HOVEDPROSJEKT:

TED-2001

Teknisk ErfaringsDatabase

FORFATTERE:
Ingar Brennmoen
Trond Hedalen
Øystein Hauklien
Erik Kvam

Dato: 23. mai 2001

Sammendrag av hovedprosjekt

Tittel:	TED2001	Nr.:
		Dato: 23.05.2001
Deltaker(e):	Ingar Brennmoen	
	Trond Hedalen	
	Øystein Hauklien	
	Erik Kvam	
Veileder(e):	Ivar Farup	
Oppdragsgiver:	Telenor Telecom Solutions TTS-DL-DP/IT	
Kontaktperson:	Roar Wilhelmsen	
Stikkord (4 stk)	Erfaringsdatabase, Web, Perl, Sybase	

Antall sider:	Antall bilag:	Tilgjengelighet (åpen/konfidensiell):åpen

Kort beskrivelse av hovedprosjektet:

Oppgaven har bestått i å kravspesifisere, designe og kode en erfaringsdatabase, og dens brukergrensesnitt. Utvikling har foregått etter RUP-modellen.

Denne databasen brukes internt i Telenor Nett, og inneholder erfaringer og varslinger for systemer og nett de drifter og forvalter. Brukergrensesnittet er web-basert.

Det skilles mellom to typer meldinger som skal lagres, varslinger og hendelser. En erfaring kan f.eks være en beskrivelse av et problem som har oppstått på en enhet, og hva som ble gjort for å løse problemet. Erfaringer gjort av en person blir derfor tilgjengelig for andre. En varsling kan f.eks være info om at en sentral vil bli tatt ned for vedlikehold i ett gitt tidsrom.

Systemet har rapporter og avanserte søkefunksjoner for raskt å finn fram til informasjon. Enkelte deler av databasen er tilgjengelig for alle brukere, men det kreves pålogging for å kunne registrere info i systemet. Det kan angis forskjellige brukernivå/rettigheter for hver enkelt bruker. Disse nivåene bestemmer både hva brukeren kan se, og registrere.

Systemet går i et UNIX-miljø. Scriptspråk for websider er HTML som genereres av Perl/CGI. Data ligger lagret i sybase database.

Forord

Denne rapporten beskriver hovedprosjektet vårt, som utgjør avslutningen på et tre-årig ingeniørstudium innen datateknikk. Rapporten er ment å gi en beskrivelse av oppgaven, sluttresultatet, og hvilke arbeidsmåter vi har hatt underveis.

Kapittel 3 og 4, om design og implementasjon, er i tillegg skrevet med tanke på bruk ved eventuell videreutvikling og oppgradering av programmet.

Vi vil benytte anledningen til å rette en stor takk til vår oppdragsgiver, Roar Wilhelmsen hos Telenor, som ga oss muligheten til å jobbe med dette prosjektet, og som la ned mye innsats i bl.a. å hjelpe oss å finne krav til systemet.

Vil også takke veilederen vår Ivar Farup, for god hjelp gjennom hele prosjektperioden. Har fått god hjelp både med de formelle delene, og hjelp til å løse tekniske problemer.

Vi vil ellers takke Kjell Rune Narvesen, Terje Myrvang, Atle Noem og Marit Øvringmo hos oppdragsgiver, for god hjelp underveis med å finne krav, teste og gi tilbakemeldinger på systemet.

Vil tilslutt sende en takk til Øyvind Kolloen ved HIG, som har vært til god hjelp med en del tekniske problemer.

Gjøvik, mai 2001

Innholdsfortegnelse

Detaljert innholdsfortegnelse

1 Innledning

1.1 Oppgavebeskrivelse.

Oppgaven har bestått i å kravspesifisere, designe og kode en erfaringsdatabase, og dens brukergrensesnitt, heretter kalt Erba.

Tidligere versjon av Erba har gått på en access database, men dette har vært tungvindt å bruke, har hatt ett begrenset bruksområde og har ikke fungert hensiktsmessig. Dette var ikke noe å bygge videre på.

Erba brukes internt i Telenor Nett, og inneholder erfaringer og varslinger for systemer og nett de drifter og forvalter. Det var krav om at brukergrensesnittet skulle være web-basert, for å gi størst mulig tilgjengelighet.

Det skilles mellom to typer meldinger, varslinger og hendelser. Erba har rapporter og avanserte søkefunksjoner for raskt å finn fram til informasjon. Enkelte deler av databasen er tilgjengelig for alle brukere, men det kreves pålogging for å kunne registrere info i systemet. Det kan angis forskjellige brukernivå/rettigheter for hver enkelt bruker. Disse nivåene bestemmer både hva brukeren har tilgang til å se og registrere. Erba ligger i et sikret nett. Det er derfor ikke nødvendig å legge vekt på sikkerhet for å hindre uvedkommende tilgang til databasen, men det skal sikres mot utilsiktet bruk. Erba sender mail til systemansvarlige for det systemet det blir registrert en melding på. Det kan i tillegg sendes mail til enkeltpersoner og forhåndsdefinerte maillister.

Administreringen av Erba foregår også via web av brukere som har fått tildelt høyere rettighetsnivå. Dette gjør at Erba enkelt kan tilpasses nye systemer og arbeidssituasjoner.

Oppdragsgiver ønsket at det skulle legges stor vekt på å få brukergrensesnittet så brukervennlig som mulig, og at designet skulle lages i samarbeid med brukerne av systemet, for å sikre at det får den funksjonaliteten som trengs. Dvs. at det skulle være enkelt å bruke for nye brukere, men samtidig funksjonelt for de som bruker Erba hver dag. Dette var viktig for å få flest mulig til å bruke systemet.

Systemet går i et UNIX-miljø.
Oppsett av webserver og databaseserver gjøres av oppdragsgiver.

1.2 Faglig bakgrunn

Da vi startet med prosjektet hadde vi lite erfaring i å behandle data via websider.
Vi hadde grunnleggende kunnskaper i html og javascript, fra faget ”Programmering mot www” og noe kunnskaper fra faget ”Databaser1”.
Henviser ellers til kapittel 7, for drøfting av faglige utfordringer.

1.3 Arbeidsformer

Oppdragsgiver ønsket at vi skulle utarbeide grensesnittet på Erba forholdsvis tidlig, slik at dette kunne presenteres for brukerne på et stadium hvor det ennå var enkelt å gjøre endringer. Vi fikk skissert hvordan de hadde sett for seg at Erba kunne bli, og fikk en oversikt over hvilke krav de hadde. Oppdragsgiver ville videre at det skulle legges stor vekt på å lage brukergrensesnittet så brukervennlig som mulig, da dette var viktig for å få flest mulig til å bruke Erba. Dvs. at det skulle være enkelt å bruke for nye brukere, men samtidig funksjonelt for de som bruker systemet hver dag.

Da vi skulle velge systemutviklingsmodell, måtte vi ta hensyn til dette. I tillegg visste vi veldig lite om den programmerings-teknikken vi skulle bruke. Det var derfor vanskelig å vite hva som ville være enkelt å lage, og hvor vi ville få utfordringer. Vi valgte på grunnlag av dette å jobbe etter RUP-modellen. Den ga oss mulighet til å jobbe parallelt med kravspesifisering/designdokument og utvikling av grensesnittet. Dette ga oss også mulighet til å dele opp problemstillingen og konsentrere oss om deler av systemet.

Vi valgte å lage fire betaversjoner underveis som ble presentert for oppdragsgiver. Vi la også ut forslaget til brukergrensesnitt ut på web, slik at oppdragsgiver kunne følge med på utviklingen. Dette ga samtidig fremtidige brukere mulighet til å teste systemet, og komme med tilbakemeldinger.

1.4 Rapportens oppbygging

Vi valgte å dele inn rapporten på følgende måte:

- Kapittel 2 inneholder krav til systemet.
- Kapittel 3 viser hvordan design på brukergrensesnitt og database ble, med begrunnelser.
- Kapittel 4 beskriver hvordan programmet fungerer, dvs koding og oppbygging av scriptene. Dette kapitlet er også ment å kunne gi en så detaljert beskrivelse av systemet, i tillegg til designdokumentet og kommentarene i selve scriptene, at det skal være til hjelp ved senere oppgraderinger/videreutvikling av programmet. Dette er derfor skrevet i et noe teknisk avansert språk.
- Kapittel 5 omhandler testing og feilretting av programmet.
- Kapittel 6 beskriver nærmere hvilke arbeidsformer vi valgte, og hvordan vi opplevde disse. Vi har også med en egen vurdering av hvordan sluttresultatet ble.

2 Kravspesifikasjon

2.1 Innledning

2.1.1 Bakgrunn

Erba en database som skal inneholde hendelser/erfaringer man har i forbindelse med systemer og nett man drifter og forvalter i fagenhet Datatjenester, og muliggjøre søking i disse.

2.1.2 Kort om krav til systemet

Basen skal bygges opp slik at man kan oppdatere og registrere hendelser ved hjelp av et enkelt webgrensesnitt. Det skal også være mulig å gjøre fritekstsøk i hele basen. Det skal være brukerautentisering på flere nivå, slik at man kan skille lese, skrive og redaktøritilganger.

2.1.3 Kort om systemets omgivelser

Basen skal ligge i Telenors interne nett og skal ha grensesnitt mot intranett.
Systemet utvikles vha av Perl-script og det skal ligge på en Sybase database.

2.1.4 Systemets brukere

Systemet skal brukes av mange forskjellige brukere. Det bør være enkelt å bruke, og lages slik at så mange som mulig ser nytten av å bruke det. For å legge til info i databasen, må brukerne logge seg på med ansattnr og passord. Redaktør kan tildele brukerne rettigheter. For brukere i høyere nivåer vil funksjonalitet og effektivitet være viktigst

2.2 Funksjonell spesifikasjon

2.2.1 Funksjonell struktur

Systemet skal inneholde varslinger og hendelser/erfaringer man har i forbindelse med systemer og nett.

Varslinger er meldinger som omhandler hendelser som skjer med systemer og nett. Disse er bare midlertidige og skal varsle brukere om at noe har skjedd. Mens hendelser er meldinger om ting som har skjedd med systemene eller nettene som avdelingen vil ta vare på som erfaring til seinere bruk. Eksempel på en varsling kan være at en router vil være ute av drift 22/6-01 fra klokken 1400 –1600. Et eksempel på hendelse kan være at et av systemene har gått ned og en beskrivelse av hvorfor dette hendte og hva som ble gjort for å rette opp feilen. Hendelser skal kunne registreres av alle i det interne nettet. Disse skal videre administreres av brukere med utvidete rettigheter.

Det skal også kunne legges ut varsling tilknyttet hendelsen, eller bare enkeltstående varslinger. Det skal skilles mellom hendelser som har faq-status og ikke. I rapporter, søk og oversikter skal en bruker kun se de meldingene som han har rettigheter til å lese.

Redaktør skal kunne se alle versjoner av en melding, dvs. varslinger som er avsluttet, og originalversjonene av meldinger som er oppdatert.

2.2.2 Data spesifikasjon

2.2.2.1 Datamodell

Data som skal lagres om hendelser og varslinger:

Feltnavn	Lengde	Beskrivelse
Ansattnr	6	Angir hvem som registrerte en melding. Dette bør komme opp automatisk, da man må være innlogget for å registrere en melding.
Fornavn	25	
Etternavn	25	
E-mail	50	
Passord		Kryptert passord
Referansenummer	Integer	En melding tildeles ett referansenummer.
Tittel	50	Meldingens tittel.
System		Skal være en dropdown-liste, når meldinger registreres. Redaktør skal kunne legge til og endre systemer.
Beskrivelse	Ubegrenset	
Status/løsning	Ubegrenset	
Aksjonspunkt	Ubegrenset	
Vedlegg		Brukere skal kunne laste opp vedlegg til meldinger. Dette skal begrenses til html- og doc-filer.
Kontaktperson	50	Fritekst, da dette kan være personer som ikke er registrert i

		brukertabellen
Status	25	Status for en melding. Dropdownboks, som skal kunne endres av redaktør
Dato		Registreringsdato

Felter som kun er tilgjengelig på varsling:

Feltnavn	Lengde	Beskrivelse
Varighet		Skal angi hvor lenge varlingen skal stå. Dropdownboks med valgene 1,2 eller 3 døgn, eller ubestemt
Feilkategori	25	Dropdownboks, som skal kunne endres av redaktør

Felter som kun er tilgjengelig på hendelser:

Feltnavn	Lengde	Beskrivelse
Meldingskategori	25	Dropdownboks, som skal kunne endres av redaktør
Leverandør	25	Dropdownboks, som skal kunne endres av redaktør og enkelte brukere
FAQ	Ja/nei	Presenteres som sjekkboks. Angir om hendelsen er tilgjengelig for vanlige brukere.

Hjelpefelter/tabeller

Feltnavn	Lengde	Beskrivelse
Passord	15	Påloggingspassord for brukere
Systemlink	128	Link til side som har beskrivelse av systemet en melding gjelder. Denne skal kunne endres av redaktør.
Levlink	128	Link til side om leverandør
Feilbeskrivelse	Ubegrenset	Beskrivelse av feilkategorier
Kategoribeskrivelse	Ubegrenset	Beskrivelse av meldingskategoriene
Synlig	Ja/nei	Angir om en melding er slettet. Når en bruker sletter en melding, skal den fremdeles være synlig for redaktør
Mail	Ja/nei	Angir om den som registrerte en melding skal ha mail når hendelsen oppdateres
Systemansvarlige		Angir systemansvarlige på hvert enkelt system. Ett system kan ha flere systemansvarlige
Maillister		Det skal angis lister for hvert enkelt system, over brukere som skal motta mail, hvis den som registrer en melding vil det

Hendelser og varslinger som omhandler samme sak, skal ha samme referansenummer. Referansenummeret skal også beholdes når meldingene oppdateres. Det vil da være lettere å finne fram til aktuell melding for brukerne.

Redaktør skal kunne se alle versjoner av en melding, og skal kunne gjøre gamle versjoner synlige. Vanlige brukere vil da se flere meldinger med samme referansenummer, men sortert på tid.

2.2.2.2 Data input til systemet

Data input foregår via web. Det er forskjell på hva enkelt brukerkategori kan registrere

2.2.2.2.1 Registrer Varsling

Feltene registrert av og dato fylles ut automatisk. Det skal ellers registreres tittel, system, beskrivelse, varighet og kontaktperson. System skal angis med dropdownboks. Kontaktperson skal angis som fritekst.

I tillegg til å registrere varslingen, skal det være mulighet for å sende mail til:

- feilmottak i nett
- valgfri mailadresse
- mailliste for aktuelt system

Det skal også være mulig å angi at man vil ha mail hver gang det skjer en endring på meldingen. Når varslingen er registrert skal det automatisk sendes en mail til systemansvarlig for systemet varslingen gjelder.

2.2.2.2.2 Endre varsling

Skal i tillegg til feltene i 2.2.2.2.1 ha statusbeskrivelse/løsning, og dropdownbokser for feilkategori og status.

2.2.2.2.3 Registrer hendelse

Feltene registrert av og dato fylles ut automatisk. Det skal ellers registreres tittel, system, leverandør, meldingskategori, beskrivelse, statusbeskrivelse/løsning, kontaktperson, eksternt refnr og faq-status, og mulighet for å laste opp vedlegg. System, leverandør og meldingskategori skal være dropdownbokser. Eksternt refnr er et tekstfelt. Det skal være mulighet for å sende mail til ansvarlig på angitt system.

Dersom bruker har rettighet til å registrere varsling, skal han kunne angi om hendelsen også skal legges ut som varsling. Brukere med rettighet til å endre hendelser, skal ha tilgang til feltene meldingskategori og status.

2.2.2.2.4 Endre hendelser

Samme felter som i 2.2.2.2.3. Det skal i tillegg være mulighet for å registrere aksjonspunkt og status, samt ha mulighet for å legge ut hendelsen som varsling.

2.2.2.2.5 Personalia

Brukerne skal ved første gangs pålogging oppgi ansattnr, personalia og passord. Skal senere kunne endre personalia og passord.

2.2.2.2.6 Administrering

Redaktør skal kunne endre tekst på forside, forandre på felter som vises som dropdown-bokser, dvs. status, feilkategori, meldingskategori, leverandører og systemer.

Det skal registreres en beskrivelse på hver feilkategori. Systemer og leverandører skal registreres med en link som viser mer informasjon. Disse sidene eksisterer allerede, slik at bare linken skal lagres.

Redaktør skal også kunne vedlikeholde brukertabellen, angi systemansvarlige og redigere mail-listene til hvert enkelt system. En person kan være systemansvarlig på flere systemer, og det kan være flere ansvarlige på et system.

Brukere med rettighet til det, skal kunne endre maillistene til systemene, og info om leverandører.

2.2.2.2.7 Endre passord

Brukerne skal kunne endre passord.

2.2.2.3 Data output

Alle steder hvor det vises oversikt over meldinger og detaljer, skal det være mulig å klikke på aktuelle felter for å få opp detaljerte beskrivelser av disse, der dette er registrert. Dette gjelder:

- Registrert av – viser personalia, og link til intern telefonkatalog.
- Systemer – link til side som har beskrivelse av systemet. Dette er en side utenfor Erba.
- Leverandør – link til side som har informasjon om leverandøren. Ligger utenfor Erba.
- Feilkategori – viser beskrivelse av feilkategorien.
- Meldingskategori – viser beskrivelse av kategorien.

En bruker får kun se det han har rettigheter til.

2.2.2.3.1 Framside

Teksten på denne siden skal kunne endres fra redaktørsiden. Denne skal også vise en oversikt over alle varslinger. Disse skal default sorteres på dato. Det skal være mulig å klikke på

overskriftene i tabellen for å sortere etter fra ønsket kriterium. De feltene som skal vises er: Referansenummer, tittel, status, dato, registrert av, feilkategori og system.

Alle varslinger skal vises, da det anslagsvis vil være ca 10 varslinger inne i gjennomsnitt. Dersom man klikker på en varsling, skal man få fram detaljer om varslingen (2.2.2.3.2)

2.2.2.3.2 Detaljer om varsling

Viser alt som er registrert om varslingen, som pålogget bruker har rettighet til å lese.
De som har adgang til å endre varsling, skal kunne klikke på en link for å endre/oppdatere varslingen (2.2.2.2.2) .

2.2.2.3.3 Oversikt over hendelser

En tabell som viser de sist registrerte hendelsene. Hendelsene skal default sorteres på dato. Det skal være mulig å klikke på overskriftene i tabellen, for å sorteres etter fra ønsket kriterium. De feltene som skal vises er: Referansenummer, tittel, status, dato og system.

2.2.2.3.4 Detaljer om hendelser

Viser alt som er registrert om hendelsen, som pålogget bruker har rettighet til å lese.
De som har adgang til å endre hendelse, skal kunne klikke på en link for å endre/oppdatere hendelsen (2.2.2.2.4).

2.2.2.3.5 Intern mail

Skal inneholde en standard-tekst, link til framside, referansenummer og tittel på meldingen.

2.2.2.3.6 Redaktør

Skal se alle versjoner av meldinger, dvs også gamle versjoner av meldinger som er oppdatert. Redaktør skal kunne gjøre originalversjoner av oppdaterte meldinger synlig igjen, slik at brukere får se alle versjoner.

2.2.2.3.7 Søk

Skal ha en enkel søkemulighet, som hele tiden skal vises. Dette skal være fritekstsøk. Det skal også være en mer avansert søkefunksjon. Det søkes kun i materiale pålogget bruker har rettigheter til.

2.2.2.3.8 Rapporter

Man skal kunne hente ulike rapporter fra databasen. Disse skal kunne hentes med utplukk på system, leverandør og status.

- System rapport skal komme som en tabell og skal inneholde feltene Refnr, Tittel, Status og Dato. Systemets navn skal stå i overskrift. Refnr skal inneholde link til den aktuelle hendelse hvor man får en detaljert utskrift. Dersom bruker har privilegier skal det også komme opp en link på detaljutlegget hvor man kan trykke for å editere på den aktuelle hendelse. De ulike overskriftene på feltene i tabellen skal være linker slik at man sorterer på aktuelt felt dersom man trykker på et av disse. Default sortering er dato.
- Leverandør rapport skal vise alle hendelser hvor aktuell leverandør er involvert. Denne rapporten skal ha feltene Refnr, Tittel, Eksternt refnr, Status og Dato. Leverandørens navn skal stå i overskrift. Leverandørens navn har link til side hvor man har lagt leverandørinfo. (Linken administreres via redaktørsiden). De ulike overskriftene på feltene i tabellen skal være linker slik at man sorterer på aktuelt felt dersom man trykker på et av disse. Default sortering er dato.
- Status rapport skal vise alle systemer med valgt status. Feltene som skal være med er Refnr, Tittel, System og dato. De ulike overskriftene på feltene i tabellen skal være linker slik at man sorterer på aktuelt felt dersom man trykker på et av disse. Default sortering er dato.
- Siden skal ha link til Søkesiden

2.2.2.3.9 Kvitteringer

Brukere skal få kvitteringer etter å ha registrert eller endret meldinger. Her skal det stå hvilket referansenummer meldingen fikk.

2.2.2.3.10 Systemansvarlige

Systemansvarlige skal ha en egen side hvor de får opp oversikt over de siste meldingene på de systemene de har ansvar for.

2.2.2.3.11 Glemt passord

Skal gi mulighet for å få oversendt passordet til mailadressen som er registrert på aktuelt ansattnr.

2.2.3 Overordnede operasjonelle systemkrav

2.2.3.1 Normal operasjon

2.2.3.1.1 Modus og kontroll

Brukerne skal inndeles i følgende kategorier:

Brukerkategori	Beskrivelse
Kat 0	dette er brukere som ikke har logget inn kan se på hendelser og varlinger med faq-status kan søke i disse
Kat 10	må logge inn med ansattnr og passord kan registrere ny hendelse kan se på hendelser og varlinger med faq-status kan søke i disse
Kat 20	Ikke definert ennå
Kat 30	kan registrere/endre varsling kan registrere hendelse kan se på hendelser og varslinger med faq – status kan søke i disse
Kat 40	kan registrere/endre varsling kan registrere/endre hendelse kan se på alle hendelser og varslinger kan søke i alle hendelser og varslinger kan endre leverandører og maillister
Kat 50	kan registrere/endre varsling kan registrere/endre varsling kan se på alle hendelser og varslinger kan søke i alle hendelser og varslinger kan gi brukere utvidet tilganger kan redigere innhold i drop-down bokser har tilganger til hendelser som er ”slettet”, altså er usynlige for andre

Bruker som ikke er logget inn har rettighet 0.

Pålogging skal skje vha ansattnr og passord. Første gang en bruker logger inn, må vedkommende oppgi fornavn, etternavn, mailadresse og ønsket passord. Det gis da automatisk rettighet 10.

Redaktør(rettighet 50) skal kunne endre rettigheter til brukerne.

2.2.3.1.2 Ytelse

Det vil maks bli 10000 registreringer pr år. Ytelse vil ikke bli noe problem på dette systemet. Det er også nok lagringsplass tilgjengelig.

2.2.3.1.3 Sikkerhet

Databasen blir liggende i et sikret nett. Det er derfor ikke nødvendig å legge inn noen spesielle sikkerhetsfunksjoner for å hindre uvedkommende å få adgang til basen. Man må imidlertid sikre seg mot utilsiktet bruk av basen.

2.2.3.1.4 Oppstart

Systemet skal igangsettes hos oppdragsgiver 10. mai. Prosjektgruppa skal videre være tilgjengelig for hjelp til igangsetting.

2.2.3.2 Operasjon i feilsituasjoner

Ved feilsituasjoner skal det først skrives ut feilmelding på norsk, for å gi vanlige brukere en ”forståelig” feilmelding. Deretter skrives ev. feilmeldingen fra databasen ut, for å gi systemansvarlig et hint om hva som er feil.

Alle feilmeldinger skal skrive ut link til mailadresse til ansvarlig, og en oppfordring om å varsle denne, og angi hvilket feilnummer feilmeldingen har.

Det skal lages en oversikt over hvilke feilmeldinger som kan oppstå. Hver enkelt feilmelding gis et referansenummer, slik at systemansvarlig raskt kan finne fram i denne oversikten over feilmeldinger, og se hvor feilen oppstod.

2.2.4 Funksjonelle krav

2.2.4.1.1 Varsling/hendelse

En hendelse kan kopieres til varsling, det skal samtidig være mulighet for å slette hendelsen (gjøre den usynlig).

Når en hendelse kopieres til varsling, vil status bli stående. Når en varsling blir konvertert til hendelse, får den status 1 (som vil være ny), og feltet kategori settes til 1 (som vil være ”Avsluttet varsling”). Hendelsen skal ha samme refnr som varslingen hadde. Hendelsen skal da ikke ha FAQ status. Når en ny hendelse registreres, skal det være mulig å samtidig legge den ut som varsling.

Det skal være mulig med tids-stempling på varslinger. Når en varsling ”går ut på tid” skal den gjøres om til hendelse.

2.2.4.1.2 Status

En ny varsling eller hendelse vil automatisk få status 1. Statusene kan endres i redaktørsiden.

2.2.4.1.3 Hendelseslogg

Når en hendelse oppdateres, beholdes refnr, mens den får et nytt løpenr. Den gamle hendelsen får da ikke FAQ-status. Feltet Lopenr vil ikke vises i grensesnittet. Hvis redaktør mener at de endringene som er gjort underveis skal publiseres, kan han gi meldingen FAQ-status. Vanlige brukere vil da få se flere meldinger med samme referansenummer, men som er sortert på dato.

2.2.4.1.4 Meny

Denne skal endres ut fra hvilke rettigheter bruker har. Hvis man ikke har logget inn, skal det vises ett felt for pålogging på menyen.

2.2.4.1.5 Backup

Det skal lages backupscript av databasen som dumper transaksjonslogg hver annen time og databasen skal dumpes en gang pr. døgn.

2.3 Begrensninger

2.3.1 Software design begrensninger

2.3.1.1 Software standarder og språk

Databasen skal lagres i Sybase v.11.9.x.x
Scriptspråk er Perl 5.x
Scriptene baseres på følgende tillegsmoduler for Perl: DBI og DBD::Sybase

2.3.1.2 Software kommunikasjonsstandarder og grensesnitt

Grensesnittet skal være web-basert, og skal gå i et internt nett hos oppdragsgiver.
Sidene designes utfra en skjermoppløsning på 1024*768

2.3.1.3 Toleranser og marginer

Dette systemet vil være så lite, at det verken blir plassproblemer eller ytelsesproblemer.
Det vil bli lagret maksimalt 10000 poster pr. år.

2.4 Installasjon/dokumentasjon

2.4.1 Dokumentasjon

Systemet skal leveres med en online hjelpefunksjon. I tillegg skal det levers en detaljert beskrivelse av hvordan systemet og hvert enkelt script fungerer, slik at det skal være så enkelt som mulig å gå inn å gjøre forandringer.

2.4.2 Krav til utvidelsesmuligheter

Scriptene skal skrives, og dokumenteres på en slik måte at det skal være enklest mulig å oppgradere systemet i ettertid.

