger skal gjøres ved hjelp av SMS, e-mail eller vanlig post. For å sikre integriteten i løsning må det også opprettes en sikkerhetsmekanisme for pålogging.

Som nevnt skal løsningen kunne brukes i andre typer bedrifter som kan ha behov for andre typer saker. For å løse dette må det benyttes en modulbasert tilnærming under utviklingen slik at det enkelt kan legges til og fjerne moduler.

I fremtiden kan det være aktuelt med en desentralisert versjon av løsningen, dette bør det tas høyde for under utviklingen. Ettersom systemet skal utvikles i moduler skal det i utgangspunktet være en enkel sak å tilpasse løsningen til en desentralisert versjon.

3 Prosjektorganisering

3.1 Ansvarsforhold

- Paul M. Lunde er prosjektleder og ansvarlig for sammensetting av dokumentasjon.
- Rino W. Falstad er ansvarlig for visuelle planer, skjemaer og dokumentasjonsdiagrammer.
- Simen Backstrøm har ansvar for utstyr og oppretting/vedlikehold av nettsiden.

Utover ansvarsområdene gruppemedlemmene har blitt tildelt ovenfor har vi valgt å dele opp utviklingsprosessen i tre deler som hvert gruppemedlem har hovedansvar for. Disse områdene blir igjen delt opp i mindre deler som i hovedsak blir utviklet av hovedansvarlig. De tre områdene og hovedansvarlig er som følger:

- Konfigurasjon og administrasjon - Paul M. Lunde
- Saker - Simen Backstrøm
- Møte - Rino W. Falstad

3.2 Rutiner og regler

- Alle gruppemedlemmer skal totalt bruke 30 timer i uka på prosjektet. Av disse skal 15 timer tilbringes sammen på rom A018B to "fellesdager", tirsdager og fredager.
- Alle gruppemedlemmer skal føre logg med timeantall og arbeidsoppgaver som er utført.
- Ved fravær på "fellesdager" skal vedkommende melde fra så fort som mulig, samt begrunne fraværet.
- For å sikre lik dokumentasjon, skal det benyttes maler som er tilgjengelige for alle i gruppen via felles lagringsområde.

3.3 Øvrige roller og bemanning

- Tom Røise, høgskolelektor HiG, er veileder for gruppen i prosjektperioden.
- Stian Bakken, fagsjef hos oppdragsgiver Popkorn, blir støttespiller under innlæring av Visual Studio, .NET og C#.

- Kai Arne Aspeli, kontaktperson hos Popkorn, vil med jevne mellomrom bli informert om fremgangen under prosjektperioden.
- Johan Fisher, bestiller av løsningen, konsulteres under prosjektperioden for å få riktig informasjon om fagområdet.

4 Planlegging, oppfølging og rapportering

4.1 Hovedinndeling av prosjektet

Det er gitt klare føringer på hvilke hovedfunksjoner løsningen skal ha fra Løiten Almenning og Popkorn, likevel er det ønskelig med en systemutviklingsmodell som er åpen for endringer underveis. Ettersom Popkorn har ønske om å være med på prosessen i form av fremskrittsvurdering med jevne mellomrom er en iterativ modell å foretrekke. Som tidligere nevnt skal løsningen være modulbasert. Dette, sammen med de foregående punktene, leder oss til å bruke Scrum som utviklingsmodell, men med 14 dagers iterasjoner istedenfor 30 dager som er hva modellen anbefaler. I tillegg til iterasjonene gir Scrum føringer på at hver sprint skal gi fungerende programvare, og ved starten av ny sprint velger Product Owner høyprioritets moduler fra Product Backlog. Dette samsvarer veldig bra med hva både vi og oppdragsgiver ønsker.

