HOVEDPROSJEKT:

S.A.F.E

Samling Av Fagbeskrivelser Elektronisk

FORFATTERE:

Mehrab Afradi
Fredrik Skarderud
Tor Bjørke
Ole Kasper Olsen

Dato:

19.5.2003

SAMMENDRAG AV HOVEDPROSJEKT

Tittel:	SAFE – Samling Av Fagbeskrivelser Elektronisk	Nr. : 2
		Dato : 19.5.2003
Deltakere:	Tor Bjørke	
	Fredrik Skarderud	
	Mehrab Afradi	
	Ole Kasper Olsen	
Veileder:	Frode Haug	
Oppdragsgiver:	Høgskolen i Gjøvik	
Kontaktperson:	Kjetil Ophus	
Stikkord (4 stk)	fagbeskrivelse, dynamikk, eksport, database	

Antall sider: **137**	Antall bilag: **8**	Tilgjengelighet (åpen/konfidensiell): **ÅPEN**

Kort beskrivelse av hovedprosjektet:

Hovedprosjektet har gått ut på å lage et system for Høgskolen i Gjøvik som omhandler produksjon, arkivering, registrering, presentasjon og kvalitetssikring av fagbeskrivelser. All relevant informasjon for å håndtere dette ligger lagret i en database der all data blir registrert med systemets Administrasjonsmodul og Registreringsmodul. Dette er to selvstendige applikasjoner som begge er plattformuavhengige. Systemet har en presentasjonsmodul som brukes til å søke frem å presentere eksisterende fagbeskrivelser på Web, og ved hjelp av systemets eksporteringsmodul kan fagbeskrivelser eksporteres til forskjellige filformater.

Med dette systemet har oppdragsgiver nå et verktøy som sikrer kvaliteten på fagbeskrivelser og som letter arbeidet med produksjon av studiehåndboka.

Forord

Kjære leser av denne prosjektrapporten for S.A.F.E (Samling Av Fagbeskrivelser Elektronisk). Det er et par ting vi gjerne vil gjøre deg oppmerksom på før du virkelig setter i gang med lesingen av rapporten.

Det første er at denne rapporten er et resultat av fire hardtarbeidende personers innsatsvilje hvis moral og styrke har blitt holdt oppe av et ukjent antall kaffekopper på sene kvelder og netter. Etter at dette prosjektets problemstilling ble gitt fra HiG, valgte vi denne og har siden jobbet mer eller mindre over egen evne for å komme i mål slik at vi kunne levere et verdig produkt til arbeidsgiver.

For det andre vil vi trekke fram de personer som har vært til inspirasjon gjennom hele utviklingsprosessen og gitt oss styrke til å fullføre blant annet det som du nå, kjære leser, sitter og holder resultatet av. Vi føler at vi skylder disse en stor takk, for uten disse ville veien frem blitt mye tyngre og ikke fullt så lystig.

Takk til Frode Haug for uvurderlig veiledervirksomhet.
Takk til Ivar Aaen for veggpryd og tidvis beskyttelse mot skulende øyne.
Takk til oppdragsgiver Kjetil Ophus for inspirerende ideer.
Takk til Øivind Kolloen for teksnisk støtte innen SQL server og Linux.
Takk til Orange Juice for lån av flagg.
Takk til Nils Opsahl ved HiGs IT-tjeneste for servicevennlig innstilling.
Takk til HiG for lån av grupperom.

Gjøvik, 19.5.2003

Tor Bjørke

Ole Kasper Olsen

Fredrik Skarderud

Mehrab Afradi

1 Innledning

1.1 Organisering av rapporten

Rapporten er delt inn i ni hovedkapitler med underkapitler som det fremgår av innholdsfortegnelsen der hvert nytt hovedkapittel skal starte på en ny side.

Kapittel 2 detaljerer hvilke krav som settes til systemet ved hjelp av en standard kravspesifikasjon. Kapittel 3 beskriver designet av systemet. Her inngår hvordan vi har grafisk utformet systemet, og hvordan underliggende funksjonalitet er designet. Videre går vi over på hvordan vi har implementert og produsert systemet i kapittel 4. Kapittel 5 tar seg av testprosessen og kvalitetssikringshenseende. I kapittel 6 er det satt sammen informasjon og veiledning angående installasjon av safe, før vi i kapittel 7 diskuterer og drøfter ulike sider ved utviklingsprosessen og måten vi løste oppgaven på. I kapittel 8 kommer en konklusjon av prosjektarbeidet og til slutt er alle vedlegg til rapporten samlet i kapittel 9.

1.2 Definisjon av oppgaven

1.2.1 Bakgrunn

Høgskolen i Gjøvik, heretter HiG, har lenge ønsket å finne en måte for å lette arbeidet med produksjon av studiehåndboka samt å få samlet og satt studiehåndboka i system. Studiehåndboka består av en samling av alle fagbeskrivelser, både for avdelingen for teknologi og avdelingen for helse, til hjelp for den enkelte student og lærer. Den inneholder i tillegg studieplaner, lover, regler, generell informasjon og viktige datoer. Studieplanen dokumenterer det faglige innholdet i studiene ved HiG, i tillegg til varighet, omfang og lignende.

Et av hovedproblemene per i dag er når fagansvarlige skal skrive sine fagbeskrivelser for studiehåndboka. De skal følge en mal for dette, men det viser seg at denne malen ofte er blitt mistolket. Dette fører til misvisende fagbeskrivelser og til mye redigering og retting for administrasjonen. Selv om mye av det som er feil blir rettet så er det ofte at fagbeskrivelsene ikke blir helt feilfri og dette kan føre til uoverensstemmelser mellom fagansvarlig og leserne av studiehåndboka.

De rutinene som inngår i produksjon av studiehåndboka per i dag er ikke gode nok og saksgangen er tungvinn. Dette innebærer at fagbeskrivelsene må gjennom mange ledd før den eventuelt blir godkjent og kan inkluderes i studiehåndboka. Dette er både ressurs- og tidkrevende. Ofte har alt dette også gjort at tidsfrister for endelig fagbeskrivelse har vært vanskelig å holde. Dette er naturligvis ikke til fordel for noen av de involverte partene, og noe HiG ønsker å gjøre noe med.

1.2.2 Problemområde

Hvordan lage fagbeskrivelser som er mest mulig fri for skrivefeil og har en mest mulig konsekvent ordbruk. Å ha en fagbeskrivelse som tilfredsstiller disse kravene er veldig viktig siden fagbeskrivelsen til et fag nesten kan regnes som en kontrakt mellom studenten og skolen. Feil og mangler i denne kan føre til uoverensstemmelser som vil være negativt for alle parter.

Det eksisterer systemer som mer eller mindre er tiltenkt å løse dette problemet, men disse er ofte veldig spesialiserte og lite egnet til å håndtere endringer. Andre løsninger er mer

manuelle som det å følge en mal utdelt på papir eller at man skriver hele fagbeskrivelsen uten noen form for mal i det hele tatt.

De fleste av dagens skoler er utstyrt med pc-er til de fleste ansatte og nettverk som binder disse sammen. Derfor vil det være naturlig å benytte seg av dagens teknologi til å både lage og lagre fagbeskrivelser elektronisk, og deretter samle disse på en sentral database. Hvis man gjør dette vil det også være mulig å drive søk i databasen.

1.2.3 Avgrensning

HiG ønsker å benytte seg av dagens teknologi for å forenkle arbeidet med å håndtere fagbeskrivelser siden dagens system ikke tilfredsstiller skolens krav. Dette innebærer et nytt system som benytter skolens nettverk for registreringen av fagbeskrivelser. Videre ønsker de elektroniske versjoner av disse lagret sentralt på en server for å kunne søke i både nye og gamle versjoner, samt å lette prosessen for å få en ferdig papirversjon av studiehåndboka. De elektroniske fagbeskrivelsene skal kunne presenteres på Web for alle interessenter som studenter, lærere og potensielle nye studenter. Her skal det også være mulig å søke seg frem til både nye og gamle fagbeskrivelser.

1.2.4 Oppgavedefinisjon

Det vi skal ta tak i er å lage et system som kan registrere fagbeskrivelser på en slik måte at fagansvarlig blir tvunget til å velge mellom forskjellige valg i de forskjellige feltene i en mal på en slik måte at antall forekomster av feil og mangler i fagbeskrivelser vil bli vesentlig mindre i forhold til dagens ofte ukonsekvente ordbruk. Systemet må også ta hånd om oppbygging av den malen som registreringen av fagbeskrivelser skal følge og lagre denne sentralt slik at de som skal registrere fagbeskrivelser kan hente ut og bruke den. Systemet skal på en god måte ivareta arkivering av fagbeskrivelser og maler for senere oppslag. For å gjøre fagbeskrivelsene tilgjengelige etter registrering skal systemet kunne eksportere fagbeskrivelser til fil, og publisere fagbeskrivelser på Web.

1.3 Rapportens målgruppe

Målet med denne rapporten er å reflektere prosjektets gang og produktets kvalitet for å gi leseren et innsyn i måten vi har løst oppgaven på og hva som ble resultatet. Det er også en rapport som skal gi de som skal anvende produktet et innsyn og en bedre forståelse for hvordan produktet virker og oppfører seg. Målgruppen for rapporten vil derfor være oppdragsgiver, faglig veileder, de fagansvarlige ved HiG og ansatte i studieadministrasjonen som har ansvaret for studiehåndboka.

Rapporten vil innholde en del tekniske elementer som egentlig ikke er beregnet for brukerne av systemet. Dette er for de personene som vil komme til å drifte systemet i fremtiden. Vi vil også ha i bakhodet at produktet, i fremtiden, kanskje vil komme til å videreutvikles av studenter ved HiG. Derfor er det viktig å gi et så godt som mulig innblikk i hvordan gruppa har kommet frem til det ferdige produktet for at det videre skal være mulig å videreutvikle applikasjonen på en best mulig måte.

1.4 Studentenes faglige bakgrunn

Alle studentene på hovedprosjektgruppa går 3-årig dataingeniørutdannelse ved Høgskolen i Gjøvik, med spesialisering i systemutvikling. Alle har derfor HiGs standard faglige programmeringsballast med grunnleggende og objektorientert programmering i C++ i tillegg til algoritmiske metoder. I tillegg har alle på gruppa hatt valgfaget Programmering mot WWW, som blant annet består av programmering i JavaScript og Java, og alle har også

hatt valgfaget Grafiske brukergrensesnitt og har derfor tidligere kjennskap til Borlands produkter. Andre fag som er ytterst relevante for dette prosjektet er Databaser I og Databaser II som til sammen gav god innføring i oppbygging av databaser og databasespørrespråket SQL.

To av gruppens medlemmer har i tillegg til overstående faglige bakgrunn også hatt valgfaget Klient- og serversideprogrammering mot WWW hvor programmeringsspråkene Java, ASP og PHP ble gjennomgått.

Alle på gruppa har altså tidligere hatt erfaring med Java som programmeringsspråk, men ingen har erfaring utover små, enkle og grunnleggende Java-programmer.

Gruppa har tidligere vært sammen om å utføre et prosjekt i Objektorientert Systemutvikling ved HiG, og har derfor god kunnskap til hverandre. Denne kunnskapen har vist seg nyttig siden vi da har kjent til hverandres sterke og svake sider som har ført til at blant annet ved tildeling av forskjellige oppgaver har det vært mulig å sette de rette personene til å gjennomføre disse oppgavene.

Ingen av gruppens deltakere hadde erfaringer med utviklingsprosjekter fra arbeidslivet, da dette prosjektet ble startet.

1.5 Valgte arbeidsformer

For å samle informasjon som er nødvendig for å kunne løse oppgaven er intervjuer vesentlig. Siden oppdragsgiver er HiG og vår kontaktperson for oppdragsgiver befinner seg på skolen var det mulig å opprettholde en nesten kontinuerlig dialog. For å gjøre det noe enklere for oss tok vår kontaktperson for skolen selv ansvar for å videreformidle alle våre spørsmål vi hadde til de aktuelle personene på skolen og så gi oss tilbakemelding på de svarene vi fikk.

Av skolen fikk gruppa tildelt grupperom i kjelleren i A-bygget som blir delt sammen med en annen gruppe. Her har mesteparten av arbeidet pågått etter at vi fikk etablert utviklingsmiljøet. Dette tok noe lenger tid enn ventet siden det utstyret som vi fikk lånt av IT-tjenesten på HiG ikke tilfredsstilte våre behov. Det ble derfor vedtatt å gå til innkjøp av to maskiner som skal selges etter prosjektets slutt.

Selv om gruppa ikke hadde noe særlig erfaring med de verktøy som skulle brukes ved utvikling av prosjektet, ble det å tilegne seg kunnskap og å lære å bruke disse verktøyene av likevel en forholdsvis grei prosess. Måten vi tilegnet oss denne kunnskapen var stort sett ved å finne lesestoff, informasjon og ikke minst ”prøv og feil” metoden.

Vi har for det meste sittet to og to sammen når vi har jobbet med prosjektet og dette har vært en viktig faktor for måten vi har valgt å jobbe på. Under arbeidet med prosjektrapporten har det blitt mer individuell jobbing for å effektivisere skrivingen av denne. Selv om vi ikke har jobbet med det samme til enhver tid har vi for det meste oppholdt oss på samme rom med løpende diskusjoner og dialog slik at alle store avgjørelser ble tatt på en mest mulig enhetlig måte.

Rapporter og dokumenter har ikke alltid blitt utarbeidet i fellesskap men de har alltid blitt gjenomgått med alle medlemmene i gruppa og eventuelle feil har blitt endret, hvis det var noen som hadde noe å utsette på ting, før det ble levert til de personene som skulle ha det.

1.6 Terminologibruk og praktisk oppsett

Utover i rapporten vil det dukke opp en del faglitterære uttrykk og engelske ord som vil være vanskelig for en del personer å forstå og derfor trenger nærmere forklaring. Disse og andre ord og uttrykk som er av en slik karakter at leseren kan ha problemer med å forstå dets fulle betydning vil være samlet og beskrevet i vedlegget terminologiliste.

Der det er mulig har vi prøvd å benytte oss av det norske språket, men enkelte ord og uttrykk i dataverden har enten ikke noe tilsvarende på norsk i det hele tatt eller så er det ofte dårlig beskrivende i forhold til engelske. Dette grunnet at dataverden er et internasjonalt fellesskap som bruker engelsk som et naturlig felles språk. I tillegg florerer det av gode og dårlige akronymer og forkortelser som er i vanlig bruk. Disse er også listet opp i terminologilisten.

Selve oppsettet av rapporten vil følge prinsippene beskrevet under.

Skrifttyper og kapitteloverskrifter vil være som følger:

Stil	Størrelse	Skrifttype	Andre attributter
Kapitteloverskrift nivå 1	18	Garamond	Fet
Kapitteloverskrift nivå 2	16	Garamond	Fet, kursiv
Kapitteloverskrift nivå 3	14	Garamond	Fet
Kapitteloverskrift nivå 4	12	Garamond	Fet, kursiv
Kapitteloverskrift nivå 5	12	Garamond	Fet
Normal skrift	12	Garamond	Ingen

Kildekode i rapporten er skrevet med fonten Courier New, skriftstørrelse 10. Java-kode vil bli vist på lys rød bakgrunn, PHP-kode på blå bakgrunn, CSS-kode på gul bakgrunn og HTML-kode på grønn bakgrunn. Alle med svart ramme. Se under for eksempel.

```
Java
PHP
CSS
HTML
```

Figurtekst vil være skrevet med fonten Garamond, skriftstørrelse 10 og fet. Denne teksten skal være oppbygd med teksten ”Figur” så nummer på kapittel og hvilken figur i rapporten det er med bindestrek mellom og til slutt en liten beskrivelse av figuren. Eksempelvis, ”**Figur 3-2 dette er en beskrivelse**”. Figurtekster vil man finne midtstilt under figuren.

Henvisninger til andre kapitler vil skrives med teksten ”se kapittel” og så kapittelnummeret. Eksempelvis, ”se kapittel 1.2.1”.

Der det er referert til vedlegg vil teksten ”se vedlegg” brukes, etterfulgt av hvilket vedlegg det dreier seg om. Eksempelvis, ”se vedlegg A”.

Der det er naturlig å referere til en URL vil disse settes opp i en fotnote for å unngå lange adresser i selve teksten.

2 Kravspesifikasjon

2.1 Innledning

Da vi startet arbeidet med prosjektet hadde vi allerede hatt et prosjekt i objektorientert systemutvikling der vi skulle utføre de to første fasene i RUP som for det meste går ut på å lage en kravspesifikasjon ved å bruke UML modellering. Oppgavebeskrivelsen vi hadde i dette prosjektet var den samme som for dette hovedprosjektet med unntak av noen små endringer. Det vil si at da vi startet med hovedprosjektet så hadde vi allerede en kravspesifikasjon der mye kunne gjenbrukes. Likevel så var det en del nye aspekter som hadde kommet inn etter at prosjektet i systemutvikling var ferdig, og gamle aspekter som måtte forandres.

Ingen på gruppa har noe nevneverdig erfaring med skriving av tradisjonell kravspesifikasjon utover den øvelsen vi har fått i forbindelse med vår utdanning innen systemutvikling. Dette er på grunn av at denne måten å formulere en kravspesifikasjon på ikke lenger er så enerådende som den har vært før. Det har kommet flere og flere alternativer til hvordan man vil legge opp malen til en kravspesifikasjon.

Det å skrive en fullstendig kravspesifikasjon i starten av prosjektet er ikke helt i samsvar med den utviklingsmodellen som vi har valgt. Derfor har vi valgt å bruke mye av det vi fikk som resultat i det forrige prosjektet til grunnlag for vår kravspesifikasjon her. Dette gir oss en kravspesifikasjon som ikke følger noen spesiell mal, men som likevel dekker de vesentlige kravene til systemet, klart og tydelig.

2.2 Brukerscenario

2.2.1 Brukerprofiler

Dette er en beskrivelse av de forskjellige brukerne av systemet og hvilke krav som stilles til dem. Kravet til brukerne vil variere rundt de forskjellige delene av systemet siden den enkelte bruker kommer til å bruke forskjellige deler av systemet. Det er viktig å få gitt en god beskrivelse av de enkelte brukergruppene for å få klarhet i hvilke kunnskaper og erfaringer de har slik at det vil være mulig å utvikle et brukergrensesnitt og en funksjonalitet som vil gjøre det lettest mulig for den enkelte bruker å benytte systemet.

2.2.1.1 Fagansvarlig

Betegnelsen brukes for faglærere som skal registrere fagbeskrivelser. Fagansvarlige vil være hovedbrukere av registreringsmodulen som brukes for registrering av fagbeskrivelser. Disse har alt fra veldig mye til veldig lite kunnskap og erfaring innen databruk. Dette skyldes at det er faglærere ved HiG som ikke bruker data i sitt daglige arbeid i det hele tatt mens andre både underviser og bruker data som et sentralt hjelpemiddel i det dagligdagse liv.

Et punkt man bør ta spesielt hensyn til når det kommer til fagansvarlige er at når de tidligere har fylt ut malen for fagbeskrivelser, der alle punkter i praksis har fungert som fritekstfelter, har hvert enkelt fagansvarlig har fylt ut denne malen på sin egen måte. Når de nå vil bli tvunget til å følge en mye en strengere mal så vil mange føle at de ikke får uttrykt seg på den måten de ønsker. Det er da en stor sjanse for at disse vil komme med kritikk av systemet. Da er det viktig at man ikke etterkommer denne kritikken uten å analysere den grundig først.

Siden denne brukergruppen varierer så mye innen kunnskap og erfaringer kan det ikke stilles noen spesielle krav.

2.2.1.2 Avdelingsstyre

Skolen har to avdelinger; avdelingen for teknologi og avdelingen for helsefag. Hver av disse avdelingene har et avdelingsstyre. Avdelingsstyrets oppgave er blant annet å vurdere og eventuelt godkjenne fagbeskrivelser. Fagbeskrivelsene kan ikke benyttes uten en slik godkjenning. Det stilles ikke noen spesielle krav til denne gruppen brukere siden de i veldig liten grad vil være i kontakt med og bruke systemet.

2.2.1.3 Studieadministrasjon

Studieadministrasjonen har overordnet ansvar for fagbeskrivelser og studieplaner. Det er representanter for studieadministrasjonen som skal fastsette malen for fagbeskrivelser ved HiG. Studieadministrasjonen har ansvaret for sammensetting og trykking av den årlige studiehåndbok.

Innenfor denne gruppen brukere har alle en del erfaring innen bruk av data og forskjellige datasystemer ettersom dette inngår i deres daglige arbeid. Dette er veldig nyttig siden den delen av systemet som denne gruppen bruker kommer til å være noe mer komplisert enn resten av systemet. Det stilles ingen spesielle krav bare allmenn kunnskap om bruk av datasystemer.

2.2.1.4 Web-bruker

Siden fagbeskrivelser skal være tilgjengelig for alle på Web, vil denne brukergruppen bli stor og bestående av både ansatte ved HiG, kommende og eksisterende studenter, og andre som kan ha interesse av å ha innsyn i skolens fagbeskrivelser og studieplaner.

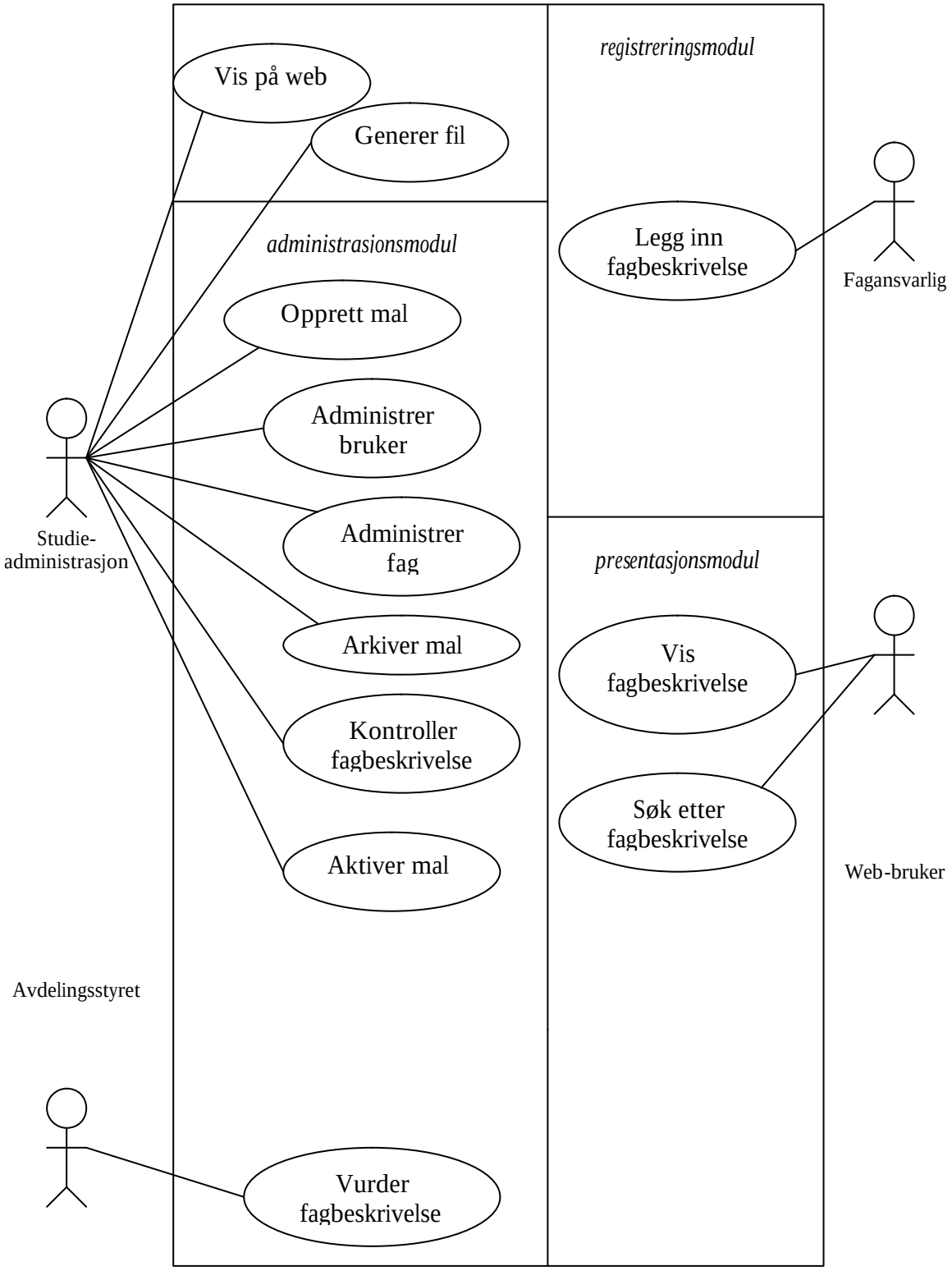
Det eneste kravet som stilles til denne brukergruppen er at de har kjennskap til bruk av en nettleser og hvordan man navigerer på Web.