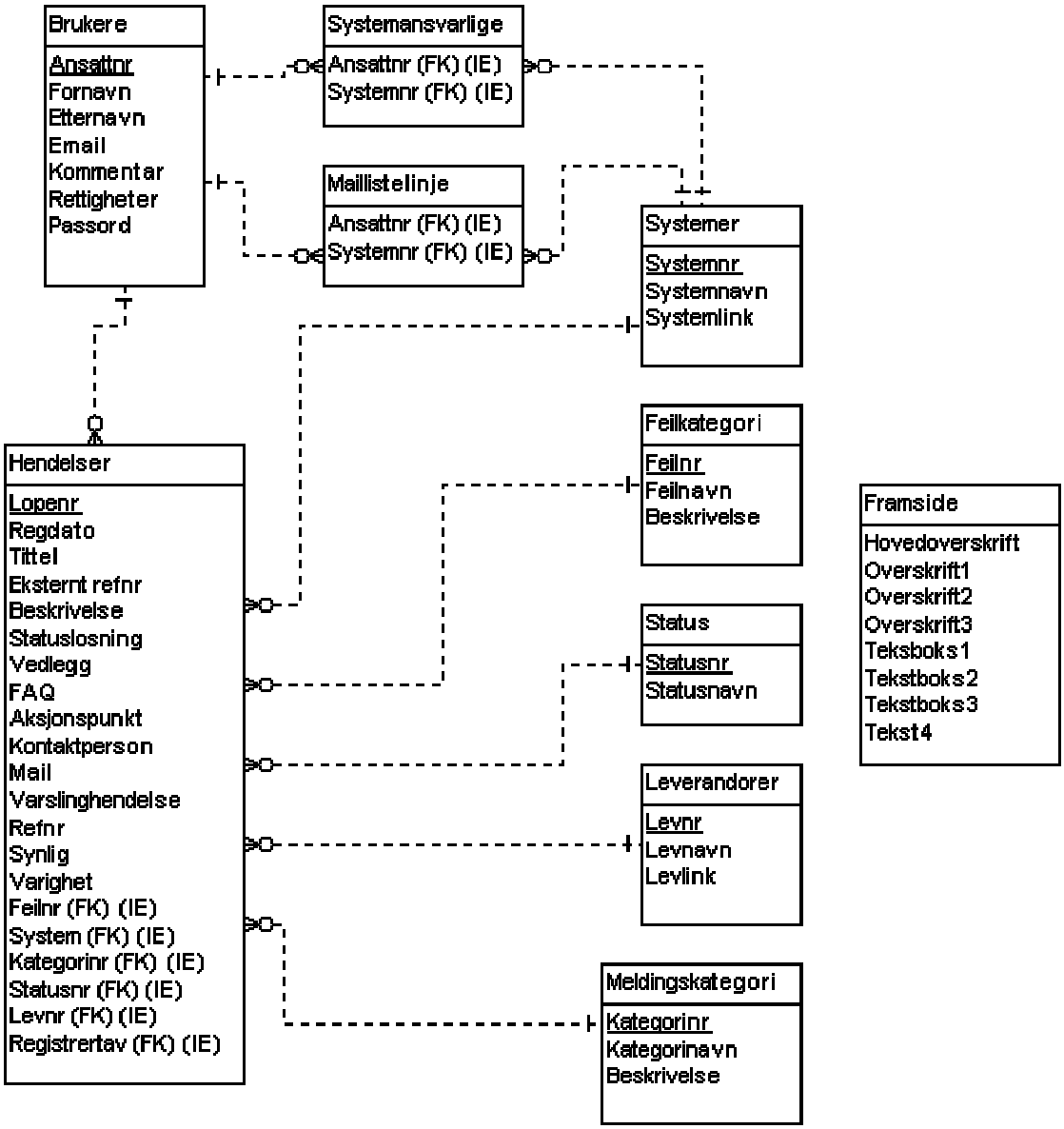
2.4.3 Installasjon

Ferdig produkt leveres på CD-ROM. Denne skal inneholde perl-script, dump av database og dokumentasjon. Systemet skal igangsettes hos oppdragsgiver 10 mai, og prosjektgruppen skal være tilgjengelig for support fram til prosjektets slutt 23. mai

3 Designdokument

3.1 Database

3.1.1 Databasemodell



3.1.2 Tabellforklaringer

Vi legger alle meldinger, dvs. både varslinger og hendelser i samme tabell, og bruker feltet Varslinghendelse for skille mellom de. Kommer tilbake til dette under avsnitt 3.1.3. Tabellene Feilkategori, Status, Leverandorer, Meldingskategori og Systemer brukes for å gi innhold til de respektive dropdownboksene. Maillistelinje holder rede på mail-listene til hvert enkelt system. Systemansvarlige holder rede på systemansvarlige på hvert enkelt system. Hadde først lagt opp systemansvarlig som et attributt under tabellen System, men fikk deretter et ønske om å kunne angi flere systemansvarlige på hvert system. Måtte derfor legge denne infoen ut i en egen koblingstabell.

I tabellen Framside blir det lagret kun en post, og det er innholdet på framsiden.

3.1.3 Feltforklaringer

Lopenr (løpenummer)

Det er en fordel at en melding har samme referansenummer etter at den har blitt oppdatert. Dessuten ønsket oppdragsgiver at den gamle versjonen av meldingen fremdeles skal ligge i databasen, slik at redaktør kan se alt som har foregått. Må da ha ett et eget felt som nøkkel. Dette er et ”autoincrement”-felt i databasen, dvs. att nye poster automatisk får et nytt lopenr. For å unngå forvirring, vises ikke lopenr i grensesnittet.

FAQ

FAQ-feltet angir om en melding har FAQ-status eller ikke. 1=FAQ-status, 0=ikke FAQ-status

Varslinghendelse

Dette angir om en melding er en varsling eller hendelse. 1=hendelse, 0=varsling

Synlig

Dette brukes for å angi om en melding er sletta eller ikke. Når vanlige brukere sletter en melding, virker det som at den forsvinner, men redaktør kan fremdeles se den. 1=synlig, 0=sletta,

Varighet

Dette angir hvilken dato en varsling skal gjøres om til hendelse.

3.1.4 Funksjonsmåte

Vi vurderte å lage en tabell for varslinger, og en for hendelser, men da det er flere felter som er like, enn felter som er ulike, valgte vi å legge varslinger og hendelser i samme tabell. Felter som kun er tilgjengelig på varsling:

- Tids-stempel
- Feilkategori

Felter som kun er tilgjengelig for hendelse:

- Meldingskategori
- Leverandør
- FAQ

Som tidligere nevnt, brukes feltet Varslinghendelse til å angi om posten er en varsling eller hendelse. Når vi registrerer en ny hendelse, gis det ingen verdi til feltet Tids-stempel. Feilkategori er en fremmednøkkel, og må ha en verdi. Men i og med at dette feltet aldri brukes, setter vi bare inn verdi 1. Tilsvarende for varsling, og de feltene som ikke skal vises der.

Når man registrerer en hendelse, har man også mulighet for å legge ut en varsling med samme innhold. Vi velger da å kopiere hendelsen til en ny post i databasen, og gjøre om feltet som viser om den nye posten er varsling eller hendelse.

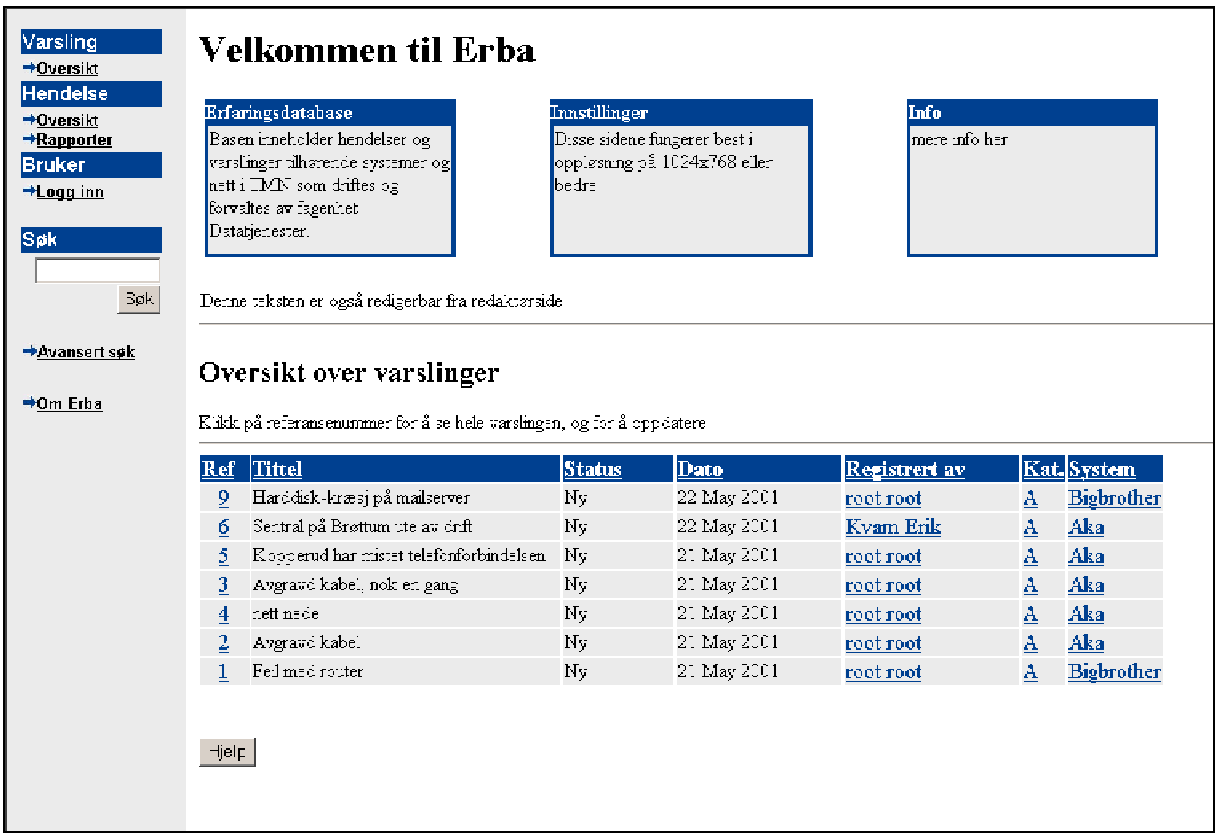
Når en hendelse oppdateres, beholdes refnr, mens den blir tildelt en nytt løpenr. Den gamle hendelsen får da FAQ-status 0. Hvis de endringene som er gjort underveis skal publiseres, kan FAQ-status settes lik 1 igjen. Vanlige brukere vil da få se flere meldinger med samme refnr, men som er sortert på dato.

Etter ønske fra oppdragsgiver, gis det ikke mulighet for å slette i databasen. Når feltet ”Synlig” i tabellen Hendelser settes lik null, vil en melding være utilgjengelig, bortsett fra brukere med høyeste brukerkategori(redaktør). I hjelpetabellene er det lov å endre navn på feltene, men ikke slette.

Oppdragsgiver foreslo å ha en egen tabell for hendelseslogg. Denne skulle inneholde originalversjonene av meldinger som var oppdatert. Men i og med at så godt som alle felter måtte være med i denne tabellen, valgte vi å ordne dette vha spørringer, i stedet for å kopiere feltene over i en ny tabell.

Det ble vurdert å legge telefonnummer til brukere i databasen, men da det finnes en intern telefonkatalog på oppdragsgivers nett, legges det i stedet link til denne, på de sidene det vises info om brukere. Man slipper da arbeid med å vedlikeholde disse numrene.

3.2 Brukergrensesnitt



Figur 3-1 Denne viser framsiden i Erba, og menyvalgene man har når man ikke er logget inn.

3.2.1 Standarder

3.2.1.1 Generelt

Brukergrensesnittet er laget på grunnlag av disse ønsker fra oppdragsgiver:

- skal være enkelt å sette seg inn i for nye brukere.
- skal være effektivt for avanserte brukere.
- hvit bakgrunn.
- bruke Telenors farger.
- skal være enklest mulig å endre i ettertid.

3.2.1.2 Skjermoppløsning

Sidene er designet med tanke på bruk i oppløsning på 1024x768. Det skal i denne oppløsningen ikke være nødvendig å scrolle i sideretningen.

3.2.1.3 Referansenummer

Referansenummer brukes for å identifisere en melding, og vises derfor overalt i systemet.

3.2.1.4 Tabeller

Alle tabeller sorteres default på dato, med de nyeste meldingene øverst. Man kan klikke på overskriftene i tabellen for å sortere etter ønsket felt.

I oversiktene vises det 20 linjer i hver tabell. For å bla fram og tilbake, brukes pilene under hver tabell. I tabellene på sidene rapporter, mine systemer og søk vises 50 linjer om gangen.

3.2.1.5 Beskrivelse av linker.

Det er noen linker som er like gjennom hele systemet:

- Navn på bruker som har registrert meldingen: Linken åpner siden som viser oss info om bruker som har registrert eller endret varslingen eller hendelsen. Linken åpner en side som viser fornavn, etternavn, mailadresse og en link til telefonkatalogen internt i Telenor.
- System: Linken åpner en web side til det systemet som varslingen gjelder.
- Feilkategori : Linken åpner en side som viser informasjon om feilkategori.
- Meldingskategori: Linken åpner en side hvor man ser en tabell over meldingskategorier og mer informasjon om hver enkelt av disse.
- Leverandør: Linken åpner et nytt vindu med en side som redaktør bestemmer. Gjerne web-siden til leverandøren.

3.2.1.6 Registrering og endring av hendelser og varslinger

Disse sidene inneholder en standard topptekst, der informasjon som ikke skal tastes inn av bruker vises. Dette gjelder dato og brukernavn på mye meldinger. Ved endring av meldinger, vises også hvem den originale meldingen er registrert av, og hvilket referansenummer meldingen har.

Store tekstområder:

Sidene registrer ny - og endre varsling/hendelse består noen større tekstområder. Ved varsling gjelder det: beskrivelse og statusbeskrivelse og løsning, mens for hendelser gjelder det feltene: beskrivelse, statusbeskrivelse/løsning og aksjonspunkt. Vi har valgt disse størrelsene ut i fra hva vi har fått vite av oppdragsgiver om hva som er gjennomsnittet av det som vil bli skrevet her. Vi har fått høre att det kan variere mellom noen linjer og opp til en A4 side.

Plassering av felter:

Plassering av felter er gjort mest mulig med tanke på hvordan brukerne skal fylle ut disse, og for å unngå scrolling. Felter som man ikke alltid trenger å fylle ut, vil stå nederst på siden, mens felter som skal være utfylt hver gang står øverst. Data fra hjelpetabellene, System, Feilkategori, Status, Meldingskategori og Leverandører velges vha dropdownbokser. Alle dropdownbokser er i utgangspunktet blanke, slik at bruker bevist må velge et element fra dropdownboksen.

3.2.1.7 Kvitteringer

Brukere skal motta kvitteringer når man har registrert eller endret meldinger. Dette skal være et popup vindu. Vinduet er mindre enn web-siden og åpnes midt på skjermen. Her står det at man har fått lagt meldingen i databasen og viser referansenummeret meldingen fikk. Valgte å bruke popupvindu da vi mener dette gir bedre oversikt, og man slipper å innom så mange forskjellige sider i hoved-framen.

3.2.1.8 Rettigheter

Alle sider og tabeller viser kun de meldingene og funksjonene pålogget bruker har rettighet til.

3.2.1.9 Meny

Menyen er en frame som hele tiden vises på venstre side på skjermen. Vi valgte denne løsningen for at det skal være enklest mulig å bevege seg raskt innenfor systemet. Brukeren har hele tiden en oversikt over hvilke mulighet han har.

Denne framen endres avhengig av hvilket rettighetsnivå aktuell bruker har. Ellers er den statisk.

Vi har valgt å samle linkene som har med hverandre å gjøre, slik at det som har med varsling å gjøre ligger under overskriften varsling, alt som har med hendelser å gjøre ligger under overskriften hendelser osv. Pilene foran linkene ble valgt for å angi at dette er valgmuligheter. Det skilles dermed lettere mellom overskrifter og valgmuligheter.

Viser til Figur 3-1 for bilde av menyen, som den vises når man ikke har logget inn. Figur 3-2 viser menyen for brukerkategori 40, og Figur 3-5 viser menyen for kategori 50.

3.2.2 Beskrivelse av sider

3.2.2.1 Startside

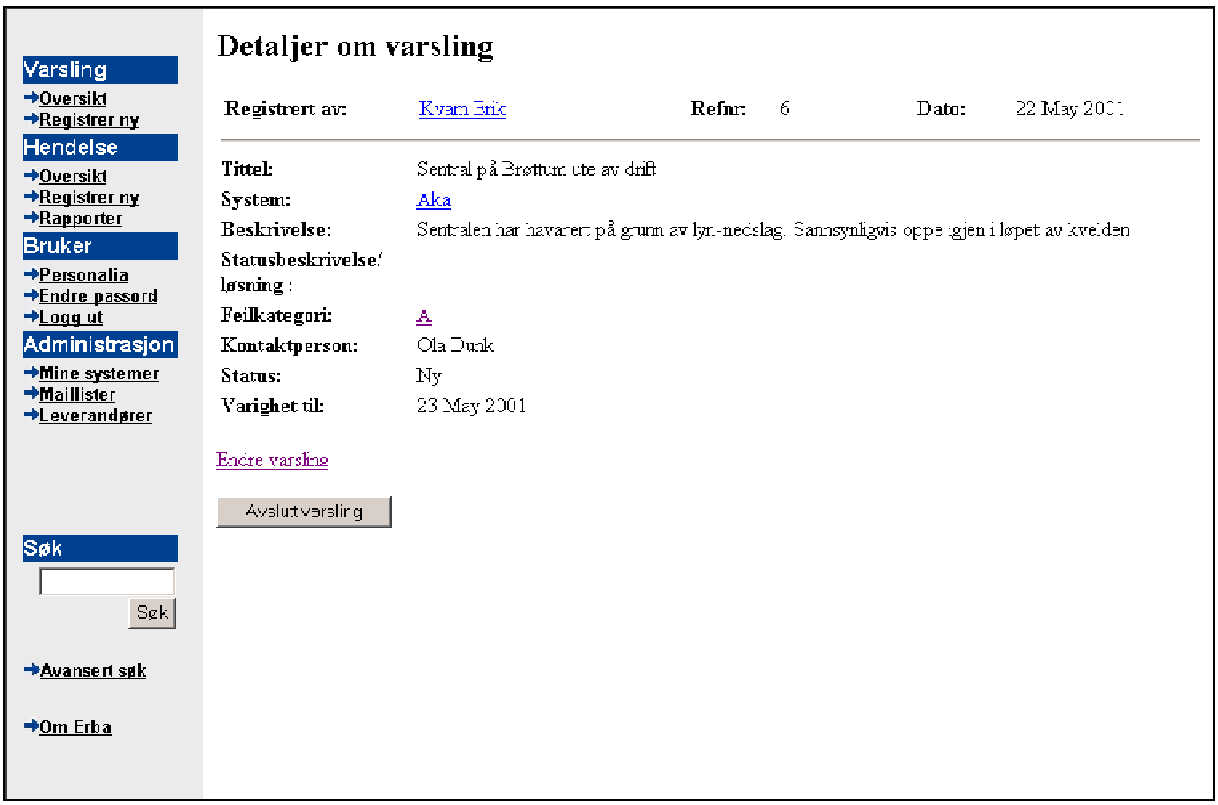
Viser til Figur 3-1 for bilde.

Denne siden er delt i to deler. All tekst på øvre halvdel av siden er redigerbar via en egen redaktørside (3.2.6.2). Formålet med denne er å kunne skrive inn en velkomsttekst, og ellers nyttige opplysninger.

Denne er inndelt i en hoverdoverskrift, tre tekstbokser med overskrift, og et tekstområde på undersiden. Dette var ikke et krav fra oppdragsgiver, men er gjort for å gjøre sidene litt mer "spennende". Dersom det ikke er registrert noen tekst i boksene, vises de ikke.

På den nedre halvdelen vises oversikt over varslinger.
Tabellen inneholder de samme feltene, som skissene fra oppdragsgiver. I tillegg satte vi inn et felt "system", etter ønske fra oppdragsgiver i møte 21. feb.
Trykker man på referansenummeret til varslingen vil man komme til siden som viser detaljer om valgt varsling (3.2.2.2).

3.2.2.2 Detaljer om varsling



Figur 3-2 Viser meny og detaljer om varsling slik det vises for brukerkategori 40

Formålet med denne siden er å vise detaljer om en varsling på en oversiktlig måte.

Det er også mulighet for å gå videre for å endre varsling. Denne linken kommer kun opp for de med brukerkategori 30 eller høyere. Brukerkategori 40 og over, har i tillegg til dette en knapp for å avslutte varslingen.

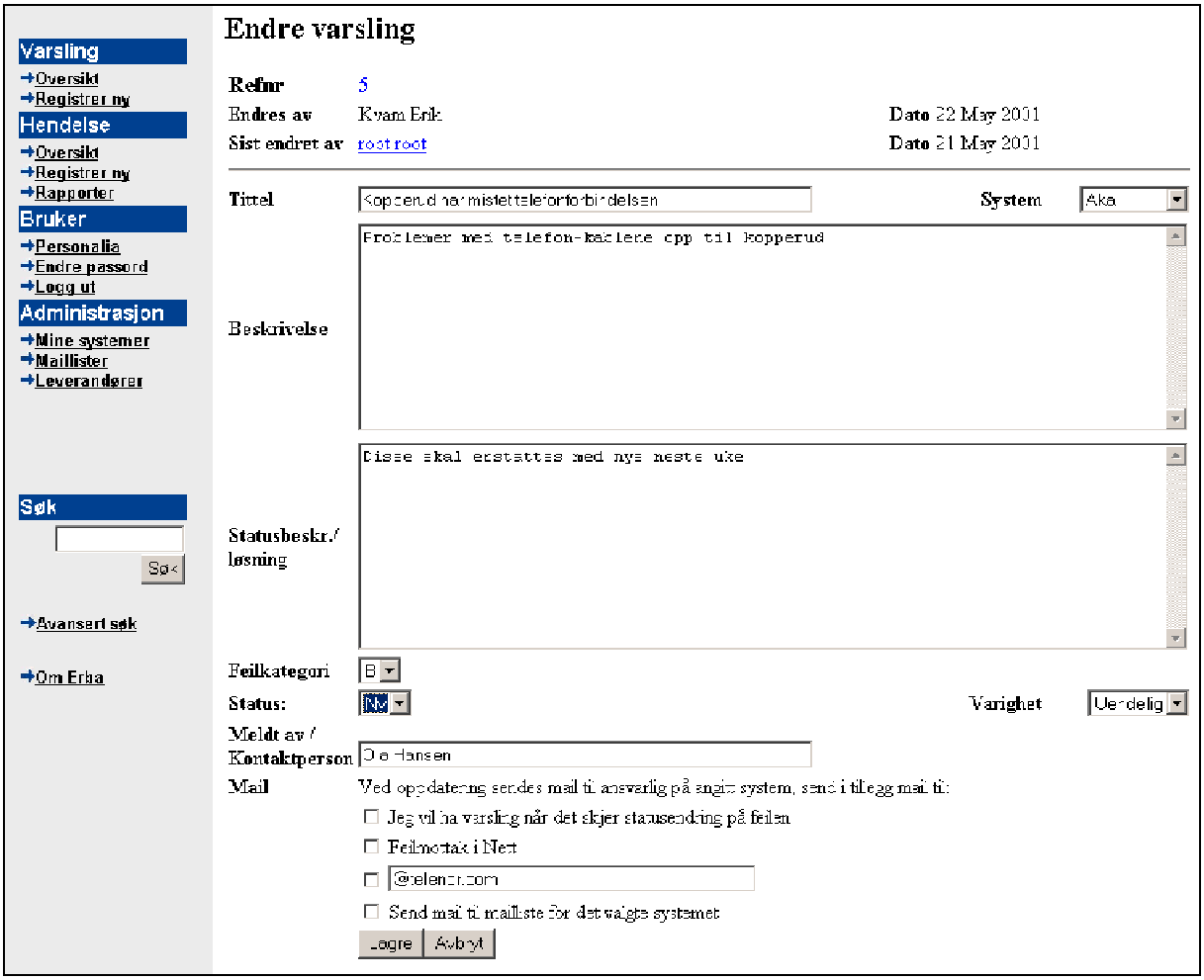
3.2.2.3 Registrer varsling

Siden brukes for å registrere en ny varsling.
Ved varsling skal det registreres tittel, system, beskrivelse, varighet og kontaktperson.

Det skal være muligheter til å motta mail når noen gjør forandringer på varslingen som er registrert. Dette gjelder bare den første gangen meldingen blir endret. Det er også muligheter til å sende mail til Feilmottak i nett, en valgfri person og mailliste for aktuelt system. For å gjøre dette brukes avkrysninsbokser. Systemet sender automatisk mail til systemansvarlig for valgt system, når varslingen registreres.

Varighet på en varsling kan angis i en dropdownboks. Valgte dette pga. at det er en enkel måte å sikre at riktige verdier blir angitt. Kan velge mellom ett, to eller tre døgn, eller uendelig. Feltene tittel, system og beskrivelse må være utfylt før man lagrer varslingen ellers vil man få feilmelding om hva som mangler. Varslingen får automatisk status nr 1 og feilkategori nr 1.

3.2.2.4 Endre varsling



Figur 3-3 Endre varsling

Denne siden brukes for å endre og oppdatere en varsling. I tillegg til feltene i Registrer varsling, har vi statusbeskrivelse/løsning, feilkategori og status. Varighet hentes ikke fra databasen, dette må velges på nytt, slik at man får angitt varighet fra da varslingen blir endret.

Varslingen som er endret får samme referansenummer. Varslingen som er endret, slettes ikke fra databasen, den gjøres usynlig for vanlige brukere.

3.2.2.5 Oversikt over hendelser

Siden viser en tabell over alle hendelser som er registrert og som er FAQ meldinger. Det vises kun 20 hendelser om gangen. Ligger det flere hendelser i databasen gis det mulighet for å bla frem og tilbake. Feltene som vises er referansenummer, tittel, status, dato og system. Klikker man på referansenummeret får man opp detaljer om hendelsen.

3.2.2.6 Detaljer om hendelser

Siden viser alle detaljer om en hendelse. Har man en brukerkategori 40 eller høyere får man opp en link nederst på siden til endre hendelse (3.2.2.8).

Brukerkategori 50, vil få mulighet til å endre faq – statusen eller å slette/gjenopprette meldingen. Grunnen til at vi la inn denne muligheten, er at redaktør skal kunne gå inn å gjøre slettede meldinger synlige og kunne endre faq-status uten å oppdatere hendelsen.

3.2.2.7 Registrer hendelse

Siden brukes for å registrere ny hendelse.

Her skal man ha mulighet til å fylle inn feltene tittel, system, meldingskategori, leverandør, beskrivelse, statusbeskrivelse/løsning, eksternt referansenummer, legge inn vedlegg og kontaktperson. Man kan velge at meldingen skal ha FAQ - status, legges ut som varsling og sende mail til systemansvarlig på valgt system.

Feltene system, leverandør, meldingskategori, status er laget som dropdownbokser. Vedlegg-feltet er et felt hvor man kan velge hvilken fil som skal legges med som vedlegg.

Feltene hvor man kan velge FAQ-status, legge hendelsen ut som varsling og sende mail til ansvarlig er avkrysningsbokser. FAQ–status og feltet for å sende mail til ansvarlig er default valgt.

Tekstfeltene er gjort like stor, unntatt eksternt referansenummer som bare skal være 10 tegn langt. Dette er gjort for å få mest mulig likt over hele skjermbildet slik at det blir mest mulig likt.

3.2.2.8 Endre hendelse

Siden brukes for å endre en hendelse.

I tillegg til feltene angitt i 3.2.2.7 er det mulighet til å endre på feltene status og aksjonspunkt.

3.2.2.9 Personalia om bruker

Denne siden skal brukes for å endre data om seg selv som bruker på systemet. Man skal kunne endre fornavn, etternavn og mailadresse om seg selv. Ansattnr kan ikke endres, da det vil gi feil i databasen. Tekstfeltene er valgt å ha samme størrelse, fordi vi mener dette er mest oversiktlig.

3.2.2.10 Endre passord

Denne siden brukes til å endre passordet til pålogget bruker. Her er bruker nødt til å oppgi gammelt passord, legge inn det nye passordet to ganger for å lagre det nye passordet.
Vi har gjort dette mest mulig likt vanlig standard.

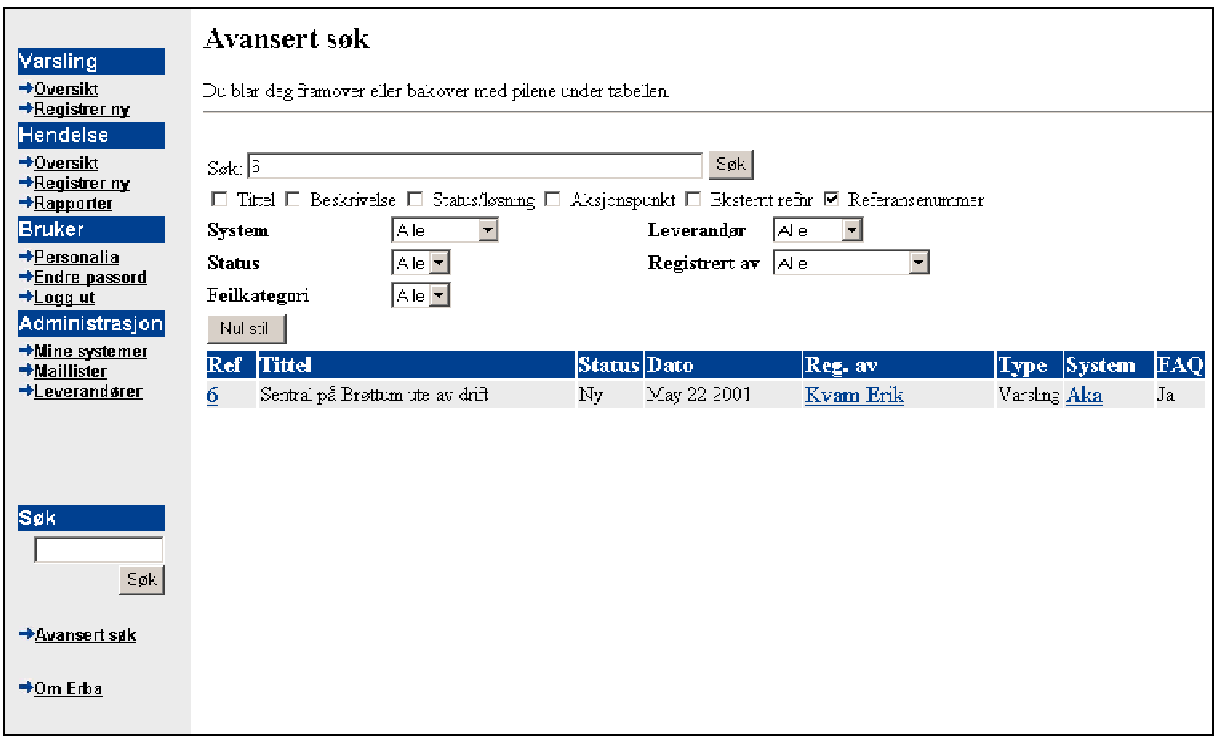
3.2.3 Søkefunksjoner

Funksjonene søker kun i de feltene pålogget bruker har rettighet til å se. Det søkes med *søkeuttrykk*, dvs at man kan søke ved å oppgi kun deler av et ord, og det søkes også på ord som står midt inne i setninger.

3.2.3.1 Søk i meny

Det var et ønske fra oppdragsgiver, å ha en lett tilgjengelig søkefunksjon, som skulle være et fritekstsøk. Menyen var derfor et naturlig sted å plassere denne.

3.2.3.2 Avansert søk



Figur 3-4 Avansert søk

Her skal man kunne angi nøyaktig hvilke felter man skal søke i, for mest mulig presist treff. Brukerkategori 50 har også mulighet for å søke blant meldinger som er slettet.

Resultat-tabell viser Referansenummer, Tittel, Status, Dato, Registrert av, Varsling/hendelse og System.

Kategori 30 og over kan se meldinger som ikke har FAQ-status. Disse har i tillegg et felt som viser FAQ-status på meldinga. Kategori 50 kan også se meldinger som er slettet. På disse står det ”sletta” i feltet for FAQ-status.

3.2.3.3 Rapporter

Dette er en side hvor bruker har mulighet til å finne fram rapporter. Det er to dropdown-bokser. Den første boksen angir om det er system, leverandør eller status man ønsker rapporter fra. Her er det system som står som default, siden dette er det som kommer til å bli brukt mest.

Den andre dropdown-boksens innhold avhenger av hva som blir valgt i den første. Hvis f.eks systemer er valgt, vil de systemene som ligger inne komme opp i denne boksen.
Rapportene vises på standard form med sortering vha å trykke på overskrift, viser 50 felter per side, og har linker til beskrivelser av feltene

3.2.4 På logging og utlogging.

Disse sidene er to små vinduer som vises når linkene aktiveres. Vinduene popper opp på skjermen når linken aktiveres. Vinduene åpnes midt på skjermen. Valgte denne måten, da dette er en vanlig påloggingsporsedyre.

3.2.4.1 Pålogging

Vi har i tillegg til felter for å oppgi ansattnr og passord 4 knapper:

- Glemt passord?: Denne åpner en ny side i samme vinduet som pålogging. Her kan man angi ansattnnummer, hvis man har glemt passord. Det genereres da ett nytt passord, som sendes til den e-mailadressen som er registrert på aktuelt ansattnr.
- Ny bruker: Her lukkes påloggings vinduet og det kommer opp en ny side i hovedframen. Denne siden er helt lik personalia om bruker, se avsnitt 3.2.2.9, bortsett fra at man må legge til passord.
- Avbryt: lukker påloggingsvinduet uten å gjøre noen forandringer.
- Logg inn: lukker påloggingsvinduet og åpner brukermenyen på nytt med riktig rettighet

Skulle noen av tekstfeltene ikke være utfylt, vil man få melding om dette.

3.2.4.2 Logg ut

Denne linken er kun aktiv når bruker er pålogget. Når man aktiverer linken vil det åpnes et nytt vindu slik som ved pålogging. Her får man spørsmål om man virkelig vil logge seg ut. Man kan velge mellom alternativene ok eller avbryt. Velger man ok så vil man logge seg ut av systemet og få opp menyen for brukerkategori 0 og framsiden.

3.2.5 Sider for kategori 40

3.2.5.1 Mine systemer

Siden viser en tabell med oversikt over alle varslinger og hendelser som ligger i databasen som omhandler systemer bruker er systemansvarlig for. Oversikten viser meldinger som er tilgjenglig for alle og meldinger som er bare tilgjenglig for systemansvarlig eller redaktør, dvs slettet.

Feltene som vises er referansenummer, tittel, type, status, dato, system og FAQ. Type viser om meldingen er hendelser eller varslinger. FAQ-feltet viser ”sletta” hvis meldingen er usynlig. Trykker man på referansenummeret til meldingen vil man få opp detaljer om varslingen eller detaljer om hendelse ettersom om det er varsling eller hendelse. Se detaljer om varsling (3.2.2.2 ovenfor) eller detaljer om hendelse (3.2.2.6 ovenfor).

3.2.5.2 Maillister

Vi kan her sette opp maillister for de forskjellige systemene.

Siden består av to dropdownbokser, en for system og en for brukere. Selve maillisten er en liste. Det skal være mulig å legge til eller slette personer fra denne listen.

For å legge til en person på en mailliste, må man først velge et system. Deretter kan man velge å legge til brukere.

3.2.5.3 Leverandører

Vi fikk etterhvert ønske om at katagori 40 skal kunne redigere info om leverandører. Denne siden er beskrevet sammen med redaktørsidene (3.2.7).

3.2.6 Sider for kategori 50 (redaktør)

3.2.6.1 Brukere

Denne siden viser en tabell med oversikt over brukere på systemet. Den er default sortert etter ansattnummer, men det er muligheter for å klikke på overskriftene til tabellen for å få den sortert etter ønsket kriterium. Feltene som vises er ansattnummer, etternavn, fornavn, mailadresse og rettigheter.

Det er mulighet for å gå inn å endre detaljer om hver enkelt bruker. Rettighetene kan velges vha. en dropdownboks.

Vi fikk et ønske underveis om å legge til mulighet for å skrive en kommentarer om bruker, denne kommentaren kan kun sees av brukere i kategori 50. Dette kan for eksempel brukes for å angi hvorfor en gitt bruker har andre rettigheter enn en standardbruker.

Velger man å oppdatere vil endringer lagres i databasen og man vil få oversikt over brukere igjen. Velger man å avbryte vil man bare kommer tilbake til oversikten over brukere.

3.2.6.2 Redigere framside

Siden brukes av redaktør for å endre framsiden. Viser til avsnitt 3.2.2.1 om beskrivelse av framsiden. Her kan man redigere overskriften, de tre tekstboksene og tekstområdet under. Lar man en tekstboks stå tom, vil ikke denne vises på framsiden.

Det er lagt vekt på at denne siden skal være mest mulig lik den øvre delen av framsiden slik at man lett forstår hva man redigerer.

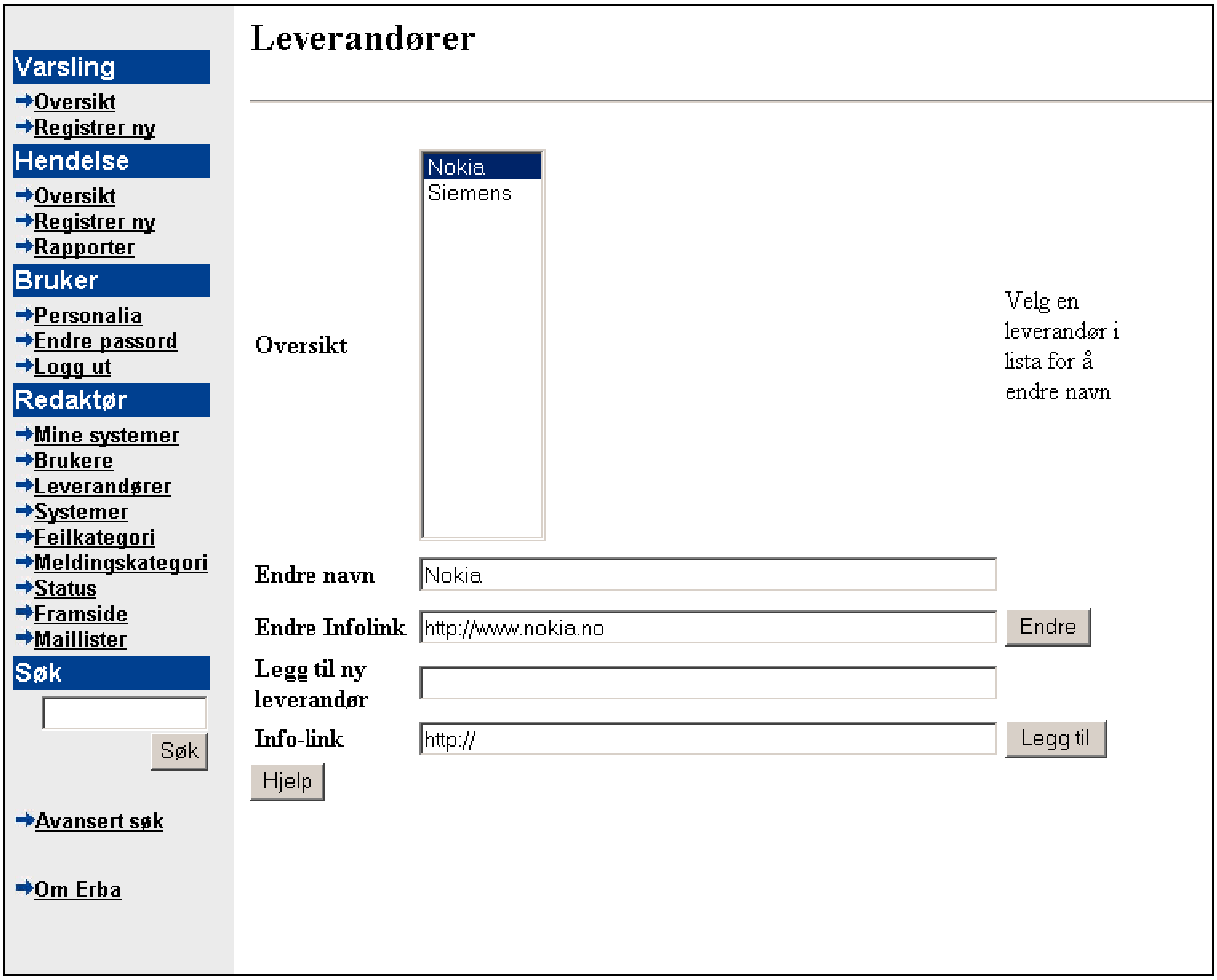
3.2.7 Leverandører, systemer, feilkategori, meldingskategori og status.

Litt generelt:

Sidene som ligger under dette avsnittet er kun tilgjenglig for redaktør. Sidene brukes til å legge data inn i ”hjelpetabellene” i databasen. Man kan legge til nye poster, og endre eksisterene poster. Alle sider inneholder en liste. Blir noen i listen merket vil få opp informasjon om disse i endre-feltene. Skal man legge til nye poster har man egne felter, legg til ny. Vi har valgt å lage disse feltene mest mulig likt for å gjøre det enkelt å lære seg å bruke.
Kategori 40 har rettighet til å endre leverandører, ellers er sidene tilgjengelig kun for kategori 50.

Viser her litt mer informasjon om hver enkelt side:

3.2.7.1 Leverandører



Figur 3-5 Viser meny for brukerkategori 50, og siden leverandører

Oppdragsgiver ønsket at i tillegg til bare navnet til leverandøren, ville man også at det skulle være mulig å legge til en informasjonslink til leverandøren. Dette vil ofte være for eksempel hjemmesiden til leverandøren.

3.2.7.2 Systemer

I tilegg til informasjonslink, er det et ønske at man skulle bestemme hvem som var systemansvarlig for et gitt system. Siden vil i tillegg til det som er nevnt i punkt 3.2.7 bestå av feltene legge til systemansvarlig og systemansvarlig(e).

For å legge til systemansvarlige merker man av systemet. Informasjonen om dette systemet kommer i endre feltene. Videre velger man hvem som skal være systemansvarlig ut i fra dropdownboksen legg til systemansvarlige. I dropdownboksen ligger alle brukere som ligger i databasen.

For å slette en fra listen, merker man en systemansvarlig og klikker på knappen slett.

3.2.7.3 Feilkategori og meldingskategori

Velger å beskrive disse likt fordi de er helt like. Sidene er nesten like leverandørsiden (3.2.7.1), bortsett fra at i stedet for informasjonslink har vi lagt til et tekstområde hvor det er mulig å legge til en liten beskrivelse om valgt feilkategori eller meldingskategori.

3.2.7.4 Status

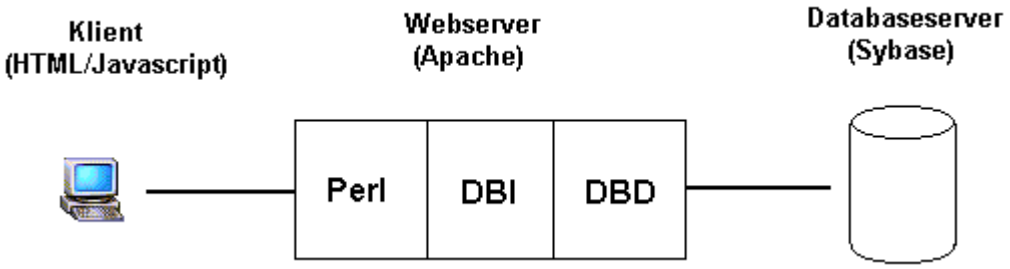
Siden har en liste som viser oversikt over status, man kan også endre eller legge til ny status.

4 Implementering

4.1 Programvareplattform.

4.1.1 Valgt programvare

Scriptene er laget for UNIX/LINUX, og bruker tilleggsmodulene DBI og DBD. Disse skulle brukes for å kommunisere med sybase, etter ønske fra oppdragsgiver. Vi har i tillegg brukt ett tilleggs-script, cgi_lib.pl. Dette har bl.a. en del nyttige funksjoner i forbindelse med filopplasting. Viser til avsnitt 4.4.2.6 for nærmere beskrivelse av dette.



Figur 4-1

4.1.2 Alternativer

Vi ytret ønske ovenfor oppdragsgiver om å kunne bruke PHP, da dette er et nytt, moderne og kraftigere utviklingsverktøy enn Perl/CGI. Vi ble også anbefalt dette etter samtale med faglærer i faget Klient og Serversideprorammering, og veileder. Oppdragsgiver undersøkte litt omkring dette, og fant ut at det ikke finnes drivere for solaris platform som takler php mot sybase. Oppdragsgiver har dessuten god kompetanse på Perl, og ønsket derfor dette språket mht. senere endringer i systemet.

4.2 Utviklingsmiljø

For å utvikle web-sidene brukte vi Windows og Macromedia Dreamweaver. Alle skjemaer er tegnet vha Dia for linux. For resten av utviklingen valgte vi linux og emacs. Vi satte opp en egen server med Sybase og Apache Webserver.

Vi lagde et eget perlscript for å konvertere html-sidene fra Dreamweaver, til et format som var lettere å implementere i perl. Se konverter.pl (4.6.1.1).

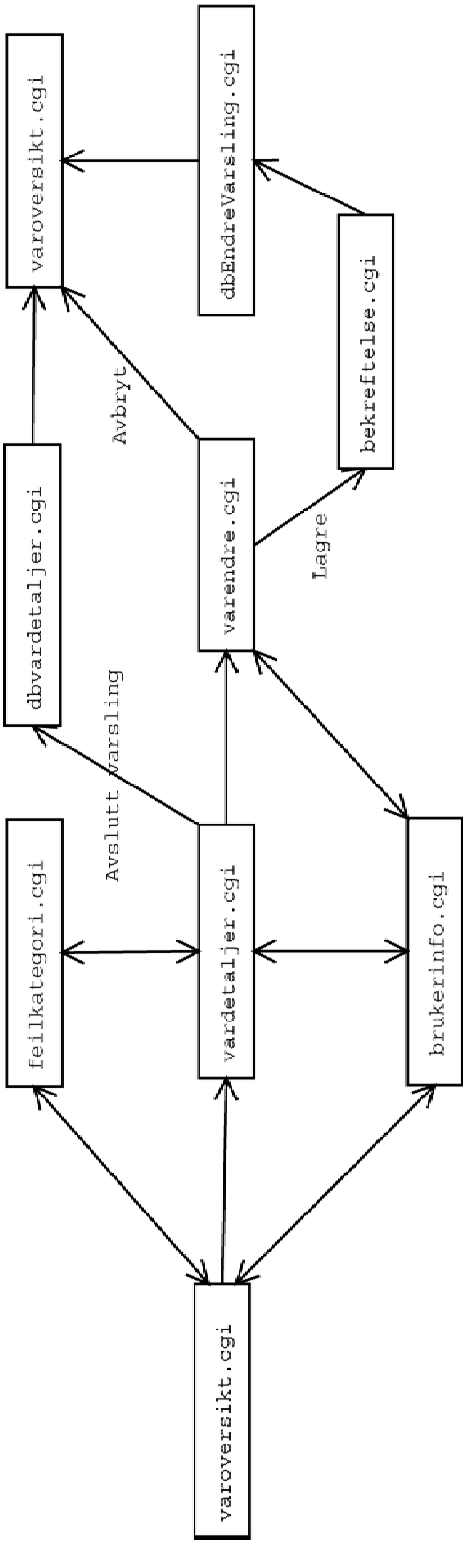
Ingen av oss hadde noe særlig erfaring med linux fra før, men vi valgte å bruke det som operativsystem og utviklingsmiljø, da det var enklest å få tak i programvare her. Dette var også en fordel da sluttsystemet skal gå i et UNIX-miljø, og vi uansett hadde kommet bort i en del UNIX/LINUX-kommandoer.

4.3 Scriptoversikt

De følgende flytskjemaene viser hvordan de forskjellige scriptene tilkaller hverandre. Figurene har samme rekkefølge som menyvalgene, slik menyen er for brukerkategori 50.

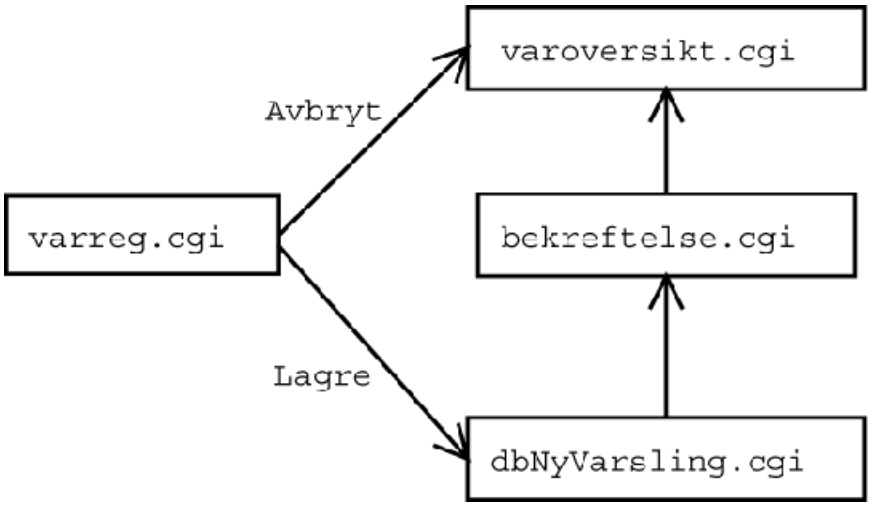
4.3.1 Varsling

Oversikt



Figur 4-2

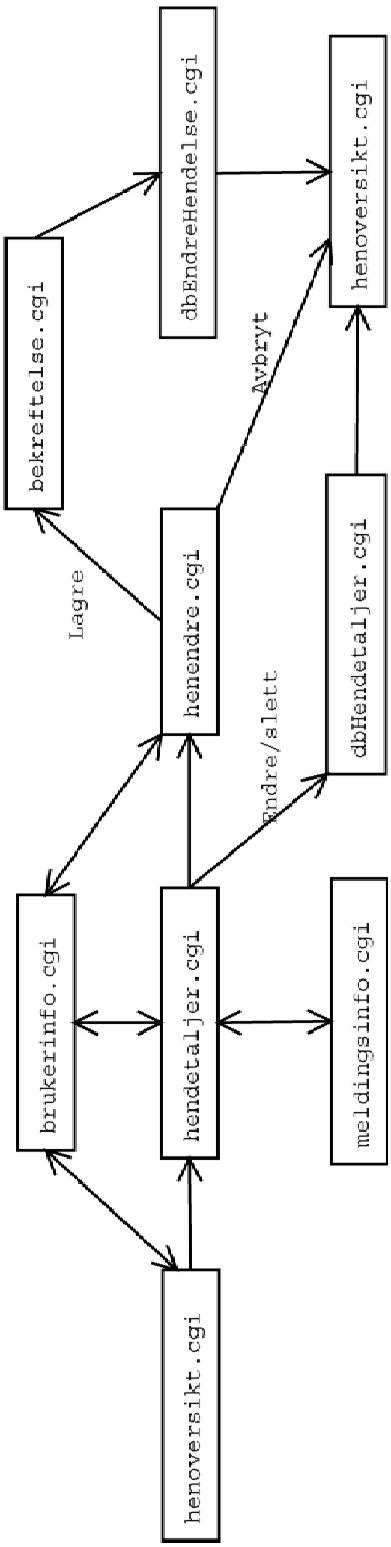
Registrer Ny



Figur 4-3

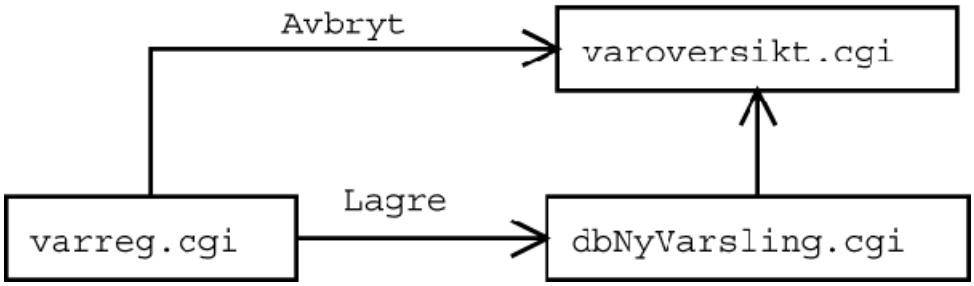
4.3.2 Hendelse

Oversikt



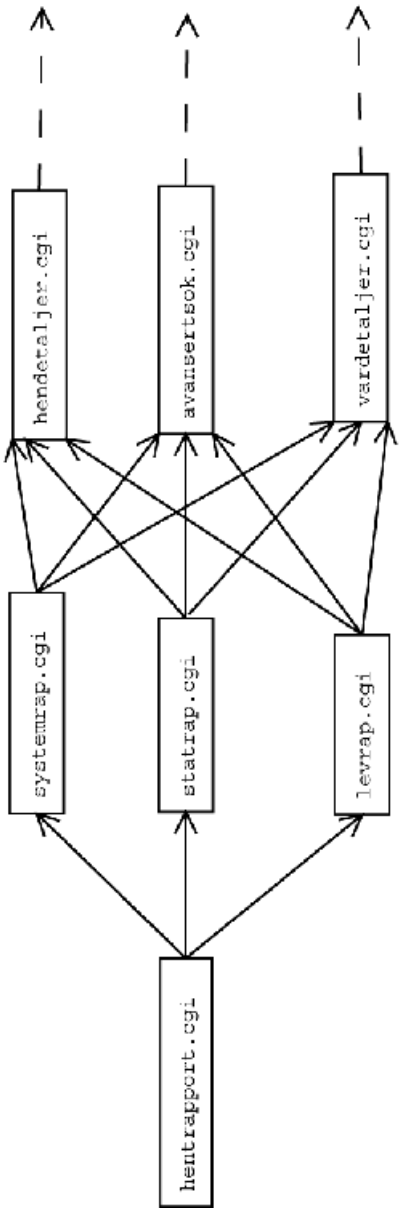
Figur 4-4

Registrer Ny



Figur 4-5

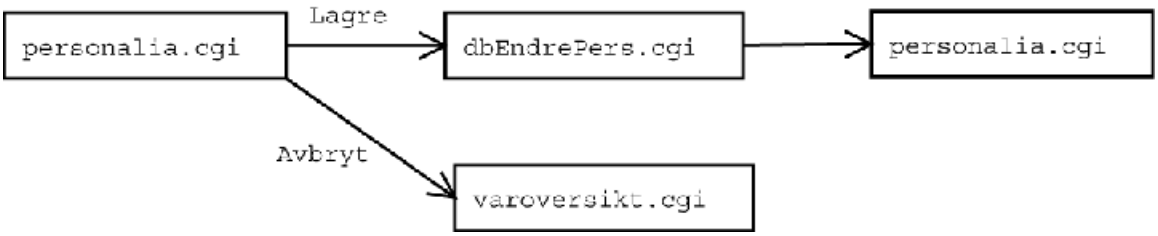
Rapporter



Figur 4-6

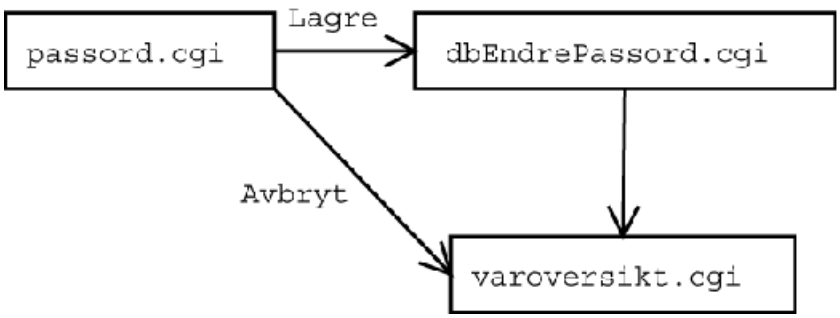
4.3.3 Bruker

Personalia



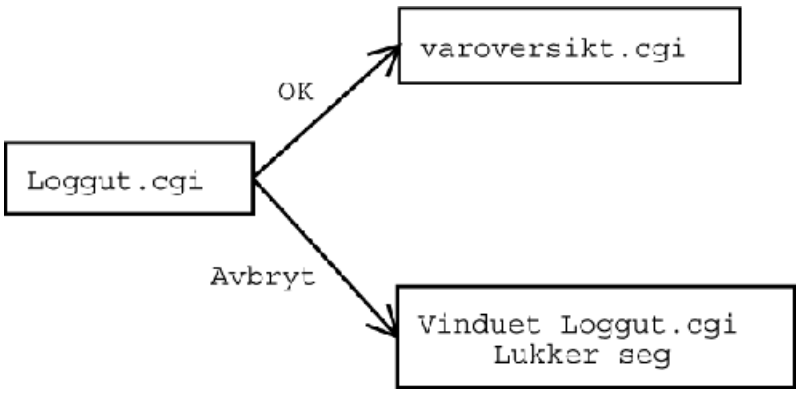
Figur 4-7

Endre Passord



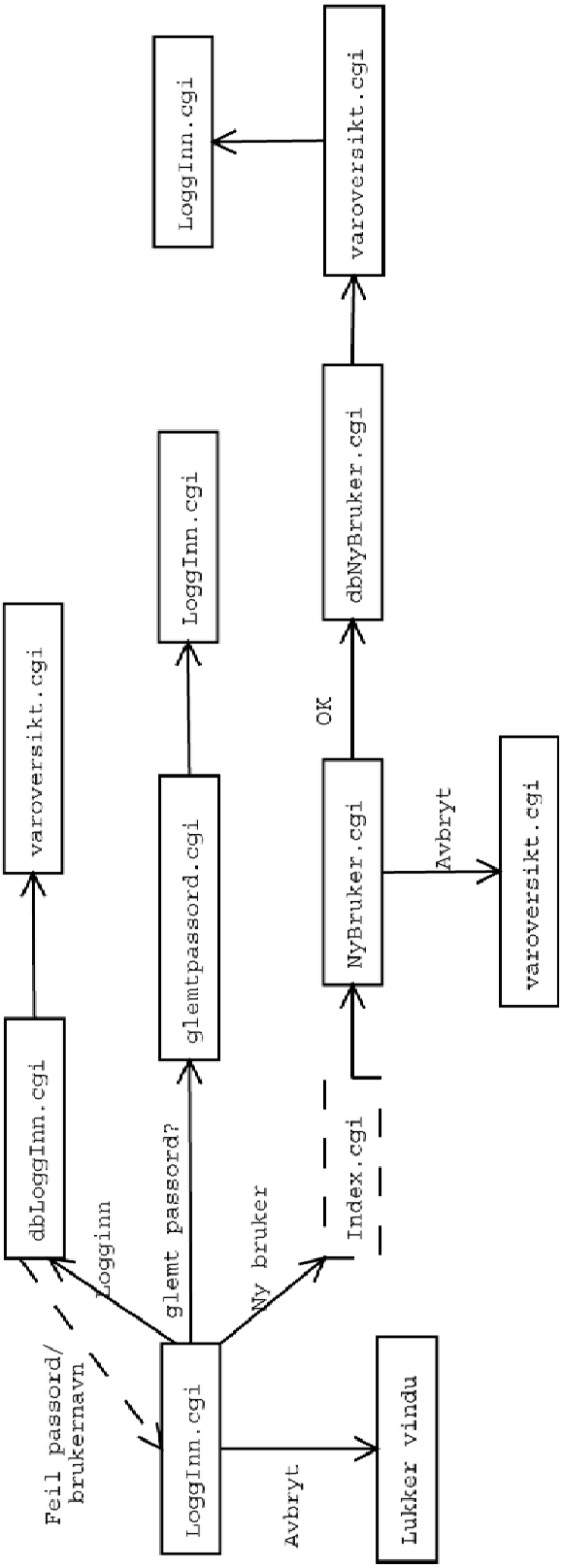
Figur 4-8

Logg Ut



Figur 4-9

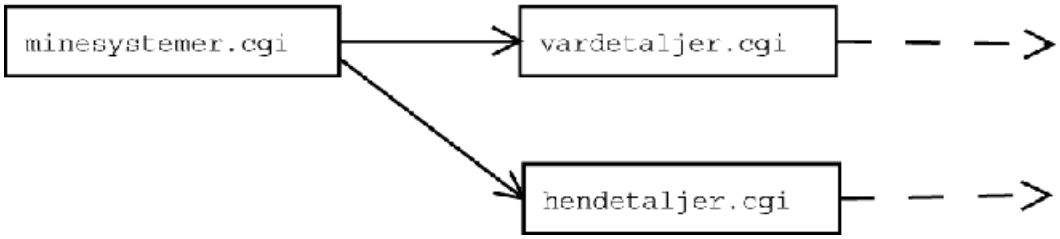
Logg Inn



Figur 4-10

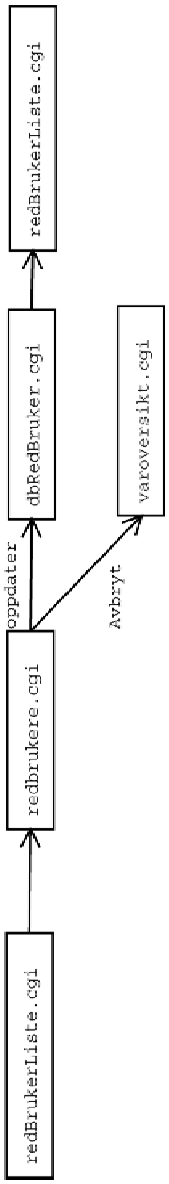
4.3.4 Redaktør

Mine systemer



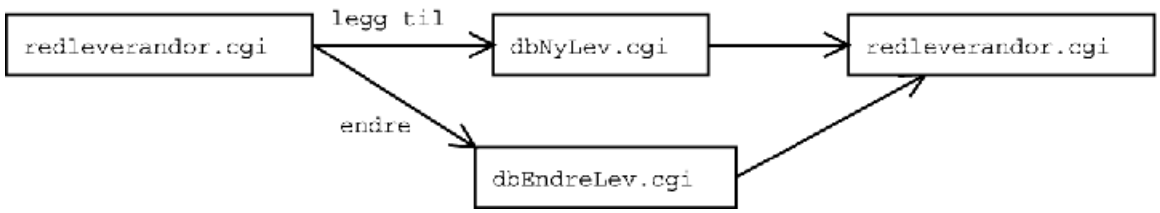
Figur 4-11

Brukere



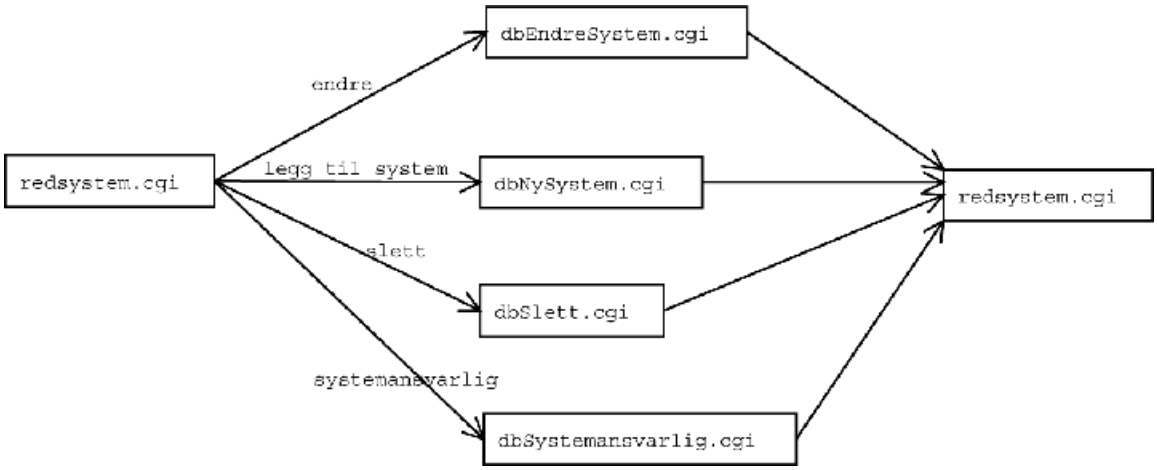
Figur 4-12

Leverandører



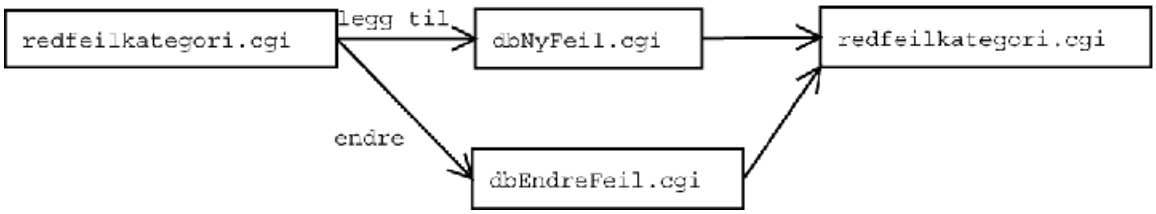
Figur 4-13

Systemer



Figur 4-14

Feilkategori

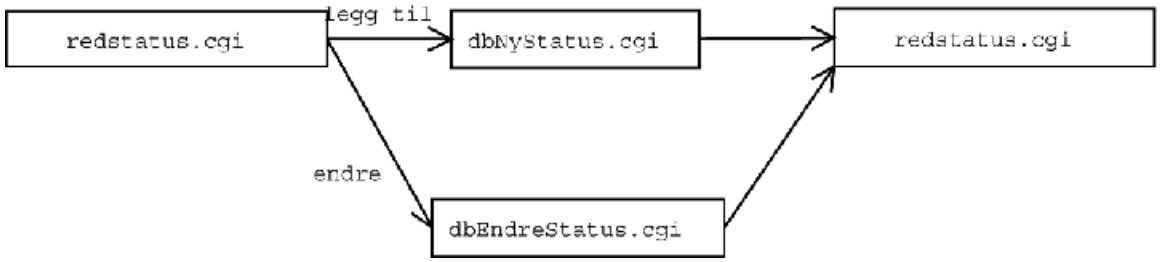


Figur 4-15

Meldingskategori

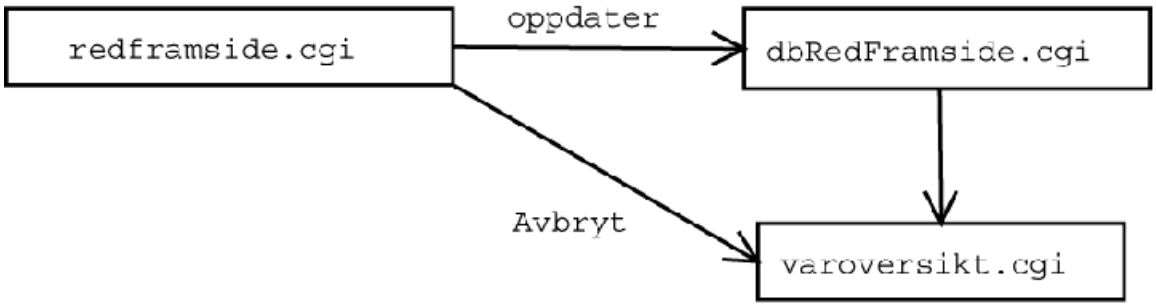
Figur 4-16

Status



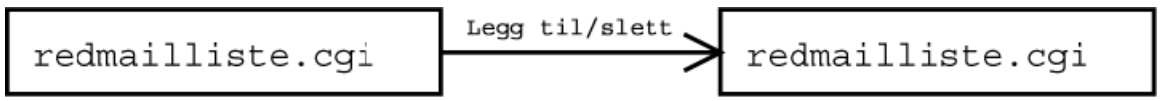
Figur 4-17

Framside



Figur 4-18

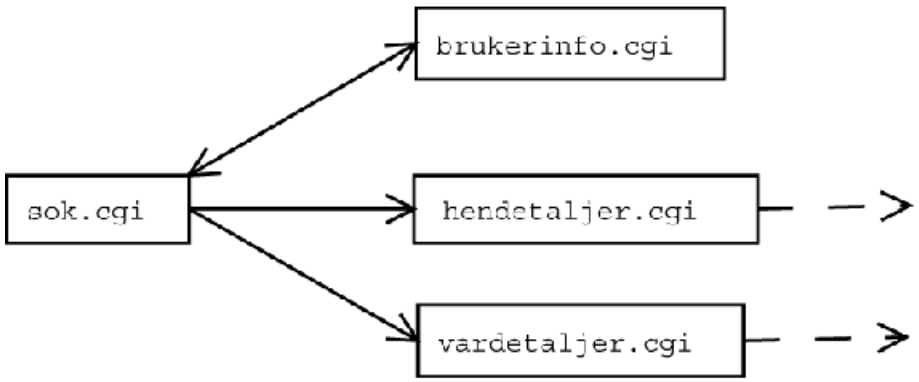
Maillister



Figur 4-19

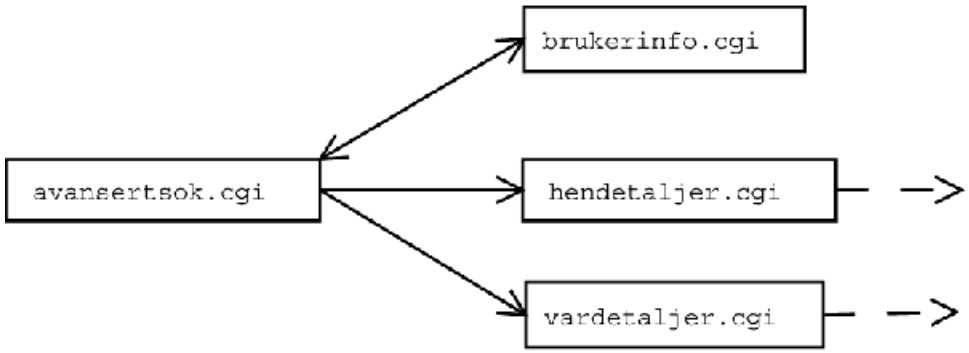
4.3.5 Søkefunksjoner

Søk



Figur 4-20

Avansert Søk



Figur 4-21

4.4 Scriptbeskrivelser

Dette del-kapitlet er beregnet på systemansvarlig, og for bruk ved videreutvikling av systemet. Det er derfor skrevet i et teknisk avansert språk, for å kunne beskrive de fleste detaljene.

De viktigste uttrykkene er beskrevet i 4.5.

4.4.1 Standarder

4.4.1.1 Rettigheter

Det brukes en cookie for å kunne bestemme hvilken rettighet pålogget bruker har. Det utføres sjekk på rettighet i hvert enkelt script, hvor dette er nødvendig. På den måten er man sikker på at ingen klarer å kjøre et script ved bare å skrive inn scriptnavnet i nettleseren, uten at de har rettigheter til å kjøre scriptet. For å sjekke rettigheter, inkluderes scriptet getcookie.dta (4.4.2.3)

4.4.1.2 Tabeller

I noen tabeller vises det 20 poster på hver side, på andre vises 50 poster på hver side.
Dette ordnes ved at det sendes med hjelpevariabler vha. get, som angir hvilke felter som skal vises. Når man trykker for å få opp neste eller forrige side, tilkaller den aktuelle siden seg selv, og det sendes med to variabler, en som angir hvor visningen skal starte og en som angir hvor visningen skal slutte. Spørringen kjøres på nytt hver gang.

De fleste tabeller kan sorteres ved å trykke på overskriftene i tabellen. Disse overskriftene er da vanlige linker, som tilkaller den samme siden, og sender over ønsket sorteringskriterier vha get-funksjon.

4.4.1.3 Dato

Datoer lagres i databasen med både dato og klokkeslett, for at meldinger som er skrevet på samme dag også blir sortert riktig. Ved utskrifter skrives derimot ikke klokkeslettet ut. Dette tas bort vha split-funksjonen i perl.

4.4.1.4 Feilmeldinger

Alle script har en egen funksjon, som tar seg av feilmeldinger. Script som vanligvis ikke skriver ut noe, har en funksjon som heter ”htmldie” som genererer en ny html-side med

feilmeldingen på. De andre sidene bruker en noe enklere funksjon, som blir kalt ”feilmelding”, da disse scriptene allerede har html-kode fra før. Alle feilmeldinger har forskjellige referansenummer. Systemansvarlig har en egen oversikt over alle disse.

4.4.1.5 Stiler

Størrelser og farger på bokstaver og linker angis i en egen fil vha. CSS. Dette ligger i filen stiler.css.

4.4.1.6 Forskjellige script-typer

Dette er en beskrivelse av scriptenes oppbygning og funksjonalitet. Scriptene er delt opp i 4 hoveddeler. Disse 4 delene er:

4.4.1.6.1 Require-script

Disse scriptene er laget for å gjøre det enklere å redigere og videreutvikle systemet senere. Dette er script som utfører en bestemt oppgave og returnerer en verdi tilbake til hovedscriptet. Med hovedscript menes det scriptet hvor det beskrevne scriptet er tilkalt fra eller inkludert i. Disse oppgavene er blant annet dekoding av post- og getforespørsler fra en html-form, definering av systemvariabler, kopling mot database, henting av html-cookie osv.. mer spesifikt om dette i 4.4.2.

Disse scriptene må returnere en true (ikke 0) verdi for at de skal bli kjørt. For å unngå feil på grunn av dette er ”require”-scriptene kodet på en slik måte at de returnerer -1 i stedet for 0 ved feil. Med dette unngår vi at scriptene stopper å kjøre og vi kan dermed gi brukeren en feilmelding i stedet.

4.4.1.6.2 Database-script

Dette er script som jobber mellom grensesnittscriptene og databasen. Disse utfører innlegging av poster i databasen, oppdateringer, spørringer og sending av mail. Disse mottar data fra en html-form og bruker ”require”-scriptene til å behandle disse dataene før de blir lagt inn i databasen. Disse scriptene er bygd opp på en DBI-modul og DBD-modul mot databasen. DBI (DataBase Interface) og DBD (DataBaseDriver) er de nye modulene for perl mot database. DBI jobber mellom perl og DBD mens DBD jobber direkte mot databasen. DBD er en databasedriver som er spesifikk for hver enkelt database. Nærmere beskrivelse av dette kommer i 4.4.2.4.

Til slutt i disse scriptene kommer subrutiner som sender mail og skriver ut feilmeldinger (HTMLdie). Se 4.4.1.4.

4.4.1.6.3 Grensesnitt-script

Dette er scriptene som viser grensesnittet mot brukeren og genererer html-sidene. Her har vi valgt å bruke ”print” funksjonen i perl for å skrive ut html-kode i stedet for å bruke CGI-modulen til perl. Dette ga oss mulighet til å utforme html-sidene i dreamweaver først og så konvertere disse over til perlkode etterpå. For å få konverteringen effektiv, kodet vi et script som gjorde dette for oss.

Disse scriptene bruker htmlhead.dta (4.4.2.5) for å skrive ut head i html-formen. Så kommer javascript-delen som brukes til å sjekke at påkrevede felter (for eksempel for- og etternavn, tittel, mail osv..) har en verdi og at disse verdiene er gyldig. Vi har valgt å bruke et regulært uttrykk, kodet i javascript, til å sjekke et emailadresse er gyldige. Videre har vi valgt å bruke javascript til å bestemme hvilke sider som html-formen skal tilkalle, da dette bestemmes av hvilke verdier som er tastet inn (for eksempel redaktørscriptene hvor man endrer og legger inn nye poster i samme script). Websidene til system- og leverandør-info åpnes i egne vinduer vha. javascript-kode.

Når htmlhead- og javascript-delen er avsluttet så kommer perlkode for opphenting av data fra databasen. Så skrives body og html-formen ut.

Til slutt kommer subrutiner for feilmeldinger.

4.4.1.6.4 Vedlikeholds-script

Vi har valgt å kalle disse scriptene vedlikeholdsscript fordi dette er script som kjøres av cron i linux regelmessig. Dette er script som tar seg av transaksjons- og databasedump av databasen samt endrer varslinger til hendelse når varigheten på varslingen har ”gått ut på dato”.

Databasedump.pl og endreVarHen.pl kjøres en gang i døgnet mens transactiondump.pl kjøres annenhver time.

4.4.1.7 Scriptnavn

Vi brukte de første bokstavene i filnavnene for å angi hvilken del av systemet scriptet gjelder.

For eksempel har scriptet som lager skjermbildet for å endre hendelse navnet ”henendre.cgi”, og det scriptet som tar seg av databasekommunikasjonen er ”dbhenendre.cgi”.

Require-scriptene har endinger på .dta, for eksempel postinn.dta.

Vedlikeholds-scriptene har ending på .pl. Dette for at de skal kjøres av perl.

4.4.2 ”Require”-script

4.4.2.1 postinn.dta

Dette er det mest essensielle script. Dette scriptet brukes til å motta og konvertere alle post-data fra en html-form. Scriptet sjekker på om ”REQUEST_METHOD” fra formen er ”post” og om”CONTENT_TYPE” er av typen ”^application/X-www-form-urlencoded$”. Hvis disse er ok så leses ”STDIN” inn i variabelen $innlinje. Denne variabelen splittes så opp i par ut fra ”&” og så inn i en assosiativ array som er kalt for ”$Field”. Her blir navnet til input-feltene i html-formen lagt inn som navnet på ”feltene” i arrayen, noe som gjør det enkelt når vi skal ta ut verdiene igjen og legge dem inn i databasen.

Hvis et script ikke mottar post og postinn.dta er inkludert, så blir det opprettet en variabel med navn $Field{’ip’}. Dette er gjort for å gjøre feilsjekking i scriptene lettere og for sjekke at scriptet virkelig får en postforespørsel. Dette scriptet er inkludert i alle ”Database-script” unntatt dbEndreHendelse.cgi og
dbNyHendelse.cgi. Disse to scriptene kommer vi tilbake til senere.

Kildekode vedlagt, se 4.6.2.4.

4.4.2.2 getinn.dta

Dette er nesten det samme scriptet som postinn.dta, men det sjekker om ”REQUEST_METHOD” fra formen er ”get”. Dette er et script som blir inkludert for å sende ”enkle variable” mellom forskjellige script for å få mottakerscriptet til å utføre en bestemt handling. Variabelen som inneholder verdien(e) er kalt for $inn. Denne variabelen blir satt til -1 hvis scriptet ikke er tilkalt med ”get” metode.

4.4.2.3 getcooki.dta

Dette er scriptet som henter ut cookien ”Ted” fra nettleseren til brukeren. ”Ted” er navnet på cookien som blir opprettet når brukeren logger seg på systemet. Hva som blir lagt inn som verdi i denne cookien vil vi ikke oppgi pga sikkerhetsmessige grunner.

Dette scriptet blir inkludert for å finne rettighetene til brukeren og dermed kunne sjekke om vedkommende har rettigheter til å kjøre et bestemt script, og kunne bestemme hvilke linker som skal vises til brukeren i menyframe.cgi (4.4.4.2). Dette scriptet er også laget for å kunne takle ansattnr som starter med en eller flere nuller.

Dette scriptet returnerer innholdet i cookien tilbake til hovedscriptet.

4.4.2.4 dboppl.dta

dboppl.dta er scriptet som innholder systemsvariabler og koblingen mot databasen. Grunnen til at vi valgte å samle systemvariablene i et script var for at installasjon og konfigurasjon/tilpassing til oppdragsgivers system skulle gjøres enklere.

Disse variablene er blant annet:

- startside (path'n til der hvor systemet er installert)
- sysadm (email-adresse til systemadministrator)
- fil_dir (path'n til der hvor vedlegg skal lastes opp)
- mail_adresse (path'n til hvor på systemet sendmail er installert)
- telefonkatalog (path'n til oppdragsgivers interne telefonkatalog)

Her gjøres også som sagt koblingen mot database hvor det angis server-, database-, passord og brukernavn til databasen.

Den viktigste variabelen her er $dboppl som innholder databasenavnet og databaseeieren (dbo).

4.4.2.5 htmlhead.dta

Htmlhead.dta er scriptet som innholder alle opplysninger som skal inn i head i html-siden på "Grensesnitt"-scriptene. Her inkluderes title, meta og css-style.

Dette scriptet ble laget uten å avslutte head. Dette fordi at vi skal kunne sette inn javascript-kode til feilsjekking i scriptene uten å måtte endre dette scriptet. Dette fører til at all javascript-kode og </head> blir kodet i de respektive scriptene.

4.4.2.6 cgi_lib.pl

Dette er en egen perl-modul som vi har lastet ned fra www.cgi-lib.berkley.edu for opplasting av filer fra en html-form og tilpasset vårt system. Modulen er copyright-beskyttet av

Copyright (c) 1993-1999 Steven E. Brenner

men det gis rettighet til bruk av dette så lenge alle endringer som er gjort blir dokumentert, copyright overholdt og at det opplyses hvem som har copyright på modulen og hvem som har utviklet den.
De endringene som er gjennomført er:

- cgi_lib'filnavn er lagt til får å kunne kjøre sjekk på medsendt fil i mottakerscript (ref til hovedscript)
- path'n til writefiles blir satt i hovedscriptet ved hjelp av en global variabel som er definert i dboppl.dta (4.4.2.4) (ref 2)
- sjekk på refnr. Må være med da dette scriptet blir brukt både på innlegging av ny hendelse(refnr generert i hovedscript) eller ved endring av hendelse(refnr er medsendt til hoved-script fra henendre.cgi ()). (ref 3)
- Refnr blir påført først i filnavnet før innlegging i database.(ref 4)
- Sjekk på om filnavn er medsendt for å unngå at bare refnr blir lagt inn i vedleggskatalogen.(ref 5)
- filstrengen blir fjernet fra filnavnet (c:\et\filnavn\her\ja.txt => ja.txt). Dette gjøres for både Linux- og Windows-strenger.(ref 6)

- Blanke felter i filnavnet(Windows) blir omgjort til underscore(_). (ref 7)
- Arrayen som returneres er gjort om til assosiativ array for lettere innlegging i database. (ref 8)
- Sjekk på om hovedscriptet er tilkalt med post-metode fra html-formen. (ref 9)

referanser i parantesene er plassert helt i begynnelsen av linjen over der hvor endringen er gjort. eks:
ref 1
endring er gjordt her.....

Dette er kopi av det som står i toppen av scriptet.

Denne modulen benyttes i dbEndreHendelse.cgi (4.4.3.1) og dbNyHendelse.cgi (4.4.3.2) for kunne laste opp filene som vedlegg til hendelsene. For å kunne benytte html-taggen ”File” må html-formen sin enctype være ”multipart/form-data” og det støtter ikke postinn.dta (4.4.2.1) scriptet vårt. Dermed måtte vi finne en annen metode for dette og vi ble enige om å bruke cgi_lib.pl scriptet til dette formålet. Dette fordi det var forholdsvis enkelt å endre filnavnet og å implementere med det vi allerede hadde.

Eneste endringen er at denne modulen returnerer arrayen ”%in” i stedet for ”%Field” som postinn.dta og getinn.dta returnerer.

4.4.3 Database-script

Felles for alle database-scriptene er at de inkluderer dboppl.dta for kobling mot databasen. Når det registreres en ny post i en tabell blir det brukt autonummer på id-feltene (for eksempel Lopenr i hendelsestabellen). Dette autonummeret er ikke helt stabilt i Sybase. Her er ikke stabilt ment som at det ikke fungerer, men for hver gang serveren startes på nytt blir autonummeret øket med 5 000 000. Dette er måten autonummer i Sybase er bygd opp på. Samtidig inkluderes postinn.dta (4.4.2.1) for mottak av data fra html-formen og dermed sjekking om scriptet mottar postforespørsel fra hovedscriptet. Hvis ikke det mottas postforespørsel skrives det ut en feilmelding om at brukeren ikke har tilgang til siden. Dvs at brukeren prøver å laste siden på feil måte.

4.4.3.1 dbEndreHendelse.cgi

Dette scriptet har som oppgave å endre en hendelse som allerede er lagt inn, dvs legge inn en ny hendelse med samme referansenummer i databasen. På denne måten blir ingen innlagte hendelser i databasen slettet eller endret.

I dette scriptet er cgi_lib.pl inkludert for innlasting gav filer. Så blir variabelen for hvor vedleggene skal lagres sendt med inn til cgi_lib.pl (4.4.2.6). Etter dette tilkalles funksjonen som parser input strengen fra html-formen.

Så sjekkes det returnerte filnavnet om det inneholder noe mer en bare referansenummeret til hendelsen. Hvis det kommer et noe mer etter refnr så betyr det at det er medsendt en fil og dette filnavnet blir lagret i databasen. Filen har på dette tidspunktet allerede blitt lastet opp til

vedleggskatalogen på serveren.

Etterpå sjekkes det om brukeren har krysset FAQstatus slik at meldingen skal vises for i oversikten. Hvis FAQstatus er 1 så settes FAQstatus til den endrede hendelsen til 0 og den nye får FAQstatus 1. Dermed vises den nye og den gamle forsvinner fra oversikten.
Så kommer sjekk på om den endrede hendelsen var registrert med Mail = 1. Det vil si at vedkommende bruker som registrerte hendelsen ønsker tilbakemelding når det skjer en endring på hendelsen.

Etter dette har vi valgt å kjøre sjekken på om brukeren vil legge ut hendelsen som varsling. Dette gjøres da med standardverdier på varighet (satt til 2 døgn, her bruker vi DATEADD()-funksjonen i Sybase), Faq settes til 1, Statusnr settes til 1 (Dette er den øverste i lista over statuser), Feilnr settes til 1 og Mail settes til 0 (dvs at brukeren ikke får mail ved endring).
Så kommer det sjekk om brukeren som registrerer hendelsen vil sende mail til den/de som er systemansvarlige. Her kjøres det en spørring som henter ut alle registrerte brukere fra systemansvarlige på det valgte systemet og mailadressen til disse legges så i en variabel skilt med komma.

Deretter registreres selve hendelsen i databasen. Vi har valgt å gjøre dette til slutt hvis det skulle oppstå en feil ved for eksempel sending av mail, oppdatering av gammel hendelse el. Da slipper man at det blir sendt mail til for eksempel systemansvarlige uten at hendelsen ble registrert.

Til slutt, før subrutinene, blir siden henoversikt.cgi (4.4.4.12) tilkalt og her blir det verdiene ok og refnr til hendelsen medsendt. Når henoversikt.cgi mottar ”ok” så kommer det opp en bekreftelse på at hendelsen er registrert og refnr til hendelsen.

Subrutinene som i dette scriptet er

- SysAns: denne subrutinen setter variablene $et (mottaker), $es (subjektet), $eb (innholdet i mailen) i mailen for så og tilkalle subrutinen Mail.
- Mail: denne bruker linux sin Sendmail til å sende mailen. Variabelen $mail_adresse innholder path’n til katalogen hvor Sendmail ligger og er angitt i scriptet dboppl.dta. Mail mottar hvem mailen skal sendes til, subjektet (tittel på mail) og body’n (innhodet i mailen) fra de andre subrutinene. Hvis ikke mail kan bli sendt blir subrutinen HTMLdie tilkalt for å gi brukeren en feilmelding.
- HTMLdie: Denne er beskrevet i 4.4.1.4.

4.4.3.2 dbNyHendelse.cgi

Dette scriptet har som oppgave å registrere en ny hendelse i Hendelse-tabellen i databasen. Dette er veldig likt dbEndreHendelse.cgi (4.4.3.1). Har derfor valgt å bare ta med forskjellene mellom dem.

Den største forskjellen er at vi gjør en spørring mot database for å finne refnr. Dette trengs ikke ved endring av en hendelse da refnr ikke skal endres. Vi har valgt å gjøre en spørring på dette isteden for å lagre det siste refnr i en variabel i databasen, fordi da er vi sikret at vi hele tiden får et refnr som er 1 større en det forrige. Fordi Sybase har dårlig måte å håndtere autonummer på (nevnt under 4.4.3), kunne vi heller ikke bruke det, dette ville blitt helt

meningsløst.

Vi sjekker heller ikke om det er medsendt et filnavn da dette er en helt ny hendelse og det ikke er en gammel hendelse med samme refnr å ta hensyn til.

Ellers er dette scriptet helt likt som dbEndreHEndelse.cgi.

Kildekode vedlagt, se 4.6.2.2.

4.4.3.3 dbGlemtPassord.cgi

Dette scriptet har som oppgave å generere et nytt passord og sende det til brukeren. På grunn av at passordet blir lagt kryptert inn i databasen så vil det ikke være mulig å hente det opp igjen for å sende det til brukeren. Derfor valgte vi å generere et tilfeldig passord og sende det til brukeren og samtidig oppdatere databasen.

Øverst i scriptet inkluderes postinn.dta og dboppl.dta. Så kommer en sjekk om scriptet mottar en post-forspørsel fra hovedscriptet. Hvis ikke, så vises en feilmelding.

Deretter leses tegn og bokstaver inn i en variabel som brukes til å generere det nye passordet. Til å generere det nye passordet brukes RAND()-funksjonen i perl for hente ut tilfeldige bokstaver/tegn fra strengen. Så krypteres det nye passordet og databasen blir oppdatert.
Så hentes mailadresse til brukeren opp og det sendes mail til vedkommende.

Subrutinene til dette scriptet er:

- HTMLdie: Denne er beskrevet i 4.4.1.4, men er utvidet med en ”ok”-knapp for å gi brukeren mulighet til å komme tilbake til LoggInn.cgi da dette scriptet er åpnet i et popupvindu uten verktøylinje.
- Mail:Dette er den samme som er brukt i 4.4.3.1.

4.4.3.4 dbEndrePassord.cgi

Dette scriptet har som oppgave å gi brukeren mulighet til å endre sitt eget passord. Scriptet mottar det gamle og det nye passordet fra html-formen, henter opp det eksisterende passordet til brukeren fra database og sjekker om dette stemmer med det brukeren tastet inn. Hvis passordet stemmer så krypteres det nye passordet, databasen oppdateres og brukeren får en melding om at passordet er oppdatert.

Hvis passordet ikke stemmer så gis det en feilmelding om dette og brukeren kan trykke seg tilbake til passordsiden.

Subrutiner til dette er:

- FeilPassord: Enkel html-side som genererer en feil melding om at passordet til brukeren var feil og en knapp for mulighet til å komme tilbake til passord.cgi.
- PassordOk: Enkel html-side som gir brukeren tilbakemelding om at passordet er opppdatert og en knapp slik at brukeren kommer tilbake til varoversikt.cgi.
- HTMLdie: Er beskrevet i 4.4.4.4.34.

4.4.3.5 dbNyVarsling.cgi

Dette scriptet har som oppgave å registrere en ny varsling i Hendelse-tabellen systemet. Script som blir inkludert er postinn.dta og dboppl.dta.

Først kommer en sjekk på om scriptet mottar post-forespørsel fra hovedscriptet. Hvis ikke får brukeren en feilmelding.

Dersom dette er i orden, sjekkes det om brukeren har krysset av for at han ønsker å få mail når det skjer en endring på denne varslingen, og feltet Mail i Hendelses-tabellen blir satt til 1.

Refnr blir generert på samme måte som i dbNyHendelse.cgi (4.4.3.2). Så settes varslingen inn i databasen. Vi har brukt DATEADD() til å bestemme varigheten på samme måte som i dbNyHendelse.cgi (4.4.3.2). Feltene Eksterntrefnr, Statuslosning og Aksjonspunkt blir satt blanke da dette er felter som ikke er i bruk for varsling. Når varslingen så er registrert i databasen blir det sjekket hvilke mail-alternativ brukeren har krysset av for.

Først sjekkes det om det skal sendes mail til oppdragsgivers feilmottak og i så fall blir subrutinen FeilMottak tilkalt og her blir tittel på varslingen datoen medsendt.
Så sjekkes det om brukeren har krysset av for feltet ”Sende mail til”. Her kan brukeren skrive inn en emailadresse som han/hun ønsker skal få melding om at denne varslingen er registrert. Her tilkalles subrutinen SendMail og det sendes med mottaker emailadresse og tittel på varslingen.

Så blir det sjekket om brukeren vil sende mail til alle som er registrert i maillisten for dette systemet. Scriptet gjør her en spørring mot Systemansvarlige-tabellen og plukker ut alle emailadressene som er registrert på dette systemet. Disse legges så fortløpende inn i en variabel skilt med komma og medsendt til subrutinen MailSjekk sammen med tittelen på varslingen. Før subrutinen MailSjekk tilkalt sjekkes det om variablelen som emailadressene ble lagt inn i er en tom streng. Dette for å unngå feil under sending av mail hvis det ikke er registrert noen i maillisten på det angitte systemet.

Når det gjelder sending av mail til systemansvarlige kan ikke brukeren velge dette. Dette gjøres uansett og det gjøres en spørring mot database for å plukke ut mailadressene til de som er registrert på det valgte systemet. Her gjøres det også en sjekk på om det er noen registrert som systemansvarlige på valgt system for å unngå feil ved sending av mail.

Til slutt, før subrutinene, blir siden varoversikt.cgi tilkalt med verdiene ok og $RF for å få varoversikt.cgi -siden til å gi brukeren tilbakemelding om at varslingen er registrert.

Subrutiner i dette scriptet er:

- Mail: Dette er samme subrutine som er brukt i 4.4.3.1. Alle subrutinene nedenfor unntatt HTMLdie bruker denne subrutinen. Avsender (bruker) på mailen sendes med fra hovedscriptet varreg.cgi (4.4.4.34).
- SysAns: Setter opp mailen som skal sendes til den/de som er systemansvarlige for det valgte systemet. Det blir alltid sendt mail til systemansvarlige for det valgte systemet. Sendingen av denne mailen har ikke brukeren mulighet til å bestemme.
- FeilMottak: Dette er en subrutine som setter opp mailen som skal sendes til oppdragsgivers feilmottak. Her blir alle feil registrert.
- SendMail: Denne setter opp mailen som skal sendes til den emailadressen som brukeren har skrevet inn i feltet ”Sende mail til” i varreg.cgi (4.4.4.34).
- MailSjekk: Denne setter opp mailen som skal sendes til maillisten knyttet til det valgte systemet. Mer om maillisten kommer i 4.4.4.26.
- HTMLdie: Er beskrevet i 4.4.1.4

4.4.3.6 dbEndreVarsling.cgi

Dette scriptet har som oppgave å endre en allerede registrert varsling i database. Dette på samme måte som i dbEndreHendelse.cgi (4.4.3.1) hvor det blir registrert en ny hendelse med endringene i stedet for å endre den eksisterende. Scriptet er veldig likt dbNyVarsling.cgi (4.4.3.5) og vi har valgt å kun kommentere forskjellen på de.

Selve registreringen av den nye er lik, men den gamle varslingen blir oppdatert der Faq settes til 0. Dette gjør for at det skal ikke vises varslinger med like refnr på varoversikt.cgi (4.4.4.33). Disse får samme refnr.

Det sjekkes om Mail-feltet i den gamle varslingen er satt til 1. I så fall hentes emailadressen til den brukeren som registrerte den gamle varslingen, fra databasen, og det sendes det mail til vedkommende om at varslingen er endret. For å sende denne mailen tilkalles subrutinen

EndretVarsling på samme måte som de andre subrutinene som setter opp mail (4.4.3.5).

Kildekode vedlagt, se 4.6.2.3.

4.4.3.7 dbNyBruker.cgi

Dette scriptet har som oppgave å registrere en ny bruker i systemet.

Her settes en tempvariabel til 1. Denne blir endret til 0 dersom spørringen mot databasetabellen Brukere finner en eksisterende bruker i tabellen med samme ansattnr. Da får brukeren beskjed om at det eksisterer en bruker med dette ansattnr. Til dette brukes subrutinen HTMLdie med en knapp på, slikk at brukeren får mulighet til å komme tilbake til NyBruker.cgi (4.4.4.18) og registrere seg på nytt.

Dersom brukeren ikke eksisterer i databasen krypteres passordet og brukeren blir lagt inn i databasetabellen Brukere. Scriptet varoversikt.cgi (4.4.4.33) blir tilkalt sammen med ansattnr

til den nye brukeren. Dette gjør at LoggInn.cgi (4.4.4.14) kommer opp med ansattnr til brukeren i Feltet ”Ansattnr”. Mer om dette i varoversikt.cgi (4.4.4.33).

4.4.3.8 dbRedBruker.cgi

Dette scriptet oppdaterer Brukere-tabellen i databasen med de endringer som redaktøren gjør på hver enkelt bruker. Her kjøres en update-spørring som oppdaterer brukeren ut fra Ansattnr som er medsendt fra hovedscriptet. Alle feltene unntatt Ansattnr blir oppdatert Dette skal aldri endres.

Til slutt tilkalles redBrukereListe.cgi (4.4.4.21).

Subrutinen som er brukt her er HTMLdie (4.4.1.4).

4.4.3.9 dbNyFeil.cgi

Dette scriptet setter inn en ny post i Feilkategori-tabellen i databasen. Den kjører en ”insert into”-spørring mot databasen og legger inn den nye feilkategorien i databasen.

Så tilkalles scriptet redfeilkategori.cgi (4.4.4.23). Dette er den siden dette scriptet ble tilkalt fra.

Subrutinen som er brukt her er HTMLdie (4.4.1.4).

4.4.3.10 dbNyLev.cgi

Dette scriptet er kodet på samme måte som dbNyFeil.cgi (4.4.3.9) bortsett fra at den nye posten blir satt inn i Leverandor-tabellen og scriptet som tilkalles er redleverandor.cgi (4.4.4.24).

4.4.3.11 dbNyMelKat.cgi

Dette scriptet er kodet på samme måte som dbNyFeil.cgi (4.4.3.9) bortsett fra at den nye posten blir satt inn i Meldingskategori-tabellen og scriptet som tilkalles er redmeldkategori.cgi (4.4.4.27).

4.4.3.12 dbNyStatus.cgi

Dette scriptet er kodet på samme måte som dbNyFeil.cgi (4.4.3.9) bortsett fra at den nye posten blir satt inn i Status-tabellen og scriptet som tilkalles er redstatus.cgi (4.4.4.28).

4.4.3.13 dbNySystem.cgi

Dette scriptet er kodet på samme måte som dbNyFeil.cgi (4.4.3.9) bortsett fra at den nye posten blir satt inn i Systemer-tabellen og scriptet som tilkalles er redsystem.cgi (4.4.4.29).

4.4.3.14 dbredmilliste.cgi

Dette scriptet er kodet på samme måte som dbNyFeil.cgi (4.4.3.9) bortsett fra at den nye posten blir satt inn i Maillistelinje-tabellen og scriptet som tilkalles er redmailliste.cgi (4.4.4.27).

4.4.3.15 dbEndreMelKat.cgi

Dette scriptet har som oppgave å endre en eksisterende post i Meldingkategori-tabellen. Det blir kjørt en ”update”-spørring på de feltene til den posten som skal endres. Id-nummeret på denne posten blir medsendt fra hovedscriptet.

Så tilkalles scriptet redmeldkategori.cgi (4.4.4.27). Dette er det samme scriptet (hovedscriptet) som dette scriptet ble tilkalt fra.

4.4.3.16 dbEndreLev.cgi

Dette scriptet er kodet på samme måte som dbEndreMelKat.cgi (4.4.3.15) bortsett fra at den endrer en post i Leverandor-tabellen og at scriptet som blir tilkalt er redlevarendor.cgi (4.4.4.24).

4.4.3.17 dbEndreSystem.cgi

Dette scriptet er kodet på samme måte som dbEndreMelKat.cgi (4.4.3.15) bortsett fra at den endrer en post i Systemer-tabellen og at scriptet som blir tilkalt er redsystem.cgi (4.4.4.29).

4.4.3.18 dbEndreStatus.cgi

Dette scriptet er kodet på samme måte som dbEndreMelKat.cgi (4.4.3.15) bortsett fra at den endrer en post i Status-tabellen og at scriptet som blir tilkalt er redstatus.cgi (4.4.4.28).

4.4.3.19 dbEndreFeil.cgi

Dette scriptet er kodet på samme måte som dbEndreMelKat.cgi (4.4.3.15) bortsett fra at den endrer en post i Feilkategori-tabellen og at scriptet som blir tilkalt er redfeilkategori.cgi (4.4.4.23).

4.4.3.20 dbEndrePers.cgi

Dette scriptet er kodet på samme måte som dbEndreMelKat.cgi (4.4.3.15) bortsett fra at den endrer dine egne data i Brukere-tabellen og at scriptet som blir tilkalt er personalia.cgi (4.4.4.20). Her sendes ditt eget ansattnr tilbake til personalia.cgi, slik at personalia.cgi åpnes med opplysningene om deg selv.

4.4.3.21 dbHentDetaljer.cgi

Dette skriptet har som oppgave å endre Synlig og Faq til en registrert hendelse. Dersom Synlig er satt til 0 blir hendelsen ”slettet” fra systemet. Det vil si at kun redaktør og systemansvarlige for det valgte systemet på hendelsen som har mulighet til å se denne hendelsen.

Dersom Faq er satt til 0 blir ikke hendelsen vist i oversikten på henoversikt.cgi.
Scriptet sjekker først om brukeren har valgt å ”slette” en hendelse. Da kjøres en ”update”-spørring mot databasen og setter Synlig = 0.

Så sjekkes det om brukeren har valgt ”gjenopprett”. Da kjøres en ”update”-spørring mot databasen og setter Synlig = 1.

Dersom brukeren har valgt ”endrefaq”, endres faqstatus til det motsatte av det hendelsen var registrert med.

Til slutt tilkalles scriptet henoversikt.cgi (4.4.4.12).

4.4.3.22 dbLoggInn.cgi

Dette scriptet har som oppgave å sjekke passordet til brukeren når han logger seg inn og opprette en cookie i nettleseren.

Først gjøres en spørring mot databasen som henter opp passord til det ansattnr som er medsendt fra LoggInn.cgi (4.4.4.14). Så fjernes alle ”space” fra det passordet som er hentet i databasen. Dette måtte gjøres fordi databasen legger på en space tilslutt i passordet. Det gikk med mye tid før vi oppdaget denne feilen som gjorde at vi ikke fikk passordet fra databasen til å stemme overens med det brukeren tastet inn. Hvorfor databasen gjør dette har vi ikke klart å finne ut av.

Så krypteres passordet fra brukeren og sjekkes opp mot det som ligger i databasen. Er disse like så tilkalles subrutinen setCookie, med verdien til cookien, som oppretter cookien.

Hvis ikke så kjøres HTMLdie.

Subrutiner som er brukt:

- setCookie: Denne genererer en html-side uten å vise noe på skjermen. Dette fordi vi bruker onload-funksjonen i html-formen til å kjøre en javascript-funksjon videre() som igjen tilkaller javascript-funksjonen setCookies() for å opprette cookien i nettleseren. SetCookies setter navnet, verdi og varigheten til cookien. Varigheten på cookien er satt til 8 timer fordi det tilsvarer en normal arbeidsdag. Hvorfor vi valgte å opprette cookien med javascript var fordi javascript har en enkel måte å sette/opprette en cookie på. Så oppdaterer vi hovedsiden til systemet. Dermed oppdaterer vi også menyframe'n slik at de linkene brukeren har rettigheter til blir vist.
  Når dette er gjort lukkes logginn-vinduet. Dermed er logginn-delen fullført.
- HTMLdie: denne er modifisert med en knapp som gjør at brukeren kan komme tilbake til LoggInn.cgi (4.4.4.14). Dette er nødvendig da dette scriptet kjøres i logginn-vinduet uten verktøylinje og mulighet for å bruke "Back"-tasten. Denne knappen vises kun vis feilen har oppstått pga at brukeren har tastet feil passord eller brukernavn. Den andre situasjonen som gjør at denne subrutinen blir kjørt er at scriptet ikke tilkalles med postforespørsel. Da valgte vi og ikke vise knappen. Dette gjør blir brukeren blir nødt til å lukke vinduet for å komme videre.

4.4.3.23 dbRedFramside.cgi

Dette scriptet har som oppgave å slette de gamle dataene om forsiden og legge inn de nye. Grunnen til at vi sletter alle data fra Framside-tabellen er fordi det aldri skal ligge mer enn en post i denne tabellen. Vi valgte å lage en tabell i databasen i stedet for å lagre opplysningene i en fil. Dette er mer oversiktlig og enklere å administrere.

Først slettes alle data fra Framside-tabellen med en "delete"-spørring. Så legges de nye opplysningene inn i tabellen igjen ved hjelp av en "insert into"-spørring.

Til slutt tilkalles scriptet varoversikt.cgi.

4.4.3.24 dbSlett.cgi

Dette scriptet har som oppgave å slette en systemansvarlig fra Systemansvarlige-tabellen. Det gjøre ved å kjøre en "delete"-spørring mot denne tabellen og slette alle postene med et bestemt systemnr og ansattnr. Ansattnr og systemnr kommer fra redsystem.cgi (4.4.4.29) som dette scriptet er tilkalt fra.

Til slutt tilkalles redsystem.cgi (4.4.4.29) med systemnr slik at det valgte systemet blir markert i systemoversikten i redsystem.cgi.

4.4.3.25 dbSystemansvarlig.cgi

Dette scriptet har som oppgave å legge til en ny systemansvarlig på et system i Systemansvarlige-tabellen. Dette scriptet mottar et systemnr og et ansattnr fra redsystem.cgi (4.4.4.29).

Så sjekkes det om det finnes en post med samme ansattnr og systemnr i tabellen fra før. Hvis den finnes, avsluttes scriptet og scriptet redsystem.cgi blir tilkalt med systemnr og verdien 1. verdien 1 blir medsendt får at redsystem.cgi skal gi brukeren melding om at denne brukeren allerede er registrert på dette systemet.

Dersom den ikke er registrert blir det kjørt en ”insert into”-spørring mot databasen.

Til slutt tilkalles scriptet redsystem.cgi med systemnr som medsendt verdi for å vise hvilket system redaktøren jobber på. Dette blir da det aktive systemet i redsystem.cgi.

4.4.3.26 dbVarDetaljer.cgi

Dette scriptet har som oppgave å avslutte en varsling. Det vil si å gjøre om en varsling til hendelse. Her settes Varslinghendelse til 1 og de feltene som er i bruk i hendelse men ikke i varsling gis standardverdier.

4.4.4 ”Grensesnitt”-script

4.4.4.1 index.cgi

Dette scriptet genererer oppsettet for websiden til systemet. Den deler opp websiden i to frame (deler). Disse to kalles ”leftframe” og ”mainframe”. Det er ikke definert noen ”border”-tykkelse mellom framene. Antall rows er satt til ubegrenset.

Leftframe er 100 kolonner bred og er plassert til venstre på websiden. Her er menyframe (4.4.4.2) plassert.

Mainframe er 592 kolonner bred og er den framen hvor alle scriptene nedenfor blir generert og vist. De eneste scriptene som ikke blir generert her er de som åpnes i et popupvindu. Dette er nærmere forklart i de scriptene det gjelder.

Dersom dette scriptet mottar verdi 1 via get så reloades varoversikt.cgi (4.4.4.33) i mainframe og menyframe.cgi (4.4.4.2) i leftframe. Her blir verdien 1 sendt med til menyframe.cgi (se 4.4.4.2, andre avsnitt, for nærmere forklaring).

Er verdien 2 mottatt så betyr det at en ny bruker skal registrere seg og NyBruker.cgi (4.4.4.18) vises i mainframe slik at brukeren kan registrere seg.

Hvis scriptet ikke mottar noen av disse verdiene så genereres varoversikt.cgi i mainframe og varoversikt.cgi i leftframe.

4.4.4.2 menyframe.cgi

Dette er et scriptet som bestemmer hvilke linker brukeren skal få tilgang til ut fra de rettighetene vedkommende har. Linkene refererer til alle hovedsidene (grensesnitt-script) i systemet, en link for hver side. Linker til sok og avansertsok.cgi er synlige for alle brukerne. Først hentes rettighetene til innlogget bruker opp fra Brukere-tabellen. Er ikke brukeren logget inn settes rettighetene til 0. Siden er bygd opp ved hjelp av if/else spørringer som bestemmer hvilke linker bruker skal få tilgang til.

Hvis dette scriptet blir oppdatert med variabelen 1 vil siden åpne LoggInn.cgi i et eget lite popupvindu. Dette skjer når en bruker kommer til en side og ikke har adgang til denne. Da får han mulighet til å logge seg inn via en link. Denne linken oppdaterer index.cgi med variabelen 1 som igjen oppdaterer menyframe med 1.

4.4.4.3 sok.cgi

Dette scriptet genererer siden som gjør en ”select”-spørring mot databasen og viser resultatet av denne i en tabell. I denne spørringen brukes ”%søketekst%” for å få treff på alle ord som inneholder søketeksten. Feltene det søkes i er Lopenr, Refnr, Tittel, Statusnavn, Regdato, Ansattnr ,Etternavn, Fornavn, Varslinghendelse, Systemnavn og Systemlink. For å unngå feil i sql-setnignen søkes det i Refnr bare hvis søketeksten består av tall. Til å sjekke dette bruker vi et regulært uttrykk.

Er meldingen i tabellen en hendelse, er refnr på meldingen en link til hendetaljer.cgi (4.4.4.9) ellers er refnr en link til vardetaljer.cgi (4.4.4.31).

4.4.4.4 avansertsok.cgi

Denne siden tilkaller seg selv når det trykkes på submit eller nullstillknapp. Informasjon om søketekst og innstillinger blir da overført vha post. Søketekst, dropdownbokser og avkrysningsbokser blir da tegnet opp på nytt, med valgte innstillinger. Kategori 50 gis mulighet til å angi om slettede meldinger skal vises. Select-spørring som kjøres mot database bygges opp linje for linje, avhengig av søkekriterier. Dersom brukeren krysser av for søk på refnr, som er en tallverdi, brukes et regulært uttrykk for å sjekke om søkestrengen består av kun tall og om det er skrevet inn noe i søkestrengen. Dette for å unngå feil under søkingen. Tom streng og andre tegn enn tall vil føre til at select-spørringens oppbygningen blir feil. Resultatet vil bli vist i en tabell. Hvis det ikke blir treff i søkingen så vises ”Ingen treff”.