Et viktig punkt er å få på plass en så detaljert kravspesifikasjon som mulig, dette oppnås i møter med Popkorn og Løiten Almenning. Når kravspesifikasjonen er på plass skal vi dele den opp i mindre moduler som skal inn i Product Backlog. Det blir avtalt møter med Popkorn hver 14. dag, slik at man kan utføre Sprint planning meeting og Sprint review meeting. I det første Sprint planning meeting vil vi, sammen med oppdragsgiver, gå gjennom Product Backlog og velge ut de viktigste modulene til den første Sprint Backlog. Etter at en sprint er fullført, skal det være et nytt møte hvor Sprint review meeting holdes først med en demo av fullførte moduler. Deretter holdes Sprint planning meeting, hvor nye moduler velges til neste Sprint Backlog. Her fravikes det litt fra modellen, når review og planning meeting holdes i samme møte, for å effektivisere prosessen på grunn av tidsrammen i prosjektperioden.

I startfasen av prosjektet er det ønskelig å få opp en prototype av det grafiske brukergrensesnittet så fort som mulig. Grunnen er at det gir en bra visuell test på om ønsket funksjonalitet er tatt med i kravspesifikasjonen. En annen grunn er at en gui-prototype vil gi et forståelig bilde av systemet for bestiller. Samtidig med utvikling av gui kommer det til å være høy fokus på utvikling av databasen ettersom hele systemet er avhengig av denne.

4.2 Plan for statusmøter og beslutningspunkter

Det er satt opp et fast møte med veileder hver uke (tirsdag) med mulighet for avlysning dersom det ikke er behov for dette. Som nevnt over skal det være et (Scrum) møte hver 14. dag med oppdragsgiver for fremvisning av fullførte

moduler og valg av nye for neste Sprint. Dersom dette ikke kan realiseres må vi benytte alternative metoder for å få på plass en ny Sprint Backlog. For å unngå tilbakeskritt i utviklingsprosessen er det viktig at disse møtene blir gjennomført like ofte som planlagt, slik at vi får tilbakemelding på punkter oppdragsgiver ønsker endret fortløpende.

De dagene gruppen har "fellesdager" skal det holdes et Daily meeting først på dagen hvor følgende spørsmål stilles:

1. Hva gjorde du sist?
2. Hva vil du gjøre til neste gang?
3. Hvilke problemer oppsto?

De dagene gruppen ikke er samlet blir det ingen fastsatt felles oppsummering på denne måten, men oppdateringer skjer eventuelt via e-mail. I tillegg skal det være et møte etter hvert Sprint planning meeting hvor Sprint Backlog deles opp i oppgaver på 4-16 timers arbeid. For å estimere arbeidsmengden i hver oppgave kommer vi til å benytte oss av en form for estimeringspoker.

5 Utviklingsmiljø

Systemet skal utvikles i Microsoft .NET rammeverk. Ettersom oppdragsgiver i hovedsak benytter seg av .NET under utvikling av egne løsninger kommer vi også til å benytte oss av dette. Ved å benytte dette rammeverket kan oppdragsgiver enkelt videreutvikle løsningen i ettertid. Vi kommer til å benytte Microsoft Visual Studio 2008 til utvikling i prosjektperioden.

6 Organisering av kvalitetssikring

6.1 Standarder og kildekode

Kildekode skal kommenteres/dokumenteres slik at det er lett for andre, både gruppemedlemmer og utviklere hos oppdragsgiver, å sette seg inn i den. All kildekode og kommentering skal skrives på norsk ettersom løsningen i førsteomgang utvikles for en norsk bedrift.

6.2 Konfigurasjonsstyring

Versjonshåndtering av kildekode kommer vi til å benytte Subversion som tilbys av IT-tjenesten ved HiG. Når det gjelder tekstdokumentasjon skal denne gjøres ved hjelp av LaTeX . Prosjektleder har hovedansvar for skriving og sammenfatning av dokumentasjonen. All dokumentasjon skal lagres på fellesområdet skolen tilbyr til dette formålet. I tillegg skal siste versjon av prosjektdokumentene ligge tilgjengelig på gruppens webside.

7 Plan for gjennomføring

Vi har valgt følge å Scrum i prosjektet vårt, men med noen prosjektspesifikke endringer som forklares og begrunnes i punkt 4.1. Ved å benytte denne modellen får vi med jevne mellomrom, hver 14. dag, et målbart resultat. Hver av disse delleveransene skal leveres/vises til oppdragsgiver for resultatvurdering. For å få mest mulig gjort i løpet av en Sprint kommer vi til å delegere oppgavene fra Sprint Backlog slik at hver av oss jobber med en mest mulig uavhengig modul. Vedlegg 1 viser et Gantdiagram med en foreløpig plan for gjennomføring av prosjektet, med oppgaver og tidsrammer for disse. Diagrammet viser et dobbeltløp, ved oppstart av prosjektperioden er det gitt rom for at endringer kan oppstå under utviklingen, mens ved slutten av prosjektperioden kommer vi til å begynne på rapporten før utviklingen av løsningen er avsluttet.

VEDLEGG G – CD-ROMMENS INNHOLD

- Bachelorrapport med alle vedlegg i PDF format.
- Visual Studio 2008 prosjekt med all kildekode.

HØGSKOLEN I GJØVIK

PROSJEKTAVTALE

mellom Høgskolen i Gjøvik (HiG) (utdanningsinstitusjon),

POPKORN AS (oppdragsgiver), og

PAUL MAGNE LUNDE

RINO WERNER FALSTAD

SIMEN TVEIT BACKSTRØM (student(er))

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

1. Studenten(e) skal gjennomføre prosjektet i perioden fra 12/1-09 til 25/5-09.

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

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

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

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

   Besvarelser med karakter C eller bedre registreres og plasseres i skolens bibliotek. Det legges også ut en elektronisk prosjektbesvarelse uten vedlegg på bibliotekets del av skolens Internett-sider. Dette avhenger av at studentene skriver under på en egen avtale hvor de gir biblioteket tillatelse til at deres hovedprosjekt blir gjort tilgjengelig i papir og nettutgave (jfr. Lov om opphavsrett). Oppdragsgiver og veileder godtar slik

offentliggjøring når de signerer denne prosjektavtalen, og må evt. gi skriftlig melding til studenter og dekan om de i løpet av prosjektet endrer syn på slik offentliggjøring.

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

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

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

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

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

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

10. Når HiG også opptrer som oppdragsgiver trer HiG inn i kontrakten både som utdanningsinstitusjon og som oppdragsgiver.

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

12. Deltakende personer ved prosjektgjennomføringen:

HiGs veileder (navn): TOM RØISE

Oppdragsgivers kontaktperson (navn): STIAN BAKKEN / KAI ARNE ASPELI

Student(er) (signatur): Paul M. Lunde dato 30/1-09

Kine Falstad dato 30/1-09

Simen Bachström dato 30/1-09

dato

Oppdragsgiver (signatur): [signature] dato 29/1-09

~~Dekan~~ (signatur): AVD. DR [signature] dato 3/2. 2009

Revidert 11.10.07, Ivar Moe

HØGSKOLEN I GJØVIK

AVTALE OM INNSYN OG ELEKTRONISK PUBLISERING AV BACHELOROPPGAVE / MASTEROPPGAVE ("BIBLIOTEKAVTALE")

mellom Høgskolen i Gjøvik ved Biblioteket, nedenfor kalt HiG, og studenten(e) som gjennomfører bacheloroppgaven / masteroppgaven, nedenfor kalt Forfatteren.

Forfatterens navn: Simen Tveit Bachstrøm
Rina Werner Falstad
Paul Magne Lunde

Studium: Bachelor i Programvareutvikling

Adresse: Slargata 17B, 2815 Gjøvik
Idunsveg 19, 2817 Gjøvik
Fauchalds Gate 20, 2815 Gjøvik

E-postadresse: siback@bachs.no
rina@falstad.net
paulmagne@puffinweb.net

Delvis overdragelse av opphavsrett

Forfatteren gir herved HiG en vederlagsfri rett til å gjøre Forfatterens bacheloroppgave / masteroppgave, arbeidet titulert som

Styresaks database (SSD)

tilgjengelig på Biblioteket både i trykt og i elektronisk form.

