0.1 Forord

Hovedprosjektet er en obligatorisk del av bachelorgraden, og er et større studentprosjekt som gjennomføres ved Høgskolen i Gjøvik.

Prosjektet har resultert i en programvare som kan assistere et mindre antall Bygger'n-bedrifter i å samarbeide om å gjøre innkjøp.

En takk til oppdragsgiver som gav oss mulighet til å gjennomføre dette prosjektet og som har tatt seg tid til å ha møter med oss.

En spesiell takk til Tom Røise som har kommet med gode råd til gruppen når vi har trengt hjelp.

Innhold

0.1 Sammendrag

Dette er et hovedprosjektet ved HIG som har gått ut på å lage en webapplikasjon for innkjøpssamarbeid beregnet på Bygger'n butikkene i Valdres. Poenget med denne applikasjonen er å oppnå lavere innkjøpskostnader ved at bedriftene samarbeider om innkjøpene. Et slikt samarbeid vil føre til at varene kan bestilles i større kvanta og bedriftene oppnår dermed kvantumsrabatter. I tillegg vil det gi de forskjellige bedriftene mulighet til å utnytte hverandres rabattavtaler.

Gruppen har brukt PHP som programmeringsspråk, og for lagring av dataene er databasen MySQL benyttet.

Utviklingen har foregått i en tidsperiode på ca 5 måneder. I starten av prosjektet ble det holdt flere møter med oppdragsgiver hvor ønsket funksjonalitet ble avklart. Gruppen har likevel stått veldig fritt i utviklingen av funksjonene. Mot slutten av prosjektperioden ble applikasjonen kort presentert for oppdragsgiver. Etter denne presentasjonen sitter prosjektgruppen igjen med inntrykk av at funksjonaliteten oppdragsgiver hadde forespeilet seg ved prosjektstart samsvarer godt med det ferdige produktet.

Webapplikasjonen som er laget fungerer i grove trekk på følgende måte; En bedrift legger inn en ordre i systemet, deretter melder en annen bedrift seg på denne ordren og legger inn varene de ønsker. Etter at varene er lagt inn effektuerer innkjøpsansvarlig for den aktuelle varegruppen ordren og det genereres en PDF-fil med den ferdige ordren. Innkjøpsansvarlig sender så denne ordren til leverandøren via e-post, fax, post eller telefon.

Kapittel 1

Innledning

1.1 Bakgrunn

Oppdragsgiver driver et byggevarehus, Bygger'n Bagn. I Valdres er det tre uavhengige Bygger'n-medlemmer som i dag praktiserer samkjøp, spesielt på tyngre byggevarer. Hensikten med dette samarbeidet er å utnytte rabattavtalene med leverandør på en bedre måte. Stordriftsfordeler som man oppnår ved et slikt samarbeid er blant annet større kvantumsrabatter og bedre fraktbetingelser.

Når byggevarefirmaene i dag skal bestille varer, printer de ut en ferdiglaget mal som de fyller ut og sender med faks. Denne malen er en nesten ferdig ordre som har informasjon om hvilken leverandør det skal sendes til, informasjon om bedriften som bestiller, og en liste over varer som leverandøren tilbyr. Dette papiret, sammen med en penn, tar de med seg rundt i butikken og fører opp det antall varer som bedriften trenger alt etter hvilket behov de har. Til slutt kan de påføre detaljer som eventuelle rabatter de måtte ha spesielt, eller om det er tilbud de vil benytte seg av. Slik vil de ende opp med en ferdig faks som de sender til leverandøren.

Bygger'n sentralt forhandler frem rabatter mellom leverandører og Bygger'n, men den enkelte bedrift kan like gjerne ha egne spesielle rabatter som de har forhandlet frem på egenhånd. Man har ofte en fast rabatt som ligger i bunn som kalles grunnrabatt. Så har man gjerne rabatter når man bestiller over

et visst volum, det enten være i vekt, antall eller penger. Siden leverandøren gjerne tilbyr rabatter hvis man bestiller mer fra leverandøren så prøver man gjerne å samle opp flere varer som skal bestilles for å få en bedre pris.

På en annen side så er det begrenset med lagerplass som gjør at man ikke får bestilt særlig store kvanta. Faktisk så kan byggevarer være svært plasskrevende, og ofte må man bestille mindre enheter fordi man ikke har nok lagerplass til varene. Hvis man bare har plass til 10 gipsplater på lageret, og den minste enheten fra leverandør er 40stk, så må leverandøren splitte opp en pakke for at Bygger'n Bagn skal få sine 10stk. Da kan man få ekstra avgift på fakturaen. Denne avgiften kalles for brekkasje, og er selvfølgelig noe man ønsker å unngå.

1.2 Problemområde

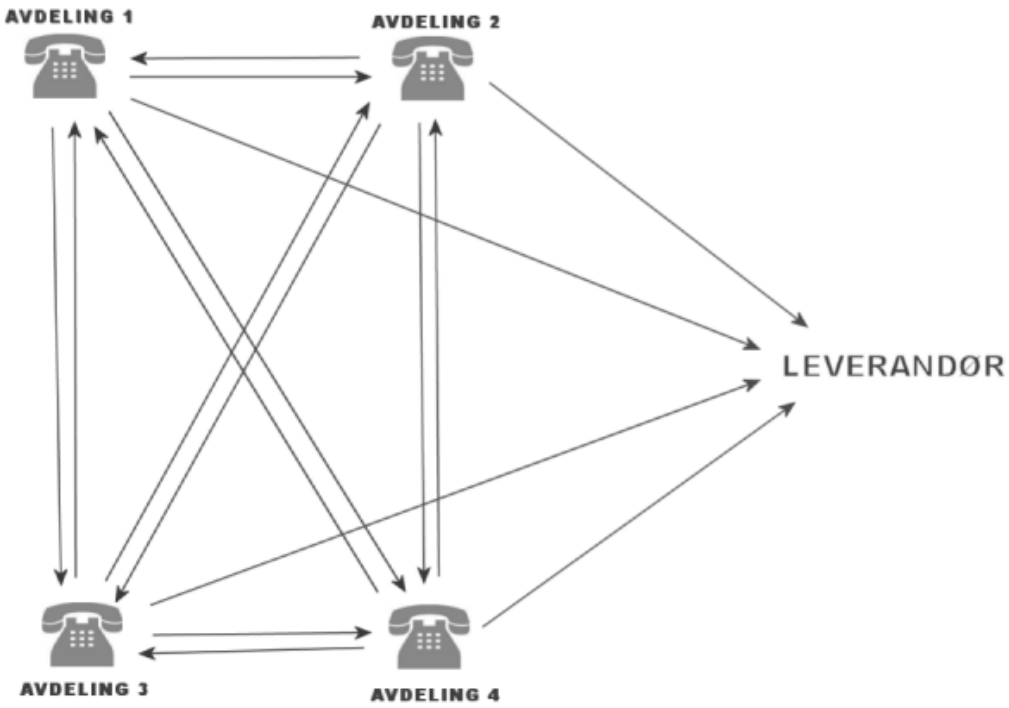
Dagens samkjøpspraksis er lite effektiv, da den baserer seg på telefonhenvendelser fra førstemann som ser at det kan være penger å spare på å samarbeide om innkjøpet. Kommunikasjon over telefon gir ikke en god oversikt over hva som foregår, og det tar tid å måtte ringe rundt til alle som deltar i ordren for å innhente informasjon, og avklare ulike ting. Det er også en fare for at misforståelser oppstår.

Samkjøpspraksisen slik den foregår i dag er derfor langt fra optimal, og man vil tape både tid og penger hvis man velger å fortsette med samme løsning.

1.3 Oppgavedefinisjon

Bakgrunnen for å samarbeide om innkjøp handler om å forsøke å redusere avgifter, samt å dra bedre nytte av rabatter. Dagens løsning med å benytte telefon har som nevnt åpenbare svakheter, og tanken er da å lage løsning som kan forsøke å gjøre det lettere å koordinere samkjøpet.

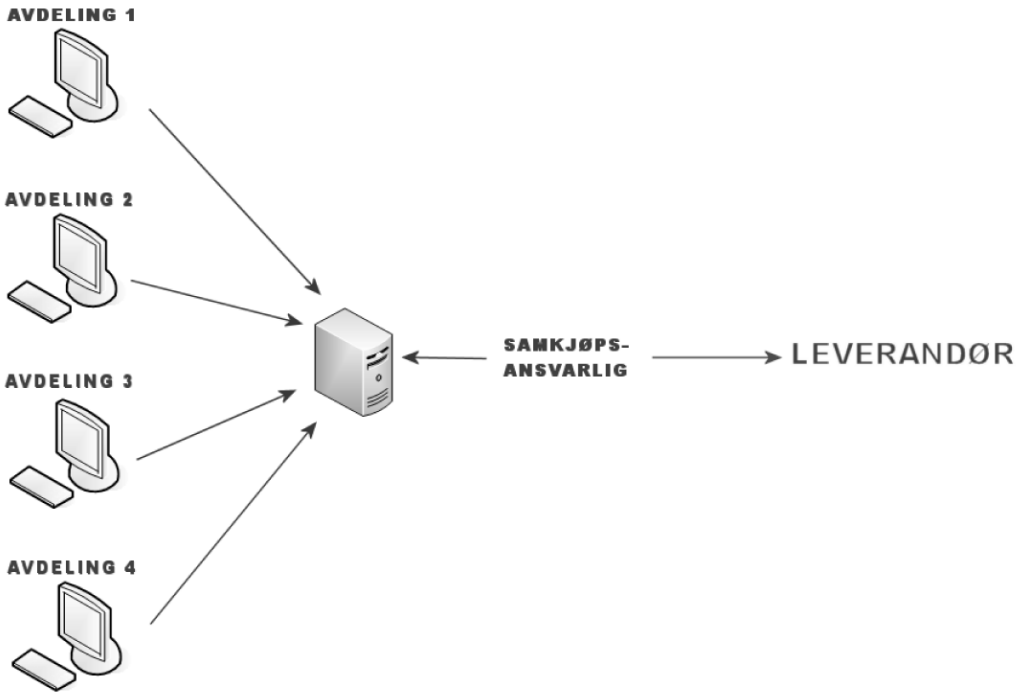
Det som er tanken er at man skal kunne lage en ordre i en dataapplikasjon, for så å legge til de varer man måtte ønske å bestille. Så skal andre bedrifter kunne se i denne applikasjonen at noen har startet en ordre, og dermed kunne gå inn i den respektive ordren og legge til de varer som de ønsker å



Figur 1.1: Situasjonen slik den er i dag, med mange telefonsamtaler på kryss og tvers av avdelingene for å koordinere et samkjøp

bestille.

Når ordren er klar for effektuering, ferdigstilles den med nødvendig informasjon og gjøres klar for oversending til leverandør, noe som kan skje med faks eller gjennom e-post.



Figur 1.2: Visjonen for samkjøpsapplikasjonen

1.4 Avgrensning

Bare det å lage et system som beskrevet over vil gi nok arbeid og avgrense prosjektet av seg selv. Skulle det bli nok tid, vil fokuset være på å lage funksjonalitet som vil støtte opp om aktiviteter rundt det å håndtere ordrer. Altså å forsterke selve poenget med applikasjonen. Vi forholder oss til situasjonen hos oppdragsgiver og systemene som brukes der i dag, og kan

med andre ord ikke ta høyde for alle andre systemer som finnes på markedet.

1.5 Målgruppe

Målgruppene for applikasjonen vil hovedsakelig være de tre Bygger'n-bedriftene som applikasjonen er tiltenkt i første omgang. Det er ingen begrensninger i systemet for å legge til flere bedrifter og brukere, eller benytte applikasjonen i andre byggevarebedrifter.

Sensor er også en målgruppe siden dette er flere personer som skal evaluere det arbeidet som er gjort i prosjektet. Siden arbeidet potensielt kan bli benyttet av andre lærere og studenter ved denne høgskolen, så vil disse på sett og vis også være en målgruppe.

1.6 Bakgrunn og kompetanse

Denne gruppen består av tre studenter som tar bachelor-graden i data og informatikk. Tor og Knut har gått bachelor data, og har spesialisert seg på driftsretningen. Frode har gått bachelor informatikk med spesialisering på systemering og programmering.

Dette prosjektet handler om å lage en webapplikasjon, og innenfor dette emnet så har gruppen veldig ulik bakgrunn.

Frode, som går informatikk, har hatt WWW-teknologi faget, Tor har erfaring med PHP og mySQL fra egne prosjekter, mens Knut har ingen kjennskap til PHP fra tidligere.

1.7 Utviklingsmodell

Systemutviklingsmodellen som skulle benyttes i utvikling av applikasjonen var Feature Driven Development, med støtte fra Use Case. Dette ble droppet tidlig da det ble problematisk å sette seg inn i dette, og bruke det fornuftig. Gruppen ønsket i stedet å fokusere sin tid på å programmere og produsere

selve applikasjonen.

Det som ble gjort tidlig var å liste opp noen hovedfunksjoner som applikasjonen skulle ha. Disse ble fordelt mellom deltagerne, gjerne etter deres evne til å fullføre dem.

Det ble i begynnelsen gjort en større diskusjon om applikasjonen og oppdaget flere ting som var uklare og manglet konkret informasjon om. Spørsmålene ble skrevet ned i et større dokument. Dette dokumentet ble så tatt med til oppdragsgiver for prosjektet, som da besvarte dem. Dette ble da brukt for å konkretisere og avklare aspekter ved de forskjellige funksjoner.

Database designet ble gjort delvis i fellesskap, men mye var opp til den personen som jobbet med den respektive hovedfunksjonaliteten. Noe av designet ble også avgjort av hvordan data fra Bygger'n Bagn sitt system så ut.

Dette er en applikasjon som ikke har utfordringer i form av å løse vanskelige oppgaver. Kruttet og hjulet er oppfunnet, og det er mer snakk om å få satt sammen ting og lage det oppdragsgiver vil ha. Det er absolutt et potensial for å forsøke å lage spennende og nyskapende ting i denne applikasjonen, men den lille tiden til rådighet fremtvinger behovet for å oppfylle oppdragsgiver sitt behov, enn andre ting.

1.8 Rammer

Prosjektet starter tidlig januar 2006, og avsluttes 24. mai 2006. Innleveringsfrist av prosjektrapporten til trykking er 22. mai, så arbeidet med prosjektet vil av naturlige grunner avsluttes denne datoen.

1.9 Øvrige roller

Oppdragsgiver: Vidar Hovdet, Bygger'n i Bagn Veileder: Harald Liodden, Høgskolelektor, Siv.ing., Høgskolen i Gjøvik

Tor ble valgt som sjef for gruppen og kontaktperson mot andre parter.

1.10 Organisering av rapporten

Rapporten er organisert på følgende måte:

- Forside
- Sammendrag
- Forord
- Innholdsfortegnelse
- Innledning
- Hovedkapitler
- Avslutning
- Bibliografi
- Vedlegg

1.11 Terminologi

ByggSAM: Navnet på webapplikasjonen som er utviklet

NOBB: Norsk byggevarebase, database over alle byggevarer som tilbys i Norge. [4]

S4: Navnet på webapplikasjonen som er utviklet [1]

PHP: PHP er et dynamisk, tolket og svakt typet programmeringsspråk hovedsakelig brukt for å utvikle dynamiske nettsider. [3]

MySQL: MySQL er en SQL-basert databasetjener. [2]

Kapittel 2

Hovedkapitler

2.1 Kravspesifikasjon

2.1.1 Eksisterende systemer

NOBB

NOBB er en database over alle byggevarer som tilbys i Norge. Hver vare har sitt unike NOBB-nummer og informasjon om selve varen og hvem som tilbyr denne varen. Denne databasen vedlikeholdes i dag av leverandørene selv, og kvalitetssikres av Norsk Byggtjeneste ODA A/S.

S4

S4 er et system for organisering av pris- og vareinformasjon. Systemet benyttes hos Bygger'n Bagn for å holde orden på vareinformasjon sentralt på Bagn. Andre datasystem hos oppdragsgiver, som for eksempel butikkdatasystemet, får sine data fra S4.

Nå bruker ikke Bygger'n Bagn NOBB direkte. S4 er derimot bygget på NOBB-systemet og kommer med en del av informasjonen som er lagret der, pluss en del tilleggsinformasjon som ikke finnes i NOBB.

S4 blir jevnlig oppdatert med vareinformasjon, og det er dette systemet vi må forholde oss til. Så langt vi har funnet ut benytter ikke S4 noe SQL

database som vi kan koble oss til. Dataformatet de benytter til ulike import og eksport er ikke dokumenterte, og er proprietære format som vi ikke får benyttet. S4 har derimot mulighet for å eksportere til Microsoft Excel, og det er via dette formatet vi har tenkt å importere data til vårt system. Det er ingen optimal løsning, men den eneste løsningen vi har å jobbe med.

2.1.2 Omgivelser

Det er tre Bygger'n-bedrifter som skal benytte applikasjonen, og disse bedriftene ligger et godt stykke fra hverandre. På oppdragsgivers oppfordring har det ikke blitt opprettet kontakt med de andre Bygger'n-bedriftene for å finne ut hva slags maskinvare de har, hva slags brukere det er, og hva de måtte mene om dette prosjektet.

Det må derfor antas at alle bedriftene besitter vanlige datamaskiner med internett-tilkobling, og at de som betjener disse vet å bruke dem på normal måte. Webapplikasjoner som nettbank, reisebestilling, og nettaviser begynner å bli utbredt. Dermed antas det også at vanlige folk ikke vil ha for store problem med å tilpasse seg slike applikasjoner.

Ingen av bedriftene har egne IT-avdelinger eller dedikerte ressurspersoner på IT etter det vi kjenner til. Det finnes personer som har interesse for data, men vi må anta at byggebransjen ikke tiltrekker seg mange med god nok kompetanse innen IT.

Skal man lage en applikasjon ut fra disse antagelsene så må det fokuseres på brukervennlighet, tilgjengelighet, og å finne en plattform som ikke er for avhengig av brukerens maskiner og programvare.

2.1.3 Funksjonalitet

Gjennom to møter med oppdragsgiver i starten av prosjektperioden, ble det avklart hva slags funksjonalitet applikasjonen skulle ha, og hva oppdragsgiver hadde sett for seg.

Det ble stilt flere krav til nødvendig funksjonalitet, men oppdragsgiver lot samtidig døra stå oppe for nyttig tilleggsfunksjonalitet som kunne dukke opp underveis. Kravene som ble satt var ikke spesielt detaljerte, så vi stod

mer eller mindre fritt til å utvikle applikasjonen som vi selv ønsket.

Det første som ble gjort var å lage en liste over funksjonalitet som applikasjonen måtte ha. Funksjonene ble drøftet, og det kom raskt til en enighet om innholdet i disse funksjonene.

Her må det også nevnes at ble bestemt at systemet må fungere uavhengig av typen nettleser.

Brukersystem

Systemet skal ha en måte å sikre at kun autoriserte brukere får tilgang. Selv om systemet ikke vil inneholde informasjon som er spesielt sensitivt, så vil man ikke la hvem som helst få tilgang til systemet. Systemet må også kunne skille ulike type brukere og tilgang til funksjonalitet.

Rettighetssystem

Til brukersystemet så skal det lages et rettighetssystem. Med et detaljert rettighetssystem, kan man endre tilgangen ned til en funksjonalitet. Man skal skille på ingen tilgang, lese tilgang, og skrivetilgang. Så setter man opp en egen rettighet for hver funksjonalitet i programmet. Slik kan man detaljert bestemme hva slags tilgang en bruker skal ha til aktiviteter med ordren, leverandøren og brukerne.

Vareregister

Dette er data som i utgangspunktet skal hentes fra bedriftens eksisterende varedatabase S4. Mulighet for å legge inn varer manuelt er også noe som kan være nyttig. Dette for å gi adgang til å legge til varer som ikke kommer via S4.

Leverandørregister

Det skal være et register over leverandører. Dette er data som kommer fra S4, men man må også ha muligheten for å legge til og endre på informasjon om leverandører.

Ordreregistrering

Den viktigste delen av systemet, nemlig å produsere ordrer, og legge til ordrelinjer. Systemet må skille på hva de ulike bedriftene bestiller, og levering

til ulike adresser. En ordrelinje skal ha informasjon om hvilken vare som skal bestilles, antall det skal bestilles, enhet det bestilles i, og pris man har på ordrelinjen. Ordrelinjen knyttes til hvilken bedrift som bestiller den og hvor varen skal leveres til. Ordren må også ha informasjon om hvem som er ansvarlig for håndtering av ordren, dato den skal effektueres, og informasjon om hvilken leverandør det bestilles mot.

Ordreoversikt

Oversikt over hvilke ordrer som er åpne og under behandling i systemet. Dette er kun ment som et oversiktsbilde, og for å la brukeren lett kunne velge den ordren han skal se nærmere på.

Ordrehistorikk

Oversikt over tidligere ordrer. Dette er oversikten over effektuerte ordrer som man skal kunne hente frem. De eldre ordrene tas vare på, i tilfelle man ønsker å nyttiggjøre informasjonen senere. Det er flere grunner til at ordren bør tas vare på i databasen. Hvis man på et senere tidspunkt har spørsmål om en tidligere ordre, så har man muligheten for å sjekke dette i systemet. Skal man senere evaluere om samkjøp har vært kostnadsbesparende, kan det være bedre å lage funksjonalitet i programvaren for å finne ut av dette. Hvis systemet ikke lagrer dataene, så må bedriftene ta vare på en masse papir som kanskje man ønsker å overføre tilbake til data på et senere tidspunkt.

Rabattoversikt

Oversikt over hva slags rabatter man har mot de ulike leverandørene. Her skriver man inn hva slags rabatt man har, og informasjon om dette.

Spesialtilbudsoversikt

Oversikt over hvilke spesialtilbud som finnes. Dette benyttes for å informere og varsle andre i systemet. Dette for å gjøre det enklere for andre å oppfatte at slike tilbud gjelder.

Effektuering

System for å effektuere en ordre. Systemet må da hindre andre å gjøre videre endringer på ordren, og kun la den som er ansvarlig for ordren ferdigstille den. Det må være mulig for andre som har rettighet til å ferdigstille en ordre

til å kunne overta om den fysiske personen plutselig ikke blir tilgjengelig.

Kontaktregister

Noen ganger ønsker man å oppgi en kontakt hos leverandør som skal ha ordren. Dette kan ha sammenheng med rabattavtaler, eller andre ting. Derfor må man kunne ha et register over hvilke kontakter som finnes hos ulike leverandører, for bruk i nettopp ordren.

Kunderegister

Dette skal holde orden på kontakt info og byggeplassadressene til kunden. Det skal ikke holde oversikt over hvor mye kunden handler for.

Import av data fra håndscanner

Oppdragsgiver har en håndscanner-enhet som kan brukes til å bestille varer. En importmulighet for denne informasjonen kan være nyttig ved bestilling.

Beskjedsystem

Et integrert beskjedsystem vil gjøre det mulig for brukerne av systemet å kontakte hverandre.

2.2 Design

2.2.1 Databasedesign

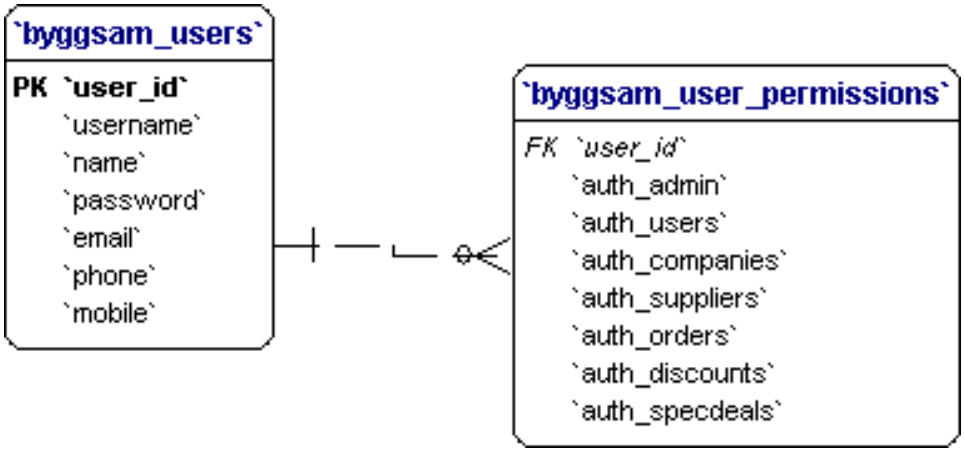
Med utgangspunkt i data fra S4/NOBB og at dette er et ordresystem, så vil det være hensiktsmessig å benytte en database i bunn til denne applikasjonen. Funksjonaliteten som er beskrevet i kravspesifiseringen forholder seg også til strukturert data, noe som gjør det lettere å designe databaseskjemaer til denne applikasjonen.

Noen tabeller som vises videre nedover i kapittelet har langt flere datafelt enn det bildet viser. Siden mange felter er selvforklarende, og andre ikke kommenteres særlig, så tar diagrammene for stor plass i dokumentet til at det er hensiktsmessig å detaljere ytterligere.

Bruker- og rettighetssystem

I utgangspunktet skulle det være et rollebasert system som skilte den vanlige bruker fra ordreansvarlig og systemadministrator. Siden det var uklart hvem som skulle være disse rollene, om det lot seg gjennomføre i praksis, og hva som skulle være rettighetene til disse rollene, ble det laget et rettighetssystem i stedet.

De deler av applikasjonen som kan organiseres i en rettighet, får en egen kode og opptil tre forskjellige rettighetsnivåer. Bakdelen med et slikt system blir når man skal opprette en ny bruker. Da må man inn og sette rettigheten til alle de ulike delene av applikasjonen, noe som kan være tungvint, hvis det skal opprettes mange brukere med administratortilgang. Det vil derfor være viktig at ikke for mange rettigheter lages, for da blir det mye jobb å administrere når man oppretter nye brukere og endrer eksisterende. Fordelen er at man kan opprette mange ulike typer brukere som har ulik ansvar i applikasjonen, og man kan endre på enkelte områder til en type bruker.



Figur 2.1: Brukere og brukerrettigheter

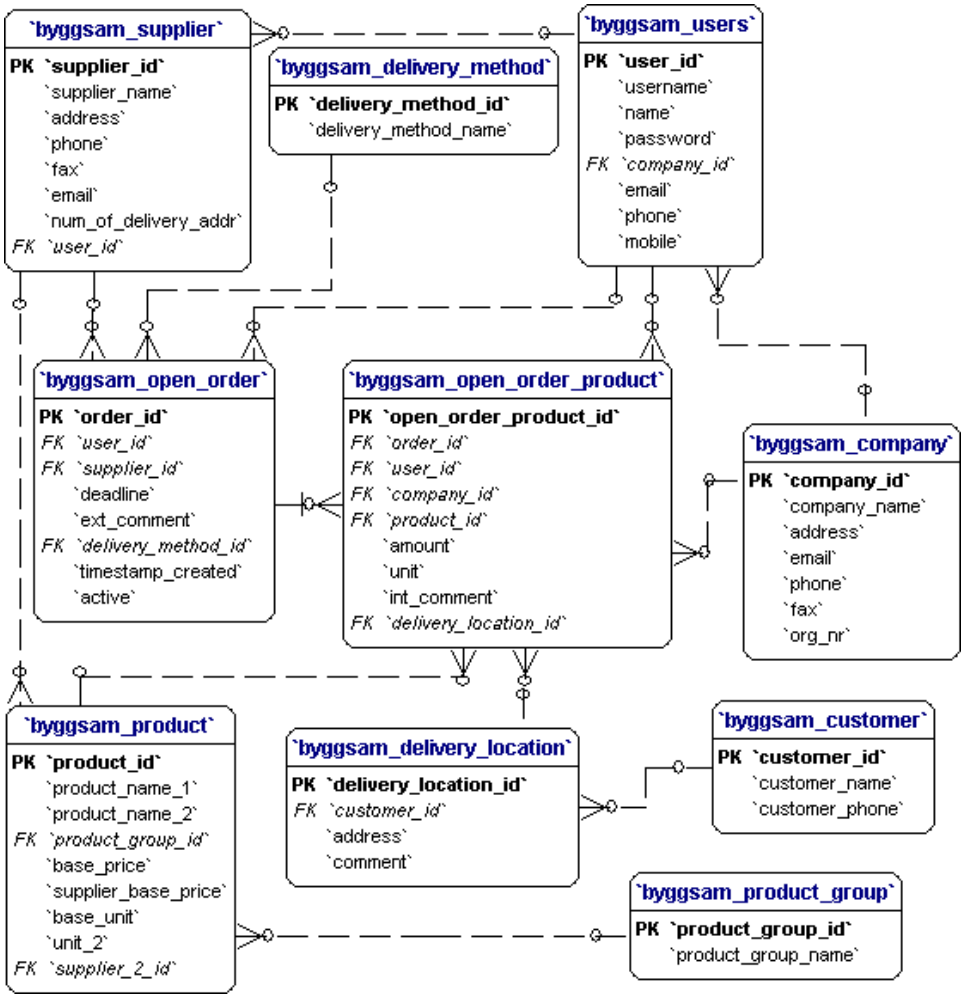
Hver bruker får et eget brukernavn og passord som de logger seg inn med. Hver bruker blir tildelt et eget id nummer som brukes internt i applikasjonen for å knytte brukerinformasjon til andre deler av applikasjonen. Resten av tabellen inneholder informasjon om brukeren, men denne informasjonen er i stor grad bedriftsrettet og valgfri.

Når en bruker opprettes så har den ingen rettighet til noe. Derfor så må

administratoren sette disse rettighetene på brukeren. Det er fra tabellen 'byggsam_user_permission' at rettighetstilgang til en funksjon blir sjekket.

Ordreregistrering

Flere bedrifter må kunne legge inn ordrelinjer, og disse må da knyttes til den bedrift som skal ha dem. Ordrelinjene må også knyttes til en leveringsadresse, som kan være noe annet enn bedriftens egen adresse. Ordren må knyttes til den som er ansvarlig for ordrehåndtering.



Figur 2.2: Åpne ordre

Som figur 2.2 viser, finnes det mange relasjoner i dette systemet. Det er to hovedtabeller for en ordre, det er 'byggsam_open_order', og 'byggsam_open_order_product'. Selve ordren knyttes til hvilken leverandør det skal bestilles fra, hvilken bruker som er ansvarlig for ordren, og hvordan ordren skal leveres til den Bygger'n-bedriften som bestiller.

Ordrelinjene knyttes hovedsakelig til hvilken bedrift som skal ha varen, og til hvilken leveringsadresse som varen skal leveres til. En vare kan like gjerne leveres rett til en byggeplass, og noen leverandører har mulighet for å frakte varer direkte til byggeplasser og privatadresser. Ordrelinjen er også selvfølgelig knyttet til produkt og hvilken bruker som har lagt inn ordrelinjen.

Vareregister

Databasen over varene ivaretas av tabellen 'byggsam_product'. Tabellen er laget med utgangspunkt i de data som mottas fra Bygger'n sitt system. I tabellen så finnes all informasjon om en vare som navn, nr, pris, vekt, og hvem som leverer produktet. Figur 2.2 viser kun et utdrag av tabellen da den har mange felt.

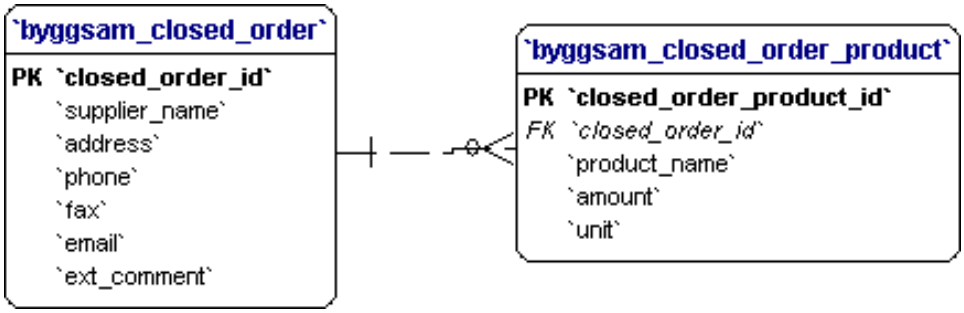
Leverandørregister

Dette registeret representeres av tabellen 'byggsam_supplier'. Data som kommer fra S4 har leverandørinformasjon på hver eneste produktlinje. Dette er skilt ut i egen tabell, fordi det er behov for å knytte leverandørinformasjon til andre deler av systemet. Ordren er knyttet til hvilken leverandør som det skal bestilles fra. Dette fører igjen til at kun varer som kommer fra den bestemte leverandøren kan legges inn i ordren.

Ordrehistorikk

Figur 2.3 viser hovedtabellene som benyttes for å ta vare på data fra ordrene, etter at de er effektuert.

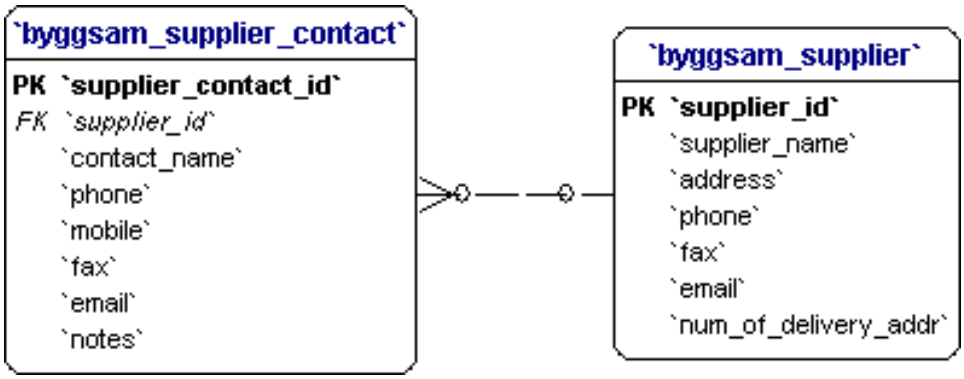
Siden det ikke er noe krav, eller det har blitt tillagt noe vekt på å ta vare på data, så er også tabellstrukturen noe enkel. Hovedsakelig kopieres data fra den åpne ordren over til den lukkede ordren, og man beholder store deler av tabellstrukturen. Noen felt endres fra å være dynamiske med relasjon til andre tabeller, til å inneholde statiske data fra disse tabellene. Noen felt har også blitt fjernet.



Figur 2.3: Effektuert ordre

Register over leverandørkontakter

For ordren sin del så kan det være hendig å påføre hvilken kontakt hos leverandøren som ordren retter seg mot. Tabellstrukturen for dette er illustrert i figur 2.4.



Figur 2.4: Leverandørkontakter

Dette er egentlig bare et enkelt register som inneholder info om denne kontakten, og hos hvilken leverandør denne personen er tilknyttet.

Beskjedsystem

Beskjedsystemet er tiltenkt å brukes til to ting. Det ene er å formidle generelle beskjeder som vises på strategiske steder som framsiden. Det andre er å sende beskjeder til personer som jobber med systemet.

Dette er ikke et avansert system som er ment å erstatte e-post, eller andre

Figur 2.5: Beskjedsystem

kommunikasjonssystemer, men mer ment som et supplement. Beskjeder som sendes til e-post har en tendens til å drukne i alle andre typer meldinger, og for å unngå å forsøple e-posten med flere meldinger, så er dette et forsøk på å begrense det.

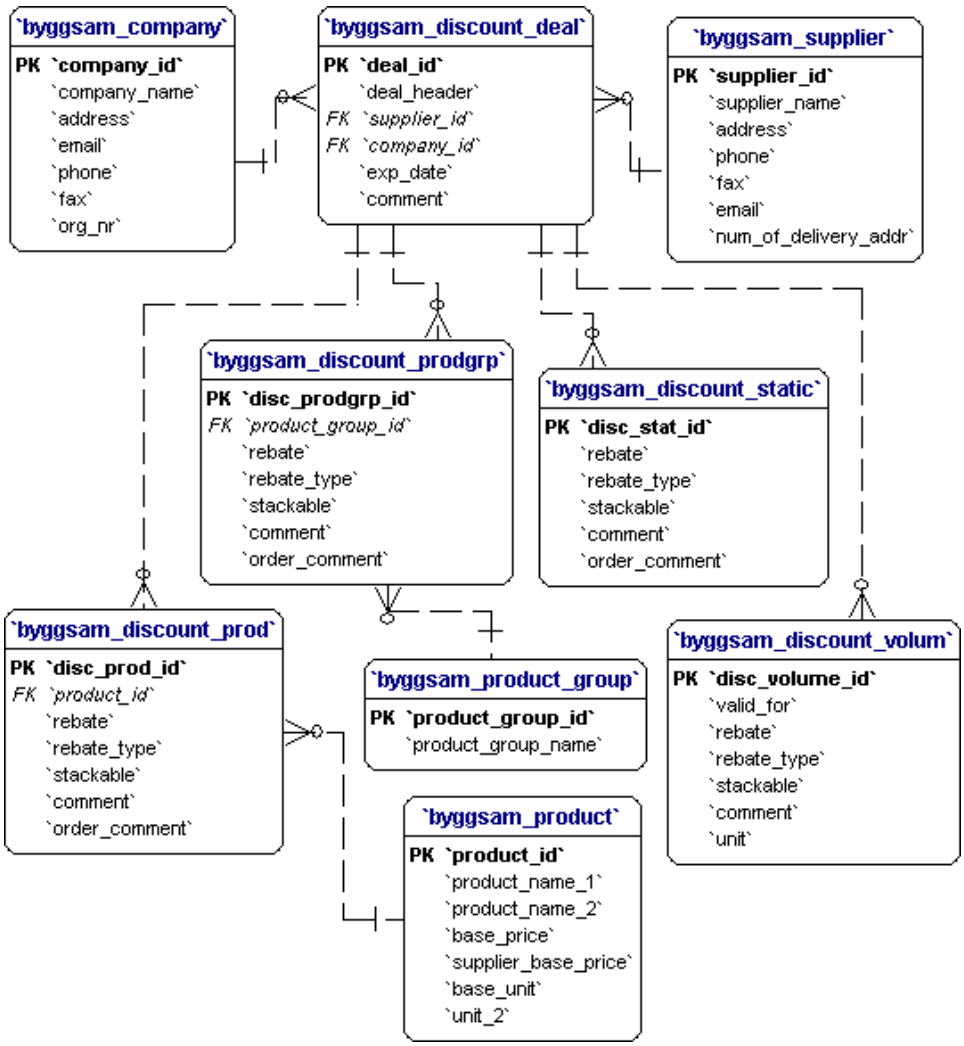
Det vil i stor grad være opp til brukerne om de syns dette er nyttig eller ikke. Men for generelle beskjeder kan det være en verdi i at de beskjedene vises i systemet, i stedet for at de har blitt sendt til andre plasser.

Rabattoversikt

Rabattavtaler som Bygger'n har med leverandører finnes bare på papiret slik det er i dag. Det å forsøke å lage et system for å overføre disse avtalene til data er komplisert, og noe vanskelig å få gjennomført. Avtalene er i stor grad individuelle med mange ulike finurligheter, det andre er å overføre dem til en generell forståelse som brukeren kan forstå.

En rabattavtale er en avtale mellom Bygger'n-bedriften og den spesifikke leverandør. Avtalen har ofte en bestemt grunnrabatt som gjelder på alle varer som bestilles. Det finnes rabatter som inntreffer ved ulik volum eller vekt, og rabatter som gjelder spesielle varer.

Hovedtabellen i figur 2.6 er 'byggsam_discount_deal'. Denne har kobling til leverandør og bedrift. Denne er ment å representere selve avtalen, og hvert rabattelement knyttes da til denne tabellen.



Figur 2.6: Rabattsystem

I dette systemet så er det 4 tabeller som tar for seg rabatt på enkeltprodukt, produktgrupper, faste rabatter, og rabatter som inntreffer ved et bestemt volum.

En rabatt kan enten oppgis i % som er det vanlige, men det kan like gjerne være et beløp. Dette representeres med 'rebate_type', som da enten settes til % eller kr. Alle rabattene har et kommentarfelt som er ment for intern informasjon til Bygger'n-bedriftene, og det er et kommentarfelt som er ment å tilføres selve ordren når rabatten benyttes. Noen rabatter kan nemlig kun benyttes om man oppgir en kode, eller en kontakt hos leverandøren. Volumrabattene som er rabatter som inntreffer ved et bestemt volum, har et felt som sier hva slags type volum det er snakk om, og et felt for når grenseverdien oppstår.

Når man ser på de 4 tabellene som har med rabatt-typene, så er ikke disse normalisert. Siden databasen ikke har støtte for fremmednøkler, så må man lage mye kode for å holde orden på relasjoner. Designet er noe enkelt, men også lett å forstå og utvide.

2.3 Implementering

2.3.1 Valg av utviklingsmiljø

Gruppen velger på et tidlig tidspunkt av planleggingsfasen å velge hva slags type applikasjon som skal lages, og hva slags språk og database som skal benyttes. Dette er fordi de valg man har tatt i designdelen er basert på dette valget.

Programmeringsspråk

Utviklingen av applikasjonen vil skje med programmeringsspråket PHP, og med databasen MySQL. Dette fordi gruppen totalt sett har best kunnskap og mulighet for raskt å sette seg inn i dette og gå i gang med arbeidet. Oppgaveteksten foreslo også å lage denne applikasjonen som en webapplikasjon, og for gruppen sin del så var det klart veldig tidlig at det skulle programmeres i PHP med MySQL som database. Det er heller ikke dukket opp ting i planleggingsfasen som skulle tilsi at PHP og MySQL er et dårlig valg.

Ved å utvikle en webapplikasjon så vil vi raskere kunne produsere et ferdig resultat, og komme langt i utviklingen med tanke på funksjonalitet. Dette er fordi man har ferdig kommunikasjonsprotokoller, brukergrensesnittkomponenter, og programvare som ligger i bunn og støtter opp om denne type program.

Alternativet kunne være å lage applikasjonen med Java, men da hadde mye tid gått med til å lage de grunnkomponentene nevnt ovenfor. Men samtidig så tilbyr Java langt større frihet til å lage grunnkomponenter slik man selv vil, og skape bedre brukeropplevelser.

En annen bakdel med Java er at 2 personer ikke har vært borti dette, og det vil ta tid før disse har lært nok Java. PHP blir derfor valgt over Java fordi gruppen mener at man kommer lengre i produksjon av selve resultatet med det valget.

Bakdelen med webapplikasjoner er at de er avhengige av de mange nettleserne som ikke gjør samme jobb likt. Nettleserne har også begrensinger på hva slags funksjonalitet man har på brukergrensesnitt nivå. Brukeren bruker den samme nettleseren til å besøke andre nettsteder, og det er klart at sikkerheten setter store begrensninger. Man ønsker ikke at ondsinnet kode skal få

gjøre hva den vil via nettleseren. Dermed så fjerner man mye funksjonalitet for å hindre mulighet for ondsinnet kode å gjøre noe skade. Det er mulig å få på plass mer funksjonalitet ved å benytte javaapplets, eller andre 3dje-parts programvare. Men ofte så introduserer disse mer kompleksitet overfor brukeren, at det kan bli vanskelig å rettferdiggjøre dette.

En annen ting med webapplikasjoner er at de kun er en serie websider. Denne stakkato måten å jobbe på setter noen begrensninger, og samtidig ødelegger noe av brukeropplevelsen. På en annen side så begynner folk generelt å bli vant med web, og vi ser ikke at det skal bli noe stort problem.

ASP er et alternativ til PHP og kunne bli brukt til denne type applikasjon. Men siden PHP ikke begrenser oss, og det ikke er avdekket at alternativene har noen spesielle tilleggsfunksjoner, så rettferdiggjør det ikke tiden det vil ta for alle å lære seg dem.

Ettersom PHP ble valgt som scriptspråk, var det naturlig å dra inn elementer fra PEAR i implementeringen. PEAR er et bibliotek med ferdiglagde klasser for diverse generell funksjonalitet, for eksempel for databasehåndtering, generering av HTML-skjemaer og generering av HTML-tabeller. Det er i første rekke pakkene DB, HTML_Quickform og HTML_Table som er benyttet under utviklingen av denne applikasjonen.

Databasen

Når valget falt på MySQL som database så er det mer et valg gjort av vane. Databasen er gratis og lett tilgjengelig. PHP har støtte for denne databasen, og den er mye brukt i mange prosjekt. De aller fleste eksempler, artikler og opplæringsmateriell bruker også MySQL, og skal man installere en webserver på egen maskin så kommer den gjerne med nettopp MySQL. Ønsker man å leie plass på webserver til å kjøre PHP applikasjoner så har ofte disse firmaene MySQL som database blant annet.

Gruppen kan ikke se at alternative databaser har noe mer å tilby, enn det som MySQL kan gjøre. Den siste tiden har Microsoft og Oracle kommet med sine gratisalternativer som kanskje gjør det mer attraktivt å prøve dem, men det er ikke et argument for å velge dem. Man kan ikke forvente i samme grad å finne støtte for andre databaser hos de firma som leier ut webplass heller. Samtidig så er det å håndtere en database noe som kanskje bør sette ut til profesjonelle uansett type. Da er det lurt å velge noe som man kan regne med alle har.

Applikasjonen som skal lages kommer neppe til å benyttes av mer en 3 personer samtidig i vanlig bruk, og aldri overstige maks 20 personer. Dette er en arbeidsmengde som alle databaser kan håndtere, og således ikke stiller noen krav egentlig. Vi har heller ikke avdekket andre spesielle krav eller behov som tilsier at vi burde vurdere andre databaser.

Versjonshåndtering

Det ble tidlig fastslått at en form for versjonshåndtering av kode og tekstdokumenter ville være både nyttig og nødvendig. Med flere personer som jobber på samme prosjekt vil det være nærmest umulig å holde oversikt over alle filer uten et eget system for dette. Ettersom Høgskolen satte opp en Subversion-tjener for hovedprosjektene, var valget egentlig gitt på forhånd. I ettertid ser vi at denne tjenesten ikke har blitt utnyttet til fulle. Gruppemedlemmene har hatt tildelte arbeidsoppgaver, og dette har i praksis betydd at man har jobbet med forskjellige filer. En jevn oppdatering mot Subversion-tjeneren har derfor til tider uteblitt. Dette medfører selvsagt risiko for dobbeltarbeid og datatap, men heldigvis har vi ikke gått på noen store blemmer her.

Backup

Vi har ikke hatt noen dedikert backup-løsning i denne perioden. I og med at vi har benyttet et versjonshåndteringssystem som Subversion, har vi allikevel hatt en sikkerhet på lagring av filer. Høgskolen kjører daglig backup på sine systemer der Subversion er installert, samtidig som hver enkelt av gruppemedlemmene til enhver tid vil ha kopier av disse filene lagret lokalt på sin maskin. Med jevne mellomrom er samtlige filer blitt kopiert ut på CD og memory stick, men det må påpekes at det ikke har vært noen bestemte rutiner for når dette har blitt gjort.

Kodestandard

Selve webapplikasjonen er utviklet med tanke på at den skal validere korrekt for XHTML 1.0 Strict. XHTML er en videreutvikling av HTML, kall det gjerne en kombinasjon av HTML og XML, og kan sees på som mer strukturert og oversiktlig enn standard HTML. Dette er en standard som flere og flere seriøse webutviklere følger, først og fremst på grunn av at det reduserer sannsynligheten for at sidene vises feil/ulikt i forskjellige nettlesere.

ByggSAM følger i sin helhet XHTML som standard, og validerer korrekt i

tilnærmet alle sammenhenger. Noen få unntak er det, PEARs HTML_Quickform-klasse har en 'feil' som genererer HTML-kode uten innhold i noen tilfeller, noe som i følge XHTML-standarden er unødvendig. De få sidene dette gjelder vil derfor returnere en XHTML-advarsel (med andre ord ikke en direkte feil) når man validerer koden.

2.3.2 Produksjon

Hovedmapper/filer

design/

Inneholder alle filer som definerer webapplikasjonens layout/utseende. Dette inkluderer både bilder og CSS stylesheets.

help/

Inneholder filene til systemets integrerte brukermanual.

includes/

Inneholder systemets kjernefunksjonalitet. Her er alle klasser, funksjoner og konstanter lagret i sine respektive filer. Hver klasse har sin egen fil, mens alle funksjoner og konstanter er samlet i hver sine filer. I tillegg finner vi en konfigurasjonsfil (config.php) som inneholder databaseinformasjon.

install/

Inneholder alle filer som blir benyttet under installasjonen av systemet.

javascript/

Inneholder all javascript-kode benyttet i systemet.

upload/

Her blir alle opplastede filer, backupfiler og systemgenererte filer lagret.

includes/config.php

Inneholder informasjon om databasen (brukernavn, passord, serveradresse og så videre). Innholdet i denne filen blir opprettet under installasjonen.

includes/constants.php

Inneholder globale konstanter.

includes/constants_db.php

Noen brukerkonfigurérbare systemkonstanter er lagret i databasen. Disse konstantene blir opprettet i denne filen.

init.php

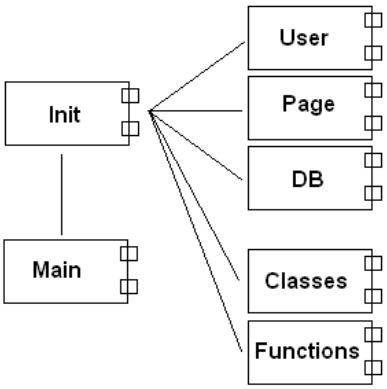
Systemets utpekte kjernefil. Denne tar seg av inkludering av alle filer som blir benyttet i systemet, det vil si alle klasser, funksjoner og så videre. Globale klasser blir opprettet, og kommunikasjon mot MySQL-databasen opprettes.

includes/functions.php

Inneholder globale funksjoner

Rammeverk

Hver webside som lages i systemet vil bestå av noen bestemte komponenter som utgjør basisen i rammeverket.



Figur 2.7: Rammeverket

Main

Dette er selve websiden. Det er her vi lager våre skjemaer, tabeller og annet innhold. Ved å inkludere Init så kan man i Main få Page til å generere hoveddeler av siden, få User til å sjekke om bruker er innlogget, og bruke DB til å hente eller legge inn data i databasen.

Init

Modulen i seg selv gjør ikke noe annet enn å sentralisere opprettelse og

gi adgang til User, Page og DB, samt gi global tilgang til noen generelle funksjoner og klasser.

Page

Dette er en klasse som har funksjoner for å generere hoveddelene til en webside. Med et par funksjonskall så kan man få på plass hele websiden. Dette er noe man gjør i Main. Vil man vise en tabell så sørger man bare får og kalle BeginPage() før tabellen, og EndPage() etter tabellen. Når man da henter frem websiden i nettleseren så er den komplett med logo, meny, bunntekst og et helhetlig design.

Kjernefunksjonalitet

Det første vi begynte med var selve kjernefunksjonaliteten i systemet. Databasetilkobling, innloggingsrutiner/sessions og webrammeverk er eksempler på funksjonalitet som kunne utvikles før vi egentlig hadde fått spesifisert noe detaljert fra oppdragsgiver.

Innloggings-scriptet ble laget med utgangspunkt i Martin Tsachevs artikkel 'Creating a Secure PHP Login Script'. Dette er en ganske avansert metode som skal sikre mot forskjellige typer autentiseringsangrep, i første rekke såkalt session stealing/session spoofing. Nøyaktig hvordan dette fungerer er beskrevet i [6].

Import av data fra S4

Som nevnt tidligere benytter Bygger'n på Bagn applikasjonen S4 for å administrere vareutvalget. Å registrere tusenvis av produkter, produktgrupper og leverandører manuelt i vårt system ville vært en uoverkommelig oppgave, og det ble derfor satt som et nødvendig krav å enkelt kunne overføre data fra S4. S4 har innebygd funksjonalitet for eksport av data. Et valg er å eksportere det i et proprietært og sperret format, et format som sannsynligvis inneholder alle data lagret i en intern database. I tillegg har man muligheten til å eksportere data ut i Excel-regneark. Det mest optimale hadde vært muligheten for å eksportere data i XML, noe utviklerne av S4 foreløpig ikke har lagt opp til. Det ble gjort et par forsøk på kontakte Advisor via epost, for å forhøre oss om det lukkede filformatet og muligheten for å benytte dette opp mot vårt system. Noe svar fikk vi aldri, så valget om å bruke eksport til Excel-fil ble tatt.

S4 eksporterer produktdataene ut i regneark med ett produkt per linje, og en kolonne for hver type produktinfo. S4 inneholder i utgangspunktet mye mer informasjon enn det som er nødvendig å importere for bruk i samkjøpsapplikasjonen, derfor må det gjøres et utvalg av kolonner ved eksport. Nøyaktig hvilke data som skal velges ved eksport er utdypt i brukermanualen.

Importscriptet går gjennom hver enkelt linje i den importerte filen, og sjekker informasjonen som ligger her mot det som allerede er registrert i databasen. Dersom varen (NOBB varenummer) på den aktuelle linjen ikke finnes, opprettes det en ny tabelloppføring på dette produktet. Dersom varenummeret allerede er registrert i databasen, utføres det en oppdatering av de aktuelle

Figur 2.8: Skjermbilde fra applikasjonen, import av data fra S4

varedataene. Tabellfelter som er ByggSAM-spesifikke, for eksempel varevekt, volum eller areal, påvirkes ikke av en slik oppdatering. Samtidig gjøres det tilsvarende sjekker mot leverandør-, produktgruppe- og enhet-tabellene, med nødvendig nyregistrering dersom noe av denne informasjonen ikke finnes fra tidligere.

Import av datafil fra håndscanner-enhet

Det ble tidlig klart at det ville være nyttig å kunne gå rundt i butikken med en håndscanner-enhet, scanne inn de varene man trenger mer av, for så å overføre dette til selve samkjøpsapplikasjonen. Oppdragsgiver hadde en bestemt modell som de brukte til dette, Intermec CK1. Denne håndscanneren genererer en datafil i txt-format med følgende informasjon:

VB;002;01;1005001;2.00
VB;002;01;1010016;2.00
VB;002;01;1010009;5.00
VB;002;01;1010004;127

Dette er 4 linjer fra en datafil som er generert av Intermec CK1. Som vi ser er det 5 felter separert av et semikolon, der de to siste feltene på hver linje angir henholdsvis produktID og antall. ProduktID er det tallet som scannes med håndscanneren, og kan enten være et EAN-nummer, et NOBB-nummer, eller leverandørens varenummer.

Figur 2.9: Skjermbilde fra applikasjonen, håndterminalinput

Prosessen for å få denne informasjonen inn i egne ordre i samkjøpsapplikasjonen, består egentlig av flere steg. Etter en kontroll på innholdet i den importerte datafila, kjøres det en rekke databasespørringer for å hente ut informasjon om for eksempel hvilke enheter som kan velges på hvert enkelt produkt, hvilken leverandør som tilbyr dette produktet og om det finnes eksisterende åpne ordre på disse leverandørene. Denne informasjonen koordineres så, og brukeren får opp hver enkelt importere linje med mulighet for å spesifisere enhet og hvor den aktuelle ordrelinjen skal plasseres (det vil si i hvilken ordre).

I utgangspunktet ble denne importfunksjonen låst opp mot nevnte håndscannerenhet, Intermec CK1. For å gjøre systemet mindre avhengig av dette, er det åpnet for muligheten for å registrere ulike typer håndscannerenheter.

Dette betyr at hvilken som helst håndterminal som genererer en datafil på

lignende format vil fungere sammen med ByggSAM. Med lignende format menes en datafil med linjer bestående av felter separert med en bestemt tekststreng. Nøyaktig hvordan en annen type håndterminal registreres, er forklart i detalj i brukermanualen.

Adminpanel

For å få tilgang til administratorpanelet er man nødt til å oppgi brukernavn og passord på nytt. Dette er gjort for å sikre systemet mot uautorisert admintilgang. Det er for eksempel ikke utenkelig at en bruker går fra datamaskinen han benytter, der han har en nettleser oppe og samtidig er innlogget i ByggSam. En hvilken som helst person kan her ta i bruk denne maskinen og overta brukersesjonen, og potensielt gjøre hva han vil dersom den innloggede brukeren har fulle administratorrettigheter. For å hindre at uvedkommende på denne måten får tilgang til administrativ funksjonalitet, må den aktive brukerens passord altså oppgis på nytt.

Dette problemet er løst ved å lagre selve filnavnet til scriptet som blir kjørt til enhver tid. Hver gang en side lastes, legges denne informasjonen inn i en session-variabel.

```
$_SESSION['currentPage'] = $_SERVER['SCRIPT_NAME'];
```

Når man så laster en admin-side, gjøres en sjekk på hvilken side brukeren kommer fra. Dersom han kommer fra en admin-side, får han fortsette uten forstyrrelse. Dersom siden han kommer fra ikke er en admin-side, avbrytes lastingen av ønsket side, og brukeren får opp et innloggingsskjema. Når brukeren så oppgir korrekt brukernavn og passord, redirectes han automatisk til adminsiden han opprinnelig ville vise.

```
if ($_SESSION['currentPage'] != $_SERVER['SCRIPT_NAME']) {
    $admin->adminLogin($_SERVER['REQUEST_URI']);
    exit;
}
```

Med andre ord; Så lenge brukeren forlater admin-området før han går fra datamaskinen han jobber på, kan han være trygg på at ingen uautoriserte personer får tilgang til administratorfunksjonalitet.

Figur 2.10: Skjermbilde fra applikasjonen, adminpanelet

Backup

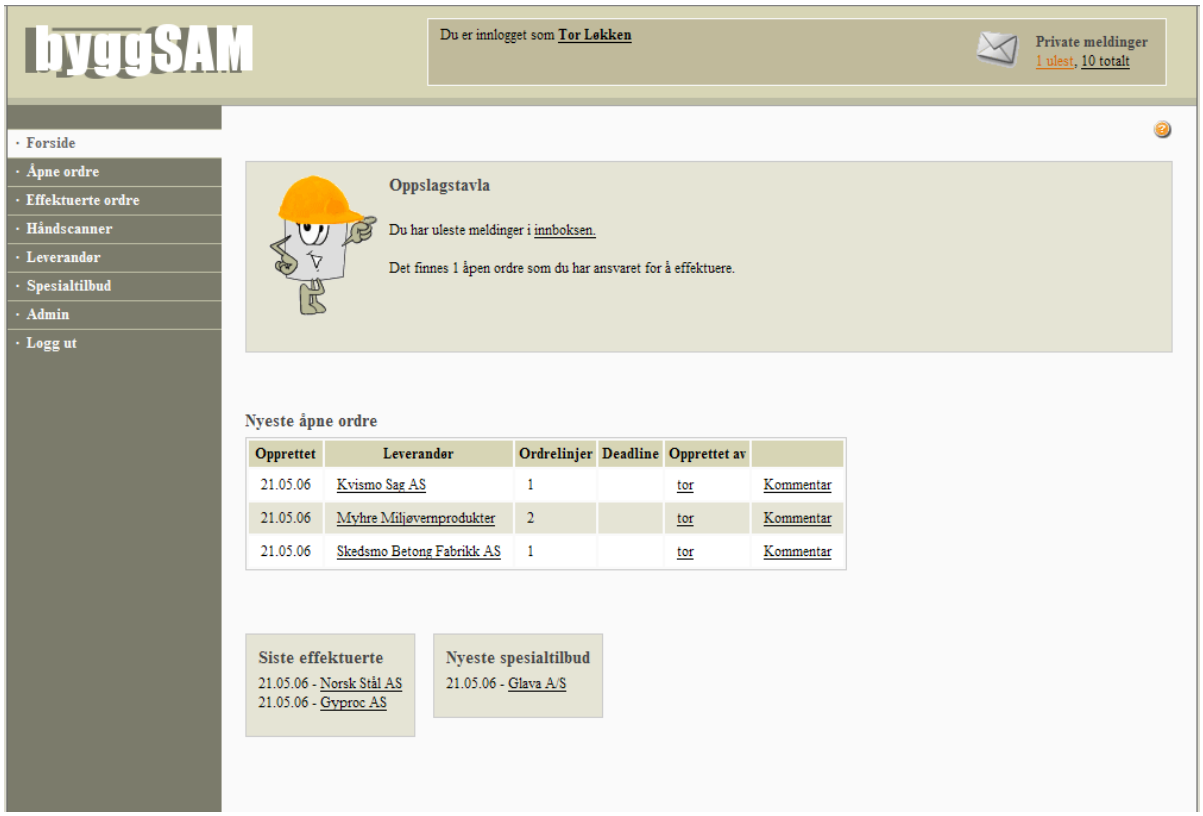
MySQL har innebygd funksjonalitet for backup i programmet mysqldump. Mysqldump kan kjøres via systemkallet 'system' i PHP. Denne metoden ble implementert og testet på 3 forskjellige server-installasjoner, alle uten hell (systemkallet returnerte ingen data). Etter en del prøving og feiling ble det avgjort at man var nødt til å lage et custom backup-script for databasen.

Ordreeffektuering

For å generere et ordredokument i PDF-format har gruppen benyttet klassen PHP Pdf Creation utviklet av R&OS [**?**] (http://www.ros.co.nz/pdf/).

Justeringer

Underveis i utviklingen ble det opparbeidet stegvis mer kunnskap og kompetanse innen PHP og programmering generelt, noe som medførte at kvaliteten på koden ble bedre og bedre. Man finner stadig vekk bedre måter å løse et problem på, både når det gjelder kodestruktur, sikkerhet og ikke minst hastighet. Som en konsekvens av dette har mye kode blitt omskrevet underveis. Et eksempel er skjemaer, som i starten ble laget manuelt på gamlemåten med HTML. Så oppdaget vi PEARs QuickForm-klasse, som byr på ferdiglaget kode for å lage skjemaer med PHP. Dette gjør skjemaene enklere og raskere å lage, i tillegg til at den har innebygd funksjonalitet for input-validering. Alle skjema i systemet er derfor laget med HTML_QuickForm.



Figur 2.11: Skjermbilde fra applikasjonen, hovedsiden

2.4 Testing

Applikasjonen inneholder i liten grad kompliserte funksjoner så testing har foregått ved at det i skjemaene manuelt er fylt inn lovlige og ulovlige verdier. Ulovlige verdier er verdier som er for store, for små eller på feil format. For eksempel må en epost-adresse inneholde alfakrøll. Når det ble funnet feil ble dette ordnet i koden med én gang og testing foretatt på nytt. I tillegg til denne testingen er koden gjennomgått i etterkant for å se etter mulige bakveier for hackere.

For å forsikre oss om at systemet kan brukes på flere plattformer, er det testet underveis i de tre mest brukte nettlesertypene på markedet, Internet Explorer, Firefox og Opera. Systemets fungerer like bra i alle disse tre.

2.4.1 Ytelsestest

Det er i første rekke import av data fra S4 og import fra håndscannerenhet som kan by på potensielle ytelsesproblemer i denne applikasjonen. Dette er prosesser med mange databasespørringer. De første testene med import av større datamengder bød på problemer. Webserveren gikk rett og slett på en timeout når datamengden og antall spørringer oversteg en viss grense. Disse testene ble utført lokalt på en laptop med en standardinstallasjon av EasyPHP, som jobbet mot høgskolens egen mySQL-server.

Det ble antatt at problemet med timeout kunne skyldes en kombinasjon av lav ytelse lokalt, samtidig som oppkoblingen mot databaseserveren naturlig nok ikke er optimal med en internett-tilkobling med begrenset båndbredde.

Webapplikasjonen ble derfor installert på en dedikert webserver hos et hostingselskap, og ytelsesforbedringene lot ikke vente på seg. Import av data fra S4 er testet uten problemer med et utvalg på 2400 produkter.

2.5 Realisering

2.5.1 Dedikert maskin eller webhotell

Vår programvare kan puttes på en egen maskin som kjører webserver, med PHP og MySQL, eller legges på et eget webhotell. Webhotell er servere som man leier plass på for å kjøre webapplikasjoner. Egne firmaer spesialiserer på denne type aktiviteter.

Dette er egentlig et spørsmål om support. Har man kunnskap nok til å sette opp egen maskin med nødvendig programvare, og supportere dette, er det greit å ha egen maskin. Det betyr at man må kunne installere og vedlikeholde operativsystem som Windows eller Linux, webserver som Apache, database som MySQL, PHP og annen nødvendig programvare. Man må kunne sikre maskinen mot innbrudd og virus, og reparere etter slike hendelser. Man må hele tiden sørge for å oppdatere programvaren, og holde seg oppdatert på utviklingen. Man må ta sikkerhetskopier ofte, og kunne tilbakeføre slike kopier når det er nødvendig. Man må kunne feilsøke i programvare og maskinvaren hvis problemer oppstår, og fikse ved behov. Det vil også være viktig at all support og drift skjer så fort at den tid maskinen er nede er så minimal at den egentlige programvaren som vår applikasjon ikke mister støtte hos brukerne.

Vi kan ikke se at Bygger'n har egne personer som er i stand til å drifte et slikt system på det nivået som trengs. Skal programvaren kunne bidra til å teste ut samkjøpkonseptet som de ønsker å prøve ut hos Bygger'n, er det en nødvendighet at i det minste det underliggende systemet fungerer. Dette blir best ivaretatt på et webhotell. Vi som studenter er heller ikke i stand til å levere en ferdig satt opp server som vi kan garantere er både sikker og stabil. Gruppen har derfor fokusert på å lage selve applikasjonen, fremfor å bruke tid på serveren.

Vi anbefaler å skaffe plass på et webhotell som har støtte for PHP og MySQL. Ved å benytte et webhotell så er hele grunnsystemet ikke lenger Bygger'n sitt ansvar. Da vil firmaet som har webhotellet sørge for sikkerhetskopiering, vedlikehold og sørge for at ting ellers er tipp topp.

Kostnadene for et webhotell er også langt mindre enn å kjøpe egen maskin. En datamaskin koster minst 7.000,- bare i maskinvarekostnader. I tillegg

kommer tid og penger på å skaffe programvare, lisenser, supportavtale, og arbeid med sette opp og betjene systemet. Muligens også utgifter til eksterne konsulenter for hjelp til ulike aspekter med et slikt system. Et webhotell med et domene får man for ca 600,- i året på det billigste.

Med tanke på at dette er et studentprosjekt og at det er en viss fare for at programvaren kanskje ikke vil bli brukt så mye, så blir det vanskelig å argumentere for annet enn webhotell.

2.5.2 Installasjon

Installasjon av systemet gjøres med et eget installasjonsscript. Nøyaktig hvordan dette skal gjøres er forklart i brukermanualen. Installasjonsscriptets funksjonalitet er inspirert av andre typer webapplikasjoner som gruppen hadde kjennskap til fra tidligere.

Kapittel 3

Avslutning

3.1 Drøftinger/diskusjoner

3.1.1 Resultatet

Hvor godt applikasjonen vil fungere som et verktøy for de tre Bygger'n-avdelingene er dessverre vanskelig å si. Grunnen til dette er at applikasjonen ikke har blitt levert ut, slik at man har fått sett det i bruk. Gruppen har heller ikke vært i kontakt med de andre brukerne og fått høre hva de ønsker av denne applikasjonen. Når produktet ikke er tatt i bruk, ikke er testet mot brukere, og heller ikke gjort en større evaluering med oppdragsgiver, så blir det vanskelig å vurdere hvorvidt det som er laget holder mål.

I forhold til listen over hovedfunksjonalitet i kravspesifikasjonen, er de fleste ting implementert. De fleste kravene har også blitt løst på en måte som gruppen er tilfreds med. Det er noen småting som mangler som å legge inn produkter manuelt. Den aller største mangelen er rabattsystemet. Store deler av rabattsystemet ble produsert, men det var ikke tid nok på slutten til å få det ferdigstilt og integrert i systemet. Det ble derfor gjort et valg å ikke ha med rabattsystemet i selve applikasjonen, siden den da ikke har noen funksjonalitet.

Det er to hovedgrunner til at rabattsystemet ikke ble ferdig. Det ene er at jobben med å finne ut hva slags funksjonalitet dette systemet skulle ha,

var noe dårlig. Det andre er at funksjonaliteten for hvordan rabattsystemet skulle integreres og benyttes i ordresystemet var noe dårlig planlagt. Begge disse tingene kunne lett blitt utbedret, men siden tiden ikke strakk til, ble dette droppet fra applikasjonen.

Til tross for dette har det vært et møte med oppdragsgiver mot slutten av prosjektet hvor applikasjonen ble raskt presentert. Inntrykket var at resultatet stort sett samsvarer med oppdragsgivers opprinnelige visjon.

3.1.2 Valg av database

Det har vært litt diskusjon i gruppen omkring valget av database på slutten av prosjekttiden. Valget om å benytte MySQL ble tatt veldig tidlig, og uten å vurdere alternativer. Nå er det slik at MySQL er den databasen som benyttes av stort sett alle som skal ha en database i kombinasjon med PHP og de var lett å falle for dette valget uten egentlig å stille spørsmålstegn om den var bra nok. Underveis i utviklingen så oppdaget gruppen at standard oppsatt MySQL ikke støtter fremmednøkler. Som standard er MySQL satt opp til å bruke MyISAM som motor, og ikke InnoDB som har støtte for fremmednøkler. Nå fortsatte gruppen å benytte MyISAM, både fordi all håndtering av relasjoner skjedde allerede i koden, og fordi det ville medføre for mye arbeid med å bytte database. Koden som benyttes forholder seg også mye til én og én tabell om gangen, og mye av databasedesignet er også forholdsvis enkelt. Derfor ville arbeidsmengden ikke blitt særlig redusert.

Helt på slutten av prosjektet oppdaget gruppen også noe om låsing i databasen, og hvordan dette kan være et problem for applikasjonen.

Lås i database benyttes når flere brukere/tråder skal ha tilgang til samme data. Dette forhindrer at flere brukere ikke ødelegger data når de skriver til databasen samtidig. MyISAM har kun mulighet for å låse selve tabellen. Nå er ikke dette et stort problem for dette systemet som stort sett aldri vil ha mer enn tre brukere inne samtidig, men i større sammenhenger med langt flere brukere så vil låsing av hele tabeller bli for tregt. Klassisk eksempel: Dette eksempelet er hentet fra boka MySQL, 3rd Edition By Paul Dubois. Published by Sams [5].

Dette er ikke et unikt eksempel. Flere artikler på internett bruker samme type eksempel når de skal snakke om låsing, men vi valgte å bruke dette

fordi det er et eksempel på både problemet og løsningen for det i MyISAM motoren.

Selger A har solgt 3 skjorter, og for å oppdatere databasen så begynner han med å hente antall skjorter fra databasen (47).

```
SELECT quantity FROM inventory WHERE item = 'shirt';
```

Selger B har samtidig solgt 2 skjorter, og begynner også med å hente antall skjorter fra databasen.

```
SELECT quantity FROM inventory WHERE item = 'shirt';
```

Selger A regner da ut nytt antall som er 47 - 3, og oppdaterer databasen

```
UPDATE inventory SET quantity = 44 WHERE item = 'shirt';
```

Selger B regner ut sitt antall som er 47 - 2 og oppdaterer databasen

```
UPDATE inventory SET quantity = 45 WHERE item = 'shirt';
```

Det som vi her ser er at databasen får feil antall. Den skulle vært 42, men er i stedet 45. Løsningen på dette er å bruke låsing.

Selger A selger 3 skjorter, og begynner å oppdatere databasen. Her vil det være nødvendig å låse tabellen for å hindre tilgang til den, før data er oppdatert.

```
LOCK TABLES inventory WRITE;
SELECT quantity FROM inventory WHERE item = 'shirt';
```

Selger B selger 2 skjorter og begynner også å oppdatere databasen. Men her så vil databasen hindre tilgang og sette selger B på vent inntil selger A låser opp.

```
LOCK TABLES inventory WRITE;
```

Selger A tar så regner ut nytt antall (47-3) og oppdaterer databasen. Når selger A er ferdig så låser han tabellen opp igjen.

```
UPDATE inventory SET quantity = 44 WHERE item = 'shirt';
UNLOCK TABLES;
```

Etter at selger A har låst opp så får selger B tilgang og kan hente antall skjorter som nå er 44.

```
SELECT quantity FROM inventory WHERE item = 'shirt';
```

Så kan selger A regne ut nytt antall (44 - 2) og oppdatere databasen med nytt antall. Når selger A er ferdig så låser han opp tilgangen.

```
UPDATE inventory SET quantity = 42 WHERE item = 'shirt';
UNLOCK TABLES;
```

Når det gjelder svakheten omkring låsing, blir dette mer en programfeil som kan utbedres. Nå vil det uansett ikke være smart å låse verken en tabell eller en hel rad i flere minutter hvis man jobber med å oppdatere en post, så her må en løsning i applikasjon på plass.

Den mest åpenbare plassen i applikasjonen hvor feilen kan oppstå er ved redigering av ordren, men her er det meningen at forskjellige personer skal ha ansvaret for sin del av ordren. Det finnes en mulighet for at brukerne kan gå og redigere data som en annen bruker jobber med, men det gjenstår å se om problemet faktisk inntreffer og blir til et reelt problem.

Det som er lærdommen fra dette er at man ikke skal velge ting blindt. Vi har lært noen nye ting om hva man skal tenke på når det gjelder database, og er derfor et par små erfaringer rikere.

3.2 Forslag til forbedringer

Applikasjonen kan forbedres på flere punkt, men før dette gjøres bør Bygger'n-bedriftene ha både testet ut applikasjonen, og gjort et større forsøk med å samarbeide om innkjøp. Slik gruppen ser det i dag, så mangler det retningslinjer og krav som har utgangspunkt i reelle behov. Derfor anbefales det å gjøre et større arbeid der Bygger'n-bedriftene tester ut dette samarbeidet og applikasjonen, samt at de ser på hvordan praktiske ting skal løses. Det er også et behov for å ha retningslinjer på hvordan man unngår misforståelser og problemer, og avklarer ansvar de ulike brukerne skal ha i dette systemet.

Gruppen har kommet over ulike ting som det ikke var tid til å implementere, men som kanskje bør vurderes å ta inn i applikasjonen. Noen eksempel på tilleggsfunksjonalitet som kan være nyttig å implementere er for eksempel å lage et fraktsystem slik at man kan få inn fraktkostnader direkte i ordren. Det kunne også være nyttig å ha muligheten til å modifisere layouten på PDF-filen. Ordren kunne muligens bli noe forbedret med å ha funksjonalitet for å se om ordren er sendt eller kun ferdigstilt, samt registrering av tilbakemeldinger fra leverandør på effektuerte ordre. Ferdigstilling av rabattsystemet og få det integrert i ordren kunne også bli vurdert, men det anbefales at det gjøres en ny kravspesifisering på dette for å få avdekket mer nøyaktig hvilken funksjonalitet dette systemet skal ha.

Siden denne applikasjonen har en lite optimal funksjonalitet for å importere data, så er dette en av de viktigste tingene som kan forbedres. Løsningen slik den er i dag er ikke god nok om man ønsker at løsningen skal vare. Her må det i såfall samarbeides med produsentene av S4 for å få tilgang til rådataene som ligger lagret i det systemet, eller eventuelt få de til å implementere en mer anvendelig eksportfunksjon, for eksempel med eksport av data i et XML-format.

Når det gjelder bedre kontroll over flere brukere som skal endre samme data, foreslår vi at disse dataene merkes med en timestamp, og når endringer skal gjøres så sjekker koden om det har vært endringer i tiden mellom data ble hentet fra en bestemt post, til oppdatering skal skje. Der man har select/update SQL-kode, bør man først låse tilgangen, gjøre select/update, så låse opp. Dette for å hindre at andre SQL-spørringer gjør endringer mellom en select- og update-spørring.

3.3 Evaluering av gruppas arbeid

3.3.1 Innledning

Før hovedprosjektet hadde aldri noen av gruppemedlemmene møttes. Alle tre hadde et ønske om å gjennomføre hovedprosjektet og etter noe konversasjon per e-post ble gruppen dannet.

3.3.2 Organisering

Tor ble valgt til gruppeleder og kontaktperson overfor oppdragsgiver og veileder. I de første månedene av prosjektet hadde gruppen noen møter med oppdragsgiver. Her fikk gruppemedlemmene en rask og grei innføring i bedriftens eksisterende datasystemer, hvordan innkjøpssamarbeidet fungerer i dag, ønsket funksjonalitet i applikasjonen og en del data som applikasjonen skulle bygge på.

Det har vært noen problemer i gruppen, da spesielt med kommunikasjon, planlegging og samarbeid. Alle de tre områdene påvirker hverandre også, noe som igjen har forsterket problemet.

Første gang gruppen hadde møte med oppdragsgiver var gruppen dårlig forberedt, noe som førte til at møtet ikke ble så konstruktivt som man kunne ønske. Etter møtet var det delte meninger i gruppen om de hadde fått god nok informasjon til å starte med utviklingen. Etterpå ble det forsøkt å diskutere ulike funksjoner som applikasjonen skulle ha, og det ble avdekket at det fortsatt var mange uavklarte spørsmål. Gruppen produserte da et dokument med disse spørsmålene, og dro tilbake til oppdragsgiver godt forberedt. Etter møtet hadde gruppemedlemmene fått gode tilbakemeldinger og arbeidet med å konkretisere og designe databasen gikk mye lettere.

Selv om dette er et eksempel hvor gruppen løste problemet, så har de fleste problemene vært uløste. Skal man ha et møte med oppdragsgiver, eller få oppdragsgiver til å komme, så må man rettferdiggjøre det med å ha noe å spørre om, eller vise til. Dårlig kommunikasjon og dårlig planlegging fører til at man ikke klarer å skape en grunn for å lage et møte. Avgjørelser som kanskje oppdragsgiver burde fått lov å være med å bestemme blir da tatt av gruppemedlemmene, eller blir utsatt.

Gruppen har lært av sine feil. Skulle det gjennomføres et prosjekt på nytt, må det være et større fokus på nettopp kommunikasjon, planlegging og samarbeid. Selv om gruppen kanskje ikke vet nøyaktig hvordan disse problemene skal forbedres, så vet de hvilke områder de må tilegne seg mer kunnskap om.

3.3.3 Fordeling av arbeidet

Arbeidsoppgavene ble hovedsaklig fordelt etter hovedfunksjonalitet ved starten av hver iterasjon så godt som mulig ut fra gruppemedlemmenes kompetanse. Tor har hovedsaklig jobbet med ordresystemet og innhenting av data fra bedriftens eksisterende database, samt systemets generelle funksjonalitet. Knut har hovedsaklig jobbet med kontakter, leveringsadresser og meldingssystem. Frode har hovedsaklig jobbet med rabattsystemet, produktgruppene og leverandørregister.

Gruppemedlemmene endte opp med følgende timeantall ved prosjektperiodens slutt.

Tor: 594 timer
Frode: 432 timer
Knut: 344 timer
Totalt antall timer: 1370

3.3.4 Subjektiv opplevelse av hovedprosjektet

Gruppen har hatt som fokus hele tiden å produsere selve applikasjonen. Gruppen har stått helt fritt i hvordan oppgaven skulle løses, og denne friheten har egentlig vært til det positive. Gruppen ville ikke ha kommet så langt i utviklingen om ukjente teknologier måtte tas i bruk, eller om gruppen måtte ha brukt mye tid på opplæring.

Prosjektet har resultert i en webapplikasjon som er klar til å bli tatt i bruk. Det har vært en konstant og jevn fremgang i produksjonen av applikasjonen, og det har egentlig ikke vært store tekniske utfordringer. Selv om programvaren har noen mangler, er den allikevel fullt kjørbar, og setter ingen begrensninger. Applikasjonen kan helt klart benyttes når Bygger'n-bedriftene ønsker å prøve ut konseptet med å samarbeide om innkjøp.

3.4 Konklusjon

Før prosjektstart hadde aldri gruppemedlemmene møttes før. Gruppemedlemmene hadde i tillegg veldig forskjellig kompetanse på programmeringsspråket som ble brukt. Dette har i seg selv gjort dette prosjektet ekstra utfordrende. Noe tid i begynnelsen ble brukt til å sette seg inn i ulike ting som skulle benyttes til prosjektet.

Resultatet av prosjektet er en applikasjon som føles langt mer fullstendig enn det som skoleprosjekter tidligere har klart å produsere. Det å ha kommet så langt gjør at troen på produktet blir sterkere, og gir en positiv selvfølelse for gruppen.

Dette produktet ble laget ved hjelp av PHP som programmeringsspråk, og MySQL som database. Dette valget har vist seg å være et godt valg, og gruppen klarte raskt å gjøre nytte av dette utviklingsmiljøet. Gruppen er enig i at om jobben skulle blitt gjort på nytt, så hadde de samme valg blitt gjort, muligens med noen endringer i selve databasen. PEAR-biblioteket som også har blitt benyttet gjorde jobben mer strukturert for oss, og har definitivt hvert til hjelp.

Bibliografi

[1] Advisor s4, http://advisor.projob.no/showpage.asp?id=59.

[2] Mysql homepage, http://www.mysql.com/.

[3] *PHP Manualen, http://www.php.net/.*

[4] Varedata oda as webside, http://www.varedata.no/.

[5] Paul Dubois. Mysql, 3rd edition.

[6] Martin Tsachev. Creating a secure php login script, http://www.evolt.org/article/php_coding_guidelines/18/60247/. 2002.

Kapittel 4

Vedlegg

- Brukermanual
- Webapplikasjon: forklaring
- Prosjektskisse
- Forprosjektrapport
- Statusrapporter
- Møtereferat
- Timelogg
- Endelig Gantt-skjema