Kildekode vedlagt, se4.6.2.1.

4.4.4.5 bekreftelse.cgi

Dette er et script som blir åpnet i lite popupvindu når en varsling/hendelse er lagt inn i databasen. Denne forteller brukeren at varslingen/hendelsen er registrert og hvilket refnr den fikk. Siden inneholder også en knapp for å lukke vinduet. Dette kodet slik at det vil bestandig bli liggende på toppen av de andre vinduene.

4.4.4.6 brukerinfo.cgi

Dette scriptet viser info om en bruker. Infoen om brukeren blir skrevet ut i en enkel tabell hvor dataene hentes opp fra databasen ved hjelp av en ”select”-spørring.

Får å finne telefonnummeret til vedkommende bruker er det lagt inn en link til oppdragsgivers telefonkatalog. Denne linken er formatert slik at det i linken sendes med fornavn og etternavn og hvilken type spørring det kjørers mot telefonkatalogens database.

4.4.4.7 feilkategoriinfo.cgi

Dette scriptet genererer siden som viser en oversikt over alle feilkategorier og beskrivelesene av disse. Disse hentes fra Feilkategori-tabellen.

Siden inneholder en link slik at brukeren kommer tilbake til forrige side.

4.4.4.8 glemtpassord.cgi

Dette scriptet genererer siden som brukes når bruker har glemt passordet. Her skal bruker taste inn ansattnr sitt. Klikker man på send, tilkalles dbGlemtPassord.cgi (4.4.3.3).

4.4.4.9 hendetaljer.cgi

Dette scriptet har som oppgave å liste opp alle detaljer på en bestemt hendelse.

Det kjøres to spørringer mot databasen, en for å hente rettighetene til innlogget bruker og en for å hente opp dataen om varslingen. Spørringen som henter rettighetene blir gjort for å bestemme hvilke linker som skal vises til brukeren. Alle med rettighet 40 og oppover får opp en link for å endre hendelsen. Denne linken tilkaller scriptet henendre.cgi (4.4.4.10) og her sendes det med Lopenr til hendelsen.

Redaktøren har mulighet til å slette en hendelse og gjøre om Faq-status. Når redaktøren utfører en av disse to mulighetene tilkalles dbHenDetaljer.cgi (4.4.3.21).

På siden vises en link til vedlegget. Trykker brukeren på denne linken åpnes vedlegget i hovedframen.

I toppen av scriptet skrives det ut hvem som har registrert hendelsen, refnr og dato. Dette er gjort bevisst for å få dette fremhevet.

4.4.4.10 HenReg.cgi

Dette scriptet genererer siden hvor brukerne med rettigheter 10 eller mer kan registrere en ny hendelse.

Det kjøres 3 spørringer mot tabellene Systemer, Meldingskategori og Leverandorer. Disse henter opp data til bruk i dropdownboksene. På spørringen som kjøres mot Meldingskategori vil ikke den første posten som leses opp fra databasen vises i dropdownboksen. Dette er fordi denne posten brukes bare til varslinger når disse endres til hendelse.

Det er mulig for bruker å legge ved et vedlegg. Dette fører til at vi må endre ENCTYPE i html-formen til ” multipart/form-data” for å kunne laste opp filer.

Når alt er fylt ut og bruker lagrer hendelsen kjøres en javascript-kode som sjekker om tittel og beskrivelse er utfylt og om det er valgt elementer fra dropdownboksene. Er dette i orden tilkalles dbNyHendelse (4.4.3.2), hvis ikke får bruker feilmelding om hva som ikke er utfylt.

4.4.4.11 henendre.cgi

Dette scriptet genererer siden hvor brukere med rettighet 40 eller over kan endre hendelser som er lagt inn i databasen. Denne siden er veldig lik HenReg.cgi (4.4.4.10) og vi har valgt og bare kommentere forskjellen mellom dem. Øverst i scriptet skrives det ut Refnr, hvem som registrer denne hendelsen og datoen når dette blir gjort (dvs deg selv og dagens dato), hvem som registrerte den gamle hendelsen og datoen for dette. Dette er gjort for å få dette fremhevet. Det er en link til brukerinfo.cgi (4.4.4.6) for se opplysninger om den som har registrert den gamle hendelsen.

Først hentes det opp alle opplysninger for den valgte hendelsen. Så kjøres spørringer mot de samme tabellene som i HenReg.cgi men i tillegg mot Status. Dette fordi at brukeren skal få mulighet til å endre status på hendelsen. De verdiene som meldingen har blir satt som default-verdier i dropdownboksene.

4.4.4.12 henoversikt.cgi

Dette scriptet genererer siden som inneholder oversikt over alle hendelser med Faqstatus 1.

Så skrives tabellen med hendelsene ut i intervaller på 20 og 20. Når brukeren trykker på pilen for å få de neste 20 genereres siden på nytt.

Hvis dette scriptet tilkalles med ok som verdi (dvs henoversikt.cgi?ok), er det registrert en ny eller endret en hendelse. Da blir scriptet bekreftelse.cgi (4.4.4.5) åpnet i et popupvindu for å gi brukeren en bekreftelse på at hendelsen er registrert og hvilket refnr denne fikk.

4.4.4.13 hentrapport.cgi

Dette scriptet har som oppgave å generere siden hvor brukerne kan velge hvilken type rapport de ønsker. Første velger man mellom leverandør, system eller status. Deretter genereres siden på nytt og det velges videre ut fra dette f.eks hvilken type leverandør, system eller status man ønsker rapport om. Javascript brukes til å tilkalle henholdsvis levrap.cgi, systemrap.cgi eller statrap.cgi (se 4.4.4.30) ut fra det som er valgt.

4.4.4.14 LoggInn.cgi

Scriptet genererer siden for pålogging av brukere. Denne siden åpnes i et lite popupvindu. Er ikke feltene ansattnr og passord utfylt, vil bruker få melding om dette. Er alt i orden tilkalles scriptet dbLoggInn.cgi (4.4.3.22) ved hjelp av javascript.

Avbryt-knappen lukker vinduet uten noen forandringer, mens knappen glemt passord åpner siden glemtpassord.cgi (4.4.4.8) i samme vindu. Knappen ny bruker lukker gjeldene vindu og tilkaller scriptet NyBruker.cgi (4.4.4.18) i hovedframen.

4.4.4.15 LoggUt.cgi

Scriptet genererer siden hvor bruker må bekrefte at han/hun logger vil logge seg ut av systemet. Når brukeren logger seg ut av systemet, slettes verdien i cookien og popupvinduet med LoggUt.cgi lukkes. Så oppdateres varoversikt.cgi (4.4.4.33) og menyframe.cgi (4.4.4.1). Velger man avbryt lukkes vinduet.

4.4.4.16 meldingsinfo.cgi

Dette scriptet generer siden hvor brukere kan få en beskrivelse av meldingskategoriene. Siden viser en tabell med alle kategorier og en beskrivelse av disse. Disse hentes opp fra Meldingskatergori-tabellen ved hjelp av en ”select”-spørring.

Siden inneholder en link slik at brukeren kommer tilbake til forrige side.

4.4.4.17 minesystemer.cgi

Dette scriptet genererer siden hvor systemansvarlige får en oversikt over alle meldinger som omhandler deres systemer. Her kjøres det en ”select”-spørring mot databasen som henter opp meldingene. Er meldingen en hendelse, er refnr på meldingen en link til hendetaljer.cgi (4.4.4.9) ellers er refnr en link til vardetaljer.cgi (4.4.4.31).

4.4.4.18 NyBruker.cgi

Dette scriptet brukes til å legge inn personalia til en ny bruker. Her brukes javascript-kode for å sjekke ut at alle feltene er fylt ut og at verdien i disse er gyldig. Ansattnr må ha en lengde på 6 tegn, passordet må skrives inn likt to ganger og være på minst 6 tegn. Er alt riktig tilkalles scriptet dbNyBruker.cgi (4.4.3.7).

Velges avbryt genereres varoversikt.cgi (4.4.4.33) i mainframe.

4.4.4.19 passord.cgi

Scriptet genererer siden hvor brukeren kan endre sitt passord. Her brukes javascript-kode til å sjekke lengde på passordene, minst 6 tegn langt, og om det nye passordet er tastet inn likt to ganger. Hvis noe er feil, vil bruker få en feilmelding. Er de tastet inn riktig tilkalles scriptet dbEndrePassord.cgi (4.4.3.4).

Velges avbryt genereres varoversikt.cgi (4.4.4.33) i mainframe

4.4.4.20 personalia.cgi

Dette scriptet genererer siden hvor bruker kan forandre personaliene sine. Her kjøres det en ”select”-spørring mot Brukere-tabellen. Det er mulig å endre på alle feltene i tabellen, unntatt ansattnr. Før endret data legges i databasen kjøres javascript-kode for å sjekke om det er fylt inn noen verdier i feltene fornavn og etternavn og om mailadressen er gyldig. Hvis noe er feil, får bruker feilmelding om hva som er galt. Ellers så tilkalles scriptet dbEndrePers.cgi (4.4.3.20).

4.4.4.21 redBrukerListe.cgi

Dette scriptet genererer en side som viser redaktør en tabell over brukere på systemet. Her kjøres en ”select”-spørring mot Brukere-tabellen. Ansattnr er en link til siden redbrukere.cgi (4.4.4.22).

4.4.4.22 redbrukere.cgi

Scriptet genererer siden hvor redaktøren har mulighet til å endre personalia og rettigheter til valgt bruker på systemet samt gi denne en kommentar. Dette kommentarfeltet har ikke brukerne adgang til.

Denne brukerens verdier hentes ut fra Brukere-tabellen vha en ”select”-spørring. Det er mulig å gjøre forandringer på alle feltene unntatt ansattnr. Dersom brukeren er root er det ikke mulig å endre rettighetene til vedkommende. Før man legger dataene inn i databasen vil

feltene bli sjekket av javascript-kode om de er fylt ut riktig og om de er gyldige. Når dette er i orden tilkalles scriptet dbRedBruker.cgi (4.4.3.8).

4.4.4.23 redfeilkategori.cgi

Scriptet genererer siden hvor brukere med brukerkategori 40 eller høyere har mulighet til legge til nye eller endre feilkategorier. All data hentes fra Feilkategori-tabellen.. Når et element i listen velges, genereres siden på nytt for å hente opp opplysningene til denne feilkategorien og de vises i endre-feltene på siden. Så kan bruker gjøre endringer i disse feltene og tilkalles scriptet dbEndreFeil.cgi (4.4.3.19). Før dette scriptet blir kjørt gjøres en sjekk med javascript-kode på om det er valgt et element fra listen.

Velger man å legge inn et nytt system, gjøres dette i feltene under. Før man tilkaller scriptet dbNyFeil.cgi (4.4.3.9), kjøres javascript-kode for å sjekke om det er lagt inn verdi i ny kategori feltet.

4.4.4.24 redframside.cgi

Dette scriptet genererer siden som brukes for å legge inn en ny framside på systemet (Figur 3-1). Her hentes all data fra tabellen Framside. Alle poster som settes inn i denne tabellen får ID=1. Dermed vil redaktøren få en feilmelding dersom ikke dataene som ligger i tabellen blir slettet og vi unngår å få flere enn en post i denne tabellen. Før ny data legges i databasen, kjøres javascript-kode for å sjekke om feltet hovedoverskrift er utfylt. Hvis dette er i orden, tilkalles scriptet dbRedFramside.cgi (4.4.3.23). Hvis ikke får bruker en feilmelding.

4.4.4.25 redleverandor.cgi

Scriptet genererer siden slik at bruker med rettighet 40 eller høyere kan legge inn ny eller endre data i Levrandorer -tabellen. All data fra denne tabellen hentes ut vha en ”select”-spørring. Disse vises i en liste. Merker man et element i listen, vil siden genereres på nytt med de valgte dataene. Før endringene legges i databasen, sjekker en javascript-kode om det er valgt en leverandør. Er dette i orden tilkalles scriptet dbEndreLev.cgi (4.4.3.16), hvis ikke vil man få melding.

Ved innlegging av ny leverandør brukes javascript-kode til å sjekke at det er tastet inn et navn på leverandøren, hvis ikke får bruker feilmelding. Hvis dette er i orden tilkalles scriptet dbNyLev.cgi (4.4.3.10).

4.4.4.26 redmailliste.cgi

Scriptet generer en side som skal gi mulighet for brukere med brukerrettigheter 40 eller høyere, til å legge til mailliste for et valgt system eller slette brukere fra maillisten. En ”select”-spørring henter alle poster fra Systemer- tabellen sortert på systemnavn. Brukeren

velger så et system og scriptet genererer siden på nytt med det valgte systemet. Dersom brukeren ikke velger et system skrives det ut en feilmelding.

Så velger brukeren en ansatt og legger til denne. Er vedkommende allerede i maillisten til dette systemet , får brukeren melding om dette.

Når man har valgt et system kan man også slette en ansatt fra listen. Dette gjøres ved at brukeren velger en ansatt fra maillistelinjen. Ansattnr til den ansatte ligger som value i listen og dette sammen med systemnr blir brukt når ”delete”-spørring mot Mailliste-tabellen kjøres.

4.4.4.27 redmeldkategori.cgi

Scriptet genererer siden hvor brukere med brukerkategori 40 eller høyere kan legge til ny eller endre meldingskategori. ”Select”-spørringen henter all data fra Meldingskategori-tabellen. Merker man et av elementene i listen som viser meldingskategoriene genereres siden på nytt med valgt verdi. Før endringer legges i databasen brukes javascript-kode til å sjekke om man har valgt et element fra listen. Hvis dette er i orden tilkalles scriptet dbEndreMelKat.cgi (4.4.3.15), hvis ikke får bruker melding om å velge et element fra listen.

Når brukeren skal legge til en ny meldingskategori kjøres javascript-kode for å sjekke at brukeren har lagt inn navn på meldingskategorien og scriptet dbNyMelKat.cgi (4.4.3.11) kjøres. Er ikke feltet utfylt får bruker feilmelding.

4.4.4.28 redstatus.cgi

Dette scriptet genererer siden hvor redaktøren kan til å legg til ny eller endre poster i Status-tabellen. Alle poster fra Status tabellen hentes vha en ”select”-spørring.

For å endre en status må det velges et element fra listen. Når det trykkes på et element i listen, brukes javascript-kode til å overføre verdien på dette elementet til feltet Endre navn. Er dette feltet tomt når brukeren trykker på endre, vil systemet gi brukeren en melding om at han må velge et element fra listen.. For å lagre endringer tilkalles scriptet dbEndreSatus.cgi (4.4.3.18).

Når brukeren skal legge til en ny status sjekker javascript-koden feltet om det er lagt til navn på statusen før scriptet dbNyStatus.cgi (4.4.3.12) tilkalles. Er det feltet ikke fylt ut vil bruker å en feilmelding.

4.4.4.29 redsystem.cgi

Scriptet genererer siden hvor redaktør kan legge til nye eller endre systemer samt legge til eller slette systemansvarlige for et valgt system. Alle poster i Systemer-tabellen hentes vha en ”select”-spørring.

Først må redaktør velge et element i listen over systemene. Scriptet vil da generere siden på nytt med det valgte systemet. Redaktøren kan nå gjøre endringer og tilkalle scriptet dbEndreSsytem.cgi (4.4.3.17) for å lagre endringene.

Skal brukeren legge til en systemansvarlig velges en ansatt fra liste og det kjøres en ”insert into”-spørring, med systemnr og ansattnr, på Systemansvarlige-tabellen når redaktøren trykker på systemansvarlige. Er det valgt et system tilkalles scriptet dbSystemansvarlig.cgi (4.4.3.25) ved hjelp av javascript-kode. Er det ikke valgt et system, gir javascript-koden redaktør en feilmelding.

Det er også mulig å slette en systemansvarlig. Dette gjøres ved å merke en systemansvarlig i listen, tilkalle scriptet dbSlett.cgi (4.4.3.24).

Får å legge til et nytt system tilkalles scriptet dbNySystem.cgi (4.4.3.13). Dette gjøres vha javascript-kode og det sjekkes samtidig om nødvendige felter er utfylt. Hvis ikke vises en feilmelding.

4.4.4.30 levrap.cgi, systemrap.cgi og statrap.cgi

Disse scriptene genererer siden hvor rapporten vises. Disse tilkalles av hentrapport.cgi (4.4.4.13), og skriver ut rapport om det som er valgt i denne siden. ”Select”-setningene bygges opp i de respektive scriptene og den henter bare ut poster hvor Synlig og Faq =1.

4.4.4.31 vardetaljer.cgi

Dette scriptet har som oppgave å liste opp alle detaljer på en bestemt varsling.

Det kjøres to spørringer mot databasen, en for å hente rettighetene til innlogget bruker og en for å hente opp dataen om varslingen. Spørringen som henter rettighetene blir gjort for å bestemme hvilke linker som skal vises til brukeren. Alle med rettighet 30 og oppover får opp en link for å endre varslingen. Denne linken tilkaller scriptet varendre.cgi (4.4.4.32) og her sendes det med Lopenr til varslingen. Brukere med rettighet 40 eller over får mulighet til å avslutte varslingen, dvs å gjøre den om til hendelse ved å tilkalle dbVarDetaljer.cgi (4.4.3.26).

I toppen av scriptet skrives det ut hvem som har registrert varslingen, refnr og dato. Dette er gjort bevisst for å få dette fremhevet.

4.4.4.32 varendre.cgi

Dette scriptet har som oppgave å vise brukeren opplysninger om en valgt varsling og mulighet til å endre denne. Øverst i scriptet skrives det ut Refnr, hvem som registrer denne varslingen og datoen når dette blir gjort (dvs deg selv og dagens dato), hvem som registrerte den gamle varslingen og datoen for dette. Dette er gjort for å få dette fremhevet. Det er en link til brukerinfo.cgi (4.4.4.6) for se opplysninger om den som har registrert den gamle varslingen.

Det gjøres spørring mot databasen for å hente opp varslingen og data om brukeren, samt hente inn alle systemer, feilkategorier og statuser til dropdownboksene. Alle verdiene til den valgte varslingen blir satt som default-verdier i de respektive feltene i html-formen.

Det brukes 4 hidden-felter i html-formen til å ta vare på Lopenr, Refnr, Ansattnr og emailadressen til brukeren.

Javascriptkoden har 2 funksjoner:

- sjekk: Denne sjekker at alle påkrevde felter er utfylt og at gyldig emailadresse er oppgitt i emailfeltet hvis dette er avkrysset før formen submittes til scriptet dbEndreVarsling.cgi (4.4.3.6).
- avbryter: Denne avslutter dette scriptet og åpner varoversikt.cgi (4.4.4.33).

4.4.4.33 varoversikt.cgi

Dette scriptet genererer hovedsiden til systemet har som oppgave å vise informasjon om systemet og gi brukeren en oversikt over alle varslinger med Faqstatus 1. Se Figur 3-1 for å få oversikt over denne siden.

Informasjonen om systemet vises i 3 like bokser i toppen av scriptet. Her leses det inn informasjon fra tabellen Framside. Dersom overskriften eller innholdet til en av boksene ikke en lagt inn i tabellen Framside vil ikke denne boksen vises.

Så skrives tabellen med varslingene ut i intervaller på 20 og 20. Når brukeren trykker på pilen for å få de neste 20 genereres siden på nytt.

Hvis dette scriptet tilkalles med ok som verdi (dvs varoversikt.cgi?ok), er det registrert en ny eller endret en varsling. Da blir scriptet bekreftelse.cgi (4.4.4.5) åpnet i et lite vindu for å gi brukeren en bekreftelse på at varslingen er registrert og hvilket refnr denne fikk.

Blir dette scriptet tilkalt med verdi større en 1 så åpnes dette på samme måte men med scriptet LoggInn.cgi i stedet og brukere kan logge inn. Dette skjer når en bruker registrerer seg for første gang og Ansattnr sendes til LoggInn.cgi og blir lagt inn i Ansattnr-feltet. Dette er gjort for å gjøre systemet mer brukervennlig.

4.4.4.34 varreg.cgi

Dette scriptet genererer siden hvor brukerne registrerer nye varslinger. Øverst på dette scriptet vises hvem som registrerer denne varslingen og datoen når varslingen registreres.
Det kjøres en spørring mot tabellene Systemer. Disse henter opp data til bruk i dropdownboksen.

Når alt er fylt ut og bruker lagrer varslingen kjøres en javascript-kode som sjekker om tittel og beskrivelse er utfylt og om det er valgt et element fra dropdownboksen. Det kjøres også en sjekk på om mailadressen i avkrysningsfeltet, send mail til, er gyldig. Er dette i orden tilkalles dbNyVarsling.cgi (4.4.3.5), hvis ikke får bruker feilmelding om hva som ikke er utfylt

Kildekode vedlagt, se 4.6.2.5.

4.4.5 Vedlikeholds-script

4.4.5.1 endreVarHen.pl

Dette scriptet kjøres en gang i døgnet og sjekker hvilke varslinger som skal gjøres om til hendelse. Er differansen mellom varslingens varighet og dagens dato 0 eller mindre gjøres varslingene om til hendelse. Til å finne ut differansen på datoene bruker vi DATEDIFF() i Sybase. Denne tar to datoer og hvilken del av datoen differansen skal sjekkes på som input (argumenter). Den returnerer et heltall (differansen) og er derfor enkel å bruke. I og med at det er noen felter i varslinger, som ikke er med i hendelser og omvendt, blir feltene i hendelsen satt inn med standardverdier.

Spørringen som kjøres er vist under:

"UPDATE $dboppl"."Hendelser SET Varslinghendelse=1, Levnr=1, Kategorinr=1, Varighet=" WHERE Varslinghendelse=0 AND DATEDIFF(dd, GETDATE(), Varighet) <= 0 AND Varighet not like "".

4.4.5.2 transactiondump.pl

Dette scriptet dumper transaksjonsloggen til databasen. I kravspesifikasjonen er det ikke beskrevet noe spesielt om hvordan dette skal gjøres så vi har bare laget et forslag til script på hvordan dette kan gjøres.

4.4.5.3 databasedump.pl

Dette scriptet dumper databasen. I kravspesifikasjonen er det heller ikke beskrevet noe spesielt om hvordan dette skal gjøres, så vi har laget et forslag til script her også på hvordan dette kan gjøres.

4.5 Ordforklaringer

- Cookie: Dette er et id-felt som legges til i nettleseren for å ta vare på informasjon. Dette identifiseres med navn, verdi og varighet i dette tilfellet.
- DBD: Står for DataBaseDriver. Denne jobber mellom DBI og databasen. Denne er spesifikk for hver enkelt database.
- DBI: Står for DataBase Interface. Dette er perl sin modul for å jobbe mot databasen.
- Get: Dette er også en måte å overføre data mellom websider på, men her blir dataene sendt via "Location" i nettleseren og de blir derfor synlige for bruker.
- HTML-form: Er en webside som inneholder felter hvor bruker kan legge inn data.
- Post: Dette er en måte å overføre data mellom websider uten at dataene er synlige.

4.6 Eksempler fra kildekode

4.6.1 Koder for bruk under utvikling

4.6.1.1 konverter.pl

#!/usr/bin/perl

```
open (INNFIL, $ARGV[0]);
open (UTFIL, ">$ARGV[1]");
print UTFIL "#!/usr/bin/perl\n";
print UTFIL "print\"Content-type: text/html\\n\\n\";\n";
print UTFIL "print\"\n";
while (<INNFIL>) {
  chomp($_);
  $linje=$_;
  $linje=~tr/"/'/;
  $linje=~s/\.html/.cgi/;
  $linje=~s/\.htm/.cgi/;
  print UTFIL "$linje\n";

}
print UTFIL"\"\;";
close(INNFIL);
close(UTFIL);
```

4.6.2 Kildekode

4.6.2.1 avansertsok.cgi

Dette scriptet er tatt med p.g.a:

- Et av få script som tilkaller seg selv.
- Viser hvordan tabellene med varslinger/hendelser er satt opp.
- Viser hvordan brukere med forskjellige rettigheter får tilgang til data.
- Oppbygning av avansert sql-setning.

```
#!/usr/bin/perl
#Testet 15.05.01 TH, IB
###################Brukergrensesnitt og funksjoner for
avansertsøk################

require 'getcooki.dta';
require 'dboppl.dta';
require 'postinn.dta';
require 'htmlhead.dta';

print"<script language='javascript'>
function forrige(form, mi, ma){   //funksjoner for å vise kun
50 linjer pr side
  document.sok.Min.value = mi;
  document.sok.Max.value = ma;
  document.sok.submit();
}
function neste(form, mi, ma){
  document.sok.Min.value = mi;
  document.sok.Max.value = ma;
  document.sok.submit();
}
function link(side){                // åpner vindu for
systeminfo og leverandørinfo
  popupWin = parent.window.open(side, 'link');
}
</script></head>";

                                                        #henter
rettigheter til bruker
$sqlbruker = $forb->prepare("select Rettigheter from
$dboppl"."Brukere WHERE Ansattnr='$nr'");
$sqlbruker->execute || die {&FeilMelding("Får ikke rettigheter
til bruker","100") };
@SqlBruker = $sqlbruker->fetchrow;

$Ant = 0;                          #variabler som brukes for å
vise 50 linjer pr side
$Min = 0;
$Max = 0;

$Antvises = 50;
```

```
if($Field{'Min'}){
$Min = $Field{'Min'};
}
$Max = ($Field{'Min'} + $Antvises);
$Neste = ($Field{'Max'} + $Antvises);

if($Field{'nullstill'}){%Field=-1;}      #sletter variabler
hvis nullstillknapp er trykt
print"
<body bgcolor='#FFFFFF'
onload='document.sok.soketekst.focus();'>
<h2>Avansert søk</h2>
Du blar deg framover eller bakover med pilene under tabellen.
<hr width='100%'>
<form name='sok' method='post' action='avansertsok.cgi'>
  <table border='0' width='701'>
    <input type='hidden' name='Min' value=$Min>
    <input type='hidden' name='Max' value=$Max>
    <tr>
      <td colspan='4'>Søk: <input type='text' name='soketekst'
size='50' value='$Field{'soketekst'}'> <input type='submit'
name='Sok' value='Søk'></td>
    </tr>
    <tr>
      <td colspan='4'>";
                                #tegner checkboxer med post-
verdier.
      if($Field{'system'}) {
     print"<input type='checkbox' name='tittel' value='1'";
        if($Field{'tittel'}){print" checked";}
        print">
        Tittel
        <input type='checkbox' name='beskrivelse' value='1'";
        if($Field{'beskrivelse'}){print" checked";}
        print">
        Beskrivelse
        <input type='checkbox' name='statuslosning' value='1'";
        if($Field{'statuslosning'}){print" checked";}
        print">
        Status/l&oslash;sning
        <input type='checkbox' name='aksjonspunkt' value='1'";
        if($Field{'aksjonspunkt'}){print" checked";}
        print">
        Aksjonspunkt
        <input type='checkbox' name='eref' value='1'";
        if($Field{'eref'}){print" checked";}
        print">
        Eksternt refnr
        <input type='checkbox' name='ref' value='1'";
        if($Field{'ref'}){print" checked";}
        print">
        Referansenummer";
      }else{                             #tegner checkboxer
med standardverdier
     print"<input type='checkbox' name='tittel' value='1'
checked>
        Tittel
```

```
        <input type='checkbox' name='beskrivelse' value='1'
checked>
        Beskrivelse
        <input type='checkbox' name='statuslosning' value='1'
checked>
        Status/l&oslash;sning
        <input type='checkbox' name='aksjonspunkt' value='1'
checked>
        Aksjonspunkt
        <input type='checkbox' name='eref' value='1'>
        Eksternt refnr
        <input type='checkbox' name='ref' value='1'>
        Referansenummer";
      }
                                          #Tegner system-
dropdowmeny
print" </td>
    </tr>
    <tr>
      <td width='113'><b>System</b> <b></b></td>
      <td width='214'>
        <select name='system'>
        <option value='alle' selected>Alle";
         $sql = $forb->prepare("SELECT * FROM
$dboppl"."Systemer");
         $sql->execute || die {&FeilMelding("Får ikke hentet
data fra database","101") };
         while(($systemnr, $systemnavn) = $sql->fetchrow){
          if($Field{'system'}==$systemnr){
           print "<option value=$systemnr
selected>$systemnavn\n";
          }else{
           print "<option value=$systemnr>$systemnavn\n";
          }
         }
                        #Tegner leverandør-dropdowmeny
print" </select>
      </td>
      <td width='97'><b>Leverand&oslash;r</b> </td>
      <td width='259'>
        <select name='leverandor'>
        <option value='alle' selected>Alle";
         $sql = $forb->prepare("SELECT * FROM
$dboppl"."Leverandorer");
         $sql->execute || {&FeilMelding("Får ikke hentet data
fra database","102") };
         while(($levnr, $levnavn) = $sql->fetchrow){
          if($Field{'leverandor'}eq$levnr){
           print "<option value=$levnr selected>$levnavn\n";
          }else{
           print "<option value=$levnr>$levnavn\n";
          }
         }
                        #Tegner status-dropdowmeny
print"
        </select>
      </td>
    </tr>
```

```
    <tr>
      <td width='113'><b>Status</b></td>
      <td width='214'>
        <select name='status'>
        <option value='alle' selected>Alle";
         $sql = $forb->prepare("SELECT * FROM
$dboppl"."Status");
         $sql->execute || {&FeilMelding("Får ikke hentet data
fra database","103") };
         while(($statusnr, $statusnavn) = $sql->fetchrow){
          if($Field{'status'}==$statusnr){
           print "<option value=$statusnr
selected>$statusnavn\n";
          }else{
           print "<option value=$statusnr>$statusnavn\n";
          }
         }
                          #Tegner registrert av-dropdowmeny
print"
        </select>
      </td>
      <td width='97'><b>Registrert av</b></td>
      <td width='259'>
        <select name='regav'>
        <option value='alle' selected>Alle";
         $sql = $forb->prepare("SELECT Ansattnr, Etternavn,
Fornavn
                                  FROM $dboppl"."Brukere");
         $sql->execute || {&FeilMelding("Får ikke hentet data
fra database","104") };
         while(($ansattnr, $etternavn, $fornavn) = $sql-
>fetchrow){
          if($Field{'regav'}==$ansattnr){
           print "<option value=$ansattnr selected>$etternavn
$fornavn\n";
          }else{
           print "<option value=$ansattnr>$etternavn
$fornavn\n";
          }
         }
                          #Tegner feilkategori-dropdowmeny
print"
        </select>
      </td>
    </tr>
    <tr>
      <td width='113'><b>Feilkategori</b></td>
      <td width='214'>
        <select name='feilkategori'>
        <option value='alle' selected>Alle";
         $sql = $forb->prepare("SELECT * FROM
$dboppl"."Feilkategori");
         $sql->execute ||{&FeilMelding("Får ikke hentet data
fra database","105") };
         while(($feilnr, $feilnavn) = $sql->fetchrow){
          if($Field{'feilkategori'}==$feilnr){
           print "<option value=$feilnr selected>$feilnavn\n";
          }else{
```

```
           print "<option value=$feilnr>$feilnavn\n";
          }
         }
                    # tegner checkboks for å vise slettede
meldinger
print"
        </select>
      </td>";
      if ($SqlBruker[0] == 50){
        print"<td colspan=2><input type='checkbox'
name='synlig' value='1'"; if($Field{'synlig'}){
print"checked";}
        print" >Vis slettede meldinger</td>";
     }else{print"
      <td width='97'> </td>
      <td width='259'> </td>";}
print" </tr>
    <tr>
      <td width='150'>
        <input type='submit' name='nullstill'
value='Nullstill'>
      </td>
      <td width='214'> </td>
      <td width='97'> </td>
      <td width='259'> </td>
    </tr>
  </table>";


                                  #bygger spørringen mot
databasen

 $sql = "SELECT DISTINCT he.Lopenr, he.Refnr, he.Tittel,
st.Statusnavn, he.Regdato, br.Ansattnr ,br.Etternavn,
br.Fornavn, he.Varslinghendelse, sy.Systemnavn, he.FAQ,
he.Synlig, sy.Systemlink
         FROM  $dboppl."Hendelser he, $dboppl."Systemer sy,
$dboppl."Status st, $dboppl."Feilkategori fk,
$dboppl."Brukere br
         WHERE he.Statusnr=st.Statusnr
         AND he.System=sy.Systemnr
         AND he.Registrertav=br.Ansattnr
         AND he.Feilnr=fk.Feilnr";
  if($Field{'system'}!='Alle'){$sql="$sql AND
he.System=$Field{'system'}"}
  if($Field{'status'}!='Alle'){$sql="$sql AND
he.Statusnr=$Field{'status'}"}
  if($Field{'leverandor'}!='Alle'){$sql="$sql AND
he.Levnr=$Field{'leverandor'}"}
  if($Field{'regav'}!='Alle'){$sql="$sql AND
he.Registrertav='$Field{'regav'}'"}
  if($Field{'feilkategori'}!='Alle'){$sql="$sql AND
he.Feilnr=$Field{'feilkategori'}"}
  if(!($SqlBruker[0] == 50 && $Field{'synlig'})){$sql="$sql AND
he.Synlig=1";}
  if($SqlBruker[0] < 40){$sql="$sql AND he.FAQ=1";}

 $sql="$sql AND he.Lopenr IN(SELECT Lopenr
```

```
                    FROM $dboppl"."Hendelser
                    WHERE Lopenr=-1";

 if($Field{'tittel'}){$sql="$sql OR he.Tittel like
'%$Field{'soketekst'}%'";}
 if($Field{'beskrivelse'}){$sql="$sql OR he.Beskrivelse like
'%$Field{'soketekst'}%'";}
 if($Field{'statuslosning'}){$sql="$sql OR he.Statuslosning
like '%$Field{'soketekst'}%'";}
 if($Field{'aksjonspunkt'}){$sql="$sql OR he.Aksjonspunkt like
'%$Field{'soketekst'}%'";}
                                    #Sjekker på om soketekst
består av bare tall
 if($Field{'eref'} && (!($Field{'soketekst'}=~
m/\D/))){$sql="$sql OR he.Eksterntrefnr =
'$Field{'soketekst'}'";}
 if($Field{'ref'} && ((!($Field{'soketekst'}=~ m/\D/)) &&
($Field{'soketekst'} ne ''))){$sql="$sql OR he.Refnr
=$Field{'soketekst'}";
}
  $sql="$sql) ORDER BY he.Regdato DESC";

                                    #utfører spørring mot
database
  $resultat = $forb->prepare($sql);
  $resultat->execute() || {&FeilMelding("Får ikke hentet data
fra database","106") };

                                    #skriver ut søkeresultat
  if (($lopenr, $refnr, $tittel, $status, $dato, $ansattnr,
$enavn, $fnavn, $varhend, $system, $FAQ, $Synlig,
$systemlink)=$resultat->fetchrow){
    print"<table width='800' border='0'>
    <tr bgcolor ='#004090'>
     <td width='40'><b class='tabelloverskrift'>Ref</b></td>
     <td width='270'><b
class='tabelloverskrift'>Tittel</b></td>
     <td width='50'><b class='tabelloverskrift'>Status</b></td>
     <td width='130'><b class='tabelloverskrift'>Dato</b></td>
     <td width='160'><b class='tabelloverskrift'>Reg.
av</b></td>
     <td width='50'><b class='tabelloverskrift'>Type</b></td>
     <td width='70'><b
class='tabelloverskrift'>System</b></td>";
   if ($SqlBruker[0] >= 40){print"<td width='30'><b
class='tabelloverskrift'>FAQ</b></td>"};
 print"</tr>";
    do{
     @Tid = split(' ', $dato);              #formaterer dato
     if(($Ant >= $Min) && ($Ant < $Max)){  #styrer at det vises
kun 50 linjer pr side
       print"
       <tr bgcolor ='#ebebeb'>
        <td width='40'>";
         if($varhend == 0){                #forskjellige linker
avhengig av om varsling eller hendelse
           print"<a href=vardetaljer.cgi?$lopenr
class='tabellink'>$refnr</a>";
```

```
         }else{
           print"<a href=hendetaljer.cgi?$lopenr
class='tabellink'>$refnr</a>";
         }
      print"</td>
      <td width='270'>$tittel</td>
      <td width='50'>$status</td>
      <td width='130'>$Tid[0] $Tid[1] $Tid[2]</td>
      <td width='160'><a href=brukerinfo.cgi?$ansattnr
class='tabellink'>$enavn $fnavn</a></td>
      <td width='50'>";if($varhend ==
0){print"Varsling";}else{print"Hendelse";}
      print"</td>
      <td width='70'><a href=javascript:link('$systemlink')
class='tabellink'>$system</a></td>";

               #skriver "sletta" hvis meldingen er usynlig.
Hvis synlig, skriv FAQ-status
      if($SqlBruker[0] == 50 && $Synlig==0){print"<td
width='30'>Sletta</td>";}
      elsif ($SqlBruker[0] >= 30 && $FAQ==1){print"<td
width='30'>Ja</td>";}

      elsif ($SqlBruker[0] >= 30 && $FAQ==0){print"<td
width='30'>Nei</td>";}
     print"</tr>";
   }
   $Ant++;
   }while(($lopenr, $refnr, $tittel, $status, $dato, $ansattnr,
$enavn, $fnavn, $varhend, $system, $FAQ, $Synlig,
$systemlink)=$resultat->fetchrow);
                                                    #Kode for
skriving av 50 recorder i gangen
print"<tr><td colspan='4'></td><td align='right'> ";
 if($Min >= 20){                                   #forrige 50
   $Forrige = $Min - $Antvises;
   print" <a
href='javascript:forrige(this.form,$Forrige,$Min)'><img
src='tbpil.jpg' width='15' height='15'
class='tabelloverskrift'></a>";
}
print"</td><td>";                                    #Neste 50
if(($Ant - $Max) >= 1){
  print"<a href='javascript:neste(this.form,$Max,$Neste)'
class='tabelloverskrift'><img src='bgpil.jpg' width='15'
height='15'></a>";
}
print"</table>";

}else{
    print"<br><font size='+1'><b>Ingen treff</b></font>";
  }

print"</form></body></html>";
$forb->disconnect();

sub FeilMelding{
    local ($msg, $feil)=@_;
```

```
    print"<body><h2>Advarsel!!!</h2>
          $msg: $DBI::errstr
          <b><br>Feilkode: $feil</b>
          <br>Send mail til systemadministrator:
          <a
href='mailto:$sysadm'>$sysadm</a><br></body></html>";
          $forb->disconnect();
          exit;
}
```

4.6.2.2 dbNyHendelse.cgi

- En av hoveddelene i Erba.
- Viser filopplasting med cgi_lib.pl
- Viser subrutinen mail og HTMdie

```
#!/usr/bin/perl
#Testet 15.05.01 TH, IB
###############Databasescript for å regisrere ny
hendelse###############

$mto = "";                              #variabel for sending av
mail
$faq  = 0;                              #faqstatus
$RF = 0;                                #refnr for hendelsen

require 'dboppl.dta';
require 'cgi-lib.pl';             #script for mottak av multi-
form/data

$sqlref = $forb->prepare("SELECT Refnr FROM $dboppl"."Hendelser
");  #Finner refnr til ny hendelser
$sqlref->execute || &HTMLdie("Advarsel!!","Finner ikke refnr
til hendelsen: $DBI::errstr<br>Feilreferanse: 84<br>Send mail
til systemadministrator: <a
href='mailto:$sysadm'>$sysadm</a>");
while(($res) = $sqlref->fetchrow){
   if($res > $RF){             #Finner det største refnummeret
     $RF = $res;
    }
};
$RF = ($RF + 1);                #dette legges på med 1

$cgi_lib::writefiles = "$fil_dir";         #Sender med path'n
til der hvor vedlegg ligger
$cgi_lib::dbref = "$RF";                   #Sender med refnr
til &ReadParse

$ret = &ReadParse;

if($cgi_lib'filnavn =~ m/\d+\_.+/){        #sjekker om det er
lastet opp filnavn og ikke bare refnr.
   $nvn = $cgi_lib'filnavn;
}

if($in{'Faq'}){                           #endrer faqstatus vis
det er valgt
  $faq = $in{'Faq'};
};

if($in{'Var'}){                           #Setter inn hendelsen
som varsling
```

```
  $insertVar = "INSERT INTO $dboppl"."Hendelser(Regdato,
Tittel, Eksterntrefnr, Beskrivelse, Statuslosning, FAQ,
Aksjonspunkt, Statusnr, Registrertav, Kontaktperson, System,
Varslinghendelse, Refnr, Synlig, Mail,Varighet,Feilnr)
VALUES(GETDATE(),'$in{'Tittel'}','$in{'Eksref'}','$in{'Besk'}',
'',1,'',1,'$in{'Bruker'}','$in{'Kontakt'}',$in{'Sys'},0,$RF,1,0
,DATEADD(dd, 2, GETDATE()),1)";
  $sql = $forb->prepare($insertVar);
  $sql->execute() || &HTMLdie ("Advarsel!!","Får ikke satt inn
hendelsen som varsling: $DBI::errstr
                                <br>Feilreferanse: 85<br>Send
mail til systemadministrator: <a
href='mailto:$sysadm'>$sysadm</a>");
};

if($in{'Mailans'}){
  $Sn;                                  #Sender mail til alle
systemansvarlige
  $syssql = "SELECT  sy.Systemnavn, br.Email FROM
$dboppl"."Systemansvarlige sa, $dboppl"."Brukere br,
$dboppl"."Systemer sy WHERE sa.Systemnr=$in{'Sys'} AND
sy.Systemnr=sa.Systemnr AND sa.Ansattnr=br.Ansattnr";
  $sysql = $forb->prepare($syssql);
  $sysql->execute || &HTMLdie ("Advarsel!!","Får ikke sendt
mail til systemansvarlig: $DBI::errstr
                                <br>Feilreferanse: 86<br>Send
mail til systemadministrator: <a
href='mailto:$sysadm'>$sysadm</a>");
  while(($Ssn,$Em) = $sysql->fetchrow){
    $mto = $mto.$Em.",";
    $Sn = $Ssn;                         #Henter alle ansvarlige
på det valgte systemet
  }                                             #sjekker om
systemansvarlig finnes
  if($mto){&SysAns($mto, $in{'Tittel'}, $Sn);} #før tilkalling
av subrutiner
}
                                        #setter inn hendelsen
$insertHen = "INSERT INTO $dboppl"."Hendelser(Regdato, Feilnr,
Tittel, Eksterntrefnr, Beskrivelse, Statuslosning, Vedlegg,
FAQ, Aksjonspunkt, Kategorinr, Statusnr, Levnr, Registrertav,
Kontaktperson, System, Varslinghendelse, Refnr, Synlig, Mail)
VALUES(GETDATE(),1,'$in{'Tittel'}','$in{'Eksref'}','$in{'Besk'}
','$in{'Stabesk'}','$nvn',$faq,'',$in{'Mel'},1,$in{'Lev'},'$in{
'Bruker'}','$in{'Kontakt'}',$in{'Sys'},1,$RF,1,0)";
$sql = $forb->prepare($insertHen);
$sql->execute() || &HTMLdie ("Advarsel!!","Får ikke satt data
inn i databasen: $DBI::errstr
                                <br>Feilreferanse: 87<br>Send
mail til systemadministrator: <a
href='mailto:$sysadm'>$sysadm</a>");

$forb->disconnect();                    #kobler fra database
print "Location: henoversikt.cgi?ok&$RF\n\n";  #tilkaller
henoversikt med verdien ok for melding til bruker
```

```
################################## Subrutiner
#####################################
#ef= hvem sendes mail fra, et= hvem sendes mail til,es=
subject, eb= body på mail

                            #formaterer mail til systemansvarlig
sub SysAns{
  local($mailtil, $mailtittel, $Sa) = @_;
  $et="$mailtil";
  $es="Systemansvarlig $Sa\n";
  $eb="Det er registrert ny hendelse fra NPDSFD -Erba på
\"$Sa\".\nRefnr: $RF\n\nTittel: $mailtittel";
  &Mail($et,$es,$eb);
}
                           #sender mail
sub Mail{
  local($et,$es,$eb) = @_;
  open(SENDMAIL, "| $mail_adresse"."sendmail -t") ||
&HTMLdie("Advarsel!!","Finner ikke sendamil:
$DBI::errstr<br>Feilreferanse: 88<br>Send mail til
systemadministrator: <a href='mailto:$sysadm'>$sysadm</a>");
  print SENDMAIL "From: $Field{'mail'}\n";
  print SENDMAIL "To: $et\n";
  print SENDMAIL "Subject: $es\n\n";
  print SENDMAIL "$eb";
  close(SENDMAIL) || &HTMLdie("Advarsel!!","Får ikke sendt
mail: $DBI::errstr<br>Feilreferanse: 89<br>Send mail til
systemadministrator: <a href='mailto:$sysadm'>$sysadm</a>");
}
                          #htmldie
sub HTMLdie {
   local ($title, $msg)=@_;
   $title || ($title = "CGI Error");
   print "Content-type: text/html\n\n";
   print "<html>\n<head>
     <title>$title</title>
     </head>
     <body>
     <H1>$title</H1>
     $msg</body></html>\n";
   $forb->disconnect();
   exit;
}
```

4.6.2.3 dbEndreVarsling.cgi

- Viser Formatering og sending av mail.

```
#!/usr/bin/perl
#Testet 15/05-2001 av TH, IB
##############################Databasescript for å endre en
varsling###########

$Endremail = 0;    #Om vedkommende som registrerte varslingen
vil ha mail ved endring
$mto = "";         #Variable hvor alle emailadressene til
maillisten blir lagt inn i
$sato = "";        #Variabel hvor alle emaladressene til alle
systemansvarlige blir lagt inn i
$dato = localtime(time());  #Henter ut dato til varsling til
bruk i sending av mail

require 'postinn.dta';
require 'dboppl.dta';

if(!$Field{'ip'}){      #sjekker om det er form-data som kommer
  if($Field{'Mailvarsling'}){     #Setter mailfeltet i
databasen lik 1 slik at mailansvarlig får tilsendt mail ved
endring av varslingen
     $Endremail = $Field{'Mailvarsling'};
  };
                            #Setter inn varslingen
  $insert = "INSERT INTO $dboppl"."Hendelser(Regdato, Tittel,
Eksterntrefnr, Beskrivelse,Statuslosning, Aksjonspunkt, FAQ,
Statusnr, Registrertav, Kontaktperson, System,
Varslinghendelse, Refnr, Synlig, Mail, Varighet, Feilnr)
VALUES(GETDATE(),'$Field{'tittel'}','','$Field{'beskrivelse'}',
'$Field{'statuslosning'}','',1,$Field{'status'},'$Field{'bruker
'}','$Field{'kontakt'}',$Field{'system'},0,$Field{'Refnr'},1,$E
ndremail,DATEADD(dd, $Field{'varighet'},
GETDATE()),$Field{'feilkategori'})";
  $sql = $forb->prepare($insert);
  $sql->execute() || &HTMLdie ("Advarsel!!","Får ikke lagt data
inn i databasen: $DBI::errstr
                              <br>Feilreferanse: 48<br>Send
mail til systemadministrator: <a
href='mailto:$sysadm'>$sysadm</a>");

  $update="UPDATE $dboppl"."Hendelser SET FAQ=0 WHERE
Lopenr=$Field{'Varnr'}";     #Oppdaterer gammel varsling
  $sqllope = $forb->prepare($update);
  $sqllope->execute ||  &HTMLdie ("Advarsel!!","Kunne ikke
finne gammelt lopenr $DBI::errstr <br>Feilreferanse: 49<br>Send
mail til systemadministrator: <a
href='mailto:$sysadm'>$sysadm</a>");


#Sjekker om den som registrerte varlsingen skal ha med om
endring
```

```
  $sqlmailtilregav = "SELECT he.Mail, br.Email FROM
$dboppl"."Hendelser he, $dboppl"."Brukere br WHERE
Lopenr=$Field{'Varnr'} AND he.Registrertav=br.Ansattnr";
  $sqlmtra = $forb->prepare($sqlmailtilregav);
  $sqlmtra->execute || &HTMLdie ("Advarsel!!","Kunne ikke hente
opp mail til registrert av. $DBI::errstr<br>Feilreferanse:
50<br>Send mail til systemadministrator: <a
href='mailto:$sysadm'>$sysadm</a>");
  ($mail, $Email) = $sqlmtra->fetchrow;
  if($mail == 1){
    &EndretVarsling($Email);
  };

  if($Field{'Feilmottak'}){
#Sender mail til feilmottak
    &FeilMottak($Field{'tittel'}, $Field{'beskrivelse'});
  };
                                                        #sender
med email til inntastet emailadresse
  if($Field{'Sendmail'}){
    &SendMail($Field{'email'}, $Field{'tittel'});
  };

  if($Field{'Mailsjekk'}){              #Sender mail til alle
i mailliste
    $syssql = "SELECT br.Email FROM $dboppl"."Maillistelinje
ml, $dboppl"."Brukere br WHERE ml.Ansattnr=br.Ansattnr AND
ml.Systemnr=$Field{'system'}";
    $sysql = $forb->prepare($syssql);
    $sysql->execute || &HTMLdie ("Kunne ikke sende mail til
mailliste. $DBI::errstr<br>Feilreferanse: 51<br>Send mail til
systemadministrator: <a href='mailto:$sysadm'>$sysadm</a>");
    while($Em = $sysql->fetchrow){
      $mto = $mto.$Em.",";                              #Henter
alle ansvarlige på det valgte systemet
     }
    if($mto){&MailSjekk($mto, $Field{'tittel'});}  #sjekker om
det er registrert noe i maillisten
  };

  $Sn;                    #Her lagres systemnavnet
                         #Sender mail til alle systemansvarlige
  $syssql = "SELECT sy.Systemnavn, br.Email FROM
$dboppl"."Systemansvarlige sa, $dboppl"."Brukere br,
$dboppl"."Systemer sy WHERE sa.Systemnr=$Field{'system'} AND
sy.Systemnr=sa.Systemnr AND sa.Ansattnr=br.Ansattnr";
  $sysql = $forb->prepare($syssql);
  $sysql->execute || &HTMLdie ("Kunne ikke sende mail til
systemansvarlig.\n $DBI::errstr<br>Feilreferanse: 52<br>Send
mail til systemadministrator: <a
href='mailto:$sysadm'>$sysadm</a>");
  while(($Ssn,$Em) = $sysql->fetchrow){
    $sato = $sato.$Em.",";
    $Sn = $Ssn;          #Henter alle ansvarlige på det valgte
systemet
  }
```

```
  if($sato){&SysAns($sato, $Field{'tittel'}, $Sn);}    #sjekker
om det er registrert noen systemansvarlige før subrutinen blir
tilkalt

  $forb->disconnect();
  print "Location: varoversikt.cgi?ok&$Field{'Refnr'}\n\n";

}else{                                                 #feilmelding
ved ikke tilgang
  &HTMLdie("Du har ikke tilgang på denne siden.","Du kan
$logginn.");
  }

##################    Subrutiner      ####################
#ef= hvem sendes mail fra, et= hvem sendes mail til,es=
subject, eb= body på mail

sub FeilMottak{                        #mail til feilmottak,
tilkaller subfunksjonen mail
  local($tittel, $besk) = @_;
  $ef="$mailbruker";
  $et="$emailfeilmottak";
  $es="Ny varsling fra NPDSFD -Erba";
  $eb="Refnr: $Field{'Refnr'}\n\nTittel: $tittel\nRegistrert
av: $mailbruker\nDato: $dato";
  &Mail($et,$es,$eb);
}
                                       #mail til systemansvarlig
sub SysAns{
  local($mailtil, $mailtittel, $Sa) = @_;
  $et="$mailtil";
  $es="Systemansvarlig $Sa\n";
  $eb="Det er registrert ny varsling fra NPDSFD -Erba på
\"$Sa\".\nRefnr: $Field{'Refnr'}\n\nTittel: $mailtittel";
  &Mail($et,$es,$eb);
}

sub SendMail{                          #sending av mail til valgt
person
  local($et, $tittel) = @_;
  $es="Ny varsling fra NPDSFD -Erba\n\n";
  $eb="$Field{'mail'} har valgt å informere deg om at det er
registrert følgende varsling:\n\nRefnr:
$Field{'Refnr'}\n\nTittel: $tittel";
  &Mail($et,$es,$eb);
}

sub MailSjekk{                        #Seneder mail til mailliste
  local($et, $tittel) = @_ ;
  $es="Ny varsling fra NPDSFD -Erba";
  $eb="Refnr: $Field{'Refnr'}\n\nTittel: $tittel";
  &Mail($et,$es,$eb);
}

sub EndretVarsling{                #Sender mail hvis bruker
ønsket tilbakemelding ved endring
  local($et) = @_;
  $es = "Endret varsling";
```

```
  $eb = "Din varsling med refnr: $Field{'Refnr'} er
endret.\n\nNy tittel er: $Field{'tittel'}. (Refnr er det
samme).";
  &Mail($et,$es,$eb);
}

sub Mail{
  local($et,$es,$eb) = @_;
  open(SENDMAIL, "| $mail_adresse"."sendmail -t") ||
&HTMLdie("<h2>Advarsel!!!,</h2><BR>Klarer ikke å sende
mail<BR>$DBI::errstr<b><br>Feilkode: 53 </b><br>Send mail til
systemadministrator: <a
href='mailto:$sysadm'>$sysadm</a><br>");
  print SENDMAIL "From: $Field{'mail'}\n";
  print SENDMAIL "To: $et\n";
  print SENDMAIL "Subject: $es\n\n";
  print SENDMAIL "$eb";
  close(SENDMAIL) || &HTMLdie("Feil oppsto ved sending av
mail<br>Feilreferanse: 54<br>Send mail til systemadministrator:
<a href='mailto:$sysadm'>$sysadm</a>");
}

sub HTMLdie {
   local ($title, $msg)=@_;
   $title || ($title = "CGI Error");
   print "Content-type: text/html\n\n";
   print "<html>\n<head>\n";
   print "<title>$title</title>\n";
   print "</head>\n";
   print "<body>\n";
   print "<H1>$title</H1>\n";
   print "$msg \n</body> \n</html>";
   $forb->disconnect();
   exit;
}
```

4.6.2.4 postinn.dta

- Eksempel på require-script.

```
            #les hele CGI strengen inn i $inn

if ($ENV{'REQUEST_METHOD'} eq 'POST') {
#sjekker postforespørsel
   if ($ENV{'CONTENT_TYPE'} =~ m#^application/X-www-form-
urlencoded$#i){
       $ENV{'CONTENT_LENGTH'} || &HTMLdie ("Ingen lengde
oppgitt sammen med POST forespørsel.");
       read (STDIN, $innlinje, $ENV{'CONTENT_LENGTH'});
   }
}
@name, @value;                                    #splitter data
fra formen
@pairs = split('&', $innlinje);
%Field;                                          #Assosiativ
array for dataene

foreach $pair (@pairs) {
       $pair =~ s/%(..)/chr(hex($1))/ge;         #gjør om fra
hex til ascii
       $pair =~ tr/+/ /;                         #bytter ut +
med space
       ($nvn, $val) = split('=', $pair);
       $Field{$nvn} = $val;                     #data legges inn
i %Field
}
if(!%Field){$Field{'ip'} = -1;}              #Setter variabel
%Field{'ip'} ved ikke
                           #tilkallt med postforespørsel

return %Field;                                   #returnerer %Field
```

4.6.2.5 varreg.cgi

- Viser eksempel på grensesnitt-script.

```
#!/usr/bin/perl
#Testet 14.05.01 TH, IB
#######Brukergrensesnitt for å registrere ny varsling#######
                                #requirescript
require 'htmlhead.dta';
require 'getcooki.dta';
require 'dboppl.dta';

print"<script language='JavaScript1.2'>
var bool = true;
function sjekk(form){              //sjekking av mail
  var emailad = form.email.value;
  var em = /^[a-zA-Z0-9_\\-\\.]+\\@([a-zA-Z0-9_\\.\\-]+\\.)[a-
zA-Z]{2,3}\$/;
if(form.system.selectedIndex == 0){
   alert('Du må velge et system!');
   bool = false;
   }                                //sjekke lengde på felt
else if(form.tittel.value.length < 1){
   alert('Vennligst tast inn tittel!');
   bool = false;
   }
else if(form.beskrivelse.value.length < 1){
   alert('Vennligst tast inn beskrivelse!');
   bool = false;
   }
else if(form.Sendmail.checked){
   if(emailad.search(em)){
     alert('Ugyldig emailadresse!');
     bool = false;
   }
}
if(bool == true) {              //submitting av form
   form.action = 'dbNyVarsling.cgi';
   form.submit();
}
bool = true;
}

function avbryter(){
  location = 'varoversikt.cgi';
}

function openHelp(){
  popupWin = parent.window.open('varregHelp.cgi', 'varregHelp',
'left=400,top=250,width=410,height=310,toolbar=0,location=0,res
izeable=0');
}
</script></head>\n";
```

```
$sqlbruker = $forb->prepare("SELECT Rettigheter, Etternavn,
Fornavn, Email FROM $dboppl"."Brukere WHERE Ansattnr='$nr'");
$sqlbruker->execute || die {&FeilMelding("Får ikke hentet data
fra database","55") };
@SqlBruker = $sqlbruker->fetchrow;

if(($nr != 0) && ($SqlBruker[0] >= 30)){
#Sjekker om bruker har adgang og er logget inn..
  $tid = localtime(time());
  @dato = split(' ', $tid);

#henter fra systemer
  $sqlsystem = "SELECT * FROM $dboppl"."Systemer";
  $syssql = $forb->prepare($sqlsystem);
  $syssql->execute || die print"<h2>Advarsel!!!</h2><BR>Feil
ved å hente data fra database<BR>$DBI::errstr<b><br>Feilkode:
56</b><br>Send mail til systemadministrator: <a
href='mailto:$sysadm'>$sysadm</a><br>";

#Skriver ut html-form
  print"<body bgcolor='#FFFFFF'
onload='document.form1.tittel.focus()';>
  <h2>Registrer varsling</h2>
    Når skjema er fylt ut sendes mail til ansvarlig for
gjeldende system. Dersom man
    krysser av for mail til feilmottak i Nett vil også FHS få
varsling om dette.
  <form name='form1' method='post'>
  <table width='700' border='0' cellspacing='2'>
   <tr>
      <td width='100'><B>Registrert av</b></td>
      <td width='200'>$SqlBruker[1] $SqlBruker[2]</td>
      <td width='30' align='right'><b>Dato</b></td>
      <td >$dato[2] $dato[1] $dato[4]</td>
   </tr>
   <tr>
      <td colspan='6'><hr width='100%' ></td>
   </tr>
        <input type='hidden' name='bruker' value='$nr'>
        <input type='hidden' name='mail' value='$SqlBruker[3]'>
   <tr>
       <td ><b>Tittel</b></td>
       <td colspan='3'><input type='text' name='tittel'
maxlength='50' size='50'></td>
       <td ><b>System</b></td>
       <td align='right'>
         <select name='system' size='1'>
          <option value=0 selected></option>\n";
          while (($snr,$systemet) = $syssql->fetchrow){
             print"<option value=$snr>$systemet</option>\n";
      }
  print" </select></td>
    </tr>
    <tr>
      <td ><b>Beskrivelse</b></td>
      <td colspan='5'><textarea name='beskrivelse' cols='84'
rows='10' wrap='VIRTUAL'></textarea></td>
    </tr>
```

```
    <tr>
      <td ><b>Varighet</b></td>
      <td colspan='5'>
        <select name='varighet'>
           <option value=36500>Uendelig</option>
           <option value=1>1 døgn</option>
           <option value=2 selected>2 døgn</option>
           <option value=3>3 døgn</option>
        </select></td>
    </tr>
    <tr>
      <td ><b>Meldt av/ kontaktperson</b></td>
      <td colspan='5'><input type='text' name='kontakt'
size='50' maxlength='64'></td>
    </tr>
    <tr>
      <td ><b>Mail</b></td>
      <td colspan='5'><input type='checkbox'
name='Mailvarsling'value=1>
        Jeg vil ha varsling når det skjer statusendring
p&aring; feilen</td>
    </tr>
    <tr>
      <td > </td>
      <td colspan='5'><input type='checkbox' name='Feilmottak'
value='1'>
        Send mail til Feilmottak i Nett</td>
    </tr>
    <tr>
      <td > </td>
      <td colspan='5'><input type='checkbox' name='Sendmail'
value='1'>
        Send mail til:<input type='text' name='email' size='40'
value='\@telenor.com'></td>
    </tr>
    <tr>
      <td > </td>
      <td colspan='5' height='29'><input type='checkbox'
name='Mailsjekk' value='1'>
        Send mail til mailliste for det valgte systemet.</td>
    </tr>
    <tr>
      <td > </td>
      <td colspan='5'><input type='button' name='Submit'
value='Lagre' onclick='sjekk(form1)'>
        <input type='button' name='Avbryt' value='Avbryt'
onclick='avbryter()'></td>
    </tr>
  </table>
  <input type='button' name='hjelp' value='Hjelp'
onclick='openHelp()'></form>";

}
else {
    print "<h2>Du har ikke tilgang her.</h2>";
    print "Du må $logginn først.";
}
```

```
print"</body></html>";
$forb->disconnect();
                                                #sSubrutine for
feilmelding
sub FeilMelding{
    local ($msg, $feil)=@_;
    print"<body><h2>Advarsel!!!</h2>
          $msg: $DBI::errstr
          <b><br>Feilkode: $feil</b>
          <br>Send mail til systemadministrator:
          <a
href='mailto:$sysadm'>$sysadm</a><br></body></html>";
          $forb->disconnect();
          exit;
}
```

5 Testing

5.1 Rutiner for feilsjekk

Vi har testet hvert enkelt script etter hvert som vi har kodet de. I tillegg har vi testet hvordan systemet som helhet fungerer med jevne mellomrom. I og med at det er liten avhengighet mellom scriptene, har det ikke oppstått noen problemer med systemet som helhet. Vi har dessuten konsekvent gjennomført at samme script skal testes av flere personer. På slutten gjennomførte vi en større test, der vi først gikk gjennom all kode visuelt. Hadde da mye større grunnlag for å se hva som ville fungere i gitte situasjoner, enn vi hadde i starten. Gikk deretter gjennom hver enkelt side og sjekket at disse fungerte i henhold til kravspesifikasjonen. Fant en del feil her, disse er dokumentert i avsnitt 5.2

Den største overraskelsen vi fikk, var en dag da databasen plutselig sluttet å fungere, uten at vi hadde gjort noen større endringer i scriptene. Plutselig feilet alle script som skulle legge info inn i databasen.
Grunnen til dette var at det området som var avsatt til transaksjonslogg var fullt. Resultatet var da å dumpe transaksjonsloggen.
Dette hadde forøvrig ingen betydning for kodingen som sådan, da vi allerede hadde bestemt at transaksjonslogg skal dumpes annenhver time. Vi hadde bare ikke laget dette scriptet ennå.

5.2 Feil som oppsto under slutt-testen:

- Søkefunksjonen i hovedmenyen: Denne feilet hvis man trykket på søk-knappen, og det ikke var skrevet inn noen søketekst. Dette skjedde fordi spørringen mot databasen feilet på de feltene som inneholdt tallverdier. Ordnet det med en test på om det er skrevet inn en søketekst før det testes på disse feltene. Denne funksjonen sorterte dessuten i feil rekkefølge, slik at de gamleste meldingene kom øverst. Dette ble ordnet i spørringen mot databasen.

- Aksjonspunkt: Aksjonspunket ble ikke registrert i databasen. Liten feil i et script var årsaken.

- Vedlegg: Når en hendelse ble oppdatert, fulgte ikke eventuelle vedlegg med. Dette ble ordnet med en kodesnutt som registrerer det gamle vedlegget på nytt igjen.

- Noen av scriptene manglet sjekk på om bruker var innlogget, og hadde riktige rettigheter. Dette ble ordnet.

- Hvis man kryssertav for å sende mail til mailliste, uten at det var registrert noen mailliste for systemet, feilet scriptene. Dette ble ordnet med en rutine som sjekker om det er registrert en mailliste før det gjøres forsøk på å sende mail. Det samme skjedde hvis man krysset av for å sende mail til mailliste. Dette ble ordnet på samme måte.

- Feltet kontaktperson viste kun det første navnet som var skrevet inn, dvs bare fornavn eller bare etternavn. Dette ble løst ved å legge til to ”fnutter” rundt variabelen i koden.

- Brukere med kategori 50 kunne endre rettigheter til alle brukere, inklusive seg selv. Hvis det da fantes kun en bruker med kategori50, og denne ble satt ned, medførte det at ingen kunne gå inn å endre dette tilbake igjen.

- Påloggingа skrev ikke ut noen melding hvis man prøvde å logge inn med et ansattnr som ikke var registrert fra før, programmet gikk bare videre uten å logge inn noen bruker. Feilen skjedde da det ble gjort forsøk på å sammenligne inntastet passord med passord registrert på dette ansattnummeret i databasen, og dette nummeret ikke eksisterte. Dette ble løst ved først å sjekke om ansattnummeret eksisterer, før det sjekkes på passord.

6 Arbeidsmetoder og resultater

6.1 Valg av systemutviklingsmodell

Oppdragsgiver ønsket at vi skulle utarbeide grensesnittet på systemet forholdsvis tidlig, da det var noe usikkerhet om hvilke krav som skulle settes til systemet. Dette ga mulighet for å la brukerne teste grensesnittet, og komme med tilbakemeldinger. I tillegg visste vi veldig lite om den programerings-teknikken vi skulle bruke. Det var derfor vanskelig å vite hva som ville være enkelt å lage, og hvor vi ville få utfordringer.

Vi valgte på grunnlag av dette å jobbe etter RUP-modellen. Denne ga oss mulighet til å jobbe parallelt med kravspesifisering og utvikling av grensesnitt og kode. Vi fikk da også mulighet til å dele opp problemstillingen og konsentrere oss om deler av systemet.

Vi valgte bevisst å ikke bruke tid på å tegne dataflytdiagrammer, use-case diagrammer og lignende hjelpemidler. Vi fikk inntrykk av at oppdragsgiver hadde god oversikt over hvilke data som skulle lagres i systemet. Da oppgaven ble presentert, fikk vi skisser av brukergrensesnittet slik som oppdragsgiver hadde sett for seg det kunne bli, og en beskrivelse av de viktigste funksjonene i programmet. Oppdragsgiver la lite vekt på å beskrive helheten i systemet og gi eksempler på hva som skulle lagres. Vi fant det derfor mest naturlig å fortsette å jobbe videre med dette. Vi kunne da diskutere vårt forslag til brukergrensesnitt og kravspesifikasjon på møtene, og la vanlige brukere få teste programmet, istedet for å diskutere skisser og diagrammer.

6.2 Planlegging

Vi valgte å lage 4 betaversjoner av programmet, en for hver av de 4 første milepælene. Disse skulle (i følge forprosjektsrapport) inneholde følgende:

Versjon 0.1b: Selve brukergrensesnittet for oppretting og innlegging, utfra hvilken brukergruppe

Versjon 0.2b: Brukergrensesnitt optimalisert.
Generering av database og kommunikasjon med database skal være påbegynt.
Har design for pålogging

Versjon 0.3b: Database optimalisert
Pålogging fungerer
Rapporter, statistikk og søk skal være påbegynt

Versjon 0.4b: Pålogging ferdig
Optimaliserte rapporter, statistikk, søk og lignende
Brukermanualer

Versjon 1.0: Ferdig testet, og debug'et
Ferdige manualer

Viser for øvrig til gantskjema i forprosjektsrapport, vedlegg A, for planlagt fordeling av oppgavene utover tid. Kapittel 6.4 viser gantskjema slik jobbingen ble i praksis.

6.3 Kort om RUP

RUP står for the Rational Unified Proscess, og er en forholdsvis ny måte å jobbe etter ved utvikling av programvare.

Arbeidet deles opp i fire faser, Inception, Elaboration, Construction og Transition, med en milepæl i slutten av hver fase.

6.4 RUP i praksis

Viser til gantskjema nedenfor for en oversikt over hvordan vi delte inn fasene. Vi valgte å legge inn noen flere milepæler enn de milepælene angitt i RUP-modellen. Dette for å gi en bedre styring, og for lettere å estimere tidsforbruk.

6.4.1 Inception 03.01. – 26.01.

Laget, i samarbeid med oppdragsgiver, oppgavebeskrivelse, mål og rammer. Dette utgjorde, sammen med framdriftsplan og kvalitetsikringssplan, forprosjektsrapporten. Begynte i tillegg å kartlegge systemets funksjoner.

6.4.2 Elaboration 26.01. – 08.03.

Vi fikk skisser for hvordan oppdragsgiver hadde sett for seg at systemet kunne bli, og en oversikt over hvilke krav de hadde til systemet. Vi tok utgangspunkt i disse og informasjonen vi fikk på møtene. Det ble jobbet parallelt med å lage grensesnitt og en kravspesifikasjon i henhold til skolens standard.

Første milepæl var 15.02. Da skulle vi ha et forslag til design. Dette ble presentert for oppdragsgiver 12.02. Grensesnittet ble da lagt ut for testing på prosjektets webside. Vi hadde et møte med oppdragsgiver 21.02, hvor vi gikk gjennom dette forslaget. Grensesnittet var da testet av flere brukere. Vi fikk positive tilbakemeldinger, men med noen endringsforslag. Viser til møtereferatet, vedlegg H, for detaljer om dette.

Snakket også litt om selve databasen, og hva som skal lagres. Vi begynte design av database i dette tidsrommet, slik at neste versjon av grensesnittet skulle være tilpasset databasen.

Ved neste milepæl, 08.03, hadde vi en ny versjon av grensesnittet ute for testing, og hadde et forslag til databasedesign. Vi hadde nå også fått databaseserveren og webserveren til å fungere, og fått begynt å teste noen enkle script.

6.4.3 Construction 09.03. – 03.05.

Tiden fremover gikk med til perl-koding, pluss noe design-endringer etterhvert som vi så hvordan denne kodingsteknikken fungerte, og hvilke muligheter vi hadde.

Vi tok nå utgangspunkt i de html-sidene vi hadde, og lagde et script for å konvertere de over til et format som var lett å videreutvikle i Perl. Resten av utviklingen foregikk nå i Emacs for Linux.

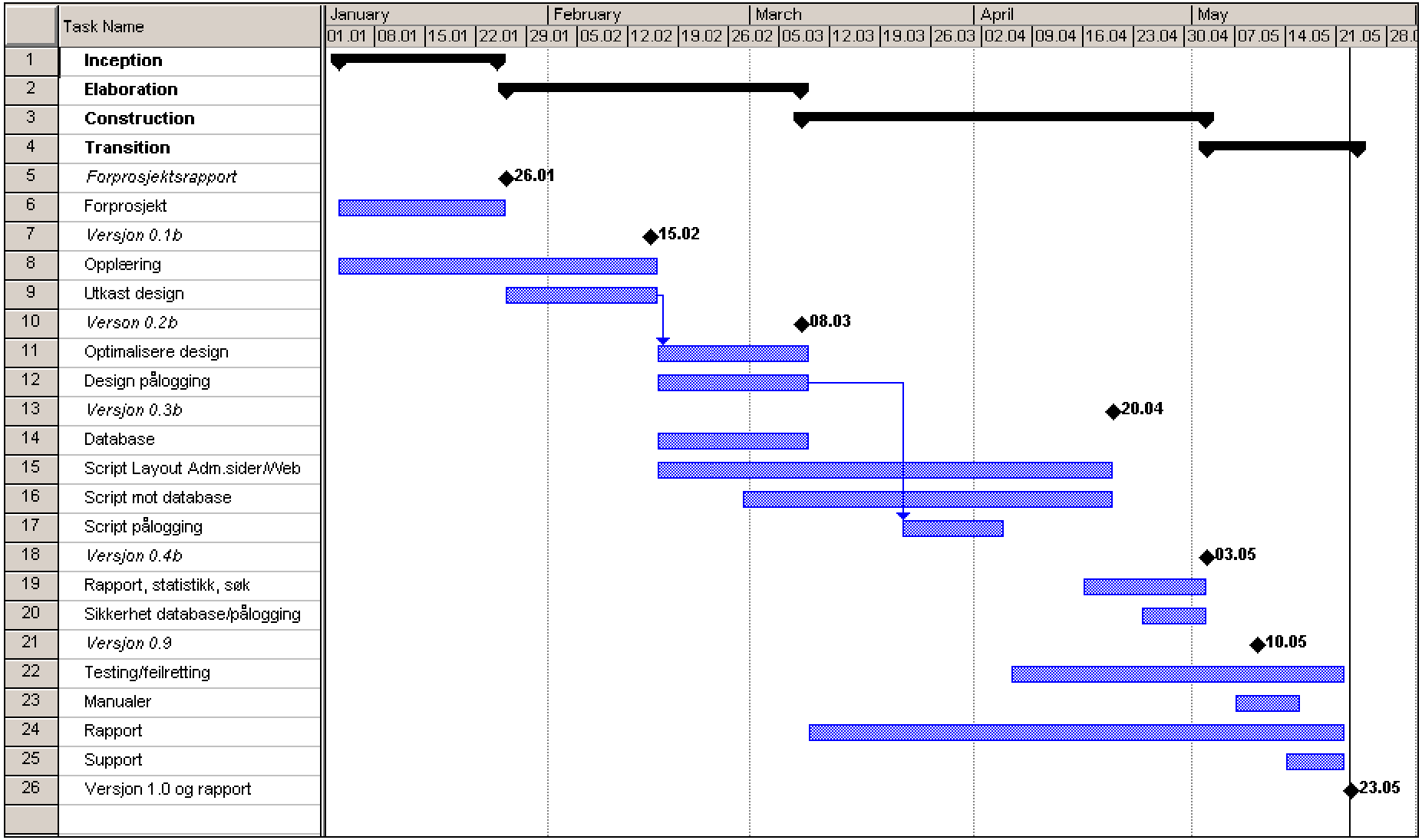
I denne perioden sprakk tidsskjemaet, pga. at det gikk med mer tid til jobbing med obliger i andre fag, og eksamener, enn vi hadde beregnet. Den milepælen vi hadde satt rett før eksamensperioden, 05.04, måtte dermed flyttes til over eksamensperioden og påske, til 20.04. Da var hoveddelene av programmet kodet. Det som gjenstod da var rapportsider, søkefunksjoner og en god del detaljer rundt omkring.

Vi hadde et møte med oppdragsgiver 23.04, hvor vi demonstrerte Erba, og diskuterte hvordan de siste detaljene skulle løses. Fikk positive tilbakemeldinger. Viser forøvrig til møtereferatet, vedlegg I.

Tiden fram mot milepælen 03.05 ble hektisk, og det manglet fremdeles enkelte detaljer i systemet da vi kom dit.

6.4.4 Transition 04.05. – 23.05.

I dette tidsrommet jobbet vi med å gjøre ferdig de siste detaljene i programmet, teste og rette feil, lage hjelpefunksjoner og skrive dokumentasjon.
Da vi kom til milepælen den 10.05., var programmet klart for igangsetting hos oppdragsgiver, men var ikke ferdig testet enda.
Siste versjon leveres sammen med prosjektrapporten 23.



Figur 6-1 Gantskjema, viser faktisk tidsforbruk

6.4.5 Oppsummering

Når det gjelder denne måten å jobbe på, synes vi den var svært vellykket. Når både systemansvarlig og vanlige brukere får teste systemet underveis er man helt sikker på at man faktisk lager et program oppdragsgiver vil ha.

Vi fikk erfare at det ble mye lettere å finne frem til krav og spesifikasjoner til systemet, når vi fikk opp grensesnittet. Har inntrykk av at dette gjaldt både oss og oppdragsgiver, da vi fikk flere nye ønsker fra oppdragsgiver, og endringer fra de opprinnelige kravene, underveis. Det var også en stor fordel å la brukerne teste grensesnittet før vi startet kodingen.

Kravspesifikasjonen ble også endret noe etter at vi hadde startet kodingen, da vi så endel nye løsninger underveis. Men vi så også at det er viktig med en god kravspesifikasjon når man begynner å kode. Da vi startet med prosjektet hadde vi så godt som ingen erfaring med skriving av interaktive web-sider. Vi hadde derfor liten mulighet for å si hva som var mulig på slike systemer, og hvilke begrensninger de hadde. Det var derfor en stor fordel å jobbe etter en modell som ga mulighet for å utvikle kravspesifikasjonen, og diskutere løsninger på problemene med oppdragsgiver, også etter at vi hadde begynt å skrive scriptene. Vi måtte også på enkelte punkt vente med noen detaljer i brukergrensesnittet, til vi var i gang med scriptingen, og så hvordan ting fungerte.

6.5 Diskusjon av resultater

Vi har lagt mye arbeid i å få systemet til å fungere slik oppdragsgiver ville. Det kostet oss mye arbeid å få opp kravspesifikasjonen og forslaget til brukerrensesnitt, da vi la stor vekt på å få funksjonaliteten og utseendet så godt som mulig.

Vi mener selv vi har lyktes med dette, og har også fått positive tilbakemeldinger fra oppdragsgiver underveis. Resultatene fra testingen viser at programmet fungerer som oppgitt i kravspesifikasjon og designdokument. De feilene vi fant ble rettet opp.

Ved den siste gjennomgangen av programmet så vi imidlertid en funksjonalitet som burde vært med. Hverken rapporter eller søkesidene har mulighet for å angi tidsrom. Dette hadde lettet bruken av programmet, da man hadde sluppet å bla så mye fram og tilbake hvis man skal finne meldinger fra et gitt tidsrom.

6.6 Egenevaluering

Vi har hatt utviklingsmiljøet vårt på sørbyen studenthjem. Her har vi hatt fire pc'er koblet i et eget nettverk, som igjen har vært koblet opp mot skolens nettverk. Vi har delt opp arbeidet i flere ansvarsområder, men samarbeidet om det meste. Dette har ført til at vi har jobbet veldig tett sammen hele veien, og fått drøftet løsninger underveis. I utgangspunktet var det meningen med ukentlige møter, men vi følte etterhvert at det ikke trengtes når vi jobbet så tett som vi har gjort. Gruppen har fungert meget godt sammen uten konflikter og samarbeidsproblemer.

Vi har også lyst til å si litt om backup-rutinene vi har hatt, siden vi synes disse har vært veldig bra. Noe som vi fikk belønning for da hele server-maskinen vår kræsjet. Det har blitt tatt backup av samtlige filer ca. annenhver dag. Det har ført til at vi har over 30 versjoner av systemet fra begynnelsen og fram til nå.

Vi har lagt ned mye arbeid i prosjektet, og har lært mye om prosjektarbeid, systemutvikling og programmering.

7 Konklusjon

Da vi startet med prosjektet hadde vi ingen erfaring i å behandle data via websider.
Vi hadde grunnleggende kunnskaper i html og javascript, fra faget ”Programmering mot www” og grunnleggende perl fra faget ”Drift og vedlikehold av flerbrukersystemer”.
Vi fikk mange utfordringer underveis. Vi måtte blant annet lære oss en programmeringsteknikk som var ny for oss alle.

Vi hadde også kun grunnleggende kjennskap til linux da vi startet. Her var det mye nytt vi måtte lære oss for å få til et skikkelig utviklingsmiljø. Dvs. å få satt opp web-serveren og databaseserveren på riktig måte, og få det til å kommuniserte skikkelig med perl-scriptene.

Vi har ellers vært innom mye av fagområdene vi har lært tidligere. Vi har hatt god bruk for kunnskapene fra fagene Databaser, Systemutvikling, og Drift av flerbrukersystemer. I alle disse fagene har vi lært mye teori, uten å få prøve det ut skikkelig i praksis. Har også brukt noen av de teknologiene vi har lært om i ”Klient og Serversideprogrammering”, men da dette kurset gikk samtidig med prosjektet, måtte vi lære oss disse teknikkene før de ble omhandlet der. Det samme gjelder kurset ”Systemutvikling 2”. Her lærte vi om RUP først etter at vi var igang med prosjektet.

Når det gjelder RUP som systemutviklingsmodell, har vi et veldig godt inntrykk. Muligheten for å fordele arbeidsflytene på tvers av fasene, og det å kunne konsentrere seg om hver enkelt problemstilling etter hvert fungerte veldig godt. Dette ga oss muligheten til å presentere det vi hadde gjort, og få nye innspill fra brukere underveis. Dermed kunne vi også utvikle kravspesifikasjonen og designdokumentet fortløpende. Dette var en stor fordel da vi var ukjent med de teknikkene vi skulle bruke, og dermed hadde vanskelig for å si hvordan ting måtte gjøres på forhånd.

Som en oppsummering føler vi at vi har lært veldig mye av dette prosjektet. Vi føler oss fornøyd med sluttresultatet, og håper brukerne også blir fornøyde.

8 Litteraturliste

Tom Negrino og Dori Smith – Visual quickstart guide, Javascript for the world wide web, 2. edition, United States of America: Peachpit Press

Arman Danesh – Internett med java og javascript, Europe: Prentice Hall Ltd.

Mark G Sobell - A practical guide to linux, England: Addison – Wesley.1999

Larry Wall, Tom Christiansen og Jon Orwant - Programming Perl, 3. edition, United States of America: O'Reilly.1999

Bill Middleton, Brian Deng og Chris Kemp – Web programming with perl5, 1. edition, United States of America: Sams.net, 2000

Erik Strom - Perl CGI programming, United States of America: Sybex Inc. 1998

D. McGoveran og C. J. Date - A guide to Sybase and SQL server, United States of America Addison Wesley 1992

Jon Tingvold, Drift av flerbrukersystemer, kompendiet, Høgskolen I Gjøvik 2000.

Sybase
http://www.sybase.com

Perl
http://www.perl.com

Høgskolen i Gjøvik. Øyvind Kolloen sine sider for faget Programmering for WWW.
http://higweb.hig.no/at/data/progwww/