2.2.2 Use Cases

Brukerens syn på systemet er ofte noe annerledes enn utviklerens. Man sier at bruker og utvikler har forskjellige konseptuelle modeller av systemet. Use Cases gir uttrykk for interaksjonen mellom bruker og systemet. De skal på en overordnet måte vise hvordan brukeren navigerer gjennom GUI-et, og hvordan brukeren oppfatter systemet, dvs. funksjonalitet brukeren ser.

2.2.2.1 Use Case-diagram

Use Case-modellen under skal vise hvilke deler av systemet den enkelte bruker ser. Viktige Use Cases beskrives hovedsakelig med tekst, men modellen under brukes til å få et overblikk.



Figur 2-1 - viser brukernes mål ved systeminteraksjon

2.2.2.2 Use Case: Legg inn fagbeskrivelse (registreringsmodul)

Primæraktør: Fagansvarlig

Interessenter og deres interesser:

- Fagansvarlig ønsker å kunne registrere fagbeskrivelsen kjapt, enkelt og greit ved hjelp av en mal
- Avdelingsstyret ønsker korrekt utfylte fagbeskrivelser for senere godkjenning

Prebetingelser:

- Fagansvarlig må være identifisert og autorisert.

Postbetingelser:

- Utkast til fagbeskrivelse ble kreert.
- Utkast til fagbeskrivelse ble lagret.

Beskrivelse (’happy-day scenario’):

1. Velger mal fra liste.
2. Velger fagkode fra liste.
3. Systemet fyller ut de felter som er mulige å fylle ut basert på sist lagrede versjon av fagbeskrivelsen.
4. Fagansvarlig redigerer aktuelle felter og lagrer.
5. Systemet kontrollerer om alle obligatoriske felter er utfylt.
6. Systemet lagrer fagbeskrivelse.
7. Systemet lagrer fagansvarlig i endringsloggen for fagbeskrivelsen.

Alternative scenarier:

2a. Fagkoden eksisterer ikke i systemet.
Fagansvarlig velger opprett nytt fag.
Systemet oppretter nytt fag med en midlertidig fagkode.

2b. Fagkoden som fagansvarlig har valgt skal ha ny kode.
Systemet oppretter faget med en ny midlertidig fagkode.

2c. Fagkoden har en godkjent fagbeskrivelse for året.
1. Systemet gir melding til fagansvarlig om at fagbeskrivelsen er godkjent og at fagansvarlig ikke får redigere på denne.
Systemet går tilbake til valg av fagkode.

3a. Det finnes ingen tidligere lagrede fagbeskrivelser for gjeldende fag i systemet.
Fagansvarlig må fylle en mal som ikke er basert på noen tidligere lagret fagbeskrivelse.

5a. Ett eller flere obligatoriske felter er ikke utfylt.
1. Systemet gir en melding til brukeren om hvilke obligatoriske felter som ikke er utfylt.
2. Systemet lagrer alle felter som er utfylt.
3. Systemet går tilbake til utfylling av felter.

Spesielle krav:

- Innfylling av skjemaet skal tvinge brukeren til å fylle ut riktig.

Mockup:
En ”mockup” er kun en idéskisse på hvordan man kan tenkte seg innleggingen av fagbeskrivelsene, og denne er kun ment for å vise hvordan registrering av fagbeskrivelser kan oppleves for en fagansvarlig. Denne mockup-en er laget i HTML og er ikke ment å gi noen antydninger til implementasjonsspråk, men kun som et eksempel på hvordan man kan styre og veilede fagansvarlig ved innlegging av fagbeskrivelsen.

Kurskode:	F164A
Fagnavn:	Databaser III
Course Title:	Databases III
Studium	bachelor i informatikk bachelor i ingeniørfag, data bachelor i ingeniørfag, bygg master i medieteknikk master i informasjonssikkerhet
Fagkategori	Grunnlagsfag
Studiepoeng:	5
Forutsetter:	F2523L - Programmering F2343F - Kantineteknikk legg til / fjern
Mål:	
Pedagogiske metoder:	forelesninger øvinger øvingsoppgaver
Evalueringsformer:	Skriftlig eksamen - 50% Prosjekt - 50% Karakterskala: A – F legg til / fjern
Pensum:	tittel: Mastering SQL forfatter: En Luring ISBN: 2341234-awd legg til fler

2.2.2.3 Use Case: Opprett mal (administrasjonsmodul)

Primæraktør: Studieadministrasjonen.

Interessenter og deres interesser:

- Studieadministrasjonen ønsker å opprette en mal for fagbeskrivelser.
- Fagansvarlig ønsker å ha en korrekt mal for utfylling.

Prebetingelser:

- Studieadministrasjonen er identifisert og autorisert.

Postbetingelser:
- Malen ble lagret.

Beskrivelse (’happy-day scenario’):
1. Velger om en mal skal bygge på en gammel mal eller være helt tom.
2. Får spørsmål om å bekrefte den nye malen.
3. Velger felter som skal være med i den nye malen.
4. Velger eventuelle valg som skal være med i feltene.
5. Setter opp eventuell avhengighet mellom valg.
6. Systemet lagrer endringer.

Alternative scenarier:
3a. Feltet finnes ikke i systemet.
Studieadministrasjonen legger til nytt felt.
Velger type for det nye feltet.
3b. Et felt skal ikke være med i malen.
Studieadministrasjonen sletter feltet fra malen.
4a. Ett felt innolder ikke de rette valgene.
Studieadministrasjonen redigerer valgene i feltet etter ønske.

Spesielle krav: Med sletting av felter, menes det at feltene ikke tas med i den gjeldende malen. Feltene er fortsatt lagret i databasen, for å forhindre at eldre fagbeskrivelser blir ødelagt.

2.2.2.4 Use Case: Vis fagbeskrivelse (presentasjonsmodul)

Primæraktør: Web-bruker

Interessenter og deres interesser:
- Studentene ønsker tilgang til gjeldende og utdaterte fagbeskrivelser via Internet, både på engelsk (ECTS) og på norsk.

Prebetingelser:
- Godkjent fagbeskrivelse må eksistere for det enkelte fag.

Postbetingelser: ingen

Beskrivelse (’happy-day scenario’):
1. Bruker velger fagbeskrivelse fra en liste, via direkte link eller via søk.
2. Systemet henter ut data, og genererer siden.
3. Systemet viser forespurt fagbeskrivelse.

2.2.2.5 Use Case: Generer fil (eksporteringsmodul)

Primæraktør: Studieadministrasjonen

Interessenter og deres interesser:
- Studieadministrasjonen ønsker en funksjon som kan ved hjelp av noen tastetrykk generere og sette sammen fagbeskrivelsene til en fil.
- Studenter og ansatte ønsker en papirutgave av fagbeskrivelser som er fri for lingvistiske og strukturelle feil.

Prebetingelser:

- Representant for studieadministrasjonen må være logget inn.

Postbetingelser:
- Fil ble laget.

Beskrivelse (’happy-day scenario’):
1. Velger hvilken mal for hvilket år det skal genereres fil(er) fra
2. Velger hvilke fagbeskrivelser som skal være med.
3. Velger hvilke felter fra malen som skal være med, og i hvilken rekkefølge disse skal stå.
4 Velger om alle fagene skal på egen fil, eller om det skal være linjeskift mellom dem.
5. Trykker på lagre og velger hvilket format filen skal være på (PDF/RTF).
6. Systemet setter sammen fagbeskrivelsene og annen relevant data som eksporteres til valgt fil.

2.2.2.6 Use Case: Administrer bruker (administrasjonsmodul)

Primæraktør: Studieadministrasjon

Interessenter og deres interesser:
- Studieadministrasjonen ønsker å holde orden på brukerne av systemet og deres rettigheter.
- Brukere ønsker å få tilgang til systemet.

Prebetingelser:
- Representant for studieadministrasjonen må være logget inn.

Postbetingelser: Ingen

Beskrivelse (’happy-day scenario’):
1. Velger en allerede registrert bruker fra systemet eller oppretter en ny bruker.
2. Systemet viser informasjon om brukeren.
3. Utfører eventuelle endringer.
4. Systemet lagrer endringer.

2.2.2.7 Use Case: Kontroller fagbeskrivelse (administrasjonsmodul)

Primæraktør: Studieadministrasjon

Interessenter og deres interesser:
- Alle ønsker en fagbeskrivelse som er mest mulig fri for feil.

Prebetingelser:
- Representant for studieadministrasjonen må være logget inn.

Postbetingelser: Ingen

Beskrivelse (’happy-day scenario’):
1. Velger mal for fagbeskrivelser.
2. Systemet returnerer fritekstfelter for valgt mal.
3. Velger felt som skal kontrolleres.
4. Systemet returnerer og viser valgt felt for alle fagbeskrivelser.
5. Kontrollerer og utfører eventuelle endringer.

6. Systemet lagrer endringer.

2.2.2.8 Use Case: Arkiver mal (administrasjonsmodul)

Primæraktør: Studieadministrasjon

Interessenter og deres interesser:

- Studieadministrasjonen ønsker å gjøre en mal utilgjengelig ved registrering av fagbeskrivelser.

Prebetingelser:

- Representant for studieadministrasjonen må være logget inn.
- Malen må eksistere i systemet.

Postbetingelser:

- Malen kan ikke lenger brukes for registrering av fagbeskrivelser.

Beskrivelse (’happy-day scenario’):

1. Velger mal for fagbeskrivelser.
2. Systemet returnerer og viser informasjon om valgt mal.
3. Bruker setter malen til utgått.
4. Systemet ber om bekreftelse.
5. Bruker bekrefter.
6. Systemet lagrer endringer.

2.2.2.9 Use Case: Godkjenn fagbeskrivelse (administrasjonsmodul)

Primæraktør: Avdelingsstyret

Interessenter og deres interesser:

- Avdelingsstyret ønsker å godkjenne gyldige fagbeskrivelser for å gjøre disse tilgjengelig for eksport og Web.

Prebetingelser:

- Representant for avdelingsstyret er innlogget.

Postbetingelser:

- Fagbeskrivelse er nå tilgjengelig for eksport og visning på Web.

Beskrivelse (’happy-day scenario’):

1. Velger mal for fagbeskrivelser.
2. Systemet returnerer og viser registrerte fagbeskrivelser for valgt mal.
3. Velger fagbeskrivelse.
4. Systemet returnerer og viser informasjon om valgt fag.
5. Godkjenner valgt fag.
6. Systemet lagrer endringer.

2.2.2.10 Aktiver mal (administrasjonsmodul)

Primæraktør: Studieadministrasjon

Interessenter og deres interesser:

- Studieadministrasjonen ønsker å la fagansvarlige ta i bruk malen.

Prebetingelser:

- Representant for studieadministrasjonen er innlogget.

Postbetingelser:

- Malen er tilgjengelig for bruk ved registrering ved fagbeskrivelser.

Beskrivelse (’happy-day scenario’):

1. Velger ikke-aktiv mal.
2. Systemet viser mal.
3. Bruker klikker ”aktivér”.
4. Systemet lagrer endringer.

2.2.2.11 Søk etter fagbeskrivelse (presentasjonsmodul)

Primæraktør: Web-bruker

Interessenter og deres interesser:

- Web-brukeren ønsker å enkelt finne fram til en gitt fagbeskrivelse vha søkeord.

Beskrivelse (’happy-day scenario’):

1. Bruker skriver inn søkekriterier.
2. Systemet søker i fagbeskrivelser.
3. Systemet viser liste over søkeresultater.

2.2.3 Supplementære krav

Her vises krav som ikke kommer klart fram ved hjelp av overstående Use Cases.

2.2.3.1 Rettighetsstyring

Systemet skal ha en administrator som har adgang til hele systemet med alle rettigheter. Administratoren har mulighet for å holde orden på adgangskontroll for systemet via administrasjonsmodulen. Brukere kan tildeles brukernavn, passord og rettigheter av administrator. Men dette vil hovedsakelig bli utført av systemet selv ved at det kobler seg opp mot en e-postserver og sjekker brukerens brukernavn og passord. Dette gjelder kun for fagansvarlige. Alle privilegerte brukere av systemet som ønsker å gjøre eventuelle endringer i systemet eller databasen må alltid være innlogget. Presentasjonsmodulen har ingen påloggingsmulighet og dermed er brukere av modulen ikkeprivilegerte brukere uten rettigheter utover søking og uthenting av informasjon fra databasen.

2.2.3.2 Brukervennlighet

Bruk av riktige farger, kontraster og skrifter er viktig for å få et behagelig skjermbilde. Dette gjøres ved å bruke GUI-standarder som brukere allerede er kjent med. Dessuten er plassering av komponenter viktig for å få et brukervennlig GUI. Brukere av presentasjonsmodulen skal kunne forvente akseptabel responstid ved fremhenting av fagbeskrivelser. Generering av webside og visning av side skal maksimum ta 10 sekunder, dvs. siden skal ha lite tung grafikk. Registreringsmodulen, administrasjonsmodulen og eksporteringsmodulen skal ha et oversiktlig utseende og en konsekvent ordbruk for å unngå å forvirre brukeren.

2.2.3.3 Implementasjonsrammer

Databaseserveren skal være en Microsoft SQL Server, da dette er et krav fra arbeidsgiver. HiG bruker allerede denne serveren til andre applikasjoner og det ville være naturlig å

kunne integrere systemet på den eksisterende serveren, eller i det minste gjøre systemet i stand til å kommunisere med andre servere. HiG har også et ønske om at systemet blir mest mulig plattformuavhengig.

2.2.3.4 Sikkerhet

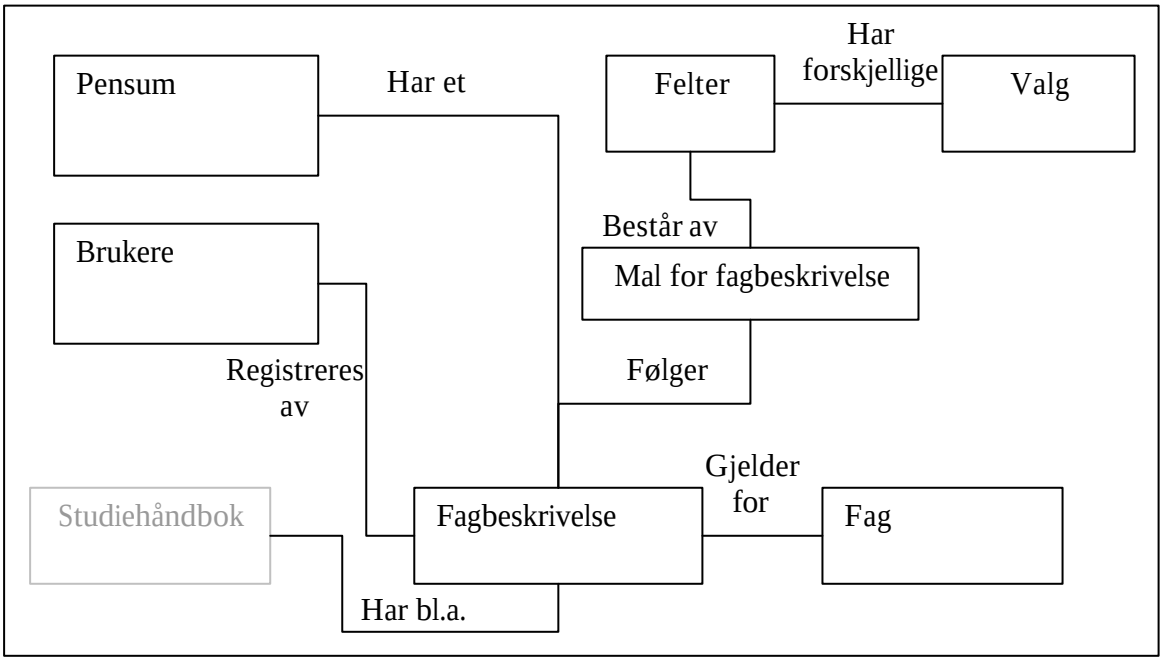
Deler av dette systemet skal være Web-basert, og kan derfor være utsatt for elektroniske innbrudd. Det er derfor viktig at man tenker på sikkerhet når man skal sette opp fysiske servere, og også programvare på disse.

De passord som lagres i databasen skal ”hashes” slik at det ikke er mulig å dekryptere passord. Det vil bli benyttet en MD5 hash til dette. Dette gjøres for å unngå at man kan gå direkte inn databasen og lese passord.

2.3 Datamodell og beskrivelse

2.3.1 Objektmodell

Denne modellen viser hvordan de forskjellige objektene i systemet henger sammen. Dette er for å gi en bedre forståelse av hva som vil være essensielt å lagre i databasen. Studiehåndbok vises med grå farge for å poengtere at denne ligger utenfor hva våres system skal kunne håndtere.



Figur 2-2 - en oversikt over de forskjellige konseptuelle objektene

2.3.1.1 Dataobjekter

Et dataobjekt er noe det kan være relevant å lagre data om i systemet. Disse er vist i sammenheng over. Under dette kapitlet blir hvert objekt nøyere beskrevet.

2.3.1.1.1 Studiehåndbok

Dette er en samling av de forskjellige fagbeskrivelsene for et gitt år, sammen med en del annen informasjon utenfor denne oppgavens rammer.

2.3.1.1.2 Fagbeskrivelse

En fagbeskrivelse kan sees på som en kontrakt mellom skolen og studentene, der fagbeskrivelsen detaljerer innholdet i et gitt fag eller kurs som skolen tilbyr. Denne kan bl.a. inneholde fagets mål og emner, angi læremidler og annen vesentlig informasjon.

2.3.1.1.3 Mal for fagbeskrivelse

Dette er en samling av forskjellige felter som til sammen utgjør en mal. Hver mal har et navn for å kunne skille de fra hverandre.

2.3.1.1.4 Felter

Et felt kan være en av flere forskjellige typer. Systemet behandler disse typene forskjellig avhengig av hvilken type det er. Som det fremgår av modellen så kan et felt ha forskjellige valg. Dette er avhengig av hva slags type felt det er, ettersom noen felttyper ikke har valg i det hele tatt.

2.3.1.1.5 Valg

Et valg er enten en tekst eller et tall. Et valg skal kunne oversettes til engelsk for at man da automatisk kan oversette det gjeldende valget. Det skal også kunne være avhengig av et annet valg slik at et valg kan påvirke dine andre valg videre.

2.3.1.1.6 Brukere

Dette er brukerne av systemet. I første rekke må det lagres brukernavn og passord slik at de kan identifiseres.

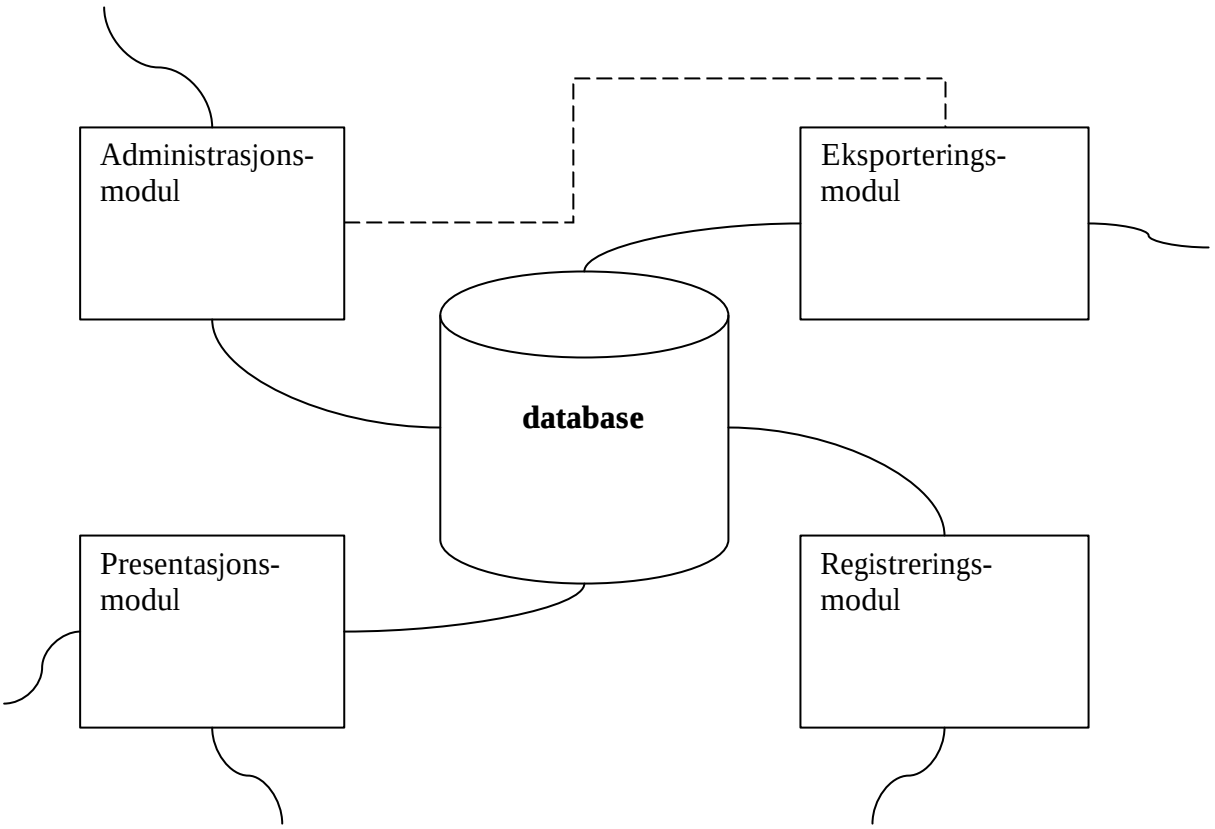
2.3.1.1.7 Pensum

Hvert fag kan ha et eller flere læremidler knyttet til seg. Disse vil bli lagret som et eget dataobjekt da de til tider kan være svært sammensatte (tittel, forfatter, forlag, etc).

2.4 Funksjonell struktur

På dette punktet kommer den implementasjonsmessige spesifikasjonen av systemet. Systemets moduler beskrives, og funksjonaliteten og egenskapene i hver modul detaljeres.

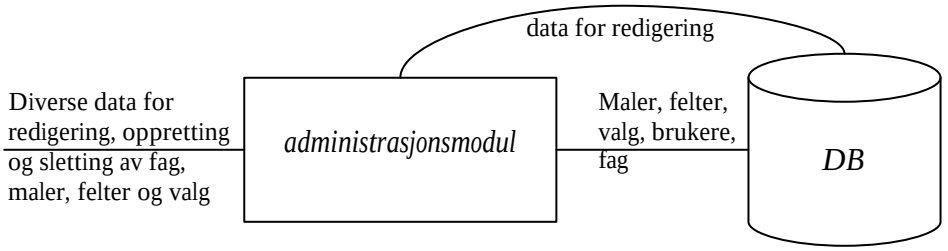
2.4.1 Overordnet modell over modulene



Figur 2-3 - modell over modulene og dataflyter

Her vises de forskjellige modulene og hvilke veier de forskjellige dataene går. Den prikkede linjen antyder at eksporteringsmodulen og administrasjonsmodulen henger svært tett sammen. De hele linjene er dataflyter og her ser vi for eksempel at presentasjonsmodulen kun henter data ut og skriver ingenting inn til databasen.

2.4.1.1 Administrasjonsmodul



Figur 2-4 - data I/O for administrasjonsmodulen

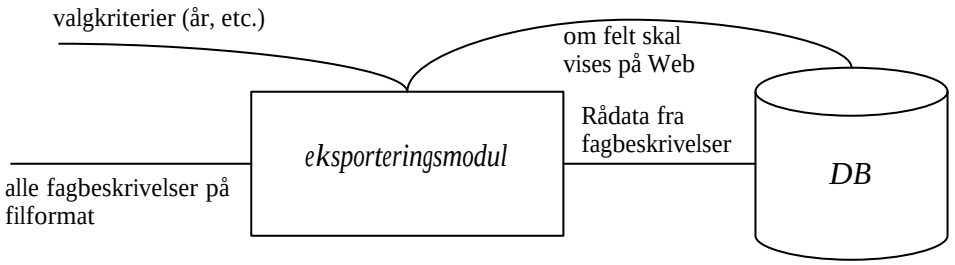
Administrasjonsmodulen vil ha mye funksjonalitet. Disse er listet nedenfor med en kort beskrivelse og forklaring.

- **Ny mal:** det skal være mulig å lage nye maler for registrering av fagbeskrivelser. En ny mal skal kunne bygges opp fra bunnen av eller bruke en annen mal som grunnlag. Hvis den bruker en annen mal som grunnlag skal den kopiere alle dens felter og de valg som hører til i disse feltene.
- **Redigering av mal:** i tilfeller hvor malen for fagbeskrivelser skal oppdateres med innlegging av nye felter, sletting av gamle felter og redigering av felter, skal administrasjonsmodulen muliggjøre dette. Man kan ikke redigere på en mal hvis

den er arkivert eller aktivert. Det er et unntak hvis den er aktivert og det er hvis man bare ønsker å legge til valg i et felt. Da skal dette være mulig å utføre.

- **Brukerhåndtering:** Administrasjonsmodulen skal ha funksjon hvor en administrator av systemet skal kunne opprette og slette brukere, samt styre tilgangskontroll og rettighetsstyring (angi om bruker er fagansvarlig og/eller administrator).
- **Godkjenning av fagbeskrivelser:** etter at fagbeskrivelsene er blitt registrert, må de godkjennes av avdelingsstyret. Systemet skal da låse fagbeskrivelsen, slik at fagansvarlig ikke lenger har mulighet for å redigere denne.
- **Kontrollering av fagbeskrivelser:** der fagansvarlig har mulighet til å skrive inn fritekst skal administrasjonen ha en funksjon som muliggjør for oversiktlig og enkel kontroll av disse fritekstene.
- **Administrering av fag:** skal gjøres av administrasjonen ved at de kan opprette og redigere på fag. Her kan de også gi fag gyldig fagkode hvis de ikke har blitt tildelt dette enda.
- **Arkivering av mal:** en mal skal kunne arkiveres for å unngå at det blir registrert flere fagbeskrivelser med denne malen. Når en mal er arkivert så er det ikke mulig å gjøre om på arkiveringen.
- **Aktivering av mal:** før en mal kan brukes til registrering av fagbeskrivelser så må den aktiveres. Etter at en mal er aktivert kan den ikke lenger redigeres. Eneste unntak fra dette er hvis man ønsker å supplementer valgene i en liste.
- **Sletting av mal:** sletting av mal er noe som sjelden forekommer. En mal kan bare slettes hvis den er redigerbar. Det vil si at den verken er aktivert eller arkivert.

2.4.1.3 Eksporteringsmodul



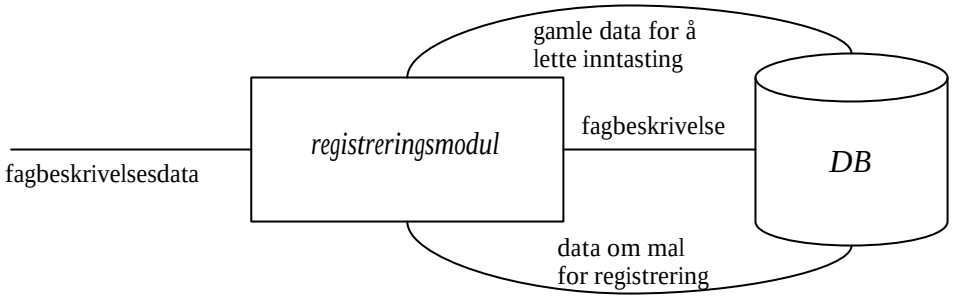
Figur 2-5 - data I/O for eksporteringsmodulen

Eksporteringsmodulen vil i hovedsak stå som en uavhengig modul i systemet, men aktivering av selve modulen vil foregå i administrasjonsmodulen og derfor vil disse modulene arbeide tett sammen. Eksporteringsmodulen vil bli en forholdsvis liten, men viktig modul bestående av:

- **Generering til fil:** eksporteringsmodulen skal generere fagbeskrivelser på filformat, slik at modulen henter ut relevante data fra databasen og setter disse sammen på en slik måte at resultatet blir en ferdigformatert fil. Bruker kan velge om hvert fag skal komme på en fil, eller alle i en.
- **Filformat:** modulen skal gi mulighet til å velge mellom RTF og PDF filformat for lagring.

- **Opsjoner:** modulen skal muliggjøre et vell av valgmuligheter. Bruker skal kunne velge hvilke som helst fagbeskrivelser og felter fra hvilket år det skulle være.
- **Publisering på Web:** det skal være mulig å velge hvilke felter i en mal som skal vises på Web for presentasjonsmodulen.

2.4.1.3 Registreringsmodul

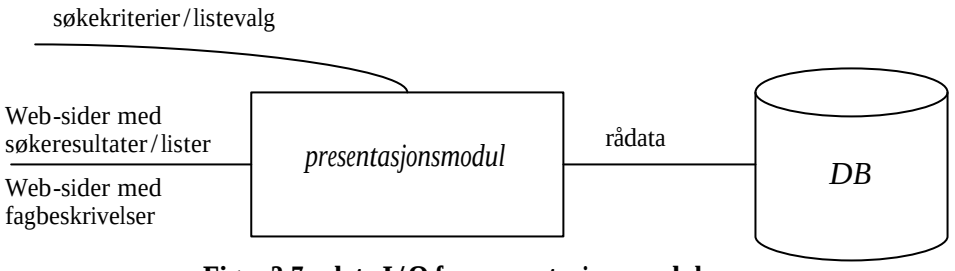


Figur 2-6 - data I / O for registreringsmodulen

Registreringsmodulen vil bestå av følgende funksjonalitet:

- **Registrering av nye fag:** dersom et nytt fag skal opprettes, eller et allerede eksisterende fag skal endre fagkode skal det være mulig å utføre dette før og under registreringen av fagbeskrivelsen.
- **Registrering av fagbeskrivelser:** all informasjon som inngår i fagbeskrivelsen til et bestemt fag skal kunne registreres og lagres ved enten å velge valg fra lister eller ved å skrive inn teksten selv. Mulighetene for å gjøre feil skal begrenses så mye som mulig ved å tvinge fagansvarlig til å velge fra predefinerte valg, og innen visse grenser.
- **Mal:** fagbeskrivelser skal følge en forhåndbestemt mal. Det vil si at modulen må hente ut denne informasjonen fra databasen og generere registreringsbildet ut fra hvilke typer felter den finner i malen.
- **Ferdigutfylling:** en fagbeskrivelse skal fylles ut automatisk så langt det er mulig med informasjon som allerede er lagret i databasen for en eldre versjon av fagbeskrivelsen. Felter som ikke kan fylles ut vil bli stående tomme klare for utfylling.
- **Sjekk av obligatoriske felter:** ved lagring skal systemet sjekke og gi beskjed om alle obligatoriske felter er utfylt. Hvis ikke alle er utfylt skal det gi beskjed om hvilke felter det gjelder. Selv om ikke alle obligatoriske felter er utfylt lagrer systemet de andre utfylte feltene.
- **Forhåndsvisning:** når en fagansvarlig har registrert en fagbeskrivelse skal det være mulig å forhåndsvise denne på Web, siden dette gir en bedre oversikt over de valg og tekster som er registrert. Dette gjøres ved at systemet åpner et nettleservindu som viser fagbeskrivelsen ved hjelp av presentasjonsmodulen.

2.4.1.4 Presentasjonsmodul



Figur 2-7 - data I/O for presentasjonsmodulen

Presentasjonsmodulen vil bestå av følgende funksjonalitet:

- **Vis fagbeskrivelse:** fagbeskrivelser skal kunne hentes ut fra databasen og det genereres en Web-side hvor denne informasjonen vises på en fornuftig og oversiktlig måte.
- **Søk:** modulen har en søkefunksjon hvor brukeren kan søke etter fagbeskrivelser, vha søkekriterier. Det skal være mulig å søke i både gamle og nye (dvs. utgåtte) fagbeskrivelser.
- **Bygger på/forutsetter:** brukeren skal kunne ved å klikke på en link, bla seg videre til det faget som et aktuelt fag bygger på eller forutsetter.
- **Kompatibilitet for gamle maler:** all presentasjon på skjerm skal være i henhold til fagbeskrivelsens mal det året fagbeskrivelsen ble registrert og ikke den nyeste malen.
- **Web-basert:** hele denne modulen skal være Web-basert, slik at brukere kan ha tilgang til fagbeskrivelser uavhengig av geografisk lokasjon.
- **Sikkerhet:** interaksjonen mellom modulen og databasen er kun enveis, nemlig å hente data **ut** av databasen. Av sikkerhetsmessige grunner vil modulen være totalt separert fra øvrige moduler.

2.5 Software- og hardwarebegrensninger

2.5.1 Software designbegrensninger

2.5.1.1 Standarder og språk

Angående programmeringen skal det settes opp en liste over rutiner og konvensjoner vi skal holde oss til, eksempelvis variabelnavn, SQL-setninger, osv. Dette bedrer leselighet og det gjør det lettere å sette seg inn i koden for andre enn forfatteren selv.

HiG ønsker også at systemet skal være mest mulig plattformuavhengig. Derfor vil administrasjons-, registrerings-, og eksporteringsmodulene utvikles i programmeringsspråket Java. Dette vil sikre lik funksjonalitet på en rekke forskjellige plattformer.

Presentasjonsmodulen genererer HTML-dokumenter ved forespørsel. Utviklingen av denne modulen skal foregå med serversideskriptspråket PHP. I de resulterende HTML-dokumenter skal HTML kun brukes til å definere dokumentets struktur, og CSS skal brukes for grafisk utforming. All HTML-kode som genereres, skal være i henhold til W3Cs[1] XHTML 1.0 Strict[2] standard, og all CSS skal være i henhold til W3Cs CSS2-standard[3].

[1] http://www.w3c.org
[2] http://www.w3.org/TR/xhtml1/
[3] http://www.w3.org/TR/REC-CSS2/

Dette for å sikre best mulig kompatibilitet med alle nettlesere – gamle, nåværende og kommende.

2.5.1.2 Softwaregrensesnitt

I dette systemet vil modulene i liten grad operere mot hverandre. De vil i hovedsak operere selvstendig mot databasen, med unntak av eksporteringsmodulen. Denne modulen vil implementeres slik at den kun kan kjøres fra administrasjonsmodulen, men den vil arbeide selvstendig mot databasen slik som de andre modulene.

2.5.1.3 Softwarepakker / verktøy

Oppdragsgiver ønsker at systemets database kjøres på en Microsoft SQL Server 2000 for å sikre kompatibilitet med andre eksisterende databaser som skolen bruker i administrasjonsøyemed. Siden kunden ønsker et Java-basert system, kommer vi til å bruke Borland JBuilder[4] som programmeringsverktøy. For utvikling av prestenasjonsmodulen, kommer vi til å bruke Macromedia Homesite[5].

2.5.1.4 Operativsystem

Siden systemet skal være plattformuavhengig stilles det ingen spesielle krav til valg av operativsystem for å kunne kjøre applikasjonen, forutsatt at det er en Java VM av nyere dato tilgjengelig.

2.5.2 Hardwarebegrensninger

Systemet trenger en eller flere sentrale servere, alt avhengig av hvordan kunden ønsker å sette opp systemet. Ideelt trenger systemet en Web-server (Apache eller Microsoft IIS) som bør ha en 300MHz CPU eller raskere og 128MB RAM, og en databaseserver (Microsoft SQL Server 2000) som bør ha en 1GHz eller raskere CPU og minimum 512MB RAM. Sistnevnte må kjøre et moderne Windows NT-basert operativsystem (Windows 2000 eller Windows XP). En løsning er å ha både SQL-server og Web-server på samme maskin, men det er mer optimalt å kjøre disse på hver sin server. Det er ingenting i veien for å bruke eksisterende servere for databasen og presentasjonsmodulen, men det må legges merke til at serveren for presentasjonsmodulen må være en HTTP-server med PHP installert. Windows-versjonen av PHP kommer med innebygd Microsoft SQL Server-støtte, mens Linux-versjonen krever at man i tillegg installerer FreeTDS[6].

Web-serveren må ha tilknytning til Internet (ikke bare HiGs intranett), slik at brukere kan få tilgang til fagbeskrivelsene som legges ut via presentasjonsmodulen. SQL-serveren trenger kun å ha tilknytning til skolens intranett, forutsatt at den har kontakt med Web-serveren. Hardwaregrensesnittet vil være standard TCP/IP Ethernet.

Web-brukerens krav til hardware er ytterst små. Denne brukergruppen trenger kun en datamaskin med Internett-tilknytning og ytelse nok til å kjøre en Web-browser. Brukerne av systemmodulene som er utviklet i Java, bør ha 300MHz eller raskere maskiner med 64MB RAM for å sikre et minimum av operasjonell standard.

[4] http://www.borland.com/jbuilder/
[5] http://www.macromedia.com/software/homesite/
[6] http://www.freetds.org/

2.6 Organisering av kvalitetssikring

2.6.1 Dokumentasjon

All kode skal kommenteres underveis, både for å forenkle endringsarbeid og også for å lette andres forståelse av koden. Vi skal benytte oss av et dokumentasjonsverktøy som automatisk kan generere dokumentasjon, som for eksempel JavaDoc[7].

Det skal lages brukerveiledning for modulene, med spesiell vekt på administrasjonsmodulen. Brukerveiledningen skal være klar, konsis og illustrert. Dette er nødvendig siden administrasjonsmodulen kommer til å inneholde en stor del funksjonalitet. Brukerveiledninger vil være å finne som vedlegg til prosjektrapporten.

2.6.2 Konfigurasjonsstyring

Systemer som CVS ble vurdert, men ble funnet unødvendig for prosjektet, siden gruppen er liten, og sammenlignet med store prosjekter i næringslivet blir det her lite kode, og dermed lettere å holde oversikt.

Vi skal i stedet bruke en liste over forandringer i begynnelsen av hver kodefil, kalt ”changelog”, hvor den som har endret noe registrerer hva som ble gjort, hvorfor det ble gjort, i tillegg til datoen.

2.6.3 Modultesting

Det skal skrives test cases for hver av modulene. Et testcase identifiserer diverse feilscenarier, hvor man beskriver hvilke feil brukeren kan gjøre. Vi kan da under utvikling kontinuerlig teste modulene opp mot disse, slik at eventuelle feil brukeren gjør ikke vil få konsekvenser for videre eksekvering av modulen. Vi skal også drive kontinuerlig testing underveis etter hvert som delene i systemet faller på plass.

2.6.4 Utgivelser underveis

Ved ferdigstillelse, men også under utvikling, av hver modul, skal modulen gjøres tilgjengelig for arbeidsgiver, slik at han kan gi tilbakemeldinger om eventuelle krav til endringer. Det vil ikke bli utgitt testversjoner tilgjengelig for alle, da det foreløpig ikke er satt opp en sentral server for systemet. Systemet vil inntil videre kjøre fra en av gruppens arbeids-PC-er.

En fullstendig testrelease av registreringsmodulen skal mot slutten av prosjektet slippes til alle faglærere for å la de teste denne og komme med eventuelle tilbakemeldinger.

[7] http://java.sun.com/j2se/javadoc/

3 Design

3.1 Alternative design

Flere alternative måter å løse oppgaven på ble vurdert og disse er kort forklart i kapitlene under. Her blir også grunner til at den enkelte løsning ikke ble valgt lagt frem.

3.1.1 Web-grensesnitt

Da vi først prøvde å danne oss et bilde av systemet og hvordan dette skulle designes, så vi for oss at hele systemet skulle fungere som en stor dynamisk webside, der lærere, administratorer og ”andre brukere” fikk forskjellige rettigheter, alt etter hva deres rolle i systemet skulle være. Administratorer hadde rettigheter til sider som hadde med administrasjon og tilretteleggelse for å registrere fagbeskrivelser, semesterplaner og annen aktuell data. Faglærere hadde mulighet til å registrere sine fag online, og ”andre brukere”, eksempelvis gamle, nye og kommende studenter skulle kunne manøvrere seg rundt i en elektronisk studiehåndbok. På dette stadiet var planen at vi skulle få laget et komplett system, som kunne håndtere alle delene av studiehåndboka, og generere en fullgod papirutgave av denne. Planen var at man med få tastetrykk ville få produsert en komplett studiehåndbok klar til trykk, samt en ferdig elektronisk versjon på web. Etter kort tid fant vi ut i samråd med veileder og oppdragsgiver at vi først og fremst skulle begrense oppgaven til det de mente var det mest vesentlige med systemet, nemlig registrering av fagbeskrivelser, samt arkivering av disse. Den endelige løsningen inneholder alle de vesentlige punktene fra oppdragsgiver, samt en del annen funksjonalitet.

Web-løsningen gikk vi fort bort fra da vi skjønte hvor dynamisk og komplisert hele systemet kunne bli til slutt. Web-teknologien ville vært for tungvindt å benytte seg av for å få til en fullgod løsning. Ved benyttelse av Web-teknologi ville nesten all prosessering av data og handlinger bli gjort på serveren noe som kan føre til stor belastning i perioder. I tillegg har ikke HTML egenskaper og funksjoner som ville gjort systemet realiserbart. Vi måtte lete etter en løsning der mesteparten av funksjonaliteten lå på klientnivå.

3.1.2 Andre vurderte løsninger

Vi var inne på både å lage en Delphi-applikasjon eller en C#-applikasjon, men vi gikk fort bort ifra dette da det var et ønske at systemet skulle være mest mulig plattformuavhengig.

3.1.3 Den endelige løsningen

Den endelige løsningen for prosjektet ble en blanding av Java og PHP. Dette vil si at vi valgte å dele opp systemet i flere mer eller mindre selvstendige systemer. Slik kunne flere deler av det komplette systemet bli angrepet på en egen selvstendig måte, og valg av implementeringsverktøy ble en egen vurdering opp mot hver enkelt del.

Java er et tilnærmet plattformuavhengig programmeringsspråk, mens PHP har en rekke fordeler vedrørende å raskt få frem resultater på web og det har god databasestøtte. Dette sammen med at alle på gruppa hadde noe erfaring innen Java fra før, ble avgjørende for dette valget.

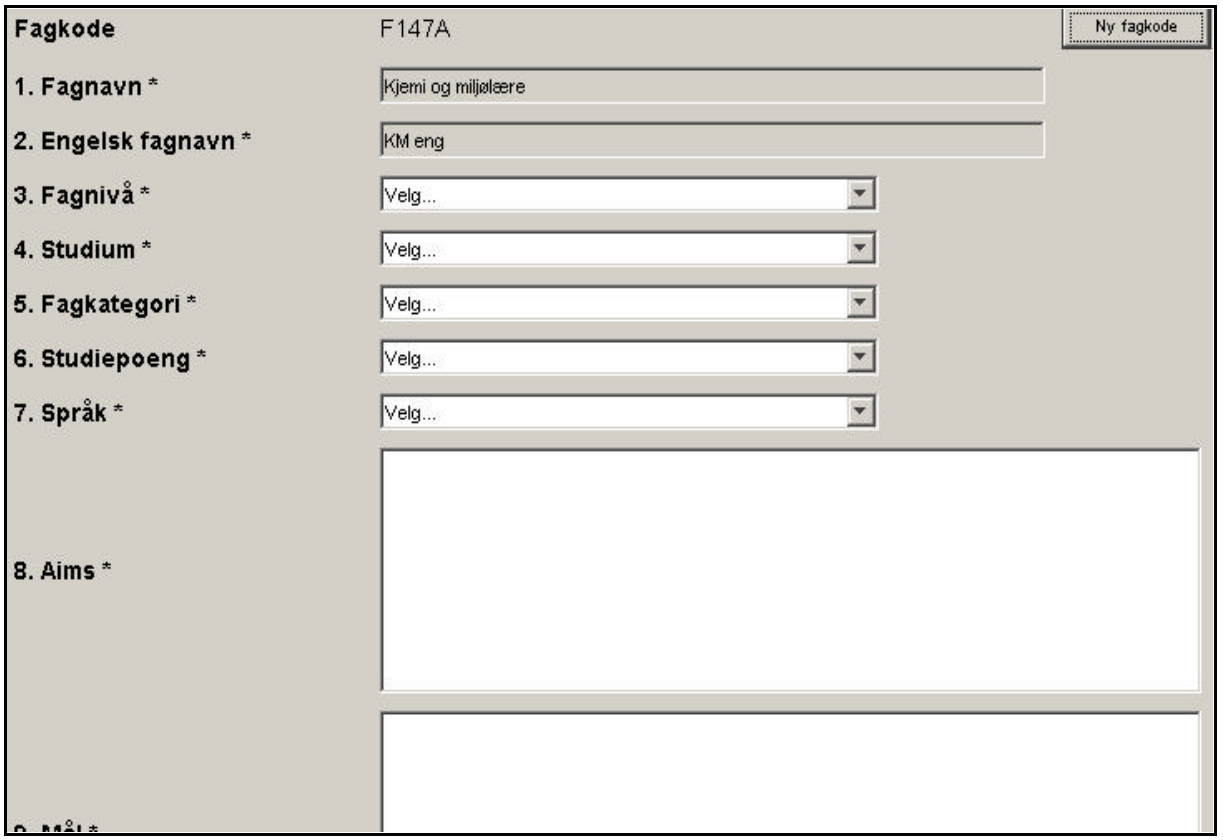
Programmet vi skulle utvikle ble først delt inn i fire moduler, der hver modul hadde hver sin faste brukergruppe. I bunn for alle modulene ligger den samme SQL databasen som alle modulene gjør spørringer og transaksjoner mot.

3.2 Grafisk design og grensesnitt

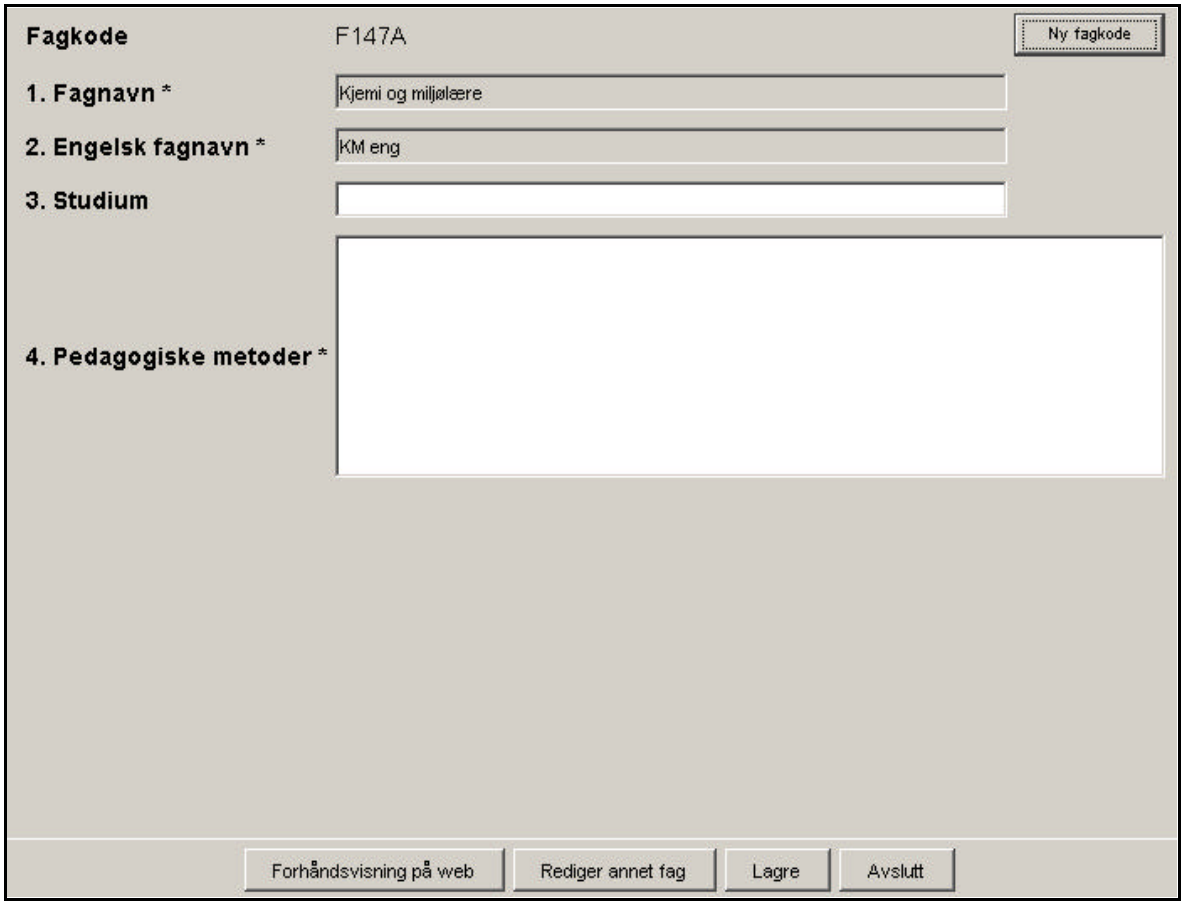
I og med at det er flere selvstendige applikasjoner som er flettet sammen i SAFE-pakken, har vi hatt muligheten til å tilpasse hver enkeltstående del av applikasjonspakken sitt grafiske brukergrensesnitt for å fungere best mulig for sitt bruk. Under vil vi gå inn i mer detalj for hver selvstendige del av systemet, hvordan det er gjort, og hvorfor.

3.2.1 Registreringsmodulen

Denne modulen er den modulen som har vært klart mest utfordrende å få et intuitivt og enkelt brukergrensesnitt på. Dette er fordi det er et grensesnitt som er nesten 100% dynamisk, og kan forandre radikalt utseende fra år til år, alt etter hvordan malen for fagbeskrivelser er utformet. Figur 1 og figur 2 viser to scenarier som eksemplifiserer hvordan registreringsmodulen kan forandre seg utseendemessig fra år til år.



Figur 3-1 – utsnitt GUI av registreringsmodulen for en mal



Figur 3-2 – GUI for hele registreringsmodulen for en annen mal

Som man ser av figurene kan en fagbeskrivelse ha forskjellige felter og et forskjellig antall av disse. Dessuten kan et felt være av forskjellig type fra en mal til en annen selv om feltet heter det samme. Dette begrenser mulighetene for å bruke en layout der man benytter seg av absolutt posisjonering av komponenter, da dette kunne ført til at komponenter falt utenfor skjermen og dessuten kan det føre til total uforutsigbarhet med tanke på hvordan det vil se ut på andre plattformer. Løsningen var å putte alle feltene inn i en tabellarisk layout. Hvert av feltene blir puttet på sin plass i tabellen når programmet starter opp før programvinduet åpnes og blir synlig. Tabellarisk layout kan sammenlignes med en tabell, der man teller opp rader og kolonner etter hvert som man trenger dem. Det vanskelige med denne måten å lage et GUI på er at man aldri helt vet hva man får, og ut over kolonnene og radene er det vanskelig å finjustere plasseringen av komponentene. Størrelse på for eksempel tekstfelt og tekstområde kan også variere fra PC til PC og operativsystem til operativsystem.

Fontene som man normalt finner på etiketter, knapper og lignende i en Java-applikasjon vil variere etter den standarden som finnes på hver enkelt PC som kjører applikasjonen. Dette kan føre til at det enkelte plasser der man forventer å finne tekst ikke finner annet enn noen prikker. Dette fordi det ikke er plass til teksten med det normale oppsettet. Dette problemet har vi løst med å spesifisere en skrifttype og en størrelse på komponentene, og sende med denne i den ferdige programpakken.

Hvert av feltene som kan forekomme i en fagbeskrivelse er i seg selv et eget lite grafisk brukergrensesnitt. Dette for å gi brukeren et mest mulig intuitivt brukergrensesnitt. Hvis feltet består av mer enn en komponent, vil hele feltet markeres med en kantlinje rundt alle

komponentene, for å markere at disse samlet utgjør et felt. Alle felter kan ha tilhørende engelske felter. Disse feltene vil oppdatere og fylle ut seg selv så sant dette er mulig. I praksis vil dette si at alle engelske felter utenom der det skal fylles inn fritekst er sperret for input fra brukeren. Man vil se at valgene i det engelske feltet blir gjort automatisk så fort man utfører en handling på det tilhørende norske feltet. For eksempel hvis man velger ”Valg 1” i et multivalgsfelt, vil automatisk det tilhørende engelske feltet velge ”Choice 1” i sitt felt. Under følger en kort forklaring rundt alle disse feltgrensesnittene:

3.2.1.1 Tekstlinje

Dette feltet er ment til å ta imot små mengder tekst av brukeren. Selv om det er tilnærmet ubegrenset hvor mye tekst dette feltet kan ta imot, er det GUI-messig utformet slik at det gir uttrykk for at det her ikke skal skrives inn for mye data. Visuelt er dette løst med å ha en enkelt linjes tekstboks. Hvis man skriver ut over tekstlinjens bredde, får man bare se et lite utsnitt av hva man skriver. Dette er også med på å fortelle brukeren at ikke mye data skal skrives.

Figur 3-3 – Tekstlinje

3.2.1.2 Tekstområde

Tekstområdefeltet er ment for å ta imot større mengder tekst. Her vil input feltet være større en hva tilfelle er ved en tekstlinje, og hvis man ønsker å skrive mer en tekstområdet visuelt tillater, vil det dukke opp rullefelter som utvider tekstfeltet. Dette feltet er laget slik at det håndterer en type for friformatering. Det vil si at bruker for eksempel kan lage seg lister, innrykk og bruke annen type virkemidler for å formatere teksten i dette feltet. I både tekstlinje og tekstområde feltene er det mulig å klippe og lime inn tekst fra andre steder. Visuelt er dette feltet mye større enn tekstlinjefeltet, dette er ment å gi brukeren en forståelse for at her er det meningen at han kan eller skal fylle ut en del data.

Figur 3-4 – Tekstområde

3.2.1.3 Nedtrekksvalg

Dette feltet er ment til å brukes der bruker kun skal ha muligheten til å velge ett valg blant flere ferdigdefinerte valg. For å få bruker til lett å intuitivt skjønne dette, benytter denne feltkomponenten seg av en rullgardinmeny (engelsk: combobox), som inneholder alle de valgene det er mulig å velge. Rullgardinmenykomponenten er en mye brukt komponent i så godt som alle grafiske applikasjoner, så intuitivt vil nok de aller fleste fort skjønne hvordan dette feltet skal brukes.

Figur 3-5 – Nedtrekksfeltet

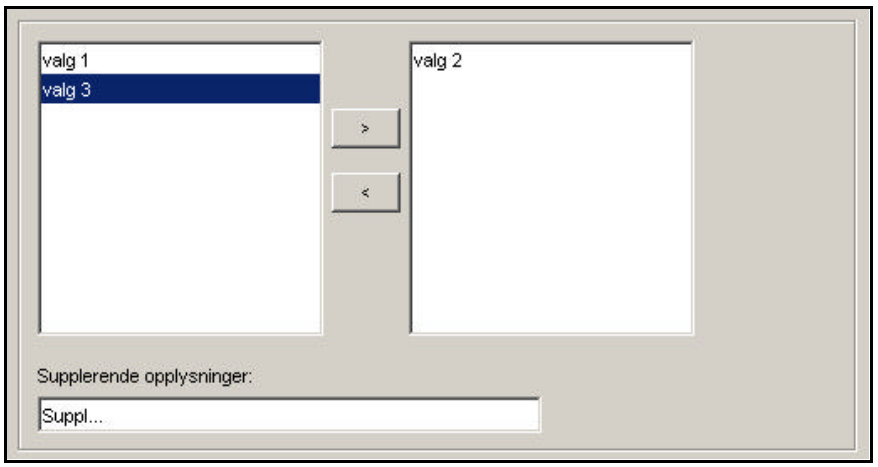
3.2.1.4 Multivalg

Multivalg er et felt tiltenkt for de situasjoner der det er ønskelig å velge flere valg blant allerede ferdigdefinerte felter. Multivalgfeltet består av to lister plassert ved siden av hverandre. Mellom listene er det plassert to knapper med piler på. Disse pilene peker mot hver sin liste. I listen til venstre finner man alle de valgene det er mulig å velge, og i listen til høyre er de valgene som er valgt for fagbeskrivelsen. For å velge valg markerer man de valg med å klikke med musen på disse. Holder man CTRL-tasten inne er det mulig å markere flere valg samtidig. Man trykker deretter på en av knappene for å sende valgene frem og tilbake mellom listene. Visuelt er dette bygget opp for å gi en rask og intuitiv forståelse av hva som kan gjøres i dette feltet. Alle mulige valg vil dukke opp i listen til venstre, dette forteller brukeren at det ikke er valgene i denne boksen som vil bli valgt med i fagbeskrivelsen. Da er det eneste logiske at man kan flytte over valg til listen til høyre. Knappenes strategiske plassering mellom listene (med pilikon på), er ment å fortelle brukeren at ved å trykke på disse kan man sende valg mellom listene. Ved presentasjoner og testing av dette feltet viser det seg at folk ikke har problemer med å skjønne hvordan dette feltet fungerer.

Figur 3-6 – Multivalg

3.2.1.5 Multivalg og tekstlinje

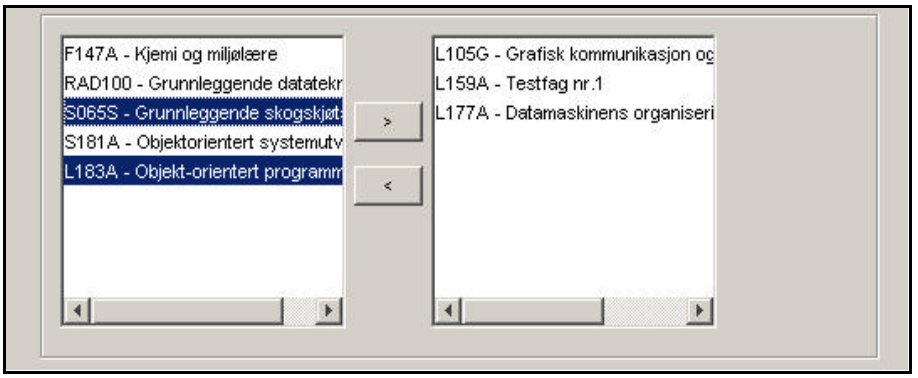
Dette feltet vil fungere på samme måte som multivalgfeltet, men her vil man i tillegg finne en tekstlinje under listene. Dette feltet har ledetekst ”Supplerende opplysninger”, og her er det tenkt at brukeren kan registrere tilleggsopplysninger. For eksempel kan dette være forklaring til valgene eller en tilleggsopplysning det ikke finnes ferdigdefinerte valg på.



Figur 3-7 – Multivalg og tekstlinje

3.2.1.6 Multivalg med fagkoder

Dette feltet oppfører seg på akkurat samme måte som et vanlig multivalgfelt, men her er mulige valg de fag som eksisterer i systemet, og som ikke er registrert som utgått.



Figur 3-8 – Multivalg med fagkoder

3.2.1.7 Multivalg med tallverdi

Dette feltet er utformet som et vanlig multivalgfelt, men i tillegg er det en liten tekstboks med ledeteksten tallverdi. Bak tekstboksen står det er prosenttegn for å indikere at her skal det oppgis en prosentverdi. Dette feltet fungerer slik at man kan knytte en tallverdi opp mot hvert enkelt valg som er valgt. Hvis ikke noen valg er markert i den høyre listen vil tekstboksen være helt grå, men når et valg blir markert der, vil valgets tilhørende tallverdi dukke opp. Visuelt vil dette fortelle brukeren at valgene har en tilhørende tallverdi. Dette vil man fort lære når man markerer forskjellige valg i listen og se at tallverdien endrer seg fortløpende etter hvilket valg som er valgt.

Figur 3-9 – Multivalg med tallverdi

3.2.1.8 Multivalg med fagkoder og tallverdi

Dette feltet fungerer helt på samme måte visuelt som multivalg med tallverdi. I praksis er eneste forskjellen at valgene vil være alle aktive fag registrert i databasen. Figur 3 -7 er i realiteten et skjermbilde av et slikt felt.

3.2.1.9 Multivalg med fagkoder og tekstlinje

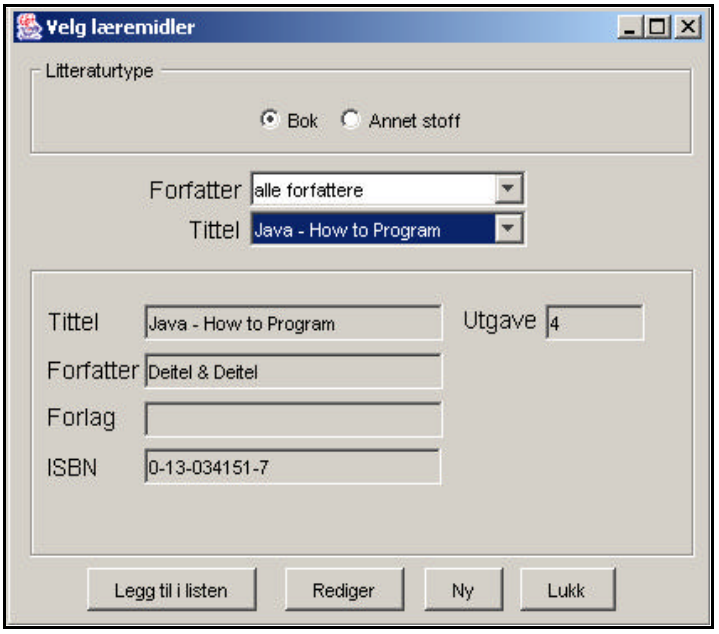
Feltet vil visuelt være likt som Multivalg med tekstlinje. Her kan man velge blant de aktive fagene i databasen, og supplere med litt tekst til slutt hvis dette er ønskelig.

3.2.1.10 Læremidler

Læremidler er et felt tiltenkt de situasjoner der det er ønskelig å registrere pensum eller støttelitteratur for et fag. Dette feltet er bygd opp av et eget, frittstående vindu, og en komponent i hovedvinduet. Komponenten i hovedvinduet består av en liste som viser oversikten over hvilke bøker og eventuelt annet stoff som er valgt (se figur 3-10). Under listen er det to knapper, en merket ”Legg til”, og en merket ”Slett”. Trykker man på slettknappen vil de merkede læremidlene bli fjernet fra listen. Merk at det hele tiden vil være to overskrifter i listen, en for bøker, og en for annet stoff.

Figur 3-10 – Læremidler

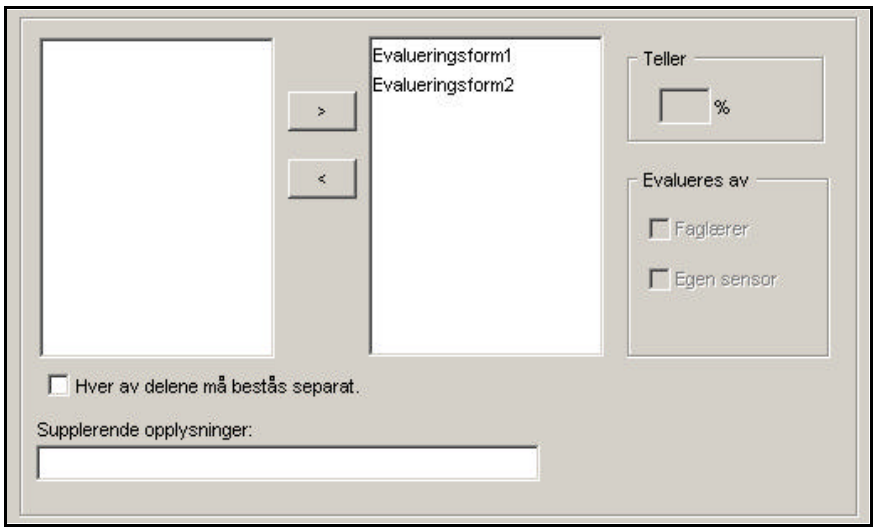
Trykker man på legg til-knappen vil et nytt vindu for å velge litteratur dukke opp (se figur 3-11). Øverst i dette vinduet finner vi to radioknapper merket ”Bok” og ”Annet stoff” samlet av en konteiner med tittelen Litteraturtype. Radioknapper er et veldig mye brukt visuelt verktøy som gjør at bare ett at de tilgjengelige valgalternativer kan velges om gangen. For brukere med litt erfaring omkring grafiske brukergrensesnitt vil de raskt skjønne at her skal man velge enten bok eller annet stoff. Videre under er det to rullgardinmenyer, en merket ”Forfatter” og en merket ”Tittel”. Som standard står det ”alle forfattere” i den øverste rullgardinmenyen. I den andre finner man som standard den første tittelen sortert alfabetisk av den litteraturtypen som er valgt. Disse boksene sitt innhold er altså avhengig av hvilken litteraturtype som er valgt øverst. Videre kan man filtrere innholdet i rullgardinmenyen ”tittel”, ved at man velger en forfatter først. Under disse boksene finner vi all informasjon som er lagret om det elementet som er valgt under ”Tittel”. Nederst er det fire knapper, disse forteller brukeren hvilket valg man har ovenfor valgte element. Det er mulig å legge til og redigere læremidler ved hjelp av knappene nederst. Tekstfeltene som i figuren under er deaktiverte aktiveres da, i tillegg til at det dukker opp to nye knapper, ”Lagre” og ”Avbryt”, slik at brukeren kan skrive inn informasjon om læremidlet. Ved lagring er tittel og forfatter obligatoriske utfyllingsposter. Om bruker skriver inn et ISBN-nummer, må dette være gyldig. ISBN-nummerets ni første siffer sjekkes da opp mot det tiende, som er en sjekksum.



Figur 3-11 - Valg og registrering av læremidler

3.2.1.11 Evalueringsformer

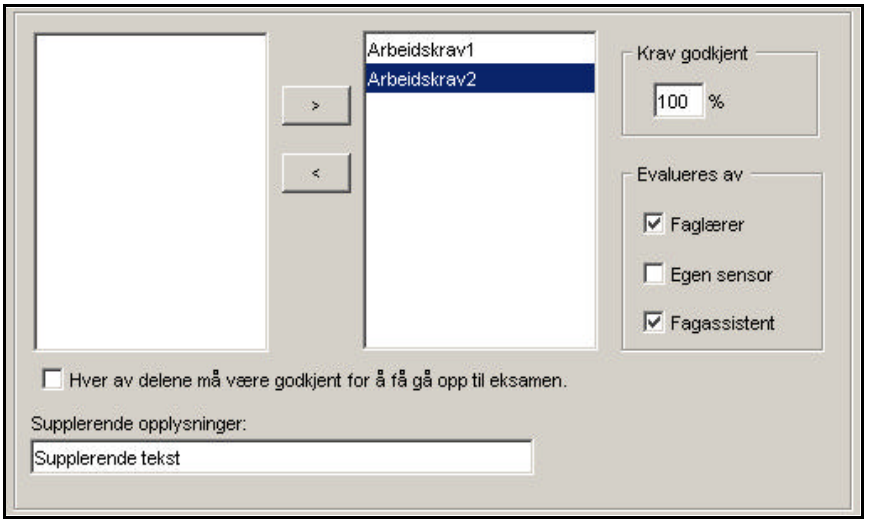
Dette feltet er et spesialfelt tiltenkt utfylling av evalueringsformer for et fag. Dette feltet består hovedsakelig av et "Multivalg med tallverdi"-felt, men i tillegg til dette finnes det her en linje for registrering av supplerende opplysninger, samt to sjekkbokser som definerer hvem som skal evaluere faget, og en sjekkboks som forteller om hver av delene må bestås separat eller ikke for å kunne gå opp til eksamen. Her er det en sjekk på at prosentverdien til sammen for alle de valgene som er valgt må være 100% før valgene kan registreres som godkjent i databasen. Dette fordi en kandidat må bli evaluert med metoder som til sammen utgjør 100% av karakteren. Det som er visuelt nytt i dette feltet er sjekkbokser, men også dette er en velbrukt standard komponent, så brukeren vil komme raskt inn i hvordan dette feltet virker. En annen spesiell ting i dette feltet er at alle ledetekstene foran sjekkboksene er hentet ut fra databasen hver gang modulen kjøres. I praksis vil dette si at disse kan endres hvis dette er ønskelig senere.



Figur 3-12 – Evalueringsformer

3.2.1.12 Obligatoriske arbeidskrav

Dette feltet er visuelt nesten identisk utformet som Evalueringsformer, men har en ekstra sjekkboks som også kan knyttes til et av valgene. Brukermessig fungerer dette feltet helt likt som Evalueringsformer, men ledetekstene er forskjellige, og summen av de valgte verdiene trenger ikke å være 100%.



Figur 3-13 - Obligatoriske arbeidskrav

3.2.2 Administrasjonsmodulen

Administrasjonsmodulen har, i kontrast til registreringsmodulen, i høy grad statisk grafisk brukergrensesnitt. Men også her er det grafiske brukergrensesnittet bygget opp av og bestående av mange mindre deler. Brukeren skal til enhver tid kunne se og ha tilgang til enkeltfunksjonalitet, og kun dette. Siden administrasjonsmodulen er en meget omfattende del av systemet, og inneholder stor grad av funksjonalitet og dermed også grafiske komponenter, er det viktig å kunne dele opp og begrense de komponenter som brukeren har tilgang til, samtidig. Dette for å hjelpe brukeren med hensyn på kompleksitet og for å ikke skape forvirring. For å kunne ivareta dette har vi basert det grafiske brukergrensesnittet i administrasjonsmodulen på et filmenysystem som vist på figur under.

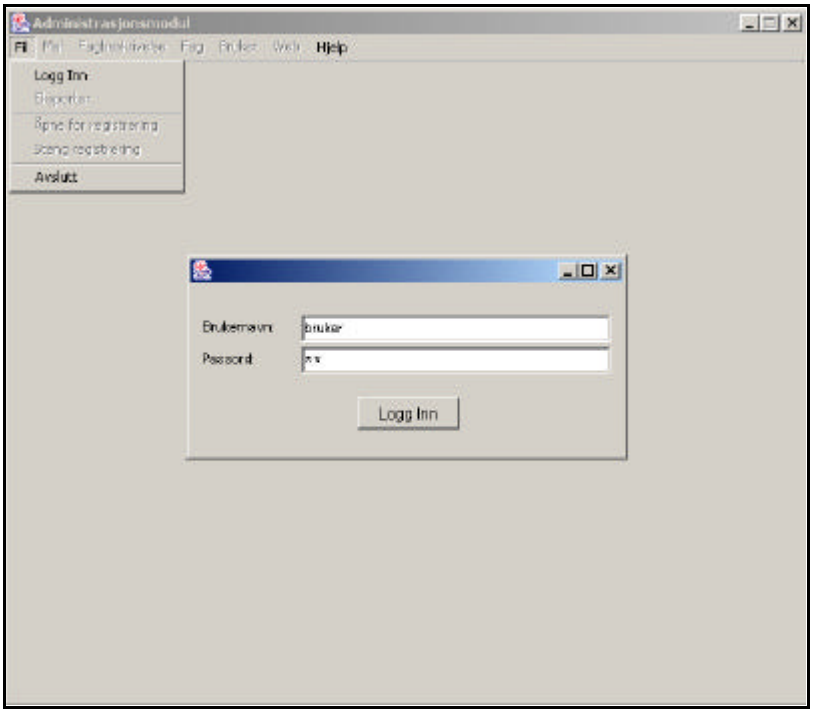
Figur 3-14 - Eksempel på filmeny

Som det fremgår av figuren over er hele administrasjonsmodulen og dennes grafiske komponenter med tilhørende funksjonalitet oppdelt på en logisk og intuitiv måte, slik at

brukeren skal kunne oppleve interaksjonen med systemet, mer eller mindre lett og forståelig. Dette bidrar samtidig til at brukeren lettere vil lære å bruke administrasjonsmodulen da komponenter vises litt og litt om gangen ved behov, i stedet for et utseende hvor alle grafiske komponenter er synlige på en gang.

For øvrig er hovedmenyen inndelt i noen få valg, som igjen har egne valg. Under vises det bilder fra administrasjonsmenyen som illustrerer denne inndelingen og viser på en klarere måte hva en slik inndeling av modulen innebærer for brukeren og hans/hennes opplevelse av modulen.

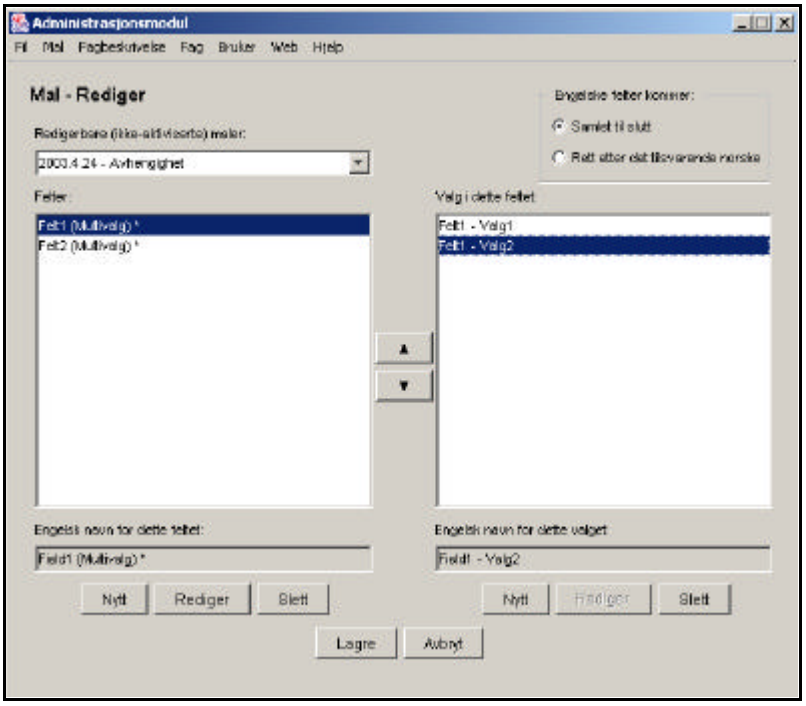
Ved oppstart er store deler av hovedmenyen inaktiv for å fremtvinge pålogging. Brukeren kan da kun logge inn, avslutte eller se ”om administrasjonsmodul”-vinduet under ”Hjelp”-menyen.



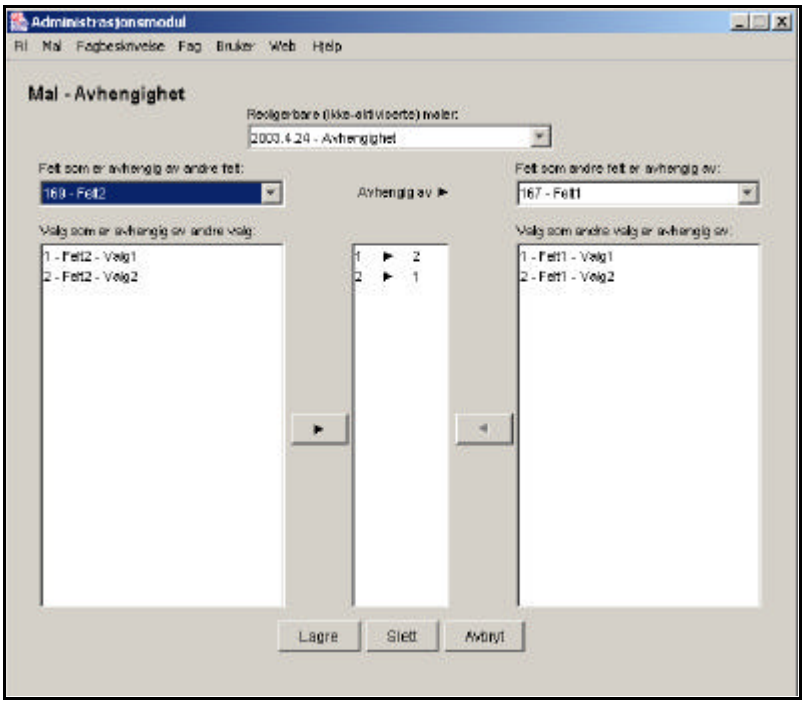
Figur 3-15 - Eksempel på begrensninger og framtvinging av handlinger

Det er under utvikling av administrasjonsmodulen og det grafiske brukergrensesnittet for denne, satt fokus på å benytte seg av de standarder og normer som finnes og ellers brukes i programutviklingsmiljøet. Et eksempel på dette er plassering av eksport i hovedmenyen. Det er naturlig å ha dette valget under fil, sammen med blant annet logg inn og avslutt. Brukeren vil anse denne plasseringen som naturlig, da denne plasseringen er i henhold til andre systemer og applikasjoner som brukeren allerede er kjent med, eller har hatt erfaringer med fra før.

All funksjonalitet som har sammenheng med en mal, som for eksempel oppretting, redigering, aktivering og oppsett av avhengighet er plassert under mal. Nedenfor vises noen eksempler på hvordan disse kan variere i både utseende og funksjonalitet, men samtidig ha en likhet i form av språkbruk, oppsett og generell inntrykk. Dette illustrerer viktigheten av at brukeren kjenner igjen hovedvinduet og har lett for å orientere seg innenfor denne.



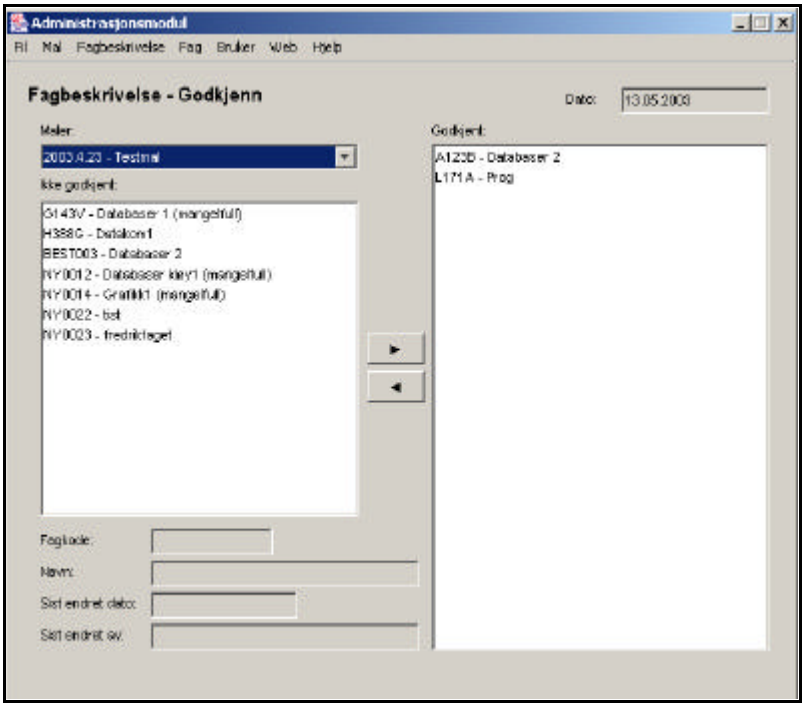
Figur 3-16 - Redigering av maler



Figur 3-17 - Oppsett av avhengighet

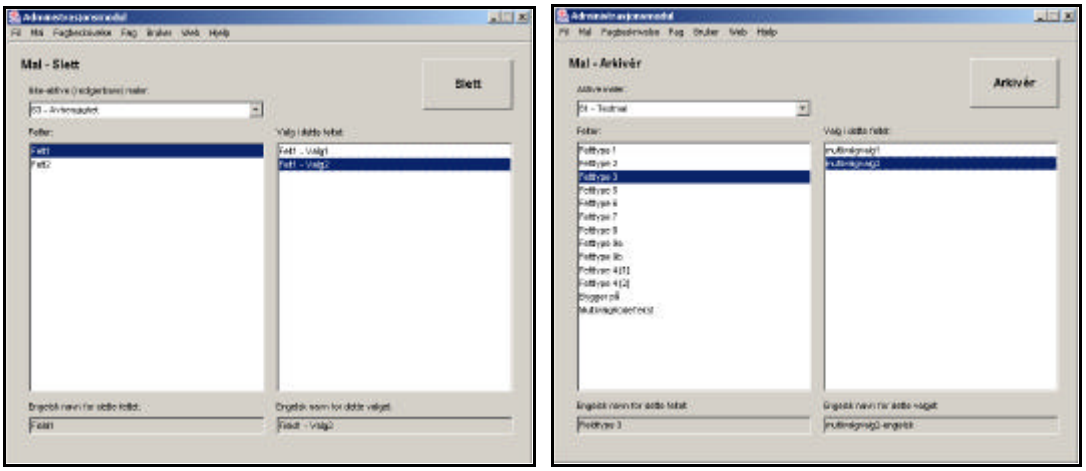
Brukeren skal kunne sette opp maler, administrere brukere og fag, og styre hele SAFE-systemet fra administrasjonssystemet. I tillegg skal brukeren kunne se, lese og korrigere fagbeskrivelser som er registrert. For å kunne lette arbeidet med dette for brukeren er det satt inn funksjonalitet og finesser som foreksempel visning av fagbeskrivelser på Web, utskrift av fritekst tatt ut fra registrerte fagbeskrivelser, mulighet for redigering og retting av disse og visning av mangler i fagbeskrivelser. Det grafiske brukergrensesnittet er satt opp på en slik måte at brukeren kan lettere se eller oppdage nødvendig informasjon. Nedenfor

er det vist et eksempel på dette, hvor de mangelfulle fagbeskrivelsene er klart merket og stjeler litt av oppmerksomheten hos brukeren.



Figur 3-18 - Mangelfulle fagbeskrivelser er klart merket

Det er under hele utviklingsarbeidet satt fokus på å gjøre det grafiske brukergrensesnittet under administrasjonsmodulen, da med tanke på de forskjellige vinduene eller utseende i hovedvinduet, mest mulig lik, både med hensyn på plassering av komponenter og språkbruk. Det er vist eksempler nedenfor og dette demonstrerer og understreker brukerens behov for å kjenne igjen applikasjonen. Det vil være enklere for brukeren å ta i bruk et slikt oppsett, og ikke minst raskere utvikle en forståelse av hvilke komponenter som brukes til hva, og hvilke funksjonalitet disse tilbyr.



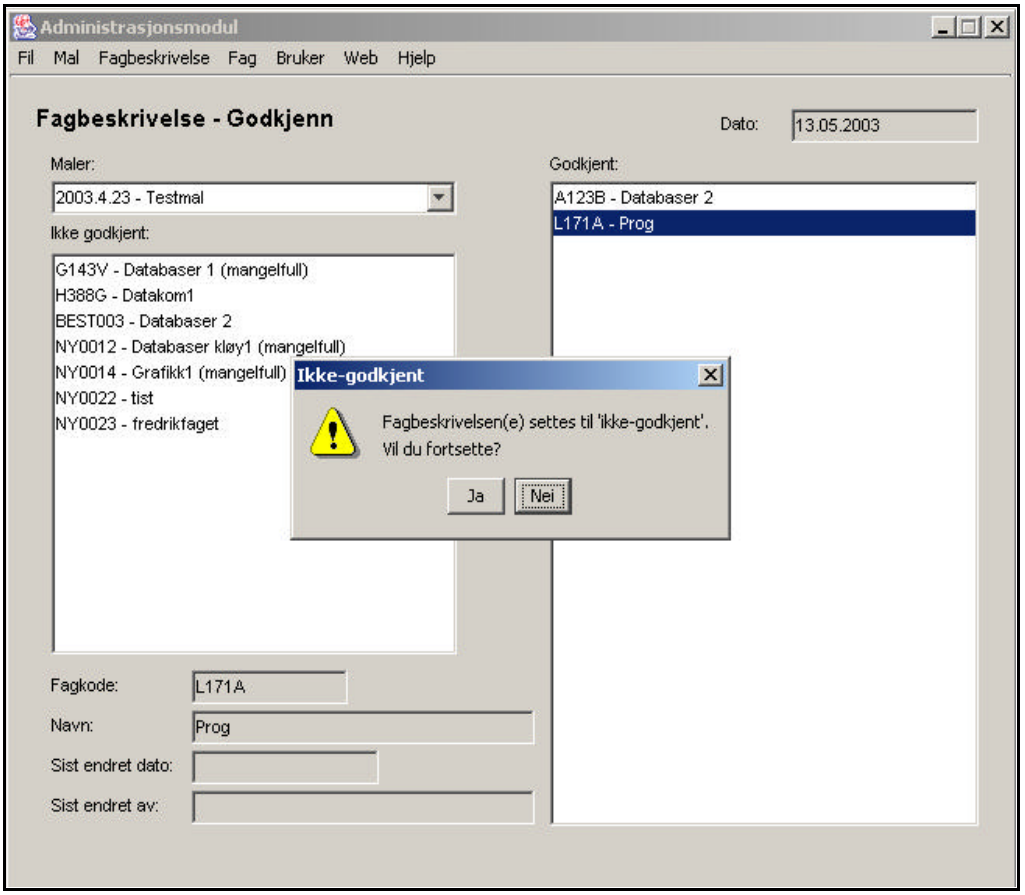
Figur 3-19 – Arkivering og sletting av mal. Illustrerer likhet ved skjermbilder.

Det satt stor fokus på å ha en brukergrensesnitt som reduserer mulighetene for feil som kan oppstå på grunn av feil bruk av grafiske komponenter, og som i tillegg krever bevisste handlinger der disse kan føre til endringer, oppsett som ikke kan reverseres og/eller som

har innflytelse på andre brukere, som for eksempel endringer i mal. Som det er illustrert nedenfor, vil brukeren alltid få beskjed dersom hun/han er i ferd med å utføre en slik operasjon og det er da naturlig at som hoved- eller førstevalg skal denne operasjonen angres eller stoppes.

Figur 3-20 - Feilmelding med standardvalget "Nei"

Det er viktig at de forskjellige komponenter har entydige bruksområder og at språkbruken er bevisst og overveid. For eksempel skal språket i feilmeldinger være i harmoni med andre meldinger og administrasjonsmodulen for øvrig.



Figur 3-21 - Likhet i språkbruk fra vindu til feilmeldinger

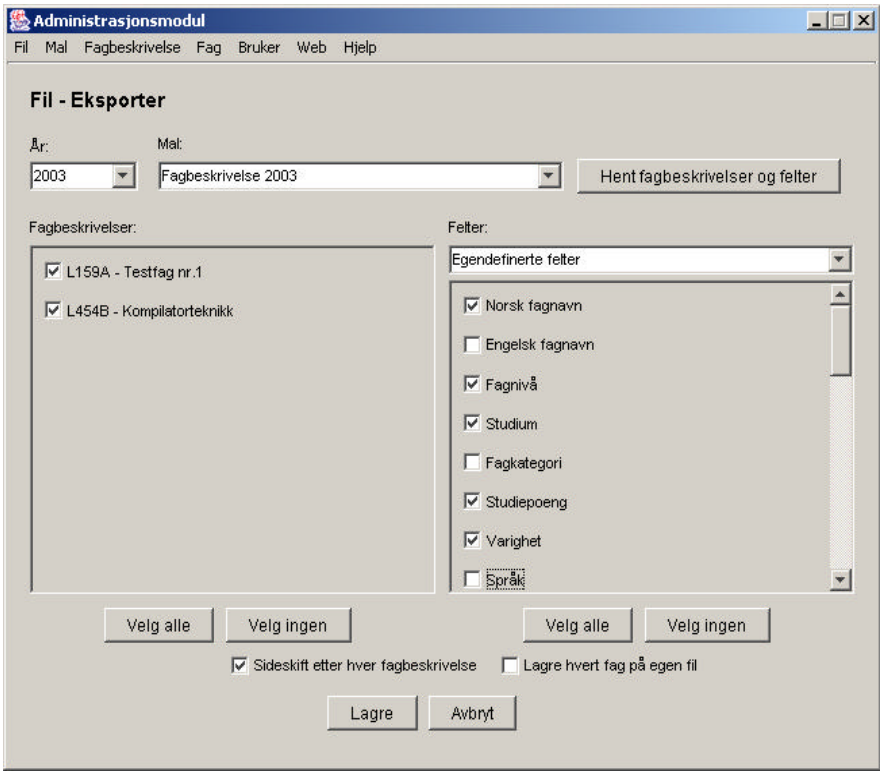
I startfasen av prosjektet, ivrige etter å endelig sette i gang programmeringen og selve utviklingsarbeidet etter opparbeiding og redigering av en lang kravspesifikasjon med de tilhørende dokumenter, ble det satt ytterst lite fokus på det grafiske brukergrensesnittet. Det var i stor grad funksjonalitet som dominerte utviklingslisten. Dette viste seg å være katastrofalt for administrasjonsmodulen da brukeren hadde vanskelig med å sette seg inn i systemet. Problematikken rundt det grafiske brukergrensesnittet av administrasjonsmodulen ble for første gang påpekt av faglig veileder. Vi hadde i stor grad

innsett problemet selv, men på grunn av lite erfaring fra tidligere, ble utviklingen fortsatt, hvor grafiske komponenter ble satt inn der det var plass. Takket være veileders innsats og hans evne til å vite hva han ville ha som en potensiell bruker av systemet, ble beslutningen tatt, all kode dumpet og arbeidet med utvikling av administrasjonsmodulen startet på nytt. Denne gang ble hele det grafiske brukergrensesnittet, ca. 95%, utviklet, før noe funksjonalitet ble implementert. Denne metoden viste seg å være meget gunstig og førte til et tilfredsstillende produkt, vel grafisk som funksjonelt. Denne beslutningen ble en viktig faktor for suksess og foruten denne kunne produktet ikke brukes av den vanlige bruker. Dette er et poeng som vi har innsett i ettertid.

3.2.3 Eksporteringsmodulen

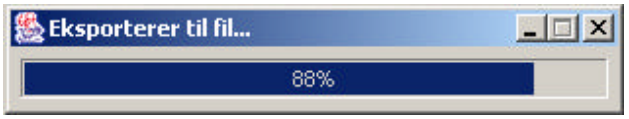
Siden denne modulen er integrert i administrasjonsmodulen, vil den også være veldig lik denne utseendemessig. Ved eksporteringen av fagbeskrivelser vil brukeren være nødt til å utføre en rekke valg. Det er viktig at brukeren, ved utførelsen av disse valgene, blir veiledet gjennom disse på en slik måte at muligheten for å gjennomføre feil blir tilnærmet lik null. Men på den andre siden er dette en modul det går an å prøve å feile litt i. Her er det ingen valg som kan føre til noe vesentlig feil i databasen og fagbeskrivelsene.

Modulen vil være delt i to der den ene delen dreier seg om eksportering av fagbeskrivelser til fil. Her kan man generere PDF og RTF filer basert på data som finnes i databasen uansett år og mal. Den andre delen vil gjøre det mulig å velge hvilke felter i en mal som skal vises på Web til enhver tid. Dette også uavhengig av år og mal. Alt i alt er det ikke så veldig mange valg og mye funksjonalitet i denne modulen, så etter kort tid vil nok brukergrensesnittet bli lett å forstå og bruke.



Figur 3-22 – Eksporteringsmodulen

Da det kan være en tidkrevende prosess å generere mange fagbeskrivelser på filformat, er det nødvendig å opplyse brukeren om hvor langt prosessen er kommet til enhver tid. Dette løses ved å benytte en framgangsindikator som vist på figur 3-23.



Figur 3-23 - Fremgangsindikator ved eksportering av fagbeskrivelser

3.2.4 Presentasjonsmodulen

Denne modulen er den eneste som ikke blir utviklet i Java, men i PHP. Det som er viktig her med tanke på det grafiske brukergrensesnittet er hvordan man kan navigere seg rundt i fagbeskrivelser, og muligheter til å søke etter spesifikke fagbeskrivelser for et gitt år.

Farger og gode kontraster er også to viktige faktorer i denne modulen. Derfor skal det være mulig å endre på disse slik at man til enhver tid kan oppnå det utseendet man ønsker. Det er spesielt fargekombinasjoner på siden, størrelse og type på fonter og den slags det vil være mulig å gjøre endringer på. Se for øvrig vedlegg G for mer detaljer rundt dette.

Vi har utviklet det grafiske grensesnittet slik at det vil være mulig å lett plassere fagbeskrivelsene inn på andre sider som en del av denne. For eksempel kan et fags hjemmeside plassere fagbeskrivelsen fremvist av presentasjonsmodulen i en egen HTML-ramme som en del av hjemmesiden.

Hvis vi ser øverst i venstre hjørnet på en fagbeskrivelse, kan vi se to forskjellige bilder der til enhver tid. Til sammen er det tre forskjellige bilder, et norsk flagg, et engelsk flagg, og en globus. Hvis man klikker på disse bildene vil visning av fagbeskrivelsen filtreres på språk. Klikkes det på engelsk flagg vil engelsk fagnavn og felter vises, klikkes det på det norske flagget vil kun de norske feltene vises, og hvis det klikkes på globusen vil både norske og engelske felter vises. Merk at det kun er de feltene som er satt til å vises på Web fra administrasjonsmodulen som vil dukke opp uansett filtrering.

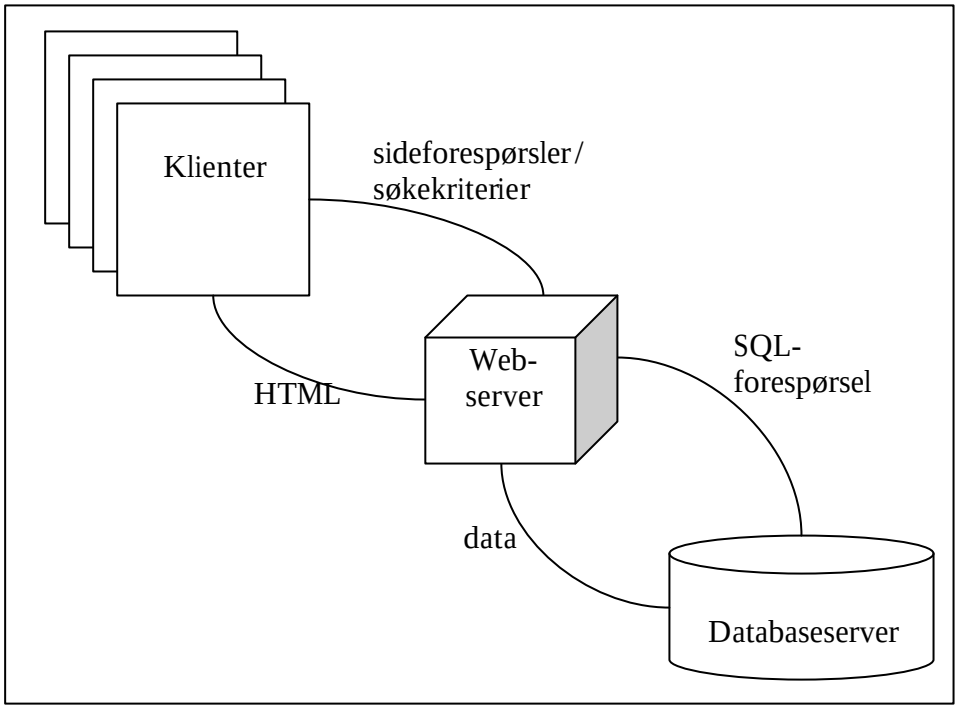
3.3 Teknisk design

Databasen er i SAFE det som binder alle modulene sammen, og det er den som tar vare på alle data som bestemmer hvordan for eksempel en fagbeskrivelse eller en mal skal se ut. I databasen ligger alt om hvilke komponenter som hører til SAFE, altså hvilke forskjellige valg og felter kan velges til en fagbeskrivelse. Dette måtte det tas hensyn til i utarbeidelsen av det tekniske designet av systemet.

3.3.1 Systemarkitektur

I systemet har vi to radikalt forskjellige klient-/serversituasjoner. Vi har presentasjonsmodulen som består av en tynn klient, en Web-server og databaseserveren. Med tynn klient menes at et minimum av prosessering ligger på klientmaskinen, mens det meste av dataprosessering ligger på en eller flere sentrale servere. I de tre Java-implementerte modulene finner vi motsatt situasjon hvor vi har det meste av funksjonaliteten og dataprosesseringen på klientene, mens serveren kun fungerer som lagringsmedium.

3.3.1.1 Presentasjonsmodulens systemarkitektur

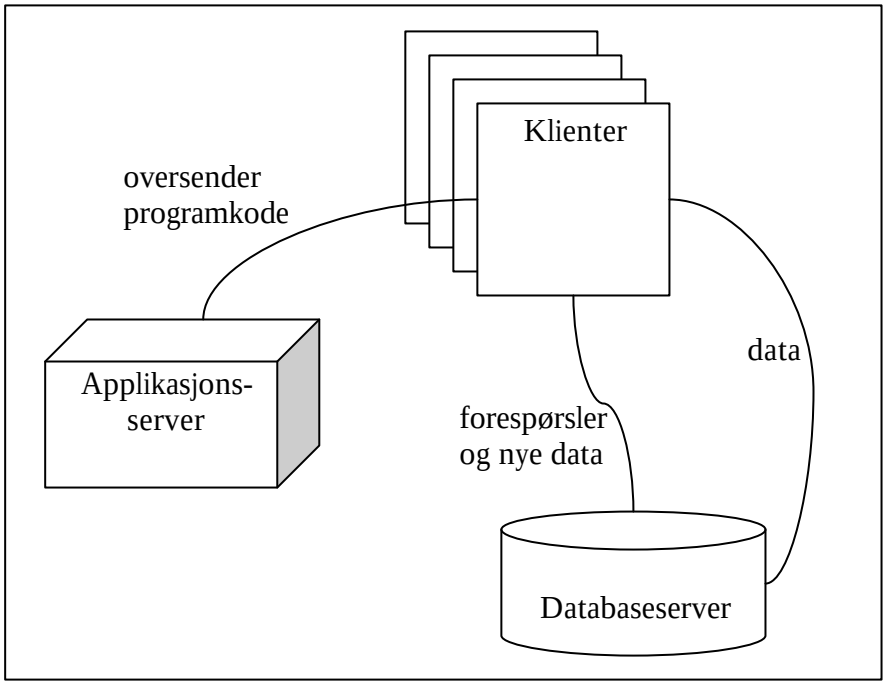


Figur 3-24 - Systemarkitektur for presentasjonsmodulen

Vi kan her identifisere en trelags systemarkitektur med presentasjonslag (klientene), applikasjonslag hvor all dataprosessering foregår (Web-server) og datalaget hvor data blir lagret (databaseserveren).

Fra figuren ser man hvordan klientene kan gjøre forespørsler mot Web-serveren, som igjen gjør forespørsler mot databasen for å hente inn nødvendige data for sammensetting av siden klienten forespurte. Siden sendes deretter til klienten som en standard HTML-side.

3.3.1.2 De Java-baserte modulenes systemarkitektur



Figur 3-25 - systemarkitektur for de Java-baserte modulene

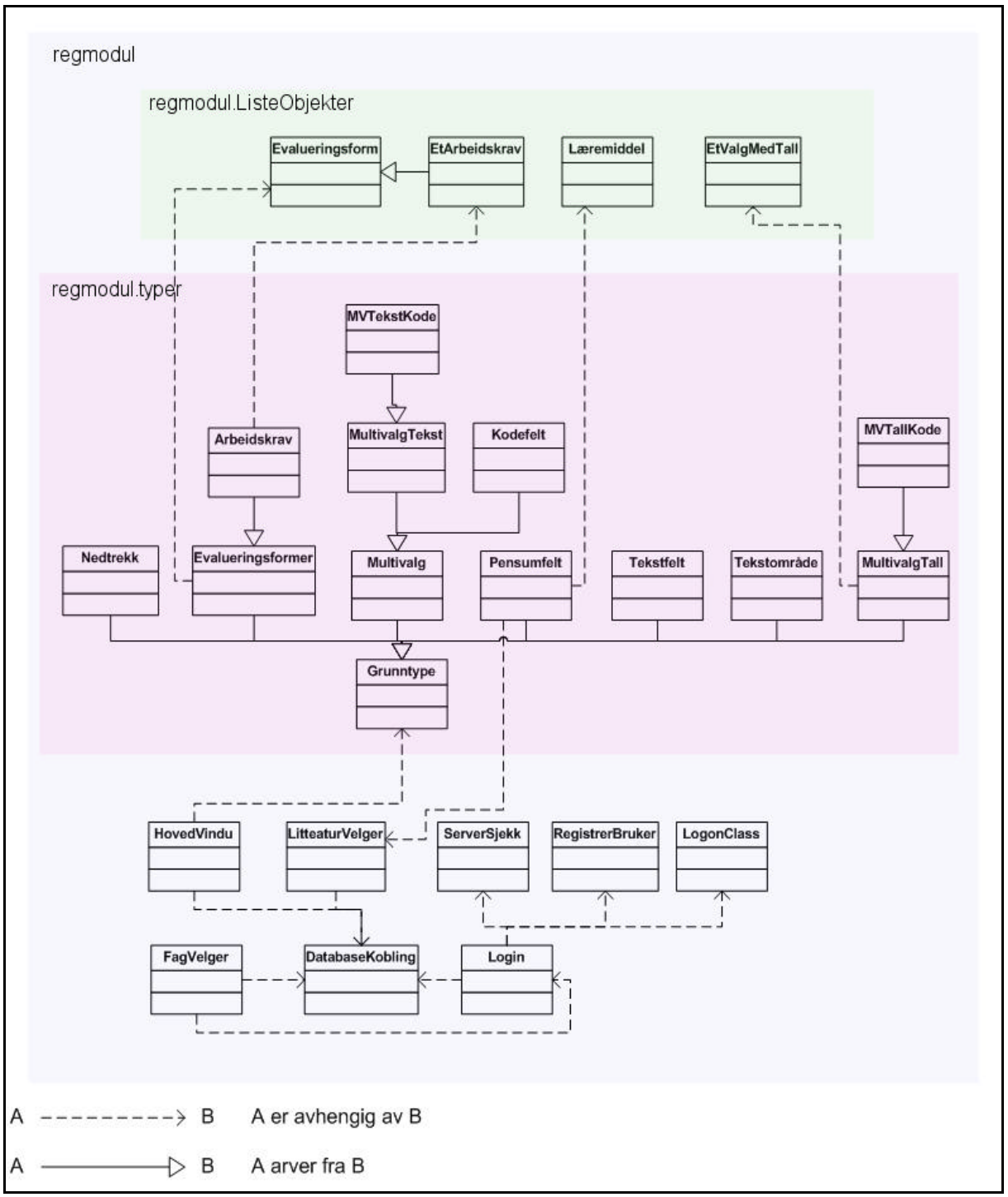
Denne arkitekturen kan minne mye om en trelags arkitektur, men applikasjonsserveren har her en spesiell funksjon. Dens funksjon er å oversende programkoden til klientmaskinens minne, slik at det fra klientens synspunkt virker som om applikasjonen kjører lokalt på deres maskin. Applikasjonen fungerer som både presentasjonslag og applikasjonslag, altså alt som har med brukergrensesnitt og dataprosessering skjer på klienten, det vil si at klienten gjør forespørsler direkte mot databasen, og at man ikke trenger en mellomstående server. Dette kalles en tykk klient, i motsetning til presentasjonsmodulens tynne klient.

Siden applikasjonen startes fra en applikasjonsserver vil den være meget enkel å oppgradere. Man kan da kun oppgradere applikasjonen på serveren, og brukerne vil benytte seg av nyeste versjon neste gang de starter applikasjonen.

3.3.2 Registreringsmodulen

Registreringsmodulen må håndtere alle typer felter som kan bli definert i administrasjonsmodulen. Da hvilken type felter man lager i administrasjonsmodulen bare blir lagret som en tekststreng i databasen, er det registreringsmodulen som må lage hele komponenten rundt et felt. Det vil si at hver type felt må håndteres på sin spesielle måte. De skal holde på forskjellig mengde og type data, og lagre resultatene slik at alle data blir korrekt hentet frem igjen neste gang fagbeskrivelsen blir åpnet. Dette løser vi med å opprette en softwareklasse for hvert felt i malen. De forskjellige typer felter i en mal blir til softwaremessige klasser spesielt designet for å håndtere akkurat sin type felt. Mer teknisk forklart så blir det i registreringsmodulen opprettet en komponentvektor som holder på alle feltene i en mal, som klasser av sin respektive type. Videre er det funksjoner i hver klasse som håndterer feltene sine endringer i denne komponentvektoren. På denne måten er registreringsmodulen meget modulær, og det vil det bli lett å implementere nye klasser senere i systemet hvis dette trengs. Hver klasse arver visse funksjoner fra en grunnklasse. Videre arver klassene av hverandre alt ettersom hvor like de er. Figur 3-26 viser et klassediagram av registreringsmodulen. Figuren er tegnet med standard UML-notasjon. De

forskjellige fargene antyder hvilke pakker klassene hører til, solid pil indikerer arv, og stiplet pil indikerer avhengighet mellom klasser. Eksempelvis så kan man si at klassen ”EtArbeidskrav” arver fra ”Evalueringsform”, dvs. at ”Evalueringsform” er ”EtArbeidskrav” sin superklasse. Videre kan man se ut i fra klassediagrammet at ”Pensumfelt” er avhengig av ”Læremiddel” og ”LitteraturVelger”.



Figur 3-26 – Klassediagram for registreringsmodulen

Klassen ”HovedVindu” er den klassen som tegner det grafiske brukergrensesnittet for registreringsmodulen. Så dette kan vel kalles hovedklassen i systemet, men siden den ikke inneholder veldig mye funksjonalitet, og at ikke all input og output går igjennom her vil den

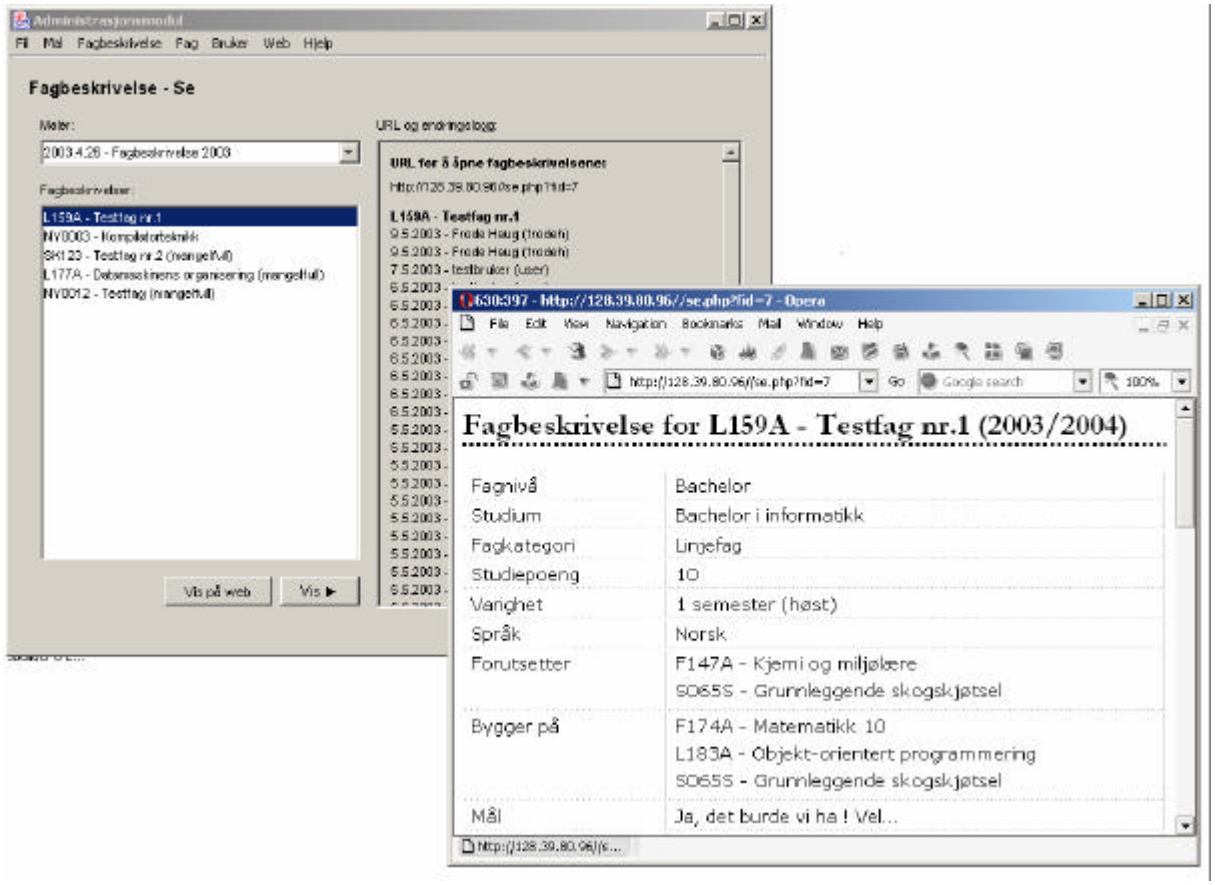
ikke opptre som noen flaskehals for data. Et av hovedpoengene med å lage systemet så objektorientert som vi har gjort har vært å fordele rollene utover de forskjellige klasser.

3.3.3 Administrasjonsmodulen

Administrasjonsmodulen er tiltenkt personer med en administrativ rolle, som for eksempel studieadministrasjonen og avdelingsstyret. Under denne modulen blir all oppsett, kontroll og vedlikehold av SAFE systemet ivaretatt.

Brukere av denne modulen har mulighet til, og ansvaret for å opprette maler for registrering av fagbeskrivelser. Dette er maler som vises i registreringsmodulen og som registreringsmodulen senere genereres ut i fra. Ved redigering av maler kan brukeren opprette diverse typer felter for den aktuelle malen. Eksempler på slike typer kan være enkle tekstlinjer, tekstfelter, nedtrekksmenyer og valglister. I tillegg til å angi hvilke valgmuligheter som skal vises i valgbare komponenter, som for eksempel lister, kan brukeren definere relasjoner og avhengighet mellom felter og valg, for å lette registreringsarbeidet for brukere av registreringsmodulen, men også for å bedre kunne styre disse for å minske og begrense feil som kan oppstå på grunn av feil bruk eller feil interaksjon med registreringsmodulen. I Administrasjonsmodulen kan maler slettes, arkiveres eller aktiveres slik at registreringsmodulen får tilgang til disse. I tillegg til dette kan hele registreringsmodulen åpnes eller stenges.

Foruten å administrere maler for systemet, kan brukere av administrasjonsmodulen se, redigere og godkjenne fagbeskrivelser. Brukeren har under ”Se fagbeskrivelser” mulighet til å se fagbeskrivelser på web og endringslogg for disse. Nedenfor vises en illustrasjon på hvordan dette kan se ut. Brukere av administrasjonsmodulen kan kontroller og redigere all fritekst som registreres under registreringsmodulen. Her har brukeren også mulighet til å skrive ut tekster med tilhørende informasjon på ark, forutsatt at datamaskinen har tilgang til en standardskriver. Under ”Mangelfulle” kan brukeren få vist hvilke fagbeskrivelser som har mangler og hvilke mangler de enkelte har. Godkjenning av fagbeskrivelser skjer på en enkel og oversiktlig måte, ved å flytte fagbeskrivelser fra liste til liste. Her er det satt fokus på at mangelfulle fagbeskrivelser og nye fagbeskrivelser uten gyldig kode ikke skal kunne godkjennes.

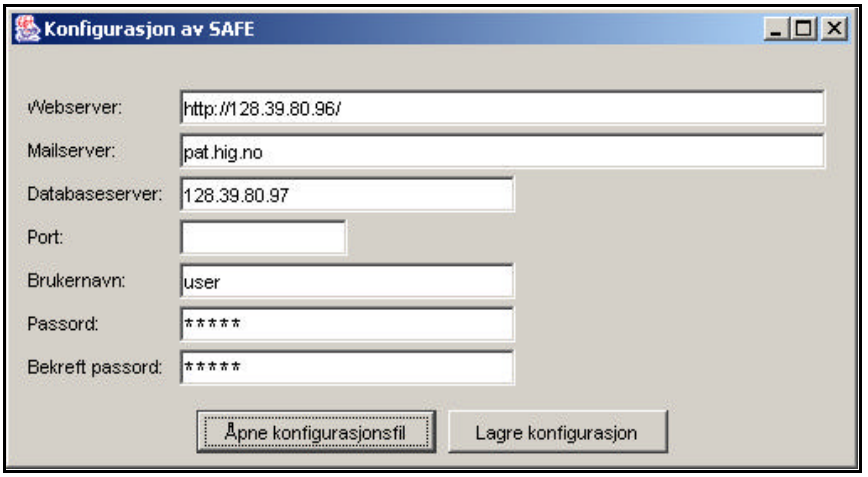


Figur 3-27 - Interaksjon mellom administrasjonsmodulen og webleseren

Administrasjonsmodulen tilbyr mulighet og funksjonalitet for administrasjon av brukere, fag og presentasjonsmodulens maler. Man kan her foreta alle de nødvendige operasjoner som oppretting, redigering og sletting av både brukere og fag. Brukeren kan sette opp maler for hva, blant annet hvilke felter, som skal vises på web under presentasjonsmodulen.

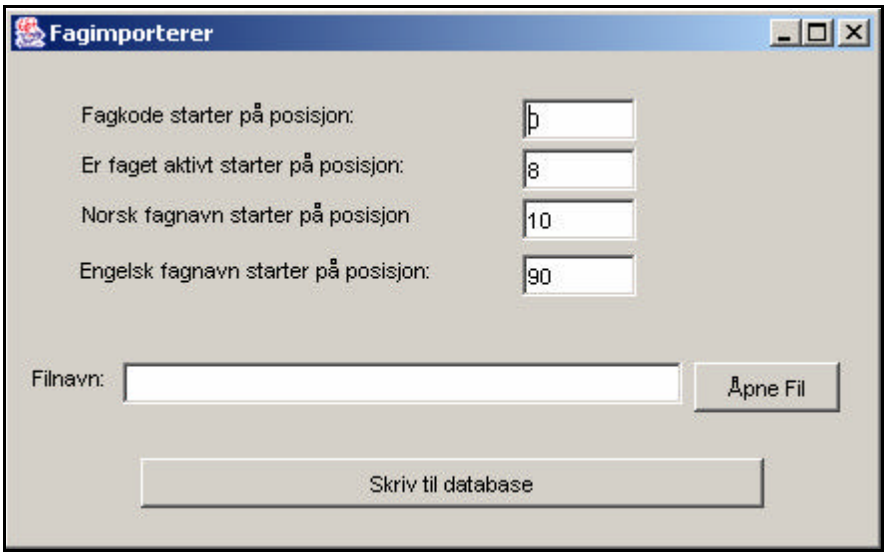
Eksportering av data, herunder fagbeskrivelser er en annen funksjonalitet som det er tatt høyde for under administrasjonsmodulen. Brukeren har mulighet til å eksportere fagbeskrivelser til en eller flere filer med PDF eller RTF format. PDF filer er ikke redigerbare, men har til gjengjeld visningsprogramvare laget av Adobe som er overlegen i forhold til vanlige teksteditorer, når det gjelder visning og funksjonalitet som er relatert til dette. RTF filer er derimot redigerbare og har egne små finesser som for eksempel autogenerert innholdsfortegnelse.

Administrasjonsmodulen består i tillegg til hovedapplikasjonen av to tilleggsdeler eller applikasjoner. Den ene applikasjonen, tilleggsapplikasjon 1, er kalt for ”SAFE Config” og inneholder funksjonalitet for å opprette, redigere og administrere konfigurasjonsfilen som brukes av administrasjonsmodulen, registreringsmodulen, presentasjonsmodulen og eksporteringsmodulen, samt av tilleggsapplikasjon 2 under administrasjonsmodulen. Konfigurasjonsfilen til SAFE inneholder informasjon og data om webserver og dennes adresse, meilserver og dennes adresse, databaseserver og dennes navn eller IP-adresse samt port, brukernavn og passord for tilgang til databasen, hvor passordet ikke lagres i klartekst av sikkerhetsmessige årsak. Nedenfor vises en illustrasjon av det grafiske brukergrensesnittet til tilleggsapplikasjon 1. Her kan brukeren åpne en allerede eksisterende konfigurasjonsfil og gjøre endringer i denne, i tillegg til å opprette nye konfigurasjonsfiler.



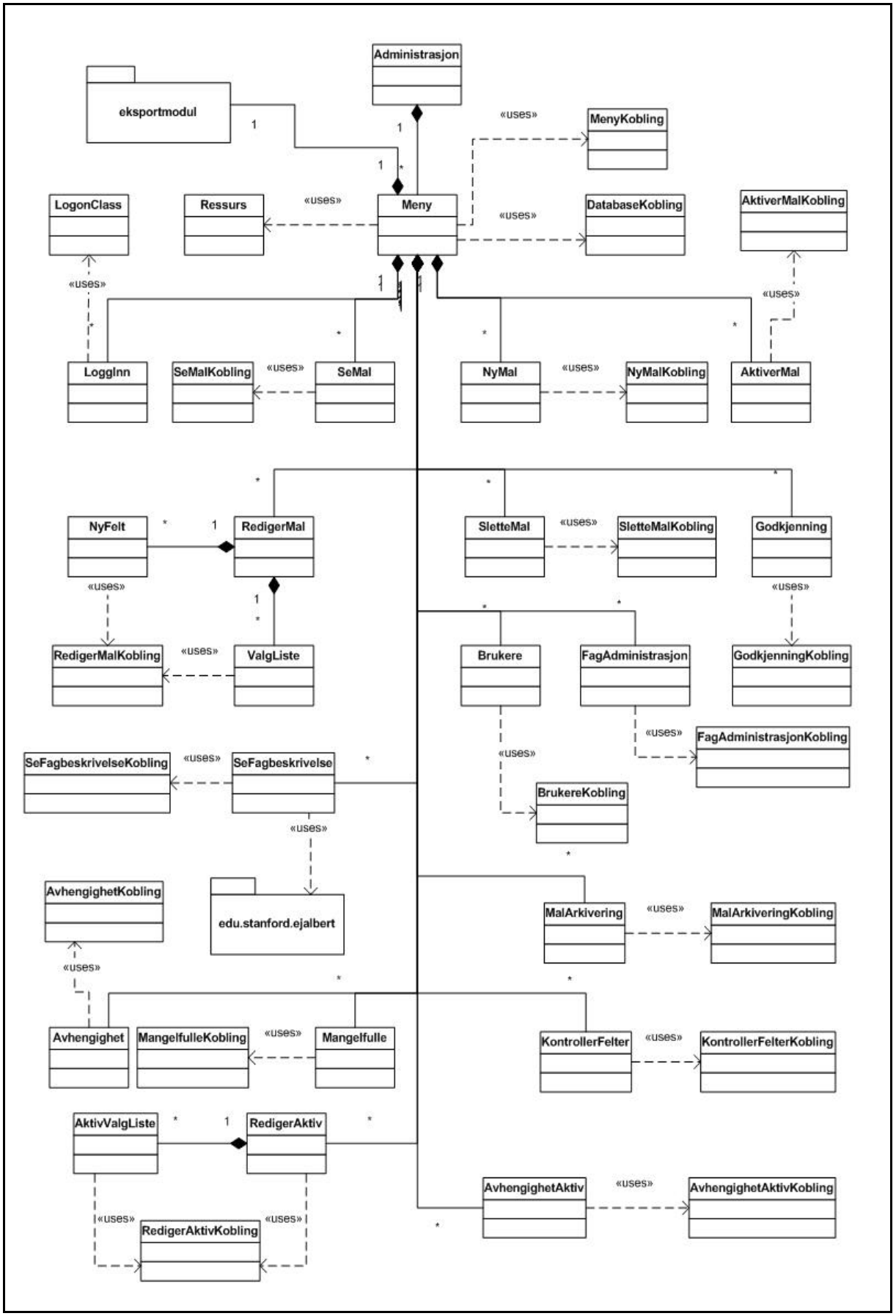
Figur 3-28 – Konfigurasjonsprogrammet til SAFE

Den andre applikasjonen, tilleggsapplikasjon 2, har fått navnet ”Fagimporterer” og inneholder funksjonalitet for å importere fagkoder og fagnavn på både norsk og engelsk inn i systemets database. Hensikten med denne applikasjonen er å lese inn de ovennevnte data fra en fil for deretter å legge disse dataene inn i SAFE -databasen. En forutsetning er at nødvendig data er eksportert til fil fra skolens MSTAS system, slik at dette kan senere importeres til SAFE -systemet. Denne tilleggsapplikasjonen er i hovedsak beregnet på bruk ved igangsetting av SAFE, da databasen mangler store deler av- eller alle gyldige fag ved høgskolen i Gjøvik. Det vil da være hensiktsmessig å bruke denne applikasjonen, i stedet for å manuelt registrere disse. Nedenfor vises en illustrasjon av det grafiske brukergrensesnittet til tilleggsapplikasjon 2.



Figur 3-29 - Grafisk brukergrensesnitt for filimport

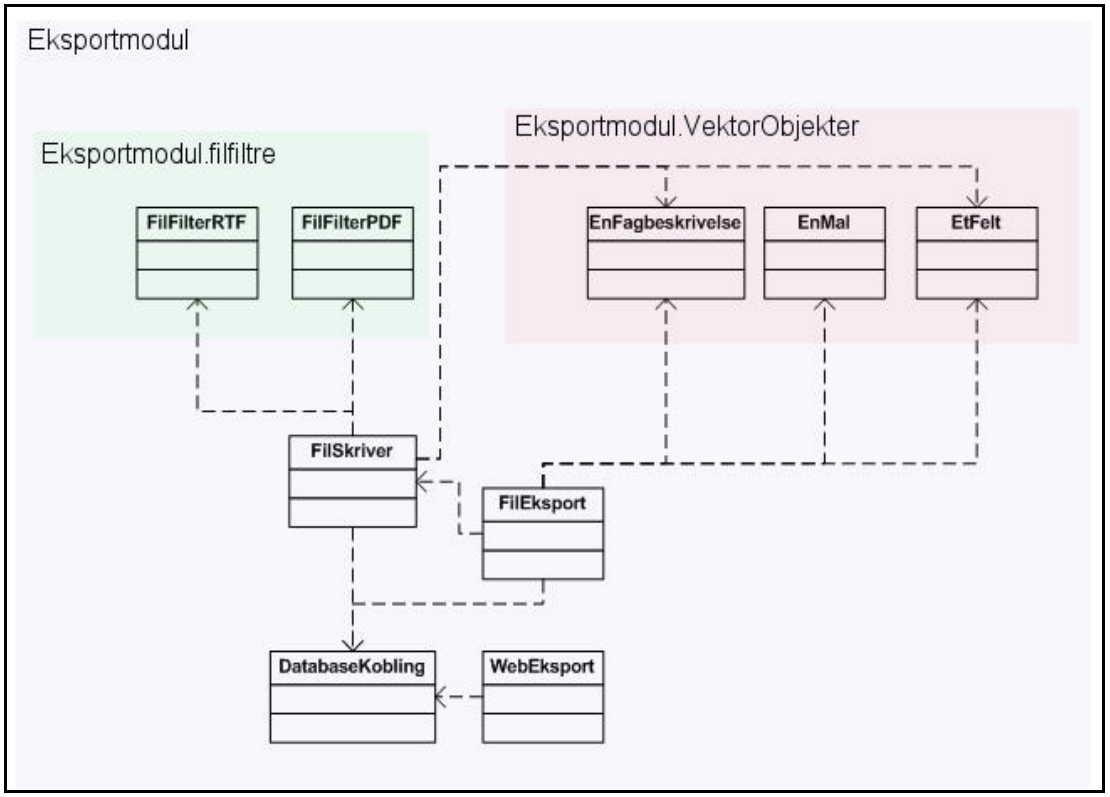
For å kunne ivareta all funksjonalitet som en så kompleks og omfattende del av systemet, som administrasjonsmodulen tilbyr, er det essensielt med oversiktlig og logisk objektorientert design. Dette for å fordele arbeidet mellom mindre deler eller klasser for på den måten å spare datamaskinressurser. Nedenfor vises en overordnet modell over klasser som inngår i administrasjonsmodulen og interaksjonen disse i mellom. Også interaksjon med eksterne klasser eller pakker er vist på figuren.



Figur 3-30 - klassediagram for administrasjonsmodulen

3.3.4 Eksportmodulen

Som nevnt tidligere er denne modulen integrert i administrasjonsmodulen. Den har to hovedoppgaver; velge hvilke felter i en mal som skal vises på web, samt kunne eksportere fagbeskrivelser til fil. Igjen har de samme prinsipper for objektorientert programmering blitt brukt, og figuren under viser et klassediagram over eksporteringsmodulen.



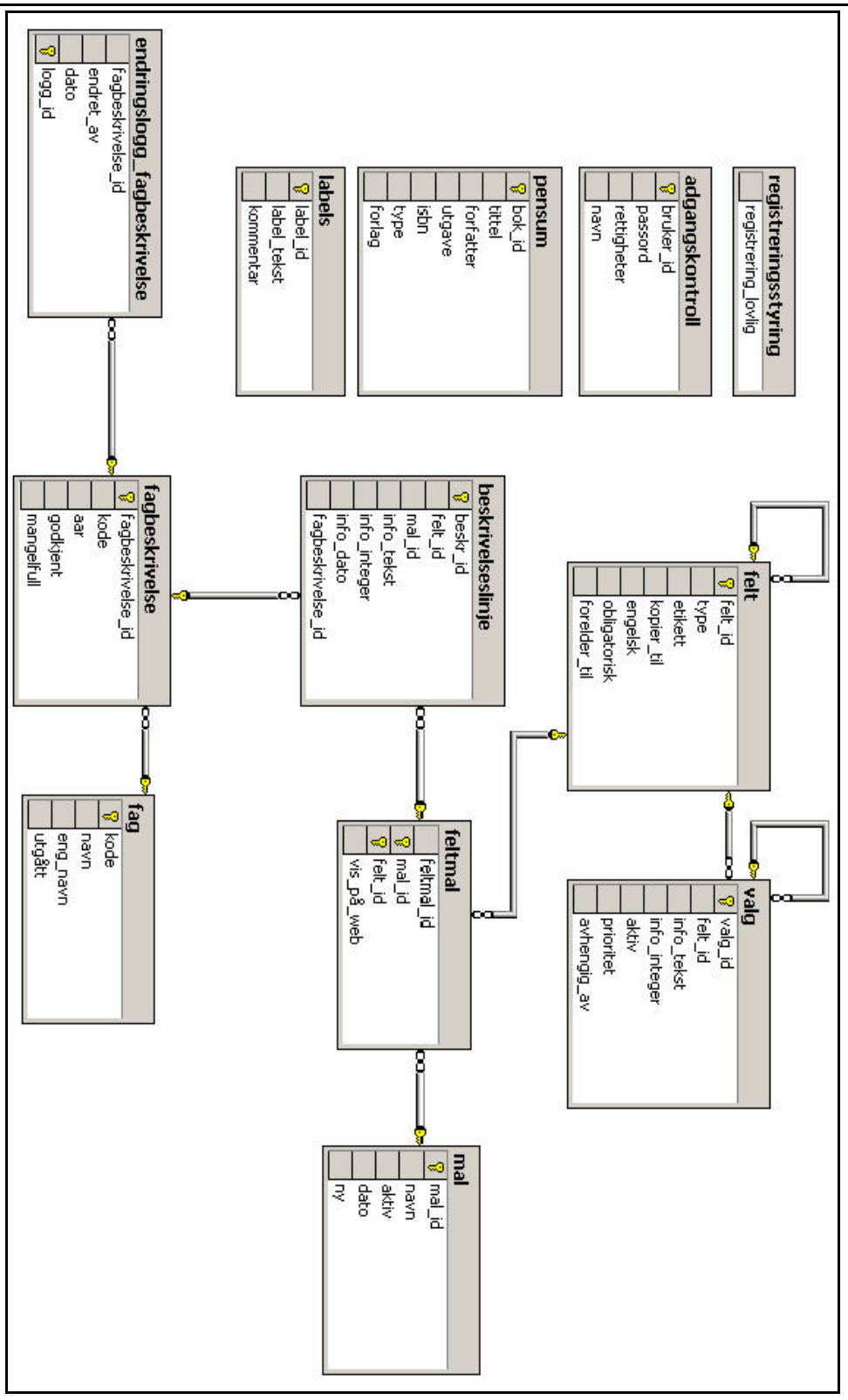
Figur 3-31 – Klassediagram for eksportmodulen

3.3.5 Presentasjonsmodulen

Denne modulen er en Web-modul, og er implementert i PHP. Teknisk har denne blitt designet for å gi en god ytelse mot databasen for raske oppslag, som igjen vil føre til rask respons til bruker. Modulen består av en del PHP-filer som arbeider sammen for å gi brukeren ønsket resultat. Som nevnt tidligere ligger det en CSS fil som bestemmer utseendet på de ferdig kompilerte HTML sidene. Denne modulen krever i tillegg til kontakt med databasen, en webserver som kan kjøre PHP-filer, slik at ferdig kompilert HTML-kode kan fremvises for brukeren. Tilgangen til å benytte denne modulen er det ingen restriksjoner på, det vil si at ingen trenger å logge seg på eller registrere seg for å benytte denne. Dette setter store krav til sikkerhet og ytelse til denne modulen. Dette har vi prøvd å ivareta, men hovedansvaret til sikkerhet og ytelse må ligge på webserveren, og dens administratorer.

3.4 Databasedesign

Databasen er selve grunnsteinen i applikasjonen, og det var derfor viktig å bruke mye tid på å utforme databasestrukturen i starten av prosjektet, og også gjøre små endringer senere der hvor det har vært nødvendig ettersom ny funksjonalitet har kommet til. Figur 3-32 viser den endelige databasestrukturen.



Figur 3-32 – databasestruktur

Under utviklingen av databasen var grunntanken alltid at systemet skulle være utrolig dynamisk. Dette førte til at vi måtte vurdere databasen veldig nøye, og komme opp med noen originale ideer til hvordan strukturen skulle bli. Vi havnet på en løsning hvor hver fagbeskrivelse er tilknyttet et enkelt fag, og hvor hver fagbeskrivelse består av en eller flere fagbeskrivelseslinjer. En linje inneholder igjen et felt, og feltets innhold. De forskjellige

feltene er definert i tabellen ”felt”, og alle felter som har valgmuligheter har sine valg lagret i tabellen ”valg”. Sløyferelasjonene i databasen angir at et felt kan kopiere sitt innhold til et annet felt (gjelder norske felter med tilknyttede engelske felter), eller være avhengig av andre felts valg. I sistnevnte tilfelle kan et valg være avhengig av hvilke valg som er valgt i et tilhørende felt.

Denne databasestrukturen muliggjør lagring av tilnærmet uendelig mange forskjellige maler som alle kan se radikalt forskjellige ut.

Tidligere hadde databasen støtte for å bruke samme fagbeskrivelse flere år for å unngå redundans i databasen, men da det ikke var aktuelt å bruke samme mal flere år på rad ble denne strukturen kassert.

Under følger litt informasjon om hver tabells hovedfunksjon for systemet.

Mal:
I denne tabellen blir alle nye maler registrert.

Feltmal:
Denne tabellen angir hvilke felter som inngår i hver enkelt mal

Felt:
Her blir alle felter som noen gang blir brukt i en fagbeskrivelse lagret

Valg:
Alle valg som på et tidspunkt har blitt registrert i databasen vil finnes her.’

Fag:
I denne tabellen lagres alle fag som tilbys og har blitt tilbudt på HiG siden dette systemet har blitt tatt i bruk.

Fagbeskrivelse:
Her defineres et fags fagbeskrivelse for hvert år og kobles opp mot rader i ”Beskrivelseslinje”.

Beskrivelseslinje:
Denne tabellen holder på all informasjon lagret om alle fagbeskrivelsene.

Registreringsstyring:
Denne tabellen inneholder bare et bit. Hvis dette bittet er satt kan registreringsmodulen startes, og lærere kan begynne å registrere sine fagbeskrivelser

Endringslogg:
Her vil alle endringer gjort på fagbeskrivelser av en bruker bli lagret.

Labels:
Denne tabellen inneholder en del tekststrenger som brukes av enkelte felttyper i registreringsmodulen og i presentasjonsmodulen. Det er meningen at administratorer skal kunne gå inn her og endre disse hvis det er ønskelig.

Pensum:

I denne tabellen vil det etter hvert bli lagret alle læremidler som kan inngå som pensum i et fag

4 Implementering

4.1 Generell implementering

4.1.1 Utvilkingsverkøy og utviklingsspråk

For de tre modulene som skal utvikles i Java (administrasjonsmodulen, registreringsmodulen og eksporteringsmodulen) har vi valgt å bruke Borland JBuilder som utviklingsverktøy. JBuilder har god funksjonalitet for å designe applikasjoners grafiske brukergrensesnitt, og i tillegg har den mange kode funksjoner som hjelper og tilrettelegger for manuell koding, noe som grunnet systemets dynamiske natur har vist seg å være høyst verdifullt.

4.1.2 Eksterne kodebiblioteker

Under utviklingen tok vi i bruk enkelte eksterne kodebiblioteker som utvider Javas grunnpakker. Disse er beskrevet under.

4.1.2.1 Microsoft SQL Server 2000 Driver for JDBC[8]

Microsofts SQL Server 2000 Driver for JDBC (Java DataBase Connectivity) åpner for at Java-applikasjoner kan kommunisere med Microsoft SQL Server 2000, altså vår databaseserver. Denne brukes av alle systemets moduler utviklet i Java.

4.1.2.2 MD5[9]

MD5 er en allsidig hashalgoritme utviklet av RSA Security. Den brukes som oftest til å teste integriteten til meldinger og andre datapakker som sendes over et nettverk ved å generere en form for sjekksum, men kan også benyttes for å hindre at passord blir lagret i klartekst en database, siden sjekksummen alltid blir den samme for samme passord. Dermed kan man teste MD5-hashen som ligger lagret i databasen mot hashen av passordet brukeren skriver inn ved innlogging.

Selv om personer med uærlige hensikter skulle få tak i den MD5-hashede versjonen av passordet, vil det være tilnærmet umulig å reversere algoritmen, for deretter å sitte igjen med passordet i klartekst, bortsett fra å programmere en applikasjon som prøver hver eneste kombinasjon av bokstaver, tall og tegn som det kan tenkes å være i passordet.

Databaseservere som MySQL og PostgreSQL har MD5-hasing av passord innebygd, men siden Microsoft SQL Server 2000 ikke har slik funksjonalitet måtte vi ty til en ekstern implementasjon. MD5-algoritmen som brukes i dette systemet, er en Java-implementasjon utarbeidet av Santeri Paavolainen[10] distribuert under GPL-lisensen.

4.1.2.3 iText[11]

iText er et bibliotek for eksportering av data til Adobe Portable Document Format (PDF) og rikt tekstformat (RTF), i tillegg til enkelte andre formater. iText blir benyttet i eksporteringsmodulen for å skrive data til fil.

iText er utviklet av Bruno Lowagie og er distribuert under LGPL/MPL.

[8] http://www.microsoft.com/
[9] http://www.ietf.org/rfc/rfc1321.txt
[10] http://www.cs.hut.fi/~santtu/java/
[11] http://www.lowagie.com/iText/

4.1.2.4 BrowserLauncher[12]

BrowserLauncher er en liten Java-klasse som automatisk kjenner igjen hvilket operativsystem applikasjonen kjøres på, for deretter å åpne operativsystemets standard nettleser. Denne klassen blir benyttet både i registreringsmodulen og administrasjonsmodulen.

BrowserLauncher er utviklet av Eric Albert og kan brukes fritt i både kommersielle og ikke-kommersielle produkter.

4.1.3 Kommentering av kode

All kode er kommentert i henhold til JavaDoc, slik at det er mulig å automatisk generere en kodespesifikasjon. I tillegg til å muliggjøre denne automatiske genereringen, er også denne måten å kommentere på meget lesbar for mennesker, og er god for å beskrive funksjoner og deres parametere, både for forfatteren av koden og andre.

4.2 Implementering av registreringsmodulen

Registreringsmodulen er den modulen som faglærerne vil måtte logge inn på. Siden det er mange ansatte på Høgskolen i Gjøvik er det ikke fristende å manuelt legge inn hver enkelt ansatt ved hjelp av administrasjonsmodulen. Måten registreringsmodulen autoriserer ansatte på, er derfor ved å verifisere brukernavn og passord opp mot skolens e-postserver. Alle ansatte har en konto på denne e-postserveren. Dette gjør at alle ansatte som har behov for å bruke registreringsmodulen vil få tilgang, mens uvedkommende vil bli hindret adgang.

Som designet av systemet spesifiserer, har hvert av de ulike feltene som kan forekomme i malen sine egne klasser. Da vi bruker Java under utviklingen var det å opprette disse som klasser også en naturlig framgangsmåte for å oppnå oversiktlig og godt forståelig kode, og ikke minst i takt med Javas ellers objektorienterte natur. Alle feltene arver enkelte variable og funksjoner fra en grunnklasse. Her blir forskjellige nødvendige funksjoner for oppretting, lagring, innsetting av gamle data og lignende deklarert. De enkelte feltklassene overstyrer deretter disse funksjonene slik at hvert felt har en funksjon som er tilpasset feltets spesifikke behov.

Dette gjorde for eksempel opprettingen av de forskjellige feltene forholdsvis grei. En Java-vektor benyttes for å holde på alle de forskjellige feltklassene. For hvert felt i den valgte malen, opprettes det et objekt av den riktige feltklassetypen. Det å ha en vektor med alle objektene gjør fellesoperasjoner som innsetting av gamle data og lagring av data enkelt. I tillegg kunne vi ved hjelp av spesielle funksjoner finne hvor gitte felter lå, slik at deres innhold kunne oppdateres. Under følger utdrag fra koden som brukes for å igangsette lagring av data. Koden utnytter det faktum at Java-vektoren (her kalt ”komponenter”) inneholder klasser som alle er underklasser av klassen ”Grunntype”.

```
    for (int i = 0; i < komponenter.size(); i++) {
      try {
         ( (Grunntype) komponenter.elementAt(i)).
                                  lagreData(prepared_insert_statement);
      }
   }
```

[12] http://browserlauncher.sourceforge.net/

Denne koden gjør at hvert felt kjører sin egen lagreData-funksjon, som er spesielt tilpasset feltet, selv om funksjonen blir tilkalt likt for alle klassene.

Koden for innsetting av data fra tidligere fagbeskrivelser følger samme idé, ved at man går igjennom vektoren med objektene, og kaller hver klasses innsettingsfunskjon. Under følger kode for å automatisk sette inn gamle data, der hvor det er tilgjengelig, for to forskjellige typer felt. Det første feltet, ”Tekstomraade”, er et enkelt tekstområde for innskriving av tekst, og det andre feltet, ”Kodefelt”, brukes ved flervalgsmulighet av fagkoder. For sistnevnte vises utdrag fra en mer avansert versjon av funksjonen da denne typen felt trenger mer omtenksom innsetting av data enn et tekstområde.

For Tekstomraade:

```
/**
 * Fyller tekstområdet med data fra eldre fagbeskrivelse
 * @param gamleData Dataset med gamle data
 */
public void setGamleData(QueryDataSet gamleData) {
  if (gamleData.getRowCount() > 0) {
    tekstområde.setText(gamleData.getString("info_tekst"));
    tekstområde.setCaretPosition(0);
  }
}
```

For Kodefelt:

```
/**
 * Fyller listen med data fra eldre fagbeskrivelse
 * @param queryGammelBeskrivelse Dataset med gamle data
 */
public void setGamleData(QueryDataSet queryGammelBeskrivelse) {
  if (!engelsk) {
    int[] selected = new int[queryGammelBeskrivelse.getRowCount()];
    int idx = 0;
    do {
      for (int i=0; i<list_alternativer.getModel().getSize(); i++) {
        if ( ( String)list_alternativer.getModel().getElementAt(i)).
                startsWith(
                  queryGammelBeskrivelse.getString("info_tekst"))) {
          selected[idx++] = i;
        }
      }
    }
    while (queryGammelBeskrivelse.next());
    list_alternativer.setSelectedIndices(selected);
    swapAlternativer(list_alternativer, list_valgte);
  }
}
```

I koden for ”Kodefelt” over bruker vi en spesiell teknikk for å sette inn gamle data. I stedet for å sette de gamle dataene direkte inn i listen som holder på valg som er valgte, for siden å fjerne disse fra alternative valg, markerer vi først de alternative valg som passer med de gamle dataene, for deretter å bruke en spesiell egenutarbeidet funksjon for å sende disse over til den andre listen. Ved å gjøre dette kjøres automatisk en oppdateringsfunksjon hvis feltet har et tilhørende engelsk felt, slik at vi faktisk setter inn gamle data i to felter samtidig.

Norske felter med engelske tilhørlige felter var, i tillegg til felter med valgalternativer som er avhengige av valg gjort i andre felter, en de største utfordringene vi støtte på under utvikling av registreringsmodulen. Med tanke på systemets fleksibilitet og dynamiske natur, var dette langt i fra trivielt å implementere, men siden vi har lagret alle objektene i en vektor kunne vi utarbeide funksjoner som lokaliserte gitte felter i objektvektoren.

Tilnærmet alle de forskjellige typer felter kan ha et tilsvarende engelsk felt. Det engelske feltet skal da automatisk oppdateres med de valg som er den engelske motparten til de valg som er gjort i de norske feltene. Som et resultat av dette, har de fleste feltklassene kode som kjøres avhengig av om feltet er engelsk eller norsk. Hvis feltet er norsk, og har et engelsk felt som skal oppdateres, kjøres en spesiell funksjon som oppdaterer det engelske feltets valg ved hver eneste oppdatering av valg i de norske feltene. Oppdateringen skjer automatisk og de engelske feltene skal i hovedsak ikke kunne endres av brukeren, unntatt der hvor det må skrives inn tekst, dette for å sikre konsekvente data.

Noe tilsvarende skjer hvis et felts tilgjengelige valg er avhengig av valg gjort i et annet felt. Feltene som styrer valgene i andre avhengige felter har egen kode for oppdatering av de valgalternativer man har i det avhengige feltet.

Flere ganger oppsto nødvendigheten av å tilegne elementer i lister mer data enn kun en vanlig, standard tekststreng. Vi oppdaget snart at i Java kan man legge hele objekter i en liste, ikke bare tekststrenger. Vi opprettet derfor egne objekter for en del typer sammensatt data som det var aktuelt å lagre i lister under registrering. Disse objektene ligger i pakken ”ListeObjekter” (for klassediagram, se kapittel 3.3.2). Ved å legge disse objektene rett inn i listene kunne vi tilegne ekstra data til elementene i listen, og vi kunne også referere til, og oppdatere disse.

Se vedlegg E for et eksempel på en feltklasse. Denne klassen implementerer de viktigste prinsippene som er brukt i alle feltklassene, nemlig egene setGamleData- og lagreData-funksjoner.

Registreringsmodulen består av i alt 6 000 linjer eksekverbar kode. Med dette menes kodelinjer som faktisk genererer kjørbar kode, altså ingen blanke mellomrom eller kommentarer.

4.3 Implementering av eksporteringsmodulen

Eksporteringsmodulen består av to hoveddeler, fileksport og webvisning.

Ved fileksport skal brukeren kunne velge mellom to forskjellige filformater, PDF og RTF. Måten vi implementerte dette på var at vi programmerte to filfiltre for Javas innebygde filvelger, et for PDF og et for RTF. Et filfilter er en Java-klasse der man kan bestemme hvilke typer filer som skal aksepteres av filvelgeren. Eksporteringsmodulen genererer filer i det filformatet som hører til det filfilteret brukeren velger i filvelgeren, noe som er vanlig framgangsmåte i de fleste programmer med mulighet for lagring av flere ulike filformater.

Det ble også opprettet en egen, dedikert klasse for skriving av filer; ”FilSkriver”. Denne klassen tar seg av alt som har med skriving til disk å gjøre. Klassen starter selve grovarbeidet, nemlig uthenting av data fra databasen og skriving til fil, i en egen Java-tråd. Dette gjøres for å kunne kjøre to funksjoner parallelt, en i hver tråd. Dermed får vi tilgang til nok CPU-ressurser slik at det blir mulig å oppdatere en framgangsindikator samtidig som grovarbeidet med å lese fra databasen og skrive til fil blir gjort. Vi så oss nødt til å ha en

framgangsindikator, da det tar en del tid å hente slike store mengder data, og i så måte er framgangsindikatoren et viktig element fra et brukervennlighetsperspektiv da brukeren fort kan tro programmet har låst seg hvis det forblir visuelt statisk over lengre tid.

Klassen benytter seg av det eksterne biblioteket iText (se kapittel 4.1.2.3) for å skrive dataene til fil. iText gjør det mulig å opprette tabeller som inneholder tekst, på samme måte som i Microsoft Word. Denne funksjonaliteten benytter vi for å bygge opp en fagbeskrivelse på filen. Det opprettes en tabell for hver fagbeskrivelse som deretter skrives til filen, etterfulgt av valgfritt sideskift.

Webvisning er en enklere del av eksporteringsmodulen hvor brukeren ved hjelp av sjekkbokser angir hvilke av en mals felter som skal vises med presentasjonsmodulen.

Eksporteringsmodulen består av i alt 2 000 linjer eksekverbar kode.

4.4 Implementering av administrasjonsmodulen

Administrasjonsmodulen er den av de fire modulene som er minst dynamisk med tanke på det grafiske brukergrensesnittet, og dermed kunne vi bygge opp og designe hele det grafiske brukergrensesnittet før vi gikk i gang med å implementere selve funksjonaliteten. Vi fikk av denne grunn mulighet til å gjøre god nytte av Borland JBuilders visuelle designfunksjonalitet som gjør det mulig å plassere ut komponenter der vi ønsker, med kun mindre begrensninger. Dette gjorde også at vi raskt kunne forandre på ting etter å ha konferert med veileder og oppdragsgiver, uten å måtte skrive om store deler av programmet grunnet underliggende, og som det senere viste seg, tung funksjonalitet.

Etter å ha fått på plass en implementasjon av brukergrensesnittet som både vi, oppdragsgiver og veileder var fornøyd med, ble arbeidet med den underliggende funksjonaliteten påbegynt. Administrasjonsmodulen kan kanskje sies å være selve hjertet i systemet, og det er av denne grunn mye funksjonalitet som skal implementeres. Administrasjon av brukere, fag, fagbeskrivelser, maler for fagbeskrivelser, eksportering og visning på web må alle ivaretas i administrasjonsmodulen. Derfor er det et bredt spekter av brukere som skal bruke denne modulen. Siden det ikke er realistisk å anta at alle brukere av administrasjonsmodulen er kyndige datamaskin brukere er det også her som i registreringsmodulen viktig å stoppe, eller i det minste begrense, feil som kan oppstå på grunn av feil bruk av funksjonalitet i denne modulen. Fordi de tre andre modulene i systemet i stor grad er avhengig av definisjoner og oppsett under administrasjonsmodulen, kan selv de minste feil føre til vesentlig skade og det er derfor desto viktigere at modulen er feilfri. Derfor har testing hele tiden vært en vesentlig del av arbeidet, både under og etter implementering av funksjonalitet i administrasjonsmodulen. I deler av denne modulen, som ved redigering av maler, skal det være mulig for brukeren å angre operasjoner som er gjort. Programmet må derfor kunne ivareta endringene internt og noe uavhengig av databasen slik at disse er reverserbare. På denne måten kan også feil begrenses, da disse kan lukes ut før lagring. For eksempel ved redigering av maler, skal maler ha felter som igjen har valg og alle disse har egne tilhørende engelske felter eller valg. Felter og valg kan være avhengig av hverandre, og alle felter og valg kan styres opp og ned i listene, slik at de kommer i riktig rekkefølge. For å kunne ivareta dette ble det brukt kode som internt i teamet har fått navnet ”The Cube”. Under vises noen kodeeksempler av denne.

```
// initialisering av ”THE CUBE”
private int[] mal_id;
private int[][] felt_id_norsk;
```

```
private int[][] felt_id_engelsk;
private int[][][] valg_id_norsk;
private int[][][] valg_id_engelsk;

int[] tempfelt_norsk = new int[felt_id_norsk[m].length + 1];
int[] tempfelt_engelsk = new int[felt_id_norsk[m].length + 1];
int[][] tempvalg_norsk = new int[felt_id_norsk[m].length + 1][];
int[][] tempvalg_engelsk = new int[felt_id_norsk[m].length + 1][];

----------
// eksempel på bruk (brukes ved flytting av felt og valg)
temp = felt_id_norsk[m][f];
felt_id_norsk[m][f] = felt_id_norsk[m][f-1];
felt_id_norsk[m][f-1] = temp;
temp = felt_id_engelsk[m][f];
felt_id_engelsk[m][f] = felt_id_engelsk[m][f-1];
felt_id_engelsk[m][f-1] = temp;
temparr = valg_id_norsk[m][f];
valg_id_norsk[m][f] = valg_id_norsk[m][f-1];
valg_id_norsk[m][f-1] = temparr;
temparr = valg_id_engelsk[m][f];
valg_id_engelsk[m][f] = valg_id_engelsk[m][f-1];
valg_id_engelsk[m][f-1] = temparr;
```

Dette er rekker av arrays med diverse dimensjoner hvor all nødvendig data om maler, felter og valg ivaretas på en slik måte at endringer kan angres, samtidig som databasen hele tiden er oppdatert med hensyn på nye felter og valg.

Under utvikling av denne modulen har brukeren hele tiden vært i fokus og vi har forsøkt å lage modulen på en slik måte at den er intuitiv eller lett å lære. Dette er selvfølgelig også viktig for at feil på grunn av feilbruk oppstår.

I denne modulen er all funksjonalitet i stor grad implementert i egne klasser. Dette er både for å ha en mer oversiktlig kode, men også for å fordele arbeidet mellom mange klasser. Dessuten blir gjenbruk av funksjoner også enklere på denne måten, da mange klasser kan bruke en og samme funksjon eller klasse til å utføre operasjoner. Det grafiske brukergrensesnittet i administrasjonsmodulen gjør det enkelt å distribuere funksjonaliteten i mange klasser. Hver gang brukeren velger noe fra menyen opprettes det en instans av den ansvarlige klassen som også inneholder panel for grafisk fremvisning. Dette panelet puttes så inn i hovedvinduet, eller menyvinduet. Dette er selvfølgelig operasjoner som er helt usynlig for bruker. På denne måten kan også hele modulen deles inn i mindre deler til implementasjon. Dette betyr at hver klasse, eller en samling av klasser kan ha ansvaret for en bestemt del av modulen. På denne måten spares det datamaskinressurser som den ene eller de få klassene kan benytte seg av. Under implementasjon av administrasjonsmodulen har det vært en målsetting å skille mellom grafiske operasjoner, funksjonelle operasjoner og databaseoperasjoner. Disse er delt i egne klasser og opprettes kun ved behov.

Administrasjonsmodulen består av en egen GUI-basert applikasjon for å kunne opprette og redigere konfigurasjonsfiler som brukes av alle modulene. Denne applikasjonen er vesentlig, da alle passord i konfigurasjonsfilene skal hashes og dermed ikke skrives i klartekst. I konfigurasjonsfilene står informasjon om databaseserver, webserver, e-postserver, porter, brukernavn og passord for disse.

Administrasjonsmodulen består av i alt 13 000 linjer eksekverbar kode.

4.5 Implementering av presentasjonsmodulen

Allerede tidlig i planleggingsfasen falt valget på PHP som utviklingsspråk for presentasjonsmodulen. PHP er et meget fleksibelt serversideprogrammeringsspråk for Web-applikasjoner, og har innebygd mulighet for å koble opp mot en mengde forskjellige databaser, deriblant Microsoft SQL Server 2000.

Under utviklingen av presentasjonsmodulen har vi konsentrert oss om å kommentere godt og bruke variable med meningsfulle navn, da blandingen av PHP og HTML har en tendens til å bli uoversiktlig, noe enkelte av gruppens medlemmer har erfart fra tidligere prosjekter.

Under vises et kodeeksempel som fremhever kommentarer og gode variabelnavn. Koden brukes for å lese et årstall ut i fra en URL til en fagbeskrivelse på formen `http://server/fagbeskrivelse.php?fagkode=H388G&aar=2003`. Hvis årstall ikke er sendt med i URL-en antas nyeste fagbeskrivelse.

```
/**
 * henter ønsket år fra URL
 * hvis år ikke er sendt med, antas det at bruker vil se nyeste
 * fagbeskrivelse for gjeldende fag
 */
if(!isset($_GET['aar']) || $_GET['aar'] == "") {
   // antar nyeste fagbeskrivelse hvis år ikke er sendt med
   $aar_query = "SELECT MAX(aar) AS aar FROM fagbeskrivelse WHERE
                          kode = '${fagkode}'";
   $aar_result = mssql_query($aar_query);
   $aar = (is_numeric(mssql_result($aar_result,0,"aar"))) ?
                           mssql_result($aar_result, 0,"aar"):die();
}
else {
   $aar = $_GET['aar'];
}
```

For å sikre både bakoverkompatibilitet og kompatibilitet med kommende Web-standarder, er all HTML som blir generert av presentasjonsmodulen basert på W3Cs XHTML 1.0 Strict-standard. Ved å følge denne standarden skilles det klart mellom HTML-dokumentets struktur og innhold på den ene siden, og stil og utseende på den andre. Dokumentets stil og utseende styres av CSS (Cascading Style Sheets, også kalt stilark på norsk), også en standard utviklet av W3C, med nordmannen Håkon Wium Lie i spissen. Med CSS kan man angi farger, posisjoner, rammer og mye annet til elementer som finnes i HTML-dokumentet, uten at man trenger å skrive dette inn i selve HTML-dokumentet. Den største fordelen med CSS er derfor at det vil være elementært å forandre farger og andre mindre designmessige ting, siden slikt kun angis ett sted, nemlig i CSS-dokumentet, og er ikke spredt rundt i flere HTML-dokumenter.

Presentasjonsmodulen er dermed visuelt sett meget tilpasningsdyktig, med tanke på at man med enkle grep kan modifisere den for å tilpasse den til skolens Web-side. Under vises eksempler på ulike implementasjoner av CSS, men med identisk HTML, som gir vidt forskjellige resultater.

Figur 4-1 – Utsnitt fra fagbeskrivelse med standard CSS

Figur 4-2 – Utsnitt fra fagbeskrivelse med alternativ CSS

Se vedlegg G for nærmere beskrivelse av hvordan man kan manipulere CSS for å endre utseende på presentasjonsmodulen uten å måtte redigere HTML- og PHP-kode.

I likhet med andre deler av systemet er også presentasjonsmodulen i høyeste grad dynamisk, og den støtter enhver mal som blir opprettet med administrasjonsmodulen.

Hvert av de forskjellige feltene blir behandlet ulikt i presentasjonsmodulen, som de også blir i presentasjons- og eksporteringsmodulen. Eksempelvis blir enkelte listefelter vist kommaseparert på en linje, mens andre blir vist som en liste. Et spesialtilfelle er de forskjellige lister med fagkoder. Disse er implementert på en slik måte at alle referanser til fag er linket direkte til det fagets fagbeskrivelse, såfremt faget har en eksisterende fagbeskrivelse. På den måten kan brukeren klikke seg fra et fags fagbeskrivelse til andre fagbeskrivelser for fag som bygger på eller forutsetter det opprinnelige faget.

Presentasjonsmodulens søkemotor er implementert på en slik måte at den basert på de søkekriterier brukeren gir, syr sammen en tildels kompleks SQL-setning som så blir sendt til databasen som en vanlig SQL-forespørsel. Resultatet fra databasen er da søkeresultatet, som forevises brukeren som en liste med linker til relevante fagbeskrivelser.

Søkemotoren åpner for å søke etter eldre fagbeskrivelser, dvs. fagbeskrivelser som er gjaldt for et eller flere år tilbake.

Under vises et eksempel på hvordan man effektivt genererer en HTML-tabell med PHP, basert på en databasespørring. Kodeeksempelet illustrerer også hvordan man kan blande PHP og HTML. Her er det gjort slik at PHP og HTML mest mulig blir skrevet på hver sin linje. Det er ikke noe i veien for å skrive både HTML og PHP på en linje, men det blir ofte uoversiktlig og til tider uleselig for andre enn forfatteren av koden.

```
<h1>Fag ved Høgskolen i Gjøvik,
<?php echo $aar; ?>
</h1>
<table>
<?php if($sprk == 1) { ?>
<tr><th>fagkode</th><th>fagnavn</th></tr>
<?php } else { ?>
<tr><th>course identifier</th><th>course title</th></tr>
```

```
<?php }
while($query_data = mssql_fetch_assoc($fagResult))
      {
      echo "<tr><td>".$query_data["kode"]."</td><td>";
      echo ($sprk == 1) ? $query_data["navn"]:$query_data["eng_navn"];
      echo "</td></tr>\n";
      }?>
</table>
```

Koden over er en forenklet versjon av den som brukes ved generering av presentasjonsmodulens listefunksjonalitet. Denne listefunksjonaliteten viser de fagene på Høgskolen i Gjøvik som har en registrert fagbeskrivelse på en oversiktlig måte. Brukeren kan velge fra hvilket årstall han eller hun vil se fagbeskrivelser, og kan også sortere alfabetisk enten etter fagkoder eller fagnavn.

Presentasjonsmodulen består av i alt 1 000 linjer kode.

5 Testing

5.1 Teststrategier

Under utvikling av prosjektet har det vært vesentlig at administrasjonsmodulen og registreringsmodulen ble utviklet noenlunde parallelt. Dette for å kunne lettere teste små og større deler av SAFE, da disse to modulene er såpass avhengige av hverandre. I registreringsmodulen kunne feil som ble generert under oppsett av maler og annet i administrasjonsmodulen oppdages på en lettere måte enn å kontrollere resultatene i databasen, eller ved tradisjonell debugging linje for linje. Samtidig kunne vi i administrasjonsmodulen kontrollere om registrert data fra registreringsmodulen ble lagret og håndtert korrekt. Det var dermed naturlig å bruke disse to modulene som testverktøy for hverandre.

Framgangsmåten ved testing av applikasjonen var å først teste enkeltklasser, for deretter å teste en gruppe av klasser, en pakke. Testing av moduler ble gjort etter at eventuelle feil var luket ut, og vi var fornøyde med integriteten til pakkene.

Etter dette kunne parallell- og sikkerhetstesting igangsettes, hvor administrasjonsmodulen og registreringsmodulen ble brukt som testverktøy.

5.1.1 Testing av klasser

Målet ved klassetesting er å sikre at klassen gjør hele det arbeidet den er tiltenkt. Det er viktig at dette blir gjort på en kontrollert, strukturert og forutsigbar måte, da klassene er byggesteinene til hele systemet, og uforutsigbare resultater her vil kunne få fatale konsekvenser senere i utviklingsarbeidet.

Til dette brukte vi blant annet konsollutskrifter for å sikre at variable inneholdt det som var forventet, og at funksjoner returnerte verdier som antatt. I tillegg til dette brukte vi også JBuilders innebygde debugfunksjonalitet som tillater kjøring av koden linje for linje, samtidig som man får oversikt over verdiene til variable, funksjoner og objekter.

Videre ble klasser som har input- og outputverdier testet og gjort robuste mot feil inn- og utdata slik at disse til en viss grad kan håndtere feilsituasjoner som kunne oppstått.

5.1.2 Testing av pakker

Etter å ha testet klassene hver for seg var det viktig å teste samarbeidende enkeltklasser, for å sikre at de fungerte godt sammen. Det var her viktig at hver klasse kunne ivareta sitt ansvar i programmet, slik at resten av pakken av klasser fikk korrekte data.

Eksempelvis hadde vi i administrasjonsmodulen pakker som består av klasser med hvert sitt ansvarsområde. En klasse kunne for eksempel ha ansvar for grafisk brukergrensesnitt og interaksjon med brukeren, en annen klasse kunne ha ansvar for databaseinteraksjon mens en tredje fungerