HOVEDPROSJEKT:

Tittel

Livesupport 1

FORFATTER(E):

Synne Kvamme Repp
Trond Viggo Bjerke
Marianne Brattrud

Dato: 12.05.04

Sammendrag av hovedprosjekt

Tittel:	Livesupport 1	Nr. : 1 Dato : 12.05.04
Deltaker(e):	Synne Kvamme Repp Trond Viggo Bjerke Marianne Brattrud	
Veileder(e):	Harald Liodden	
Oppdragsgiver:	WebDeal	
Kontaktperson:	Jan Aril Sigvartsen	
Stikkord (4 stk)	Delphi, PHP, MySQL, Livesupport	

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

Kort beskrivelse av hovedprosjektet:

Oppgaven har gått ut på å utvikle et Livesupport-system for bedriften WebDeal. Et livesupport system er et system som tillater sanntids (live) kommunikasjon mellom kundebehandler og kunde. Dette systemet består, etter arbeidsgivers ønske, av en kundemodul i PHP, en konsulentmodul i Delphi og en MySQL-database. Utfordringen for oss har vært å få opprettet en stabil kommunikasjon mellom de forskjellige modulene.

Vi har lagt vekt på å utvikle en brukervennlig applikasjon som er enkel å navigere seg fram i. Oppgaven måtte avgrenses litt, og vi har implementert den funksjonaliteten vi mener var viktigst for å få systemet til å fungere.

Vi har hele tiden vært fokusert på å få et sluttprodukt som faktisk kunne tas i bruk. Stabilitet, brukervennlighet og funksjonalitet har vært sentrale begreper under utviklingsprosessen. Selve implementeringen av programmet er det som har tatt definitivt mest tid, og vi er glade for at vi ikke valgte et mer tidskrevende, formelt prosessrammeverk.

Besvarelsen består av denne rapporten og tilhørende vedlegg. I tillegg til dette skal CD med applikasjonen, kildekoden og rapporten på digitalt format legges med.

Forord

Da vi kastet oss ut i dette prosjektet i januar 2004, var det etter nøye overveielse og vurdering av aktuelle oppgaver. Noe av grunnen til at vi valgte dette som hovedprosjekt var fordi det virket som en utfordrende programmeringsoppgave. Vi var alle av den oppfatning at programmering var mer spennende enn systemutvikling, og ønsket en oppgave der hovedvekten av arbeidet lå i selve implementeringen.

Den største utfordringen med prosjektet har nok vært vår dårlige kjennskap til Delphi, og vi ønsker derfor å takke Jan Aril Sigvartsen. Han har vært til stor hjelp under utviklingen av systemet, og har vært vår mentor innen Delphi-programmering.

Vi vil også takke hele bedriften WebDeal for muligheten til å bygge opp et slikt system, samt veilederen vår Harald Liodden som var spesielt god å ha i startfasen av prosjektet. Vi vil også takke gruppen vi har delt grupperom med, Livesupport 2, for godt samarbeid.

Til slutt vil vi takke hverandre for godt teamwork, konstruktiv kritikk, mye latter og gode diskusjoner!

Gjøvik 12. mai, 2004

Synne Kvamme Repp Trond Viggo Bjerke Marianne Brattrud

Innholdsfortegnelse

1 Innledning

1.1 Oppgavebeskrivelse

Dette prosjektet har som mål å utvikle et funksjonelt og godt Live-support system for WebDeal A/S. Systemet skal benyttes for å bedre deres kundeservice, gi kundene en bedre serviceopplevelse, samt gjøre hverdagen for WebDeals konsulenter lettere og mer effektiv.
Målet er altså tidsbesparing, økt service og økt effektivitet.

Et livesupport system er et system som tillater sanntids (live) kommunikasjon mellom kundebehandler og kunde. Det er ønskelig å gi kundene inntrykket av personlig, dedikert support, mens kundebehandleren i realiteten kan sitte og arbeide med mange ulike kunder og saker samtidig.

Figur 1.1 Livesupportsystem

Det finnes i dag en stadig voksende brukermasse som er vant til å bruke Internet som deres eneste kommunikasjonskilde mot leverandører. Mange har ett eller flere enkle spørsmål som de ønsker raskt svar på. Problemet er at kunden i flere tilfeller vil være i tvil om hvor lang tid det tar før man får svar, og kanskje også om personen som sitter på kundeservice har nettopp den kompetansen som trengs for å svare på spørsmålet.
Fordelen med et livesupport system i et miljø som det WebDeal opererer i, er muligheten for at en person på kundeservice kan yte support til flere kunder parallelt, noe man ikke kan med tradisjonell telefoni (som i dag er den eneste ”live support” vi opererer med).
Vi mener at eksisterende livesupport løsninger ikke er et alternativ for WebDeal (og mange andre) pga. følgende punkter:

- Løsningene krever egne klienter hos kunden noe som er komplisert å installere og gjør at kunden ikke vil få til å benytte systemet.
- Systemer med tilfredsstillende funksjonalitet er ikke tilgjengelig for egen installasjon. Dette fører til at man må gå via servere i USA som gir en utrolig treg løsning.
- Ingen systemer gir muligheten til å kjøre profiler der man kan treffe personer basert på språk, faglig kunnskap og lignende.

1.1.1 Avgrensninger / omfang

Vi besluttet i forkant av prosjektet å legge vekt på grunnleggende og vesentlig funksjonalitet i systemet. Da vi i løpet av den tiden vi har til rådighet i dette prosjektet vanskelig kan rekke å implementere all ønsket funksjonalitet, har vi gjort noen beviste avgrensninger. Vi besluttet å kutte ut standarsvar-søkeroboten som Webdeal i utgangspunktet ville implementere i kundemodulen. Dette virket som en svært innviklet oppgave vi ikke hadde tid nok til å løse. Oppdragsgiver ønsket også mulighet til å sende filer mellom kunde og konsulent. Dette ble nedprioritert, da vi syntes det fantes enklere måter å utveksle filer på. Nærmere beskrivelse av hva vi valgte å prioritere i prosjektet, finnes i kapittelet ”kravspesifikasjon”.

Selve konsulentdelen av systemet, altså den delen av systemet WebDeals konsulenter vil benytte seg av, skal være en Windows-applikasjon som ligger på WebDeals maskiner. Kundene skal derimot ikke trenge å installere et program på sin maskin for å benytte seg av supportsystemet, og skal heller ikke trenge en spesiell type hardware eller software.

1.2 Målgruppe

Målgruppen for det endelige systemet er WebDeal A/S og kundene deres.
Målgruppen for rapporten er vår oppdragsgiver (WebDeal), vår veileder (Harald Liodden) og sensor. Andre som eventuelt vil lese om systemet, benytte det, og kanskje videreutvikle det ved en senere anledning, vil også ha nytte av denne rapporten.

1.3 Faglig bakgrunn

To av medlemmene på gruppen, Marianne Brattrud og Synne Repp har bakgrunn fra data- og multimedieteknikk ved HIG. Denne våren fullfører de dataingeniørgraden etter 2 års påbygging, og de har vært gjennom mange fag som er relevante for dette hovedprosjektet. Det siste medlemmet, Trond Viggo Bjerke, har bakgrunnen sin fra NTNU, der han avla mellomfag innen informatikk. Han går nå 5 årig sivilingeniør ved HIG. Gruppemedlemmene har litt ulik faglig bakgrunn, noe som er en fordel da kompetansen er fordelt på ulike områder og medlemmene kan gi ulike vinklinger på metoder og problemer.

Vi har alle hatt ulike systemutviklingsfag, så gruppen har kompetanse innen ulike systemutviklingsmodeller og teknikker.

En av oss har tidligere erfaring med Delphi, noe som kom godt med i arbeidet. Denne personen hadde imidlertid ikke erfaring med klient- og serversideprogrammering, som står sentralt i denne oppgaven. Det hadde de to andre medlemmene, så vi opplevde en positiv kompetanseblanding på ulike områder i denne oppgaven.

1.3.1 Hva måtte læres?

Som nevnt tidligere hadde et av medlemmene erfaring med Delphi fra undervisningssammenheng. Det viste seg midlertidig raskt at hovedvekten av kompetansen lå innen Pascal koding, og heller var begrenset når det gjaldt arbeid mot databaser og utvikling av grensesnitt. Det ble derfor svært viktig å tilegne seg mer kunnskap om dette raskt. De andre medlemmene hadde ikke vært borti Delphi tidligere, og måtte tilegne seg alt fra grunnen av. Kunnskapen vår om bruk og drift av databaser måtte oppdateres, da vår erfaring med dette stort sett var teori gjennomgått i undervisningssammenheng. Medlemmet som manglet PHP-erfaring måtte tilegne seg noe kunnskap om dette, selv om de to andre gruppemedlemmene tok seg av PHP-programmeringen i oppgaven.

1.4 Arbeidsformer

Under arbeidet med denne oppgaven fikk vi et grupperom i kjelleren på A-bygget til disposisjon. Vi skulle dele rom med en annen hovedprosjektgruppe. Denne gruppen jobbet for samme arbeidsgiver, men med andre aspekter av WebDeals kundeservicesystem.
Store deler av arbeidet ble gjennomført når gruppen var samlet på arbeidsrommet. Vi var kun tildelt en PC fra IT-tjenesten, som grunnet manglende kapasitet raskt ble byttet ut med private maskiner. 2 stasjonære og en bærbar. Dette gjorde at vi hadde gode verktøy til å jobbe med oppgaven.
Vi har også jobbet mye hjemme med oppgaven, og har benyttet oss flittig av kommunikasjonsverktøyet Microsoft Messenger gjennom hele perioden.
Annenhver uke har vi hatt møte med vår veileder Harald Liodden. Disse ble gjennomført på Lioddens kontor på HIG. Vi har også hatt ukentlige møter med Jan Aril, vår kontaktperson hos WebDeal. Han har også vært tilgjengelig via Microsoft Messenger, noe som har bidratt til snarlig løsning av problemer.
Det faktum at vi har delt grupperom med Live-support gruppe 2, har bidratt til godt samarbeid, samt at vi har kunnet utnytte de ulike gruppemedlemmenes kompetanse på en god måte.
Vi har kontinuerlig dokumentert vårt arbeid, ført statusrapporter, møtereferater og logg over timebruken til den enkelte.

1.4.1 Systemutviklingsmodell

Vi valgte å benytte oss av inkrementell systemutvikling. Vi mente inkrementell utvikling var en fordel siden deler av systemet kunne testes ut tidlig, og bli supplert med flere moduler etter hvert. Dermed kunne vi ta lærdom av evt. feil og mangler ved tidlige inkrementer. Siden vi var såpass uerfarne innen Delphi, var dette en fordel. Vi fikk også sjansen til å teste de først utviklede, viktigste komponentene nøye, samt at vi kunne foreta en løpende vurdering om ønskene fra WebDeal kunne la seg gjennomføres. Det passet bra å bruke en inkrementell systemutviklingsmodell på dette prosjektet, siden det var enkelt å dele funksjonaliteten som skulle implementeres inn i moduler og inkrementer.

Vi har ikke benyttet oss av samme utviklingsstrategi for hvert inkrement, men har sett an arbeidsoppgavene. Det har hele tiden vært mulighet for å gå tilbake, rette opp feil og gjøre forbedringer. Vi har hatt beslutningspunkter i slutten av hvert inkrement for å vurdere om inkrementet oppfylte de ønskede spesifikasjonene. Statusrapporter har blitt skrevet underveis, og vi har forsøkt å dokumentere de fleste endringene som har blitt foretatt.

Systemutviklingsmodellen RUP ble også vurdert, men etter vår mening vil denne fungere bedre på større prosjekter og i større prosjektgrupper. Vi ønsket en utviklingsmodell der vi kunne fokusere mye på utviklingen og mindre på det formelle papirarbeidet. Dette var en nødvendig prioritering siden arbeidsoppgavene var mange og ingen av oss hadde lyst til å kutte ned på funksjonaliteten i produktet til fordel for en utvidet systemutviklingsprosess. Vi fikk inntrykket av at en stor og omfattende systemutviklingsmodell ikke var noen god idé dersom vi ville ha mer igjen for arbeidet enn bare en god rapport.

1.4.2 Roller innad gruppen

Vi begynte med en forholdsvis rigid ansvarsstruktur innad gruppen. Det viste seg derimot raskt at ansvarsområdene fløt over i hverandre. Arbeidsoppgavene ble fordelt fortløpende til den som var ledig for øyeblikket såfremt personen hadde den nødvendige kompetansen. Vi jobbet for det meste selvstendig, men jobbet også sammen som en gruppe når dette var nødvendig.

1.4.3 Øvrige roller

Jan Aril har vært en sentral støttespiller under systemets utvikling. Han har fulgt oss opp og svart på spørsmål når vi sto fast eller ville ha en annen vinkling på et problem.

Harald Liodden har hele tiden bidratt med informasjon om SQL, Delphi, rapportskriving og problemløsning.

1.5 Organisering av rapporten

Vi har inngått avtale med arbeidsgiver om at selve rapporten skal skrives på norsk, men at selve programmet, all kode og alle kommentarer skal gjøres på engelsk. Det skal tilrettelegges slik at systemet enkelt skal kunne benyttes, videreutvikles og tilpasses av konsulenter som ikke snakker norsk.
Vi har skrevet rapporten først og fremst for dem som måtte ha interesse av systemet, og forutsetter dermed en viss kjennskap til informasjonsteknologi og tilhørende lingo.

Kapittel 1 Innledning
Innledende kapittel. Generell informasjon om oppgaven og organiseringen av gruppearbeidet.

Kapittel 2 Kravspesifisering
Dette kapittelet tar for seg kravene til produktet som skal utvikles. Her får du en oversikt over hva vi har prioritert i oppgaven, samt funksjonelle og grafiske krav. Vi har også beskrevet systemets omgivelser.
Vi har delt kravspesifikasjonen inn i en del for kundemodulen og en for konsulentmodulen. Siden vi har to såpass distinkte moduler, syntes vi det var mest oversiktlig å gjøre det på denne måten. Kravene er illustrert med usecase-diagrammer, og disse kan du finne i dette kapittelet.

Kapittel 3 Design
Denne delen omhandler oppbyggingen av systemet i ganske grove trekk. Her finner du en beskrivelse av systemets overordnede arkitektur, samt en mer spesifikk beskrivelse av arkitekturen til kundemodulen og konsulentmodulen. Vi har også beskrevet hva vi la til grunn når vi lagde databasen, og hva vi har lagt vekt på i GUI-designet. Til slutt har vi skrevet litt om alternative løsninger som er vurdert for systemet.

Kapittel 4 Implementering
Vi har valgt å ha en nokså omfattende dokumentering av implementeringen, da denne har vært en stor del av prosjektet vårt. Det vil også være greit å slå opp her for den som eventuelt skal videreutvikle systemet. Først i kapittelet kommer en systembeskrivelse som gir leseren en overordnet forståelse av hva systemet skal gjøre. Deretter har vi delt inn resten av dette kapittelet i implementering av kundemodulen og konsulentmodulen. Vi har beskrevet en avgrenset del av produktet under hvert avsnitt, og sagt litt om hvilke løsninger vi har valgt når det gjelder selve programmeringsprosessen.
Du vil også finne eksempler på kode.

Kapittel 5 Kvalitetssikring og testing
I dette kapittelet står det litt om hva som har blitt gjort for å besørge kvaliteten på produktet. Vi har beskrevet hvilke kvalitetsfaktorer som har vært viktige for oss under arbeidet, og hva vi har gjort for å ta hensyn til disse. Vi har også beskrevet testmetodene vi har benyttet oss av, og hvordan testingen har blitt gjennomført.

Kapittel 6 Egenvurdering og avslutning
Dette er vår oppsummering av arbeidet. Her har vi beskrevet resultatet både av produktutviklingen og av selve gruppearbeidet. Vi har prøvd å se arbeidet vårt fra en kritisk synsvinkel, men fordi det er vanskelig å bedømme seg selv objektivt, har også arbeidsgiver fått skrive hva han synes om resultatet (se vedlegg G). Det blir også sagt litt om hvilke forbedringsmuligheter systemet har videre, og hvilke valg vi har gjort underveis.

Kapittel 7 Litteraturliste
Dette er en oversikt over kildene vi har benyttet oss av i prosjektet.

Vedlegg – Tilleggsinformasjon som ikke hører hjemme i selve rapporten.

1.6 Terminologi

Definisjon av benyttede begreper.

Ansatte	Alle de som er ansatt i WebDeal.
ActionManager	Delphi komponent som kan brukes til å utvikle layout, og lar deg bygge menyer med selvdefinerte og standard prosesser.
Brukergrensesnitt	Den visuelle delen av systemet som brukeren kommuniserer gjennom.
Chat	Populært navn på Internett-funksjonalitet som gjør at to brukere kan "samtale" med hverandre direkte ved å skrive meldinger på tastaturet.
DBGrid	Delphi komponent som brukes til å vise og manipulere poster i et datasett.
DBLookupComboBox	Lar deg bruke verdier fra et datasett til å lage en drop-down boks.
DBLookupListBox	En boks som lister tekstelementer som hentes direkte ut fra databasen.
Delphi	Windows-basert programutviklingsverktøy fra Borland International.
Drop-down boks	En liste over valgmuligheter der du kun kan velge ett element.
FAQ	Standardsvar som ligger i databasen (**F**requently **A**sked **Q**uestions).
Form	Skjema i HTML. Innledes med <FORM> og avsluttes med </FORM>.
Frames	Definerer en ramme i et rammesett på en HTML-side. Deler en webside opp i flere uavhengige deler.
GUI	Grafisk brukergrensesnitt (**g**raphical **u**ser **i**nterface).
Konsulentmodul	Den delen av systemet som brukes av de ansatte i WebDeal. Delphi grensesnitt.
Kunde	WebDeals kunder.
Kundemodul	Den delen av systemet som brukes av klienten. Kodet i PHP.
Kundeservice	Avdelingen hos WebDeal som jobber med kundeservice.
LSS	Live-support System.
Memofelt	Delphi-komponent. Tekstfelt med flere linjer.
MySQL	MySQL Et system for administrasjon av databaser som bruker et subsett av ANSI SQL.
PHP	Skriptspråk for Web-sider, basert på åpen kildekode. PHP er gratis og fungerer med andre åpen-kildekode-prosjekter, som f.eks. Web-tjeneren Apache og databasene MySQL.
Profil	Hver konsulent hos WebDeal som benytter Live-support systemet har en egen profil. Denne profilen inneholder navn, kompetanseområder, språk og annen relevant informasjon i forbindelse med Support.

Spørringer	Strenger med sql-tekst som brukes opp mot databasen for å hente ut data vi ønsker.
SQL	Structured Query Language. Strukturert spørrespråk for arbeid mot databaser.
Support	Bidra til å løse kundenes problem. Aktiv hjelp til kunder som henvender seg angående en spesifikk problemstilling.
System Administrator	Bruker med tilnærmet alle, eller alle tilganger, som administrerer et system.
URL	Internettadresse (**U**nified **R**esource **L**ocator)

2 Kravspesifikasjon

2.1 Hva skal vi lage

Vår oppdragsgiver WebDeal har hatt mange konkrete og godt definerte krav angående hva de ønsker av funksjonalitet. De har også supplert med noen nye krav underveis. Det har vært helt nødvendig med kontinuerlig samarbeid med WebDeal for å være sikre at produktet ble det de ønsker. Under kommer en kort beskrivelse av det WebDeal ønsker at vi skal utvikle.

Det som har blitt prioritert i konsulentmodulen er:

- sikker, kryptert login
- profilopprettelse
- muligheter for å hente fram profiler fra login
- mulighet til å ignorere plagsomme kunder utifra ip-addresse
- mulighet til å åpne gamle saker
- muligheten til å ha mange dialoger oppe samtidig
- tilgang til statistikkfunksjoner
- muligheten til å søke fram og hente ut standardsvar
- oversikt over kø og kundeproblemer
- mulighet til å rute sak videre til annen konsulent

Kundemodulen:

- Et enkelt panel før man får opp live-supporten der man må taste inn noen relevante data (Referansenummer / domene, språk, spørsmålets tema)
- En chat med kundeservice med info om konsulent, der kommunikasjonen foregår i tilnærmet sanntid
- Et oversiktlig og godt køsystem

2.2 Systemets omgivelser

WebDeal baserer sine systemer på Unix (FreeBSD 5.2.x: Fullstendig kompatibel med Linux) og MySQL-database. Back-end systemer på web og server kjøres i PHP og Perl mens Windows-klienter til nå har blitt programmert i Delphi. WebDeal benytter separate servere; databaseserveren kjører MySQL 4.0 og webserveren kjører Apache 2.0.

2.3 Kravspesifikasjon kundemodul

Kravene til systemet forstås best ved å definere de overordnede funksjonelle kravene først. Disse kravene har vi kommet frem til sammen med oppdragsgiver. De er skrevet på en entydig måte for å unngå eventuelle misforståelser.

2.3.1 Overordnede funksjonelle krav

- Brukerne skal til en hver tid ha mulighet til å legge inn en forespørsel til WebDeal via Internett.
- Brukeren skal kunne legge inn en henvendelse uten noen forhåndskunnskaper om hvordan systemet fungerer.
- Brukerne skal ha mulighet til å velge hvilke språk som skal benyttes under supportsamtalen fra et begrenset utvalg.
- Brukerne skal måtte velge en kategori som passer til den forespørselen de har fra et begrenset utvalg.
- Alle skal kunne legge inn en henvendelse, både allerede eksisterende kunder og ikke-kunder.
- Idet en henvendelse legges inn, skal det returneres en samtale ID som benyttes som identifikator til samtalen er ferdig.
- Det skal ikke være mulig å legge inn en henvendelse uten at noe spørsmål er stilt i ”question” feltet.
- Programmet skal håndtere all input uten at det oppstår noen feil, for eksempel at brukeren skriver inn HTML-kode.
- Når en kunde legger inn en henvendelse, skal det umiddelbart sjekkes om det finnes en konsulent tilgjengelig med rette kvalifikasjoner når det gjelder ferdigheter og språk.
- Brukerne skal hele tiden få tilbakemelding på hvilket nummer de er i køen til sin konsulent.
- Systemet skal fange opp kundenes ip-adresser og sjekke om disse er blokkert. Hvis dette er tilfelle skal ikke henvendelsen legges inn i databasen og kunden skal

få en direkte beskjed av systemet om at WebDeal ikke kan hjelpe ham på dette tidspunktet.

- Etter at brukerne har lagt inn en henvendelse, skal de ikke utføre noen flere handlinger før de får opp et samtalevindu.
- Det skal ikke spille noen rolle hvilket operativsystem og nettleser som kunden benytter seg av.

2.3.2 Krav til grafisk brukergrensesnitt

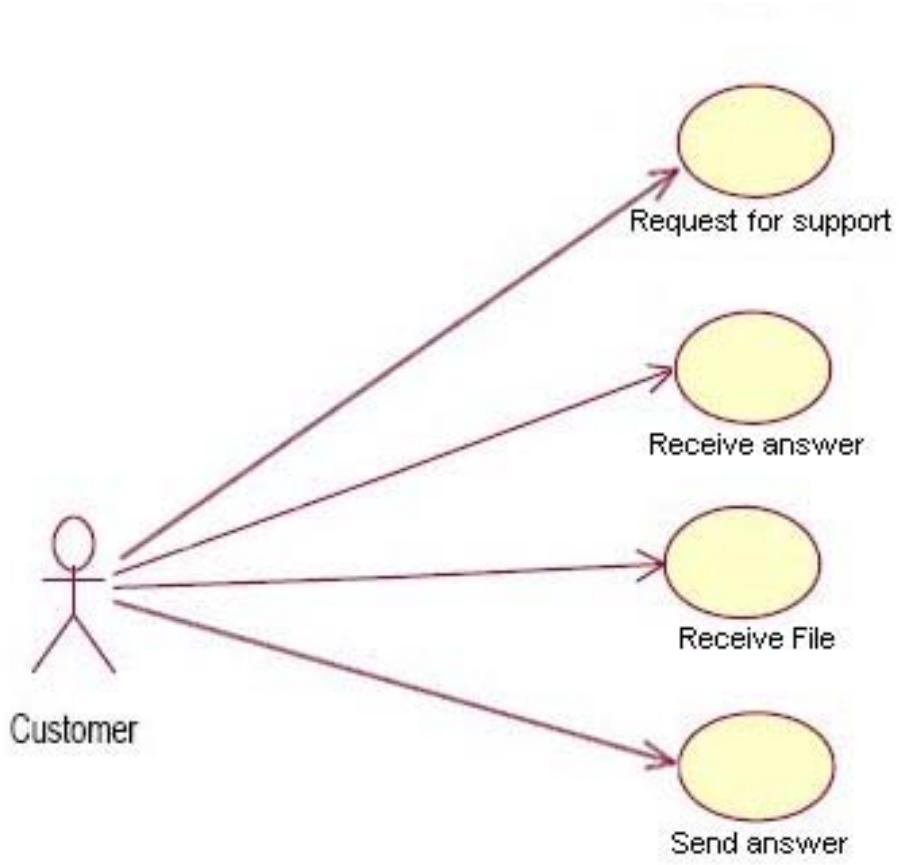
Det stilles høye krav til det grafiske brukergrensesnittet, da spesielt med tanke på brukervennlighet. Du skal altså som kunde intuitivt forstå hva som skal skrives inn i de ulike feltene og hvordan du får sendt henvendelsen til WebDeal. Det skal være lett å forstå hva de ulike komponentene gjør og navigeringen skal gå automatisk når du førstes har lagt inn henvendelsen.

Det skal legges inn noen kontroller og sjekker slik at det ikke skal være mulig for brukerne å gjøre noen feil. Det skal gis tilbakemeldinger på hva brukerne eventuelt har gjort feil og hva som må endres på.

En kunde skal også til en hver tid vite hvor i køen han ligger til sin utvalgte konsulent, slik at han vet at det er kontakt med WebDeal.

Når det gjelder designet skal vi bare lage hovedskallet, deretter skal WebDeal sin designer lage et finere design som vi skal implementere.

2.3.3 Use Case diagram



Figur 2.1 Use case kunde

2.3.4 High-level UseCase-beskrivelser

Use case	Request for support
Mål	Kunden skal få lagt inn en henvendelse til Livesupport hos WebDeal
Aktør	Kunde
Beskrivelse	Kunden skal greit kunne legge inn en hendelse uten noen form for login. De må skrive inn sitt spørsmål, velge hvilken kategori spørsmålet ligger under og hvilket språk de ønsker supportsamtalen skal foregå på. Kunden vil få innen ett minutt på om det er noen ledige konsulenter og hvilket nummer kunden er i køen til valgt konsulent.

Use case	Receive answer
Mål	Kunden skal kommunisere direkte med kundekonsulent
Aktør	Kunde
Beskrivelse	En kundekonsulent skal kunne skrive inn et svar som vises på kundesiden. Kunden skal ikke trenge å gjøre noe spesielt for at svaret skal vises. Han skal bare vente og lese teksten når den kommer opp.

Use case	Receive Standard Answer
Mål	Kunden skal kunne motta et standardsvar fra kundekonsulenten
Aktør	Kunde
Beskrivelse	En kunde skal kunne motta et standardsvar fra en kundekonsulent. Kunden skal få spørsmål om han vil motta et standardsvar og hvis kunden sier at dette er greit så sender konsulenten standardsvaret. Det blir da opp til kunden å lese standardsvarene og følge instruksjonene.

Use case	Send answer
Mål	Kunden skal kunne sende svar og spørsmål til kundekonsulenten
Aktør	Kunde
Beskrivelse	For at det skal være mulig for kunden og kundekonsulenten å kommunisere må kunden ha en mulighet til å svare på konsulentens svar. Dette gjøres ved at kunden skriver inn sin respons og sender det slik at det vises både hos kunden og konsulenten.

2.3.5 Operasjonelle krav, kundemodul

- Minimumskravet bør være ISDN-linje, men ADSL 384/128 Kbit/s nedlasting kan være anbefalt
- Optimalisert for 1024x768
- Må kjøre i IE6+, Opera 7.x
- Hastigheten bør være såpass god at kunden slipper å sitte og vente på at informasjonen lastes.
- Alle "chat" data bør være lett tilgjengelig.
- Ved "refresh" må dette gjøres på en slik måte at det ikke virker forstyrrende for kunden.
- Kundene skal ikke behøve å installere noen spesielle programmer for å kjøre kundemodulen.
- Brukergrensesnittet skal være enkelt og forståelig, og skal kunne brukes av alle som har tilgang til Internett.
- Brukergruppen vil bestå av både avanserte og lite tekniske brukere.

2.4 Kravspesifikasjon konsulentmodul

Bruksgruppen består av kundebehandlere, som normalt sett har noe høyere kunnskap enn det den generelle bruker er i besittelse av. Kravet til brukervennligheten er dermed noe lavere enn det som kreves av kundemodulen, men kravet til funksjonalitet er selvsagt høyere.

Kravene til konsulentmodulen er mange og på mange forskjellige detaljnivåer. Vi har derfor valgt å lage Use Case diagram og beskrive dette i tillegg til vanlig skriftelig beskrivelse.

2.4.1 Overordnede funksjonelle krav

Brukervennlighet:

- Systemet skal være brukervennlig og intuitivt slik at en konsulent kan bruke det i løpet av kort tid uten spesiell opplæring
- Hjelpfunksjonene skal være i form av hint når musepilen holdes over knappene.

Ved opprettelse av profil:

- Det skal til enhver tid være mulig å opprette nye profiler. Når disse profilene registreres blir de identifisert med en automatisk generert ansatt ID.
- Det skal ikke være mulig å legge inn en konsulent uten at navn og passordfeltene er utfylt.
- Passord skal krypteres med MD5 kryptering.
- Etter opprettelse av profil skal konsulenten blir returnert til innloggingsvindu.

Ved innlogging:

- Konsulentene må logge seg inn for å benytte systemet.
- En konsulent skal logge seg på systemet med brukernavnet sitt og selvvalgt passord.
- Etter vellykket login, skal konsulenten komme direkte inn i hovedvinduet, men status skal være satt til inaktiv.

Ved modifikasjon av profil:

- En konsulent skal kunne endre sin profil når han er innlogget på systemet.
- Allerede eksisterende data om konsulenten skal vises når vinduet åpnes.

- Konsulenten skal kunne lagre endringer og fortsette arbeidet uten noe restarting og ny innlogging.
- Endringene skal lagres direkte i databasen og gamle verdier skal slettes.
- Endringer som gjøres når status er satt til aktiv, skal ikke påvirke de som allerede ligger i køen.

Ved aktivisering av profil:

- En konsulent skal kunne sette sin status til aktiv, det vil si at han er tilgjengelig for henvendelser fra kunder. Etter at statusen er satt til aktiv, dukker det automatisk opp henvendelser.

Ved organisering av kø og opprettelse av samtaler:

- En henvendelse blir lagt i køen til den konsulenten som har satt sin status til aktiv, har de rette kvalifikasjonene og har kortest kø av konsulentene.
- Når en henvendelse blir lagt i køen til en konsulent, registreres det et kønummer på henvendelsen som sier hvilket nummer henvendelsen er i køen.
- Når konsulenten vil starte en samtale, skal det kun være nødvendig å trykke på en knapp og så åpnes et samtalevindu til den kunden som ligger øverst i køen. Det skal være mulig å åpne inntil 20 slike samtalevinduer samtidig.
- Hvert samtalevindu skal være instanser av ett og samme program som skal kjøre uavhengig av hverandre.
- Når en samtale startes, skal kønummet til de andre i køen automatisk telles ned med 1.
- Selv om konsulentens status blir satt til inaktiv, skal han fortsatt kunne betjene de som allerede ligger i køen.

Ved samtaler:

- En konsulent skal til enhver tid se informasjon om hvem kunden er og hva som er hovedproblemet. Det skal også vises en support ID som brukes for å identifisere samtalen.
- Det skal være mulighet for å hente standardsvar og sende disse til klienten for å lette jobben ved hyppige stilte spørsmål.

- Det skal være mulig å sende en henvendelse videre til en annen konsulent etter at samtalen er i gang.

Øvrig funksjonalitet:

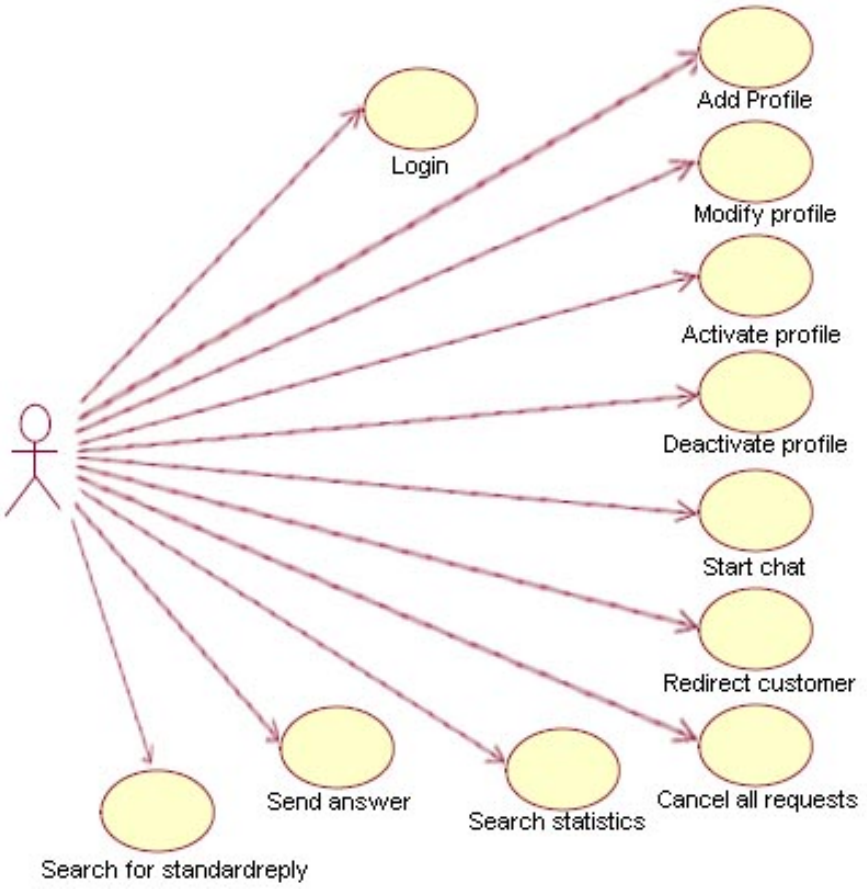
- Det skal være en mulighet for å tømme køen. Dette skjer når en konsulent trykker på en knapp i hovedvinduet. Da skal det gis beskjed til samtlige kunder om at ett problem har oppstått, og at de vennligst må prøve igjen senere.
- Det skal være mulig å lete gjennom gamle samtaler og få opp litt statistikk på forskjellige kategorier.
- Gamle samtaler skal kunne gjenåpnes, og det skal være mulig å fortsette samtalen ved å ta kontakt med kunden.

2.4.2 Krav til grafisk brukergrensesnitt

Vi tar utgangspunkt i at systemet skal være så selvforklarende som overhode mulig. Det vil si at alle knapper har passende navn og passende ikoner.

Det er mennesker med høy kompetanse innen data som skal bruke konsulentmodulen og de stiller desto høyere krav til at systemet skal væres selvforklarende. Det skal ikke være for mange navigasjonsnivåer slik at man kan gå seg bort i applikasjonen, men heller litt mer funksjonalitet i hovedvinduet.
Designet på denne modulen er det vi som skal uforme, og den skal brukes uten endringer fra WebDeal sin side. Det er derfor viktig for WebDeal at dette er pent, konsekvent og oversiktelig.

2.4.3 Use Case diagram



Figur 2.2 Use case konsulent

2.4.4 High-level UseCase-beskrivelser

Use case	Login
Mål	Konsulenten skal få logget seg på systemet
Aktør	Konsulenten
Beskrivelse	En konsulent må logge seg på systemet med navnet sitt og et selvvalgt passord. Det sjekkes opp mot databasen om brukernavn og passord er korrekt før man sendes videre inn i systemet. Hvis det er feil i brukernavn og passord gis det beskjed om dette.

Use case	Add/ Modify profile
Mål	Legge til/ endre konsulentens profiler i systemet
Aktør	Konsulenten
Beskrivelse	En konsulent skal kunne legge til sin profil med sine selvvalgte egenskaper, samt modifisere dem på samme måte. Konsulenten skal også velge brukernavn passord selv, og kan til en hver tid gå inn og endre på disse.

Use case	Activate profile
Mål	Konsulenten får aktivert sin profil og kan ta imot henvendelser
Aktør	Konsulenten
Beskrivelse	En konsulent må sette sin status til aktiv etter han er logget inn for at det skal kunne komme henvendelser inn i hans kø. Dette gjøres ved å trykke på en enkelt knapp.

Use case	Deactivate profile
Mål	Deaktiverer sin profil for å ikke lenger kunne motta henvendelser
Aktør	Konsulenten
Beskrivelse	En konsulent skal ha muligheten til å sette sin status til inaktiv ved trykke på en knapp. Konsulenten kan da ikke lenger ta imot nye henvendelser, men kan betjene ferdig de som allerede ligger i køen.

Use case	Start chat
Mål	Få åpnet et samtalevindu med kunden
Aktør	Konsulenten
Beskrivelse	For å få åpnet et chat-vindu med en kunde trykker du på en enkelt knapp og så åpnes et samtalevindu med den kunden som ligger øverst i køen til konsulenten.

Use case	Redirect customer
Mål	Sende en kundesamtale videre til en annen konsulent
Aktør	Konsulenten
Beskrivelse	Hvis en konsulent under en samtale oppdager at en annen konsulent har bedre forutsetninger for å bistå denne kunden har han mulighet til å sende kunden videre. Dette gjøres ved å trykke en knapp i chat-vinduet og skrive inn hvilken konsulent du vil sende samtalen til.

Use case	Cancel all requests
Mål	Avbryte alle henvendelsene til en konsulent
Aktør	Konsulenten
Beskrivelse	Hvis det oppstår en situasjon der en konsulent må avslutte alle sine samtaler, har han muligheten til å trykke på en ”nødstopp-knapp”. Da vil alle kundene som ligger i køen få beskjed om at den situasjon har oppstått og at de vennligst må prøve igjen senere.

Use case	Search statistics
Mål	Få statistikk over avsluttede samtaler
Aktør	Konsulenten
Beskrivelse	En konsulent skal ha mulighet på å få vist informasjon om alle samtalene som er lagt inn i databasen. Det skal være mulighet til å søke på språk, kategori og konsulent. Postene som passer til kravene vil vises.

Use case	Send answer
Mål	Konsulenten skal kunne føre en samtale med kunden
Aktør	Konsulenten
Beskrivelse	Når et samtalevindu åpnes skal konsulenten ha mulighet til å kommunisere med kunden ved å skrive inn svar og spørsmål i et tekstfelt og sende dette. Teksten vil da vises både hos konsulenten og kunden.

Use case	Search for Standardreply
Mål	Konsulenten skal lete frem standardsvar og sende dem til kunden
Aktør	Konsulenten
Beskrivelse	En kundekonsulent skal ha mulighet til å lete frem standardsvar ved å søke i en database. Han kan søke på ord som finnes i tittelen eller i selve standardsvaret. Disse vise i en liste og konsulenten har muligheten til å velge en av disse.

2.4.5 Operasjonelle krav konsulentmodul

- Optimalisert for 1024x768 pixels oppløsning
- Skal kjøre på Windows 2000 eller nyere versjoner av Windows
- Systemet skal kjøre på en datamaskin med 256 MB RAM eller mer
- Konsulenten må ha mulighet til å holde 20 chat-vinduer oppe samtidig.
- Ved "refresh" av vindu må dette gjøres på en måte som det ikke virker forstyrrende eller gir lavere effektivitet for konsulenten.
- For sikkerhets skyld må det være mulig å blokkere IP-adresser, slik at ikke en enkelt person kan oversvømme databasen med "søpledata".
- Systemet må være robust slik at det er mulig å opprette ny "session" for en bruker, legge vedkommende inn i kø og velge denne fremfor de andre i køen for å gjenoppta "chat".
- Kundebehandler skal kun behøve å installere klienten og motta innloggingsinformasjon fra systemadministrator for å kjøre klienten.

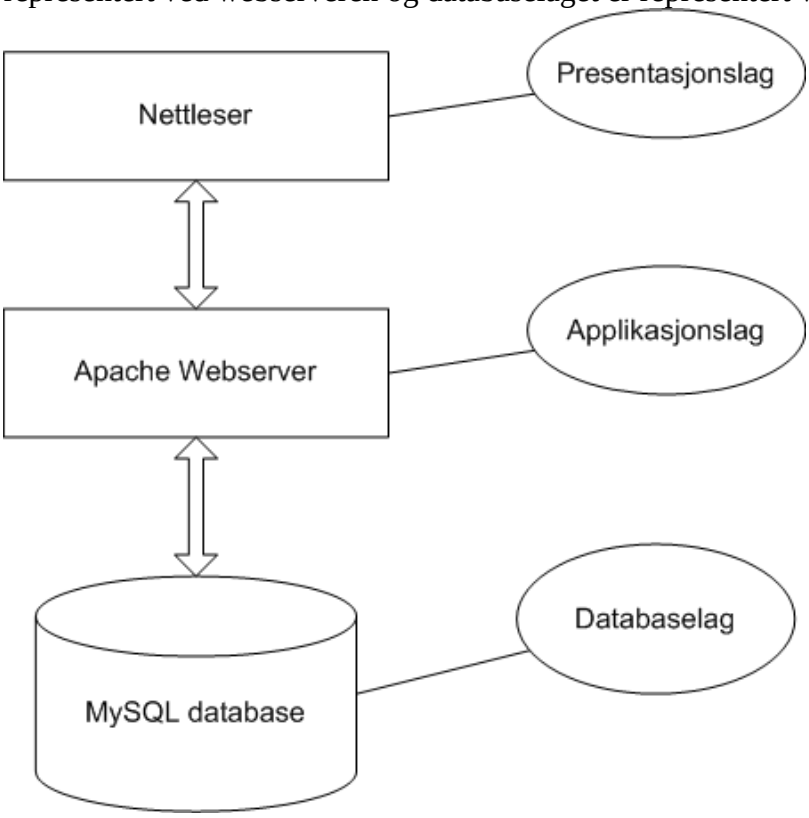
3 Design

3.1 Systemarkitektur

Systemet skal brukes av de ansatte hos WebDeal og deres kunder. Kundene er spredt over hele verden, dermed fant vi det mest hensiktsmessig å utvikle kundemodulen som en webløsning. Det eneste systemet krever av klientene, er at de har tilgang til Internet og en nettleser. WebDeal hadde som krav at konsulentmodulen skulle utvikles i Delphi, noe som medfører at denne applikasjonen må installeres som software på konsulentenes maskiner. Alle felles data holdes på en sentral database som kan nås fra alle moduler.

3.2 Kundemodulen

Kundemodulen av systemet er utviklet i PHP, HTML og JavaScript, og her valgte vi å bruke en tre-lags arkitektur med et presentasjonslag, et applikasjonslag og et databaselag (se figur 3.1). Vi benytter oss av såkalte *tynne klienter*. Denne arkitekturen har flere fordeler, bl.a. er den skalerbar, det er enkelt å foreta endringer og den krever lite av brukerens hardware.
Presentasjonslaget er representert ved brukerens nettleser, applikasjonslaget er representert ved webserveren og databaselaget er representert ved MySQL databasen.

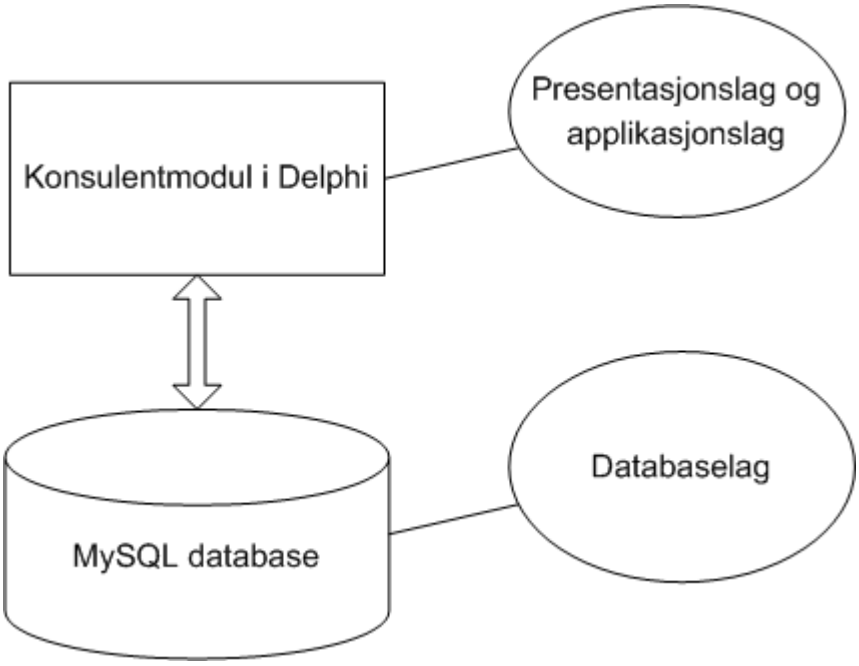


Figur 3.1 Arkitekturen til kundemodulen

Presentasjonslaget innholder grensesnittet der all visuell kommunikasjon med brukeren foregår. På applikasjonslaget ligger alle operasjonene som kundemodulen utfører, og dette laget sender forespørsler til databaselaget for å tilføye og hente ut data fra databasen.

3.3 Konsulentmodulen

I konsulentmodulen av systemet har vi brukt en to-lags arkitektur med et kombinert presentasjonslag/applikasjonslag og et databaselag. Vi benytter oss m.a.o. av en såkalt *tykk klient*. Selv om dette vil føre til mer arbeid ved evt. oppdateringer, kom vi fram til at det var den beste løsningen på oppgaven siden konsulentmodulen kun skal ligge på WebDeals egne PC-er. Kommunikasjonen mellom databasen og applikasjonslaget i konsulentmodulen er til tider svært intensiv, og dermed mener vi det er riktig å utnytte den prosessorkraften som moderne PC-er disponerer.



Figur 3.2 Arkitekturen til konsulentmodulen

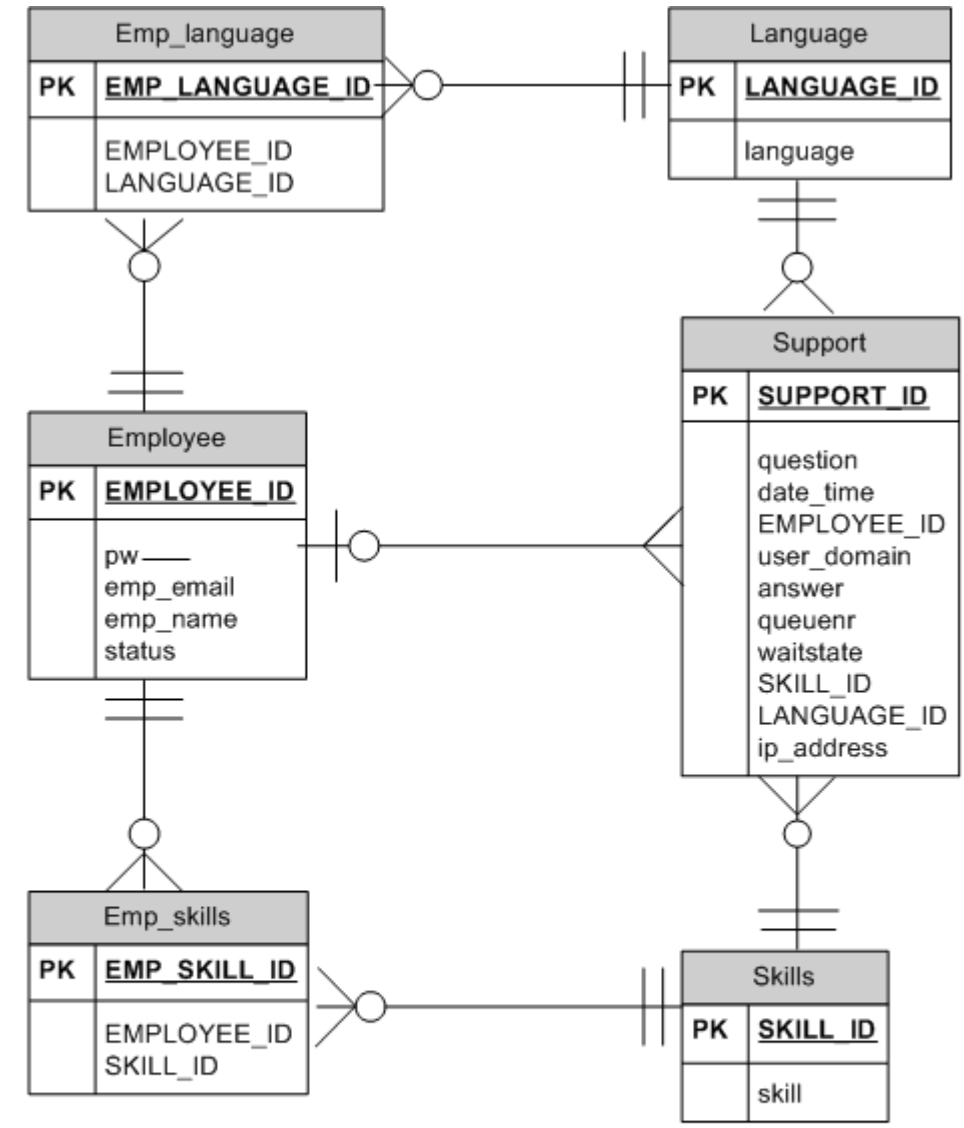
3.4 Databasestruktur

WebDeal ønsket seg et Livesupport-system der samtalene som fant sted mellom kundebehandler og klient ble lagret i en MySQL-database, og databasen ble dermed det sammenbindende ledd i prosjektet. Vi benyttet denne til å holde orden på variabler som skulle benyttes av både kunde- og konsulentapplikasjonen. Databasen er designet fra grunnen av, basert på arbeidsgivers ønsker og krav for systemet. Vi administrerte MySQL-databasen med PHPMyAdmin, via linken http://PHPmyadmin.testweb2.WebDeal.no/

Vi har hele tiden vært opptatt av at databasen skulle være oversiktlig og har prøvd å styre unna overdreven kompleksitet. Eks: I stedet for å lage en tabell for uregistrerte henvendelser, en for henvendelser i kø og en for behandlede henvendelser, har vi laget en felles tabell der en status-variabel avgjør hvor henvendelsen befinner seg i behandlingsprosessen. Vi syntes også det var viktig å unngå redundans slik at ikke databasen ble stor og uoversiktlig. MySQL er ingen relasjonsdatabase, derfor er man uansett avhengig av attributter som er felles for flere av tabellene. Vi har for øvrig passet på så vi kun dobbeltlagret de attributtene som var nødvendige for å knytte tabeller sammen.

Vi har gjort bevisste valg gjennom hele utviklingsprosessen når det gjelder tabeller, attributter og deres datatyper. For å sikre unike identifikatorer for hver post i en tabell, har vi lagt inn prosedyrer i databasen som sørger for at dette ivaretas. Dette er viktig i et flerbrukermiljø der mange kommuniserer mot databasen samtidig. Data er lagret i klartekst, bortsett fra passord-feltet i ”employee”-tabellen, som er lagret på md5-kryptert form.

Figur 3.3 viser en oversikt over databasestrukturen. Det er denne modellen som er grunnlaget for databasen vi har implementert. For nærmere beskrivelse, se vedlegg F.



Figur 3.3 ER-modell for databasen

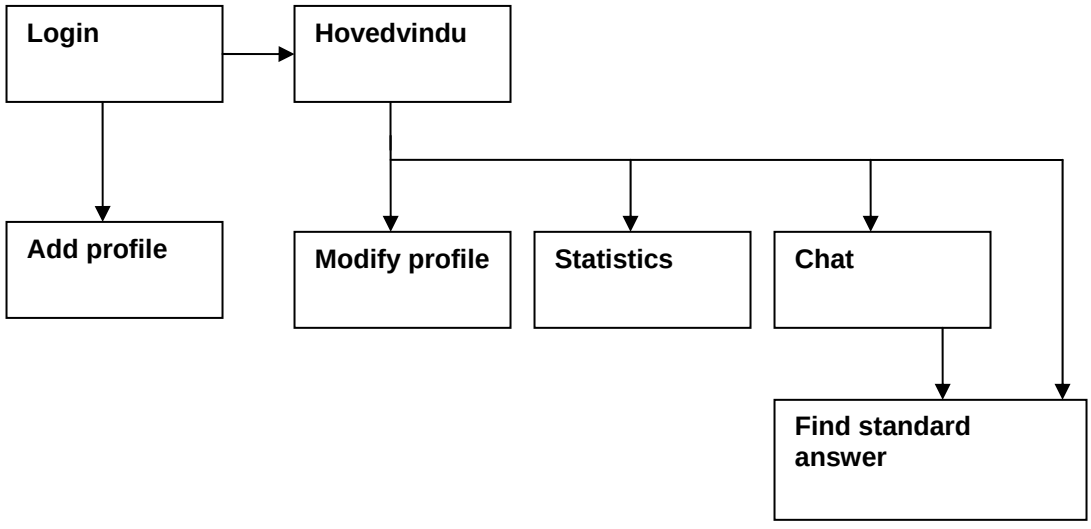
3.5 GUI design

Vi har lagt vekt på at support-programmet skal ha et utseende der brukerne intuitivt forstår hva de skal gjøre. Bortsatt fra å ta hensyn til brukervennlighet, var det ikke noe krav fra WebDeal at vi la stor vekt på grafikk og utforming. WebDeal har dessuten en egen designer som tar seg av utseendet på kundemodulen, og dermed kunne vi se bort fra designet dette.

Det var et krav til systemet at det skulle lages i Delphi og kjøres på Windows-plattformer, dermed ble det et naturlig valg å basere brukergrensesnittet på velkjente Windows-elementer. Vi var dessuten bevisste på å utnytte kunnskap som brukerne allerede besitter, slik at programmet ble enkelt å sette seg inn i. Delphi er et svært brukervennlig utviklingsverktøy når det gjelder grafiske elementer, da man kan tilføre de fleste komponenter ved hjelp av dra og slipp metoden.

Når det gjelder oppdelingen av skjermbildene i konsulentmodulen, valgte vi å ha forskjellige skjermbilder for forskjellig funksjonalitet. Vi mente dette ville være bedre enn å dytte all mulig funksjonalitet inn i et enkelt skjermbilde, og delte isteden vinduene inn i et statistikk-vindu, et hovedvindu, et chat-vindu osv. Det er mulig å ha oppe flere vinduer samtidig, noe som er en forutsetning for at programmet skal fungere som planlagt.

Det viktigste vinduet er hovedvinduet, for herfra kan du nå all annen funksjonalitet du måtte trenge gjennom standard windows-menyer. Konsulentmodulen har en ganske flat navigasjonsstruktur, der utgangspunktet tas i dette vinduet. Det er ikke på noe tidspunkt mulig å komme lenger enn tre nivåer nedover i navigasjonshierarkiet. I kundemodulen opererer vi i realiteten ikke med noe navigasjonshierarki, da man automatisk blir sendt videre fra en side til en annen.



Figur 3.4 Navigasjonskart for konsulentmodul

Vi har valgt å ha lyseblå bakgrunner til alle vinduene i programmet, da vi syns dette er en behagelig bakgrunnsfarge. Logoen til WebDeal er plassert på passende steder for å gi applikasjonen firmaprofil. Skrifttypen og skriftstørrelsen i de forskjellige vinduene tilsvarer hverandre, samt at vi har valgt å minimere bruken av forskjellige fonter. Vi mener dette hjelper til med å holde applikasjonen ren og oversiktlig. Vi har også benyttet oss av ”musepeker-hint” som skal hjelpe brukeren til å forstå hvilke muligheter hun/han har på mange av komponentene, isteden for å benytte oss av en egen hjelp-fil.

Generelt kan man si om designet at vi har vært bevisste på å utvikle en brukervennlig applikasjon, og at funksjonaliteten hele tiden har kommet i første rekke.

3.6 Alternative valg

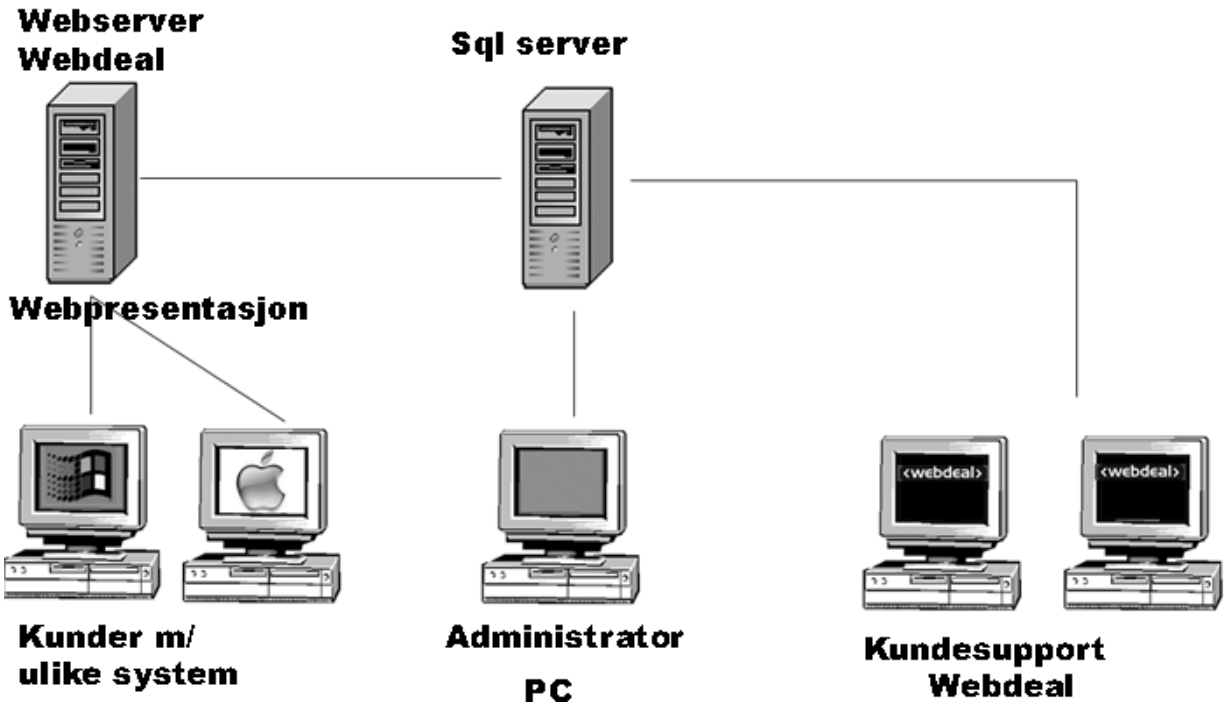
Helt i startfasen vurderte vi å programmere konsulentmodulen i PHP og benytte oss av tynne klienter også her. Hovedsakelig fordi vi ikke hadde kjennskap til Delphi og fordi PHP er et språk som fungerer godt opp mot MySQL-databaser. Etter å ha konferert med WebDeal om dette, kom det fram at det eneste reelle alternativet var tykke klienter i form av applikasjoner som skulle installeres på konsulentenes maskiner. WebDeal begrunnet dette med at databasekommunikasjonen ville gå raskere med denne metoden siden konsulentmodulen til tider jobber intensivt mot databasen. Vi sitter fortsatt med inntrykket av at dette kunne vært løst på en tilfredsstillende måte også i PHP, blant annet hadde vi sluppet alle problemene med kommunikasjonen Delphi - MySQL.

4 Implementering og koding

4.1 Systembeskrivelse

Support-programmet skulle bestå av to moduler; en del for kundebehandlerne og en for klientene. WebDeal ønsket at den delen de skulle benytte skulle lages i Delphi for å få en rask kommunikasjon med databasen. Selve databasen skulle være en MySQL-database som kunne opprettes og editeres ved hjelp av programmet PHPMyAdmin. Klientdelen valgte vi å programmere i PHP, da dette er et programmeringsspråk som har god støtte for kommunikasjon med MySQL-databaser.

Grunntanken ved systemet var at WebDeals konsulenter skulle kunne behandle forespørsler på en rask og effektiv måte, samtidig som all relevant informasjon ble lagret i en database. Klienten henvender seg vha. et webskjema der et PHP-script vil matche klientens preferanser med tilgjengelige konsulenter; deretter legges det inn en ny support-forespørsel og brukeren sendes videre til en *php*-side som skal stå og sjekke databasen for å finne ut hvor langt forespørselen er kommet i behandlingen. Konsulentmodulen vil med jamne mellomrom sjekke databasen etter nye henvendelser til den aktuelle konsulenten, og vi valgte å bruke tre forskjellige statuser for å betegne hvor langt en forespørsel var kommet i behandlingen: Uregistrert, registrert, samt behandlet. Når en kunde legger inn en forespørsel, får denne statusen ”uregistrert” (verdien 0) helt til Delphi-applikasjonen til den aktuelle kundebehandleren registrerer henvendelsen og setter statusen i support-tabellen til ”registrert” (verdien 1). Et PHP-script på klientsida vil registrere når dette skjer, og gi melding om hvilket nummer klienten har i køen. Når henvendelsen er klar til behandling og chatten åpnes, blir statusen satt til ”behandlet” (verdien 2). Når dette oppdages av klient-scriptet, vil brukeren automatisk bli videresendt til et chat-vindu, og samtalen kan begynne. Dette er hovedstrukturen til systemet, i tillegg kommer ekstra funksjonalitet i konsulentmodulen. Modulen består av flere separate vinduer (frames) som er lenket sammen til en applikasjon.



Figur 4.1 Modell av systemet

4.2 Konsulentmodulen

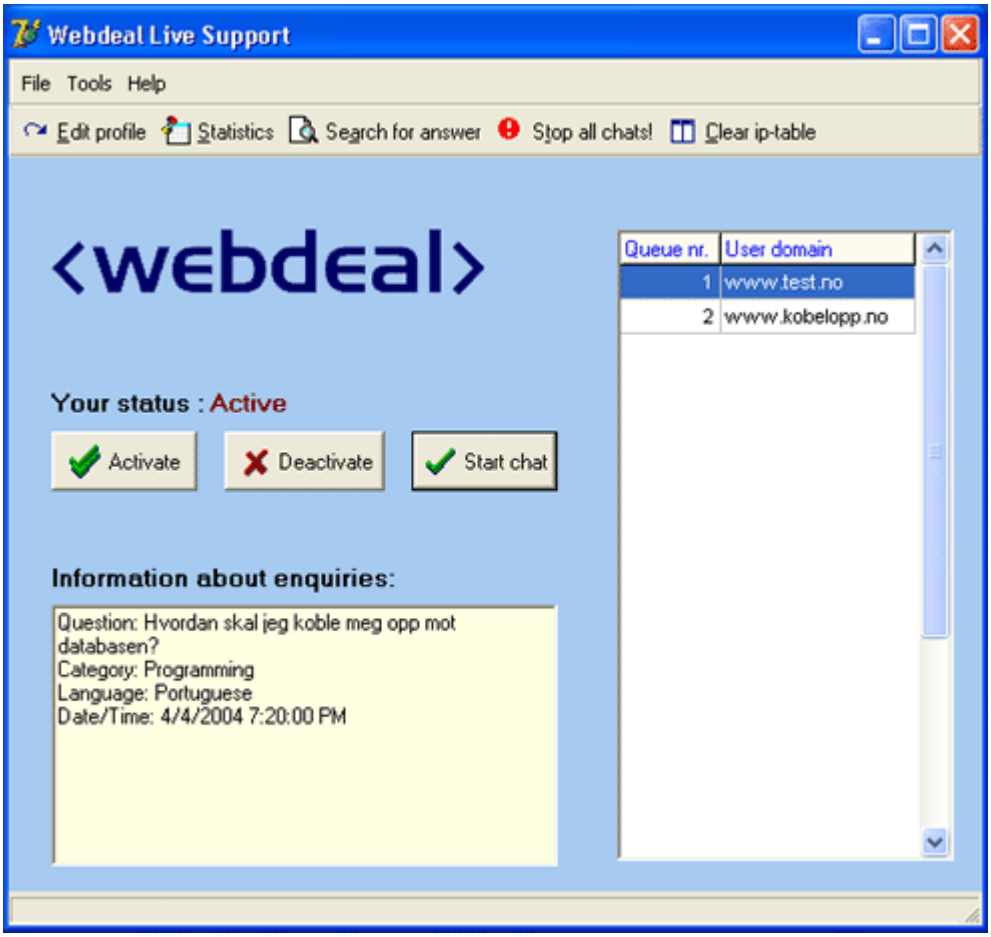
4.2.1 Programmeringsspråk

Konsulentmodulen er utviklet ved hjelp av Delphi 7.0, et Windows-basert programmeringsmiljø. I utgangspunktet ønsket vi ikke å programmere i Delphi, siden to av gruppens tre medlemmer var blanke på området. Vi mente det var en ulempe med plattformavhengighet og foreslo bruk av PHP også i konsulentmodulen. WebDeal gjorde det forøvrig klart for oss at Delphi var det eneste reelle alternativet. Delphi hadde god støtte for Interbase og Access (noe vi kjente litt til fra tidligere) men MySQL -støtte var derimot et helt annet kapittel, noe som viste seg å bli en formidabel utfordring. Etter hvert som vi har blitt kjent med dette utviklingsmiljøet, har vi lært oss å sette pris på fordeler som brukervennlig ”dra-og-slipp” funksjonalitet, mange forhåndsdefinerte komponenter og innebygd debugging. Med unntak av noen få kodeeksempler fra oppdragsgiver, har vi bygd opp all funksjonalitet i konsulentmodulen fra grunnen av.

4.2.2 Hovedvindu for WebDeal Internet Support

I dette vinduet ligger en del grunnleggende funksjonalitet som kundebehandlerne trenger for å gjøre jobben sin. Det er denne applikasjonen som holder orden på rekkefølgen kundene skal behandles i, og den vil kontinuerlig sjekke databasen etter nye forespørsler. Fra dette vinduet åpnes chatten man bruker for å konferere med kundene, samt at det

fungerer som ”portal” til tilleggsfunksjonalitet som statistikk og profilendring. Meningen er at dette vinduet skal gi brukeren oversikt over sine muligheter. Grafisk består hovedvinduet av en standard meny med diverse funksjonalitet, en tabell med oversikt over køen, en ramme med informasjon om de forskjellige henvendelsene i køen og diverse knapper.



Figur 4.2 Skjermbilde hovedvindu

Når man logges inn i systemet, henter man id’en til den ansatte fra ”login”- vinduet. Denne brukes til å finne ut hvilke av henvendelsene i databasen som er adressert til den aktuelle kundebehandleren. Når hovedvinduet åpnes, sjekkes det først om det ligger en kø og venter på den ansatte, og evt. hvor mange det er i køen. Dette blir lagret i en global integer-verdi (I) fordi programmet må vite hvilke plassering i køen den neste supportforespørselen skal få. Deretter vil en timer oppdatere vinduet hvert 10. sekund. Hver gang dette skjer kjøres en funksjon som sjekker etter nye henvendelser til den ansatte i databasen. Deretter sjekkes det om køen vil få mer enn 20 elementer, og dersom dette ikke er tilfelle legges de nye henvendelsene inn i køen på plassen som blir indikert av variabelen ”I”. ”I” blir telt opp med 1 hver gang vi gjør en slik update-operasjon.

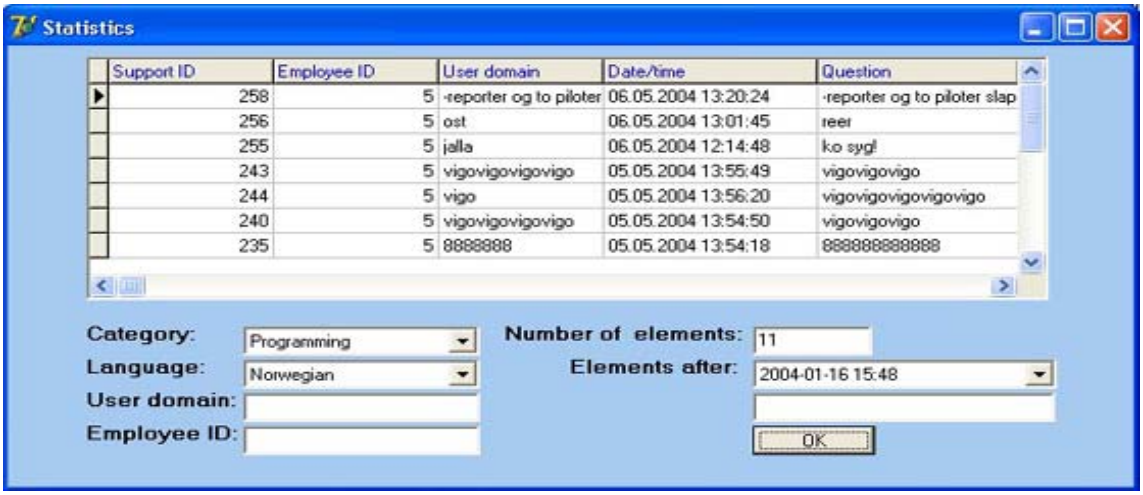
I køen ligger elementene som returneres til en DBGrid etter spørringen. Dersom man trykker på et av elementene i DBGrid'en (onColEnter), vises det informasjon om henvendelsen i et Memo-felt.

Kundebehandleren har mulighet for å sette statusen sin til aktiv eller inaktiv, alt ettersom hun/han har tid til å behandle kunder (status settes i en global variabel). Når statusen er inaktiv vil timeren være slått av og køen oppdateres ikke med flere henvendelser. Når man lukker applikasjonen vil statusen automatisk bli satt til inaktiv, og det er opp til kundebehandleren å sette statusen til aktiv når hun/han logger seg på. Dette muliggjør at man kan være inne på systemet uten å betjene kunder. Fra hovedvinduet kan man også nullstille tabellen der de ignorerte ip-adressene ligger. Det er ikke meningen at ip-adresser skal ignoreres permanent, bare for et visst tidsrom.
Hovedvinduet har også en meny der man enkelt kan nå annen funksjonalitet, som f.eks. "statistics" og "modify profile". Denne menyen er laget ved hjelp av *ActionManager*.
Fra hovedvinduet kan man også kreere og åpne opptil 20 ulike instanser av chat-vinduet.

Å få tak i hva som skjedde når man hentet ut poster fra databasen var noe av det vi strevde mest med. Det var vanskelig å forstå hvorfor dette måtte gjøres så tungvint i Delphi/MySQL (med mange forbindelsesledd og oppkoblinger). Fra før hadde vi erfaring med klient- og serversideprogrammeringsspråk som ASP og PHP, der dette gjøres på en mer oversiktlig måte. Vi ønsket for eksempel å kunne bla igjennom support-id'ene til forespørslene som enda ikke var lagt inn i køen, og strevde lenge med dette før vi fant ut at vi måtte gå omveien om en DBGrid. Vi har derfor benyttet oss mye av denne komponenten som mellomlagringsledd for uthentede verdier. Det var i det hele tatt implementeringen av køen som var "akilleshælen" til hovedvinduet. Det tok oss lang tid å finne ut av hvordan man skulle løse dette, og i den endelige løsningen benyttes databasen til å holde orden på rekkefølge og status framfor å benytte et array.

4.2.3 Statistikk

Dette vinduet gir de ansatte statistiske opplysninger om kundebehandlingsprosessen. De kan for eksempel finne ut hvor mange henvendelser innen kategorien ”database” man har hatt den siste uka, eller hvor mange henvendelser en angitt ansatt har fått fra en angitt kunde. Man har fem forskjellige valgmuligheter (språk, kategori, ansatt-ID, user domain og dato) og velger selv hvor mange av disse man vil skal telle når det gjøres et oppslag i databasen. Ansatt-ID og user domain må skrives inn i tekstfelt, kategori og språk har hver sin drop-down boks, mens vi bruker en Delphi-komponent kalt DateTimePicker for å velge dato.



Figur 4.3 Skjermbilde statistikk

Denne applikasjonen er i bunn og grunn en søkefunksjon. For hvert eneste søk gjøres en henvendelse til databasen, som deretter returnerer de aktuelle postene inn i en DBGrid. Vi har også valgt å lage et felt som viser nummeret på poster som ble returnert. Dette kan for eksempel være hendig hvis man ønsker å sammenligne hvor mange saker de ansatte har behandlet. Når vinduet åpnes gjøres også spørringer til skills/language-tabellene om hvilke kategorier og språk som skal kunne velges; dette gjør applikasjonen mer dynamisk i og med at kategorier og språk ikke er definert i selve kildekoden. Resultatet fra disse spørringene legges rett inn i to DBLookuPComboBox’er. Her brukes parameterne ”KeyField” og ”ListField” til å tilordne hvert element riktig verdi og navn.

DateTimePicker er en innebygd Delphi-komponent som gir deg muligheten til å velge mellom forskjellige datoer på en kalender. Vi måtte endre datoformatet før spørringen for at det skulle passe med ”timedate”-formatet som brukes i databasen. Vi hadde litt vansker med å finne en funksjon som gjorde dette, men fant til slutt FormatDateTime-funksjonen på en Internett-side:
Edit4.Text := FormatDateTime('yyyy-MM-dd HH:mm', DateTimePicker1.DateTime);
Edit4.Text := Edit4.Text + ':00';
DateTimePicker tilbød ingen mulighet til å inkludere sekunder i formatet, dette måtte derfor skjøtes på som en streng etterpå. Under datovelgeren er et tekstfelt som viser den

datoen som er valgt. Hvis brukeren lar dette feltet stå blankt, vil applikasjonen gjøre spørringen uten å ta hensyn til dato (brukeren må m.a.o. blanke ut dette feltet selv). Dette er ingen optimal måte å løse dette på, men datovelgeren tilbyr ingen muligheter for å velge en blank dato, dermed ble dette et greit alternativ.

Selve søkestrengen som sendes til databasen settes sammen etter at brukeren har trykt ”ok”, og er avhengig av hvor mange valgmuligheter brukeren ønsker å benytte i søket. For hvert valg brukeren har gjort skjøtes en ny streng til uttrykket som blir bundet sammen av forskjellige where- og and - klausuler.

Utsnitt av koden:

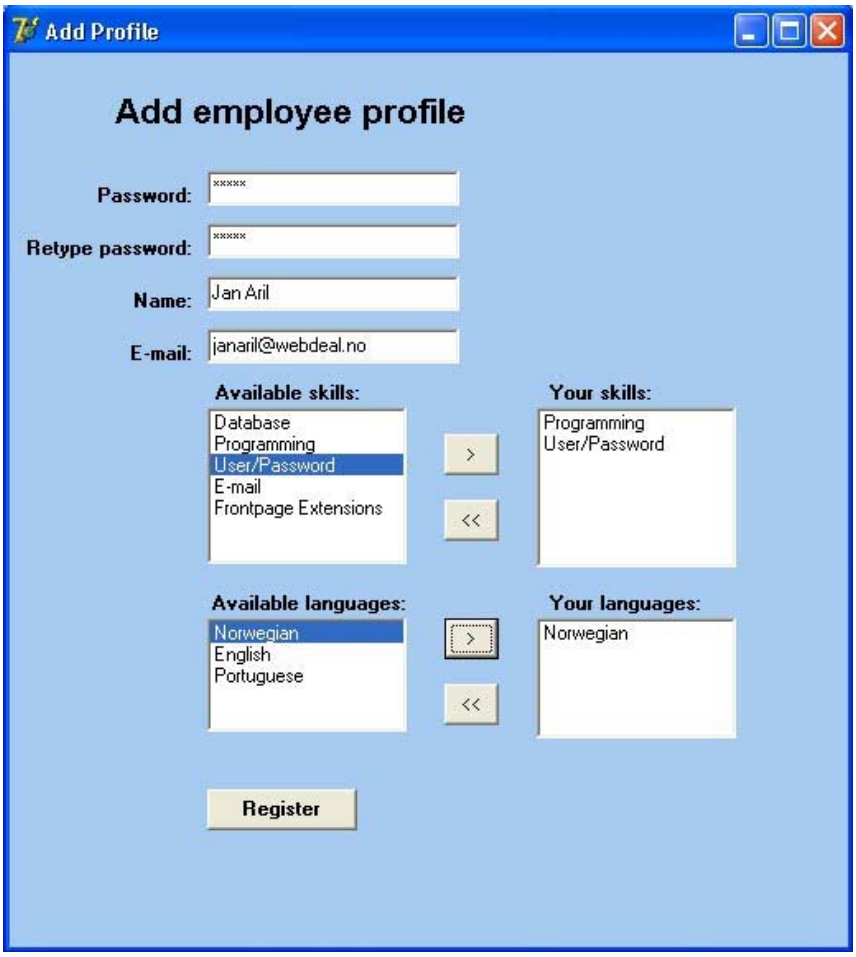
```
Temp2 := 'SELECT *';
if (StrToInt(Cat) = 0) AND (StrToInt(Lang) <> 0) then
  Temp1 := ' FROM support WHERE LANGUAGE_ID='"+Lang+ ""
else if  (StrToInt(Cat) = 0) AND (StrToInt(Lang) = 0) then
  Temp1 := ' FROM support WHERE EMPLOYEE_ID'
else if (StrToInt(Lang) = 0) AND (StrToInt(Cat) <> 0) then
  Temp1 := ' FROM support WHERE CAT_ID = '" +Cat+ ""
else
  Temp1 :=' FROM support WHERE CAT_ID='"+Cat+ '" AND LANGUAGE_ID='"+Lang+ ""';
```

*(*Her er Cat kategori og sier hvilken kategori som er valgt, mens Lang er language og sier hvilke språk som er valgt. *)*

Når uttrykket er ”ferdigbygd”, skjøtes det sammen og det utføres en spørring til databasen. Man gjør også en henvendelse der man bytter ut ”SELECT *” med ”SELECT count(*)” for å få ut antallet poster.

4.2.4 Add Profile

Denne modulen ligger tilgjengelig på innloggingsvinduet. Det vil si at hvis du ikke er registrert som bruker på systemet har du muligheten til å gå inn og registrere deg som bruker før du blir sendt tilbake til innloggingsvinduet og kan logge deg inn. Når du trykker på ”register” knappen”, blir dette vinduet automatisk lukket og ”add profile” vinduet kommer til syne. Her må du registrere deg med navn som brukes under innloggingen og et eget passord i to forskjellige passordfelt. Det testes på om det er fylt inn noe i navn-feltet og om passordene stemmer over ens. Hvis disse kriteriene ikke er oppfylt får brukerne feilmeldinger og data blir ikke lagt inn i databasen. Ellers er det krav til at man fyller inn e-post, ferdigheter og språk.



Figur 4.4 Skjermbilde legg til

I begynnelsen var både ”ferdigheter” og ”språk” avkrysningsbokser, men vi ble rådet av Jan Aril til å ha egne tabeller for disse og bruke DBLookupListBox’er isteden. Dette var et godt råd, men vi hadde store problemer med å få flyttet elementer fra en DBLookuplistBox til en annen. Dette måtte vi bare gi opp og flytter isteden elementer fra en DBLookuplistBox til en vanlig ListBox. Når vinduet åpnes leses det automatisk inn i DBLookupListBox’ene og alle ”ferdigheter” og ”språk” vil være tilgjengelige. Når

brukerne har fylt inn sin informasjon og trykker ”register” blir dataene lagt inn i databasen, vinduet lukkes og login vinduet vises.

For at dataene skal bli lag inn med rette verdier i databasen er det nødvendig å bruke en DBGrid samt noen tilleggsspørringer for å få tak i rette verdier til EMPLOYEE_ID, SKILL_ID og LANGUAGE_ID. Dette er nødvendig for at matchingen opp mot klientenes behov skal fungere på en best mulig måte.
På grunn av dette blir det fem spørringer mot databasen for å hente ut alle data vi trenger og tre for å legge den nye informasjonen inn i databasen.

4.2.5 Modify profile

Denne modulen er tilgjengelig fra hovedvinduet i applikasjonen. Den skal benyttes av kundekonsulentene for å endre data som allerede er lagt inn om dem eller legge inn mer informasjon.

Når konsulentene trykker på denne knappen hentes allerede eksisterende data om dem fra databasen og leses inn i sine respektive felt i grensesnittet. For å få til dette benyttes 5 ulike spørringer opp mot databasen. Dette kan virke som ganske mange bare for å hente ut data om en konsulent, men det er helt nødvendig for å få alle data på rett plass.
Her er det oppstod det også ett problem for her ble vi nødt til å bruke DBLookuplistBox’er for alle feltene som kan romme ferdigheter og språk istedenfor ListBox’er. Dette gjør at vi for hver gang brukerne velger en ny ferdighet eller et nytt språk så må dette legges direkte inn i databasen, selv om brukeren ikke har trykt på knappen ”save changes”, og det gjøres to nye spørringer mot databasen for å oppdatere DBLookupListBox’ene. Hvis de derimot velger å tømme boksen som holder dens ferdigheter, på grunn av at de har valgt feil ferdighet eller lignende, så vil alle registrerte ferdigheter på denne konsulenten slettes og må legges inn på nytt.

Når konsulenten er ferdig med å legge inn endringer og trykker på ”save changes” er det kun endringer i navn, passord og e-post som lagres ved hjelp av en spørring. Vinduet lukkes og vi returnerer til hovedvinduet.

4.2.6 Innloggingsvindu

Innloggingsvinduet er det første vinduet brukeren blir presentert for ved starting av Live-supportsystemet. Hensikten med innloggingsmodulen er på en sikker og effektiv måte å identifisere brukeren. Om man ikke har benyttet systemet før, og ikke har laget en profil, kan en gå rett til ”lag profil” applikasjonen av livesupportsystemet.

Ved klikk på ”Log in” sjekker programmet først om et de to obligatoriske feltene er fylt inn, og hvis ikke stopper programmet med tilbakemelding om hvilket felt som må fylles inn.

Figur 4.5 Skjermbilde innlogging

Er begge feltene fylt inn, sendes brukernavn og passord for verifisering inn til databasen. Dette gjøres gjennom SQL strengen

```
'SELECT count(*) from employee where emp_name like '" + edit1.Text + "' and pw like md5('" + edit2.Text + "') ';
```

Det som skjer her er at innholdet i username boksen sjekkes opp mot det tilsvarende elementet i databasen, altså emp_name. Disse må være identiske.
Deretter sjekkes innholdet i password opp mot de korrekte ”pw-feltet” i databasen. Dette sendes kryptert via MD5 kryptering. Passordene i databasen er også lagret med MD5 kryptering, og innholdet i password feltet vil da kunne matches til ”pw-feltet” i databasen.

Helt i begynnelsen av spørringen står count(*). Dette teller opp hvor mange korrekte autentiseringer denne spørringen gir. Dette er enten 0 eller 1.
Resultatet som gis er 0 om autentiseringen mislyktes og 1 om den var vellykket.
Dette svaret lagres i en DBgrid.

Deretter sjekkes det om count-verdien er lik 1, altså en vellykket brukerverifisering (som detaljert over). Da vises beskjed om korrekt pålogging, og hovedapplikasjonen kalles.

4.2.7 Chatvindu

I chat-vinduet vil all kommunikasjon mellom konsulent og kunde foregå, og er derfor en svært sentral del av applikasjonen. Her presenteres konsulenten for kundens problem, og kan ha en direkte dialog med kunden via et tekstsystem. Her kan konsulenten også blokkere plagsomme brukere via ip, og det er mulighet for å hente ut standardsvar fra database.

Denne applikasjonen kalles fra systemets hovedvindu. Når chat-vinduet starter kjøres en del ulike kommandoer:
Først hentes all relevant informasjon vedrørende saken chatten omhandler. Dette gjøres via en enkel SQL select opp mot databasen, der det matches med kønummer = 1 og den aktive konsulentens ID nr. Informasjonen legges i DBgrid 1.

```
SQLQuery4.SQL.Text := 'SELECT * FROM support WHERE queuenr = 1 AND EMPLOYEE_ID =
''' +IntToStr(Emp_ID)+ ''';
```

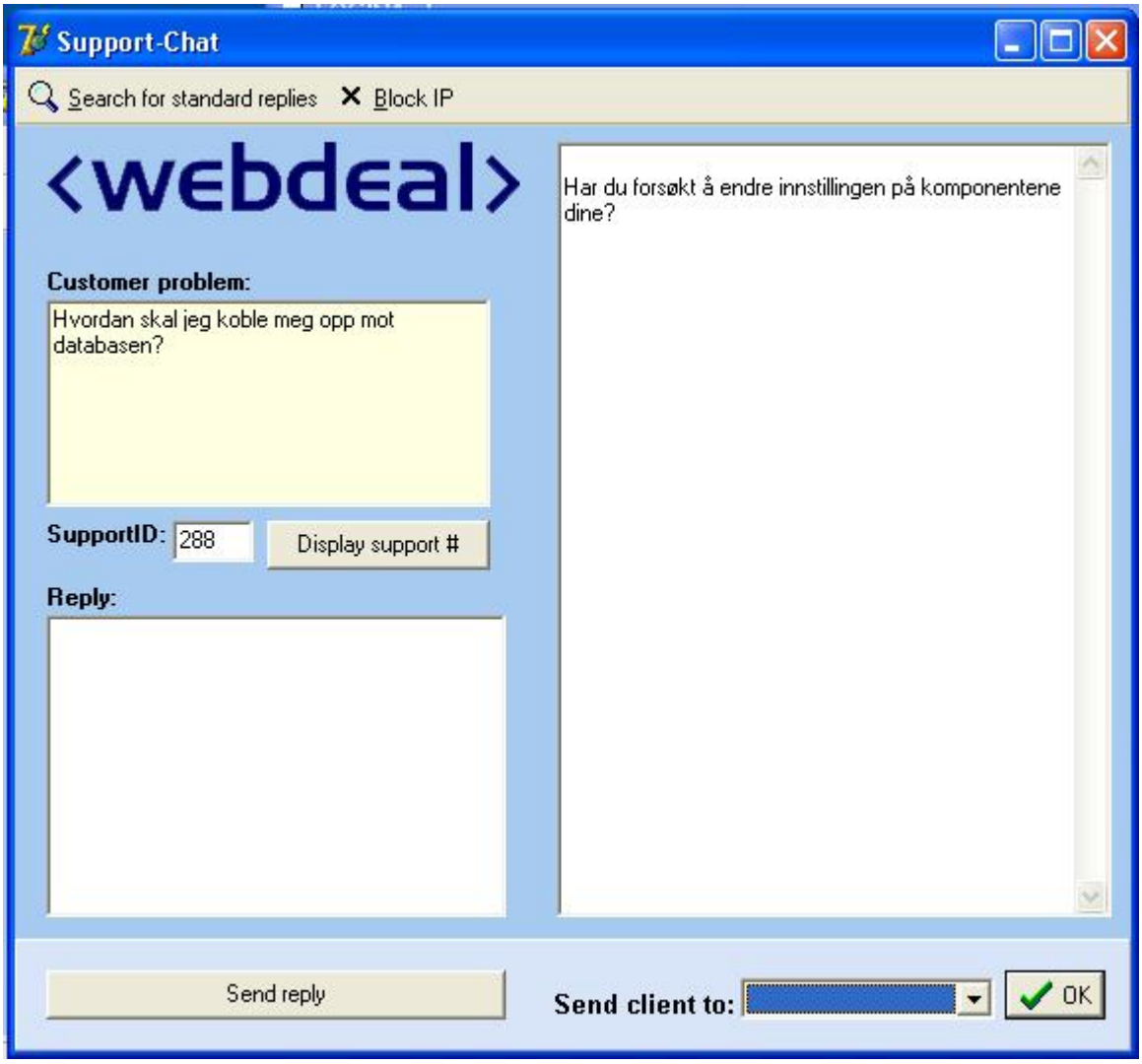
Deretter hentes support ID ut ifra DBgrid og legges i edit-boks 1.
Nå som chatten er i gang og kunden får hjelp, må køen oppdateres, slik at ventende kunder rykker fremover og får tilbakemelding om dette. Kunden som er under behandling må også få sin status satt til 2, altså under behandling, så kunden ikke fortsetter å ta plass i ventekø.

Nå er chatten i gang. Fra øyeblikket applikasjonen åpnes, begynner en timer å gå i programmet. Denne timeren kjører en refresh kommando hvert 6. sekund. (kjører koden til knapp 1, altså display support). På den måten tar det maks 6 sekund fra en melding fra kunden legges til i databasen til den vises hos konsulent.

Både det konsulenten og kunden sender til systemet lagres i et felt i databasen. Her skjøtes teksten konsulenten og kunden skriver inn på samme tekststreng. Denne teksten vises i et tekstfelt hos konsulenten og friskes opp, som nevnt, hvert 6. sekund.

Vi hadde store problemer med å vise siste innlagte tekst. Dette var fordi det var vanskelig å scrolle automatisk ned. Dette ble løst ved bruk av en SCROLLCARET-funksjon som vist under.

```
  memo1.SelStart := length(memo1.Text);
 memo1.Perform(EM_SCROLLCARET,0,0);
```

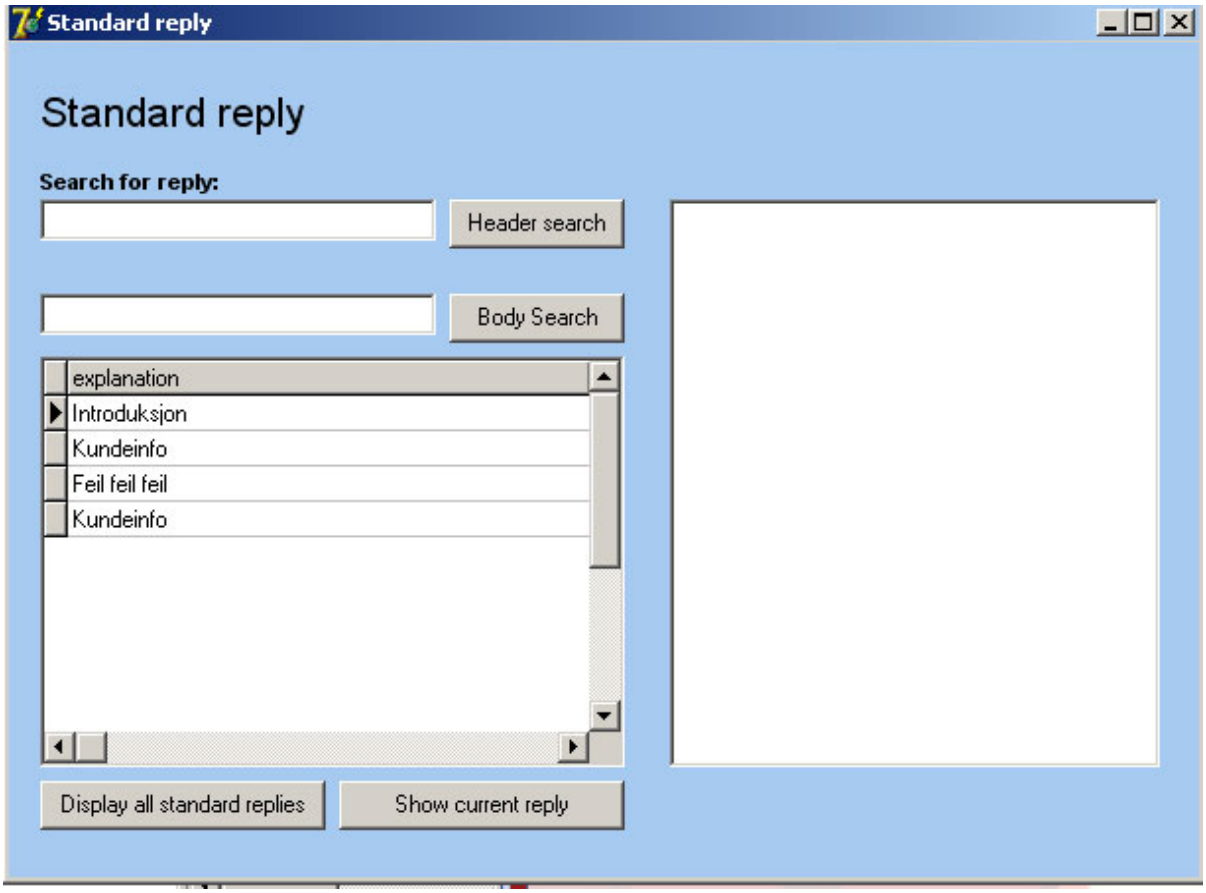


Figur 4.6 Skjermbilde chat-vindu

Det var en viss fare for at kunde og konsulent skulle skrive til tekstfeltet samtidig. Det ville da oppstå en delingskonflikt som kunne låse systemet. Dette var et problem vi måtte løse for å sikre stabiliteten i systemet. Dette gjorde vi ved å låse tabellene før skriving, og åpne de igjen etter skriving. Deretter forhindret vi flere simultane skrivinger til samme element.
Chat-vinduet har også en knapp for å hente opp standardsvarapplikasjonen. Denne detaljeres under.

4.2.8 Finn standardsvar

Dette er en forholdsvis enkel applikasjon som tillater søk i databasen etter standardsvar. Applikasjonen vil tillate konsulenter å søke etter enkeltord i emne og hovedteksten i et standsvar. Deretter kan man klippe ut deler eller hele standardsvaret og lime dette inn i chatten med kunden.



Figur 4.7 Skjermbilde standardsvar

Header search: Denne knappen og tilhørende edit-boks lar konsulent søke i emnet (overskriften) til standardsvarene. Dette er et søk som matcher all tekst som inkluderer denne strengen, altså vil et søk på ”nett” inkludere løsninger med f.eks ”nett”, ”Internett”, ”nettverk” og lignende.

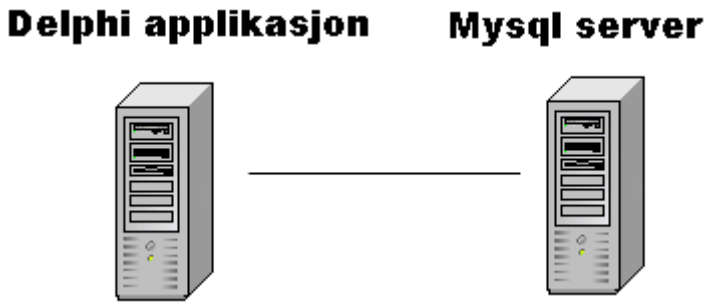
```
SQLQuery1.SQL.Text := 'SELECT explanation, stdanswer from Answers where explanation like
"%' + edit1.text +'%"';;
```

Body search: Denne søker i hovedteksten til standardsvaret. Dette er altså et identisk søk som Free match i header, med unntak av at målet er hovedteksten, ikke emnet (overskriften).

```
SQLQuery1.SQL.Text := 'SELECT explanation, stdanswer from Answers where stdanswer like
"%' + edit2.text +'%"';
```

4.3 Databasekommunikasjon Delphi → MySQL databasen

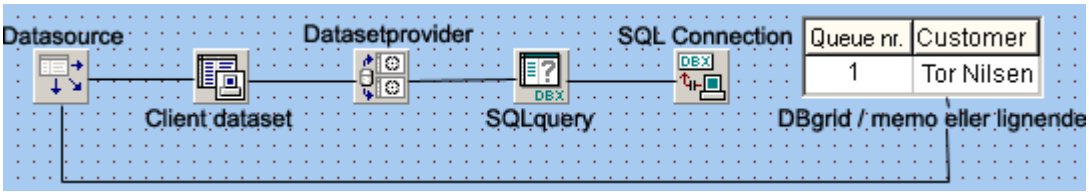
Kommunikasjonen mellom Delphi og MySQL databasen viste seg å være en av de mest krevende oppgavene under konstruksjonen av Live-supportsystemet. Det var gitt klare retningslinjer fra WebDeal om at det var de nevnte systemene som skulle benyttes. En kobling til en slik database gjennom Delphi er noe komplisert, og det kan gjøres mye enklere ved for eksempel å benytte en Access database. En av de første oppgavene som ble gjort under dette prosjektarbeidet var å få til en fungerende toveis kommunikasjon mellom et enkelt Delphi program og en MySQL database kjørende på den samme lokale maskinen.



Figur 4.8 Modell kommunikasjon MySQL - Delphi

Vi fant ut at vi kunne benytte ADO, MyODBC, eller DBExpress.
Etter å ha gjort mange søk, på ulike søkemotorer, lest igjennom mange artikler om lignende emner, fant vi ut at det i teorien var flere måter å gjøre dette på. Problemet var imidlertid å finne et eksempel, eller noen som faktisk hadde en fungerende løsning på problemet. Det var svært vanskelig å finne. Derimot hadde vi vår kontaktperson Jan Aril hos WebDeal. Han satt inne med en funksjonell løsning som han hadde kommet fram til etter svært mye arbeid.
Metoden vi endte opp med besto av en serie av DBExpress komponenter koblet opp mot hverandre i sekvens. En av oss begynte å arbeide med denne metoden, og skrev tidlig i arbeidet et Delphi-program som skulle arbeide opp mot en lokal MySQL-database. Dette var ingen enkelt jobb, og det tok lang tid å få fjernet alle de underlige problemene som oppsto. Alle komponentene måtte kobles korrekt, og ulike parametere måtte settes. I tillegg var det en utfordring å få til brukerverifikasjon på databasen gjennom dette programmet. Omsider fikk vi kommunikasjonen i gang, og en tidlig prototype av programmet kunne benyttes for å teste spørringer.

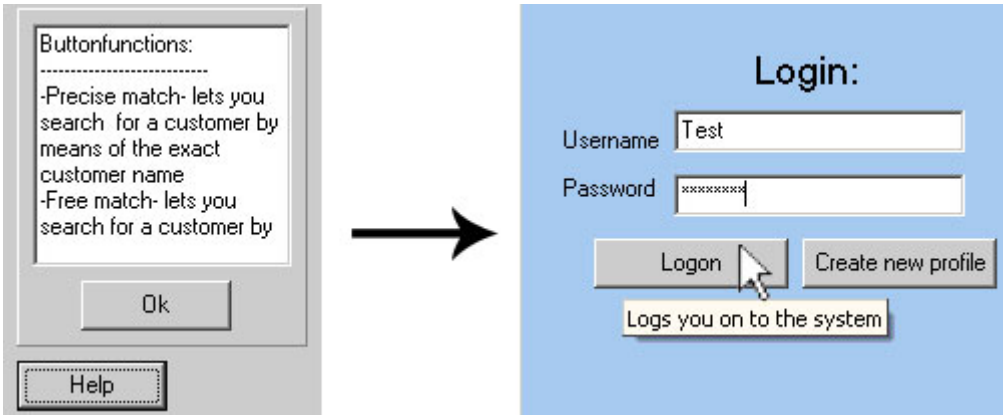
Komponentsekvensen for en enkel spørring besto av en DataSource, et ClientDataSet, en DataSetProvider. Et SQLQuery og en SQLConnection. I tillegg til dette trengtes en visningskomponent for å vise den uthentede informasjonen på skjerm.



Figur 4.9 Skjermbilde databasekomponenter Delphi

4.4 Hjelpfunksjoner

Selv om det er lagt vekt på at programmet skal være selvforklarende i størst mulig grad, hadde vi hele veien tenkt å implementere en hjelp funksjon. I de tidlige utgavene av programmet var det benyttet en enkelt hjelp knapp som viste en tekstboks med informasjon. Dette viste seg å bli noe upraktisk etter hvert som vi testet programmet, og det ble bestemt å endre dette til noe mer oversiktlig og effektivt. I det endelige programmet ble det gjort slik at en holder musepekeren over den knappen man ønsker hjelp til. Da dukker det opp en enkel beskrivende tekst om knappens funksjonalitet og bruk.



Opprinnelig hjelp **Den nye modifiserte hjelpen**

Figur 4.10 Hjelpfunksjon

Denne endringen ble gjort da navigering i en tekstboks og framleting av en enkelt knapp sin funksjon ble for tidskrevende og vanskelig. Vi kom til den konklusjon at siden dette var et program uten gjemt funksjonalitet, som dessuten bare skulle brukes innen WebDeal, var det greit å kutte ut et eget hjelpvindu.

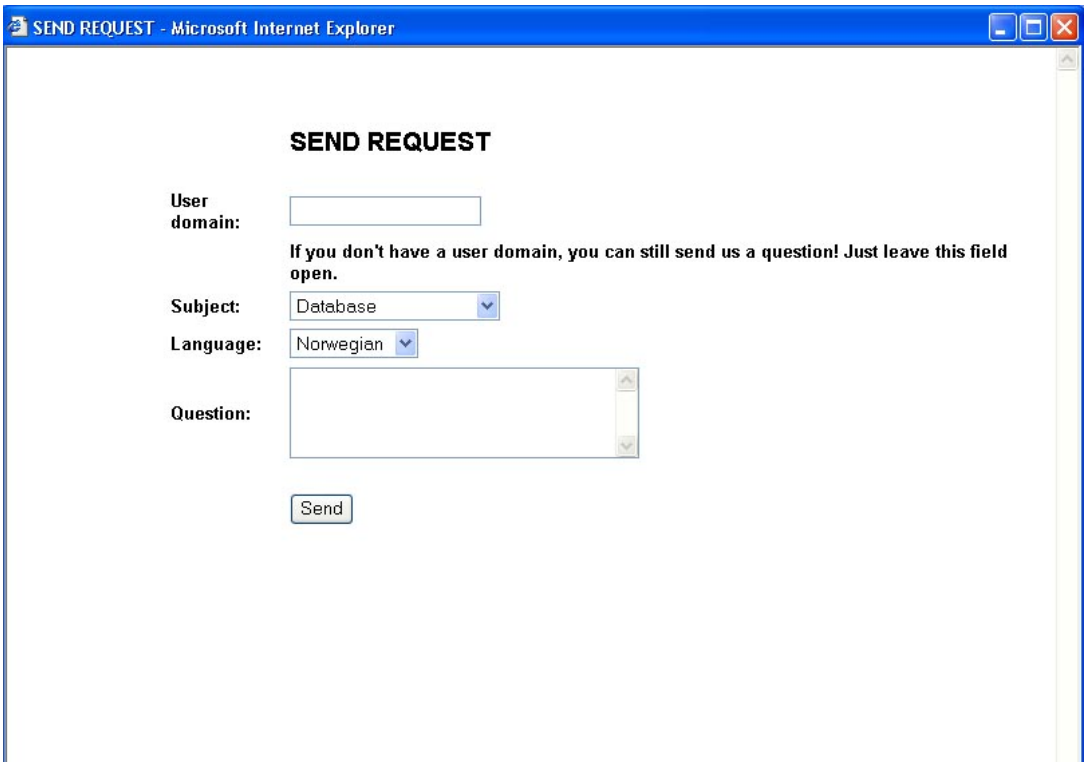
4.5 Implementering av kundemodulen

4.5.1 Programmeringsspråk

Til utvikling av kundemodulen valgte vi å bruke PHP, et skriptspråk for Web-sider. Åpenbare fordeler med PHP er at det er gratis, plattformuavhengig og fungerer med andre åpen-kildekode-prosjekter (som MySQL). To av gruppens medlemmer hadde dessuten erfaring med programmeringsspråket fra før. I tillegg har vi benyttet oss av JavaScript og HTML, to programmeringsspråk vi kjente godt fra andre prosjekter.

4.5.2 Request.php

Dette er det skjemaet som en kunde må fylle ut for å få kontakt med en kundebehandler. Her fyller man inn kundenummer, språk, kategori og spørsmålet man ønsker svar på. Vi har brukt html for å lage skjemaet. Man har muligheten til å la kundenummer være blankt; dette fordi WebDeal ønsker å ta imot henvendelser fra flere enn registrerte kunder. Språk og kategori ligger i drop-down bokser der brukeren blir tvunget til å velge ett av de angitte elementene, dette fordi denne informasjonen skal matches mot hvilken kundebehandler som har den nødvendige kompetansen.



Figur 4.11 Demonstrasjonsgrensensitt request.php

Når brukeren trykker ”send”, kaller siden seg selv og det utføres et database-søk der det sjekkes om brukerens ip-adresse finnes i tabellen med ignorerte ip-adresser. Ip-adressen til brukeren hentes ved hjelp av en PHP-funksjon dedikert til formålet. Det sjekkes også

om maskinen ligger bak en Proxy. Hvis ip-en er blant de ignorerte ip-adressene, får brukeren melding om å prøve igjen senere og ingen henvendelse registreres i databasen; i motsatt tilfelle sjekkes det om det finnes en ansatt med angitte kvalifikasjoner som er tilgjengelig for øyeblikket:

```
MySQL_select_db("live1-test", $db);
$SQL2 = "SELECT * FROM emp_skills,employee,emp_language WHERE SKILL_ID=$category
AND employee.EMPLOYEE_ID=emp_skills.EMPLOYEE_ID AND LANGUAGE_ID=$language
AND emp_language.EMPLOYEE_ID=employee.EMPLOYEE_ID AND status=1";
$result2 = MySQL_query($SQL2);
$row2 = MySQL_fetch_array($result2);
$empid = $row2[EMPLOYEE_ID];
        if($empid != 0){
        $SQL = "INSERT INTO support(USER_ID, question, LANGUAGE_ID, CAT_ID,
        date_time, EMPLOYEE_ID, ip_address) VALUES('$userID', '$question', '$language',
        '$category', FROM_UNIXTIME(UNIX_TIMESTAMP()), '$empid', '$ip')";
        $result = MySQL_query($SQL);
```

Dersom dette ikke er tilfelle, får brukeren beskjed om at ingen kundebehandler er tilgjengelig. Ellers blir henvendelsen lagt inn i support-tabellen med employee_ID lik den matchede ID'en og waitstate satt til 0. Når henvendelsen er sendt til databasen, blir brukeren automatisk sendt til en ny side (wait.php) ved hjelp av en JavaScript-funksjon.

4.5.3 Wait.php

Dette er den siden som brukeren/kunden blir sendt til etter å ha lagt inn en henvendelse i databasen. Den står og oppdaterer seg selv helt til en kundebehandler er klar til å hjelpe kunden.

Som nevnt tidligere har en support-henvendelse waitstate satt til 0 før kundebehandlerens applikasjon har registrert den. Når henvendelsen er registrert og lagt inn i køen får den waitstate satt til 1, og når henvendelsen er klar til å behandles får den waitstate satt til 2. Dette benytter vi oss av når vi står og looper på denne siden. Hver gang siden oppfriskes, gjøres det en ny henvendelse til databasen der det blir sjekket om forespørselen har fått tildelt et kønummer (se kodeeksempel). Dersom dette er tilfelle, vises en beskjed om hvor du står i køen; dette er en dynamisk verdi som telles ned fra kundebehandlerens applikasjon.

```
if ($row[waitstate] == 2){          ?>
<script language='javascript'>
location.href='index.php?id=<?PHP echo $wait_id ?>';
</script><?PHP
 }
else {
if ($row[queuenr]){
echo '<h1 align=center>You are number ';
echo $row[queuenr];
echo ' in line.</h1>';
}
```

Hver gang siden oppfriskes, sjekkes det også om waitstate har blitt satt til 2 (se kodeeksempel). Når dette skjer er kundebehandleren er klar til å starte samtalen, og man blir sendt videre til selve chat-vinduet ved hjelp av den samme JavaScript-funksjonen som i request.php. For å oppdatere siden har vi brukt en sleep-funksjon (siden "sover" i 5 sek.) og kaller deretter siden på nytt med JavaScript.

4.5.4 Klientside chat

Her foregår selve samtalen mellom klienten og kundebehandleren fra klientsiden. Med jamne mellomrom hentes ny informasjon ut fra databasen, og samtalen blir oppdatert med det siste som ble skrevet. I tillegg til det feltet som viser dialogen, har vi et tekstfelt der klienten skriver inn sine svar, samt litt informasjon om kundebehandleren.

Hvert 10. sekund oppdaterer siden seg, og innholdet i det tabell-feltet der samtalen blir lagret ("answer"), leses inn i et eget dokument for å oppdatere chatten med det siste som har blitt skrevet. Et problem var at tekst som klienten var i gang med å skrive forsvant under oppfriskingen av siden. Dette førte til at vi valgte å dele siden inn i tre frames, en dynamisk og to statiske. Den rammen som oppdateres inneholder dialogen. De andre rammene inneholder informasjon om hvem som behandler saken din og svar-feltet, og disse oppdateres altså ikke. Svar-feltet ligger i en form, og når man trykker "OK" kaller siden seg selv og innholdet blir behandlet. Før man skjøter den innskrevne tekststrengen til resten av samtalen, hentes dialogen ut fra databasen på ny, slik at man er sikker på å få med de siste endringene fra konsulentsiden.

```
$SQL = "SELECT answer FROM support WHERE SUPPORT_ID=$id";
$result = MySQL_query($SQL);
$row = MySQL_fetch_array($result);
$input=$row[answer]."\n"."> ".$input;
MySQL_query( "LOCK TABLES support WRITE" );
$SQL2="UPDATE support SET answer = '$input' WHERE SUPPORT_ID = $id";
$result2= MySQL_query($SQL2);
MySQL_query( "UNLOCK TABLES" );
```

Deretter settes det en "lås" på support-tabellen som hindrer skriverettigheter for andre enn klienten (se kodeeksempel). Dette gjør vi for å hindre delingskonflikter og overskriving. Følgende oppdateres den aktuelle raden i support-tabellen, og man kan fjerne tabell-låsen.

Vi hadde i utgangspunktet tenkt å bruke et html textarea til å holde samtalen Vi lette veldig lenge uten å finne en metode som gjorde at scrollbarene til enhver tid var scrollet helt ned, så vi måtte finne en annen metode. Dette var en tidkrevende prosess, men vi fant til slutt en funksjon som lot oss scrolle ned på onLoad i Body-tagen. Denne benyttes kun i den framen som inneholder samtalen. Dette er ingen optimal løsning, men det gjør at vi får ønsket funksjonalitet.

5 Kvalitetssikring og testing

5.1 Kvalitetsstyring

Stabilitet, brukervennlighet, funksjonalitet og effektivitet har vært vesentlige begreper gjennom systemutviklingsprosessen. Før prosjektet var i gang diskuterte vi hva som burde vektlegges, og vurderte disse produktkvalitetene til å være viktige suksessfaktorer. Gjennom hele prosjektet har vi prøvd å holde fokus på dette, slik at produktet skulle svare mest mulig til forventningene. Vi har forøvrig ikke bedrevet noen form for formell kvalitetsstyring (med dokumentasjon av rutiner); først og fremst fordi vi har forsøkt å holde mengden papirarbeid nede til fordel for selve produktutviklingen. Kvalitetsstyring er nok en betydeligere fordel i større prosjekter.

5.2 Sikring av data og backup

Gruppen har etablert backup-rutiner som har blitt fulgt gjennom hele arbeidet. Disse rutinene ble besluttet allerede i forprosjektrapporten.

5.2.1 Backup: Regler / rutiner

Det skal alltid tas backup av dokumenter før endring/skriving av ny versjon.
Alt som skrives i tilknytning til prosjektet lagres på felles områder på skoleserver.
I tillegg skal backup gjøres på:

- Floppy (ikke anbefalt)
- Egen PC
- Sende via mail til hjemme-PC (for arkivering).

En gang hver uke (mandag) skal en fullstendig backup av alle prosjektrelaterte dokumenter foretas, og lagres på CD-R evt. DVD (speiling av hjemmeområde over på cd eller dvd plate).
Søndag innen 00.00 skal alle endrede dokumenter være lagt opp på fellesområdet på server.

Disse rutinene har vært svært nyttige da det mer enn en gang har vært nødvendig å gå tilbake til tidligere utgaver av programmer og dokumenter. Vi har daglig sendt hverandre siste versjoner av våre respektive programmer via e-post, og disse har blitt arkivert hos den enkelte. Derimot har vi ikke benyttet oss mye av backup på CD eller DVD. Vi syntes ikke det virket nødvendig, siden prosjektet har ligget i oppdaterte versjoner på 4 forskjellige maskiner gjennom hele perioden.

5.3 Testing

5.3.1 Database testing

På grunn av vår avhengighet til databasen, har det gjennom hele utviklingsperioden vært viktig å forsikre seg om at spørringer og oppdateringer fra både kunde- og konsulentmodulen ble registrert av databasen uten forsinkelser og datatap. Vi har fokusert på at systemet skal være stabilt, og en pålitelig database blir dermed en avgjørende suksessfaktor.
Vi kjørte MySQL spørringer via test-programmer som vi hadde utviklet i PHP og Delphi. Vi testet alt mot lokale servere først, for deretter å teste opp mot testserveren satt opp av WebDeal for dette prosjektet. http://testweb2.WebDeal.no/
Problemene med databasekommunikasjonen er beskrevet detaljert tidligere, i kapittel 4.3.

5.3.2 Testing av moduler

Testing av kildekode har stått i fokus gjennom hele prosessen. Det har hele tiden foregått testing innad i prosjektgruppen, og med jamne mellomrom har vi hatt større ”oppryddinger”. Debugger-funksjonaliteten i Delphi har vært flittig brukt, og vi har hjulpet hverandre for å unngå å ”se seg blind” på koden. All funksjonalitet som vi kodet hver for oss, ble testet av alle gruppemedlemmene før den ble integrert i resten av systemet.
Vi har benyttet oss av både white-box testing(med fullt innsyn i koden) og black-box testing(uten innsyn).

Vi har definert noen punkter som var viktige under feiltestingen:

- Er systemet stabilt?
- Handler systemet som forventet?
- Takler systemet forskjellig bruker-input?
- Har vi testet alle forskjellige ”stier” gjennom systemet?

Etter hvert som systemet har tatt form, har vi bedrevet integrasjonstesting. De viktigste delene av Live supporten har blitt integrert først. Vi kunne dermed ha en begrenset versjon av systemet oppe å gå tidlig i utviklingsprosessen, og den viktigste funksjonaliteten ble testet over en lengre tidsperiode enn mindre viktige komponenter.

Vi har også gjennomført stresstesting av chat-funksjonaliteten. Vi har lagt vekt på å finne ut om systemet er stabilt, siden det opererer med kontinuerlige spørringer til databasen fra opptil 20 chat-vinduer.

Jan Aril fra WebDeal har med jamne mellomrom fått muligheten til å validere og verifisere produktet, og har under hele prosessen kommet med tilbakemeldinger dersom ting kunne løses på en bedre måte. I skrivende stund virker det som om alt fungerer som planlagt, men tar forbehold om at mindre feil kan åpenbare seg når systemet settes i drift.

I sluttfasen har vi lagt vekt på å teste alle stier gjennom systemet. På den måten får vi sjekket at alle hendelsessekvensene fungerer som de skal, og at systemet er robust. Etter at produktet ble ferdig fikk Webdeal en betaversjon som de kunne teste en ukes tid. Dette er en fin måte å teste om produktet tåler vanlig bruk på.

6 Egenvurdering og avslutning

6.1 Vurdering av resultatet

Vi har nå utviklet systemet vi skisserte i forprosjektrapporten. Systemet vi har konstruert, inneholder det meste av den ønskede funksjonaliteten fra WebDeal sin side. Vi føler at livesupportsystemet er godt egnet til WebDeal og deres bruk, og ser positivt på at det er komplett nok til å settes i drift i den nåværende versjonen. Systemet er godt dokumentert og kommentert, og det vil ikke være problematisk å bygge videre på dette systemet ved et senere tidspunkt.

I vårt arbeid har vi lagt vekt på en funksjonell og effektiv applikasjon som også skal være i stor grad selvforklarende og enkel å bruke. Vi føler at det endelige produktet gjør disse tingene på en god og oversiktlig måte. Applikasjonen er delt inn i ulike enheter som har hver sin funksjonalitet. Dette gjør applikasjonens ulike elementer uavhengige, og de kan modifiseres uten frykt for å gjøre endringer som ødelegger andre deler av applikasjonen.

Vi sitter med følelsen av at systemet kunne ha fått implementert enda mer funksjonalitet om tidsrammene tillot det. Systemet fungerer som det skal, men kan bygges ut til å inneholde nye funksjonsområder.

Vi har også hatt et godt samarbeid med den andre Live-supportgruppen, og vi har implementert deres standardsvar-løsning i en søkbar del i vår applikasjon. Dette gir WebDeals konsulenter ekstra ressurser i kundebehandlingen. Det gjør også at alle standardsvarene til WebDeal ligger i samme database, og forenkler administrasjonen av disse.

Vi føler at vi har oppfylt de fleste kravene vi satte i forprosjektrapporten på en god måte innenfor de gitte tidsrammene. Systemet fungerer, og det har alle de funksjonene vi satte som mål å utvikle.

På grunn av manglende erfaring innen Delphi og dårlig dokumentasjon på området Delphi/MySQL, har det vært vanskelig å finne eksempler på lignende programmer. Vi har gjort utallige søk på nettet for å finne informasjon om emnet, men med dårlig resultat. Både vi og oppdragsgiver følte ofte at vi drev nybrottsarbeid.

Når det gjelder implementeringen av PHP-delen, har denne gått relativt smertefritt. Synne og Marianne hadde god erfaring med PHP fra før, og selve skjelettkoden kom opp relativt raskt. Kommunikasjonen mellom konsulentmodulen, databasen og kundemodulen har blitt tilrettelagt etter hvert som Delphi-applikasjonen har tatt form. Vi syntes det var best å utvikle disse modulene parallelt, så vi kunne testkjøre hele kommunikasjonssekvensen under ett. Databasen er en svært sentral enhet i dette systemet, og PHP har med sin gode støtte for MySQL gjort denne kommunikasjonen enkel. Vi har benyttet oss av kontinuerlig sjekking av databasen for å hente ut ny informasjon, og det virker som om

php-sidene takler dette fint; vi har ikke hatt problemer med forsinkelser eller nettlesere som har ”hengt seg”. Vi har hatt noen vanskeligheter med å få tilbake skriverettighetene til support-tabellen etter stresstesting av chat-funksjonaliteten. Dette er for øvrig ikke noe problem så lenge oppdateringshastigheten ligger på normalt nivå.

Systemet kunne nok vært testet i større grad, og under større belastninger enn det vi har hatt tid til, det må vi ta selvkritikk på. Vi har derimot ikke hatt noen indikatorer på at systemet ikke vil tåle stor arbeidsbelastning (i form av mange simultane brukere og mange åpne kundedialoger).

6.2 Alternative muligheter og valg underveis

WebDeal ga oss temmelig fast oppgavedefinisjon som ikke ga store rom for tolkninger når det gjaldt utviklingsmiljøer og systemarkitektur. Det har derfor vært noe begrenset med muligheter for de helt store forandringene med hensyn på benyttede programmer og funksjonalitet. Det har likevel vært vurderinger som måtte gjøres underveis, og prioriteringer som måtte foretas. Vi fant tidlig ut at enkelte aspekter av oppgaven måtte nedprioriteres noe. Da spesielt delen av oppgaven som omhandlet en AskJeeves lignende søkemotor for søking i forhåndsdefinerte svar. Denne delen av oppgaven så vi på som ikke så viktig for systemets funksjonalitet, og løste dette med en applikasjon med søkemuligheter knyttet opp mot tabellen for standardsvar.

Når det gjaldt Delphi og kommunikasjonen opp mot MySQL-databasen ble det brukt en del tid på å kartlegge og teste alternativene til den metoden vi endte opp med. Disse var eksempelvis ADO og MyODBC. Det var flere som hevdet å ha benyttet ADO og MyODBC opp mot MySQL databaser, men ingen av disse kunne fremlegge noe eksempel eller noen god løsning. Vi bestemte oss derfor for å benytte DBExpress komponenter. Denne metoden fungerte godt, var relativt rask og stabil.

I utgangspunktet skulle kundene få beskjed om hvor mange minutter de måtte vente på behandling. Dette ville ha blitt noe komplisert, da det er vanskelig å beregne hvor lang tid man vil bruke på en samtale. Vi bestemte oss isteden for en løsning der kundene fikk kønummer, slik at konsulentene slapp å kalkulere tidsbruk.

WebDeal hadde i oppgavedefinisjonen ytret ønske om å laste opp filer til kunden via supportsystemet. Dette valgte vi forøvrig å nedprioritere pga. tidsmangel. En alternativ løsning er at konsulent og klient sender hverandre linker til dokumenter de ønsker å gi hverandre (dokumentene må ligge på en webserver). Dette skulle fungere bra inntil man evt. velger å bygge ut systemet med denne funksjonaliteten.

6.3 Fremtidige utvidelser og forbedringsmuligheter

Det rent estetiske ved hele systemet har ikke vært en stor prioritet for oss, og dette vil kunne gjøres noe med i etterkant. Vi fikk tidlig beskjed om at WebDeal har egen designer

som vil ta seg av designet på kundemodulen i PHP, og at dette var den delen der utseendet var viktigst. Konsulentmodulen skulle først og fremst være funksjonell og brukervennlig.

Noe som også kan forbedres ved senere tidspunkt, er søkeapplikasjonen i systemet. Denne er funksjonell og enkel, men søker kun på enkle tekststrenger og tillater kun delvis matching. Mer kompliserte søkealgoritmer kan implementeres for mer presise søk, etter hvert som denne standardsvartabellen øker i størrelse og mulighetene blir flere.

Grunnet den kommunikasjonsmetoden vi benytter mot MySQL-databasen fra Delphi-programmet, gjøres det svært mange datakoblinger under en enkelt kobling. Det kan derfor vise seg at spørringene er mer ressurskrevende enn det som strengt tatt behøves. Dette kunne vært forbedret dersom vi benyttet oss av en database som var mer velkjent i Delphi-miljøet. Vi har ikke fått muligheten til å sette flere koblingstyper mellom Delphi og MySQL-databasen opp mot hverandre med tanke på effektivitet og ressursbruk (dette er diskutert under ”alternativer og valg underveis” lokalisert over). Ved et senere tidspunkt kan det være interessant og nyttig å teste koblingene vi har gjort med forskjellige koblingstyper, slik at man kan vurdere hva som er best egnet for dette typen system. Det skal være mulig å bygge opp databasekommunikasjonen uten så mange komponenter som vi har benyttet, og det kan kanskje bidra til et raskere og mindre ressurskrevende program.

Det er en god idé å implementere søk i standardsvar i kundemodulen også. Dette kan da være plassert på WebDeals nettsider, slik at kundene i noen tilfeller løser sitt eget problem uten å spørre WebDeals konsulenter.

Vi føler også at enda mer funksjonalitet kunne vært lagt inn i programmet, hadde det vært tid til dette. Filoverførings-funksjonaliteten som WebDeal i utgangspunktet ønsket, kan legges til. Kanskje bør det også lages en mer omfattende hjelp-funksjon. Statistikk-vinduet kan modifiseres til å foreta mer kompliserte beregninger, samt at det kan være aktuelt å visualisere resultatene ved hjelp av grafikk. Det kunne også vært aktuelt å gi konsulentene muligheten til å tømme chat-feltet innimellom. Dette kunne da dumpes til en fil eller en tabell dedikert til oppgaven.
Applikasjonen kan også lett utvides til kundeadministrasjon og lignende oppgaver.

6.4 Evaluering av eget arbeid

6.4.1 Innledning

Dette prosjektet har vært en krevende og utviklende oppgave for alle gruppemedlemmene. To av gruppemedlemmene kjente hverandre fra før, og hadde jobbet en del sammen, mens det siste medlemmet tok kontakt med den allerede etablerte hovedprosjektgruppen og spurte om å få delta. Det fikk han, og gruppen ble på tre stykker som måtte bli kjent med hverandre, og deretter jobbe sammen på en hovedoppgave.

6.4.2 Arbeidet og organisering

Samarbeidet oss i mellom har gått godt og knirkefritt, det har vært få uenigheter, og de små som har vært har blitt løst raskt i fellesskap. Det var spennende i begynnelsen å se om samarbeidet ville gå godt, da deler av gruppa kjente lite til hverandre. Vi har møttes og arbeidet sammen ca 5 dager hver uke fra 2 timer og lengre. Her har vi delt ideer, jobbet med fellesoppgaver og enkeltoppgaver. Vi har hatt en fast gruppeleder som har ledet arbeidet og sørget for at progresjonen ble etter planen, samt notert møtereferater og skrevet statusrapporter. I forprosjektrapporten hadde vi temmelig klare ansvarsområder, men har samarbeidet mye mer enn antatt og jobbet mye innen hverandres ansvarsområder av oppgaven.

Når det gjelder systemutviklingsmodellen vi har brukt, inkrementell utvikling, er vi fornøyde med valget vårt. Den har fungert bra, og vi har hele tiden hatt kontroll over prosessen uten å bli ”nedlesset” av papirarbeid. Inkrementene har blitt utviklet, vurdert og evt. endret fortløpende. Vi tror dette fungerte bra både for oss og arbeidsgiver.

Vi skrev tidlig en gruppeavtale som omhandlet samarbeidet innad gruppen, og en mellom de to Livesupportgruppene (1 og 2). Det har vært kjekt å ha et felles regelverk å forholde seg til, og samtlige medlemmer på begge gruppene har holdt seg til dette. Samarbeidet både innad i gruppen og mellom gruppene har vært sosialt, lærerikt og effektivt.

Ett av medlemmene har hatt en god del problemer med ryggen under dette prosjektarbeidet, noe som har ført til at han ikke kunne delta på enkelte møter og annet oppgavearbeid. Dette har likevel ikke forhindret oss mye, siden vi jobber effektivt også på egen hånd.

Gjennom arbeidet med dette prosjektet, har vi forsøkt å føre regnskap over timene vi har brukt. Dette er totalt antall timer, og inkluderer research, nettsøk, lesing i bøker, koding og skriving av rapport for å nevne noe.
Det totale timeantallet for den enkelte, og gruppa, ble fordelt slik:

Trond Viggo Bjerke	468 timer
Marianne Brattrud	482 timer
Synne K Repp	501 timer
Totalt	1451 timer

6.4.3 Prosjekt som arbeidsform

Samtlige medlemmer føler at dette har vært en unik læreopplevelse. Det har vært veldig interessant og lærerikt å kunne delta i en slik prosess fra begynnelse til slutt. Prosjekt som arbeidsform er svært relevant for hvordan det blir å jobbe senere når man kommer ut i arbeidslivet. En kan ikke tenke bare på seg selv, da man har medarbeidere, tidsfrister, ressurser og rammer som man må ta hensyn til.

Vi har alle erfaringer med prosjekt fra tidligere studier og arbeid, men ingen av oss har deltatt i et prosjekt av denne størrelsen. Spesielt ikke der alle har en slik nøkkelrolle som vi fikk i en såpass liten gruppe. Enkelte problemer som vi trodde ville ta kort tid, tok lang tid. Arbeidsoppgaver vi trodde kun ville ta en time eller to, kunne vise seg å strekke seg over flere arbeidsdager.

Vi sitter alle igjen med følelsen av å ha blitt flinkere til å utnytte ressursene som finnes rundt oss; i andre prosjektarbeider, medstudenter/ansatte, i bøker og i elektroniske kilder. Vi har også blitt bedre programmerere; to av gruppemedlemmene har gått fra å kunne ingenting om Delphi til å programmere store deler av konsulentmodulen.

Vi håper vi får delta i lignende prosjekter i arbeidslivet, og at vi får muligheten til å jobbe med et system fra tidlig prototype til ferdig program igjen.

6.5 Veileder og oppdragsgiver

Gjennom vårt arbeid med prosjektet har vi hatt regelmessige møter med veileder og arbeidsgiver. Dette ga oss et kontaktnettverk når det gjaldt problemstillinger vi ikke klarte å løse på egenhånd. Det hjalp ofte å bare lufte problemstillingene for veileder og oppdragsgiver, slik at de kunne gi oss en ny og bedre vinkling på problemet (og alternativ løsning). Vi var ofte i kontakt med Jan Aril (på hans arbeidsplass og hjemme) gjennom Microsoft Messenger.

Veileder hadde god kompetanse innen Delphi og databaser, og dette var veldig godt å dra nytte av når vi slet med kompliserte databasespørringer og lignende.
Det var også ekstra motivasjon at vi visste at vi skulle ha møter, og følte vi måtte ha god progresjon mellom disse møtene.

Vi var så heldige å ha arbeidsgiveren vår i nærheten under arbeidet med oppgaven. WebDeal holder til på skolens område, og det var bare en kort gåtur til kontoret deres for ha møter, stille spørsmål og demonstrere nye applikasjoner.
Jan Aril hadde mye kompetanse innen de siste utgavene av Delphi, og hadde jobbet med konstruksjon og drift av lignende systemer som det vi skulle lage. Derfor var det uvurderlig å ha tilgang på hans kompetanse, synspunkter og motivasjonshjelp.

6.6 Konklusjon

Vi er nå ved veis enda av et langt og krevende prosjekt. Det var en svært interessant utfordring vi ga oss ut på ved juletider 2003.

Vi har over flere måneder utviklet et Livesupportsystem for bruk i en IT-bedrift. Dette har vært krevende og utfordrende, men også veldig belønnende.
Vi har sett systemet utvikle seg fra en idé til ferdig program. Ingen av oss hadde tidligere vært med på et prosjekt av dette omfanget, og prosessen har vært spennende og utviklende.

Å vite hvor man skulle starte på denne oppgaven var vanskelig, og tidlig så det ut som en litt vel krevende og omfattende oppgave. Etter hvert som oppgaven ble avgrenset og delt inn i mindre problemer, følte gruppen at vi fikk kontroll på prosjektet.

Prosjektarbeid er en veldig vanlig arbeidsform i bedrifter vi kommer til å jobbe i senere, og dette har derfor vært en svært verdifull erfaring å ta med seg.
Vi har på nært hold opplevd problemene som kan oppstå i et slikt prosjekt, samt frustrasjonen ved at ting ikke går som det skal. Heldigvis har vi også fått erfare at ting har løst seg, at prosjektet har gått greit og gleden ved at systemet fungerer.

Vi føler at systemet vårt gjør det vi i utgangspunktet ville forsøke å få til.
Vi kan ikke se noe hinder for at systemet blir tatt i bruk, og håper at det gjør hverdagen enklere for konsulentene hos WebDeal. Vi tror det kan føre til at kundene får en positiv opplevelse av kunde-supporten hos WebDeal.

7 Litteraturliste

[1] Marco Cantù. Mastering Delphi 6, 2001.

[2] Paul DuBois, MySQL. Second Edition, 2003.

[3] Delphi Basics. URL, 2002-2004.
http://www.delphibasics.co.uk/

[4] About Delphi Programming. URL, 2004.
http://delphi.about.com

[5] PHP. URL, 2001-2004.
http://www.php.net/

[6] PC World Norge, Mitt dataleksikon. URL, 2004.
http://www.pcworld.no/index.cfm?fuseaction=dataleksikon

[7] TECH on the Net, SQL Topics. URL, 2003-2004.
http://www.techonthenet.com/sql/index.htm

[8] Paul Kimmel. Building Delphi 6 Applications, 2001.
The McGraw Hill Companies, Inc.

[9] Marco Cantu, Mastering Delphi 7, 2003.
Sybex publishing

[10] Nyhetsgrupper:
fido7.ru.delphi.*
comp.lang.pascal.delphi.misc

[11] Hovedprosjekt LSS livesupport system. Hig, 2003

[12] Hovedprosjekt ByggPOS. Hig, 2002

[13] Hovedprosjekt ESTATICA. Hig, 2002

Vedlegg