Forfatteren er klar over og aksepterer utstrekningen og betydningen av den aktuelle rettighetsoverdragelsen og den senere publisering via Internett, slik dette fremgår av de etterfølgende sidene 2 og 3. For at denne såkalte Bibliotekavtale skal være gyldig må – og dette gjelder bacheloroppgaver – avtalen mellom HiG, oppdragsgiver og student(er), kalt Prosjektavtale, være signert av partene og vedlagt Bibliotekavtalen.

HØGSKOLEN I GJØVIK

HiGs rettigheter og plikter

HiG har vedtatt at kun bacheloroppgave / masteroppgave med karakteren C eller bedre vil bli gjort tilgjengelig i trykt og elektronisk versjon. HiG har rett, men ikke plikt til å gjøre bacheloroppgaven /masteroppgaven tilgjengelig på Biblioteket og/eller dets nettsider. Dersom HiG benytter seg av denne retten skal det som publiseres være slik det ble levert til HiG. HiG er ikke ansvarlig for å korrekturlese eller kontrollere at den innleverte versjon samsvarer med eventuelle tidligere versjoner eller utkast.

HiG får ikke råderett over bacheloroppgaven / masteroppgaven utover det som uttrykkelig fremgår av denne avtalen.

Forfatterens rettigheter og plikter

Forfatteren skal følge de retningslinjer som til enhver tid gjelder for publisering ved HiG.

Forfatteren skal ved eventuell inngåelse av avtaler med andre om overdragelse av rett til å publisere bacheloroppgaven / masteroppgaven, alltid sørge for å ivareta og beskytte HiGs rettigheter etter denne avtalen.

Forfatteren garanterer at han/hun er opphav til bacheloroppgaven / masteroppgaven og har fullstendig råderett. Det samme gjelder materiale som er lagt ved eller på annen måte er koblet til bacheloroppgaven / masteroppgaven, for eksempel som vedlegg eller gjennom lenking eller annen teknisk framgangsmåte. Materiale som er innhentet fra andre kilder skal være referert i litteraturlisten, og Forfatteren har ikke opphavsrett til disse.

Forfatteren garanterer at han/hun ikke har kunnskap eller mistanke om at bacheloroppgaven / masteroppgaven inneholder materiell som kan anses å stride mot gjeldende norsk rett eller inneholder lenker eller andre koblinger til slikt materiale.

Dersom HiG skulle bli gjort erstatningsansvarlig overfor en tredjepart på grunn av at Forfatteren ikke oppfyller sine plikter og garantier etter denne avtalen, er Forfatteren forpliktet til å holde HiG fullt ut skadesløs.

HØGSKOLEN I GJØVIK

Papirutskrifter m.m.

HiG har rett til å publisere bacheloroppgaven / masteroppgaven på Internett på en slik måte at det er mulig å ta utskrift av dokumentet. HiG skal også ha rett til å ta enkeltstående papirutskrifter og andre kopier av bacheloroppgaven / masteroppgaven til internt bruk ved HiG.

Opphør av avtalen

HiG har en ubegrenset rett til å avbryte publiseringen av bacheloroppgaven / masteroppgaven.

Forfatteren har rett til skriftlig å si opp avtalen. HiG skal fjerne bacheloroppgaven / masteroppgaven fra sine sider på Internett senest 6 måneder etter mottakelse av oppsigelsen. HiG skal likevel være forpliktet til fjerne avhandlingen raskere, dersom Forfatteren oppgir særlige, saklige grunner for dette.

Denne avtalen er utstedt og undertegnet i to likelydende eksemplarer, hvorav partene beholder hvert sitt.

Jeg har lest og akseptert den overstående avtalen med HIG, her ved Biblioteket.

Gjøvik, 15.05.09

For HiG

HØGSKOLEN I GJØVIK
BIBLIOTEKET
Postboks 191
2802 Gjøvik

Forfatter

Simen Bendiksen
Forfatter

Paul M. Lunde
Forfatter

Kristian Fahstad
Forfatter

Revidert 11.10.07, Ivar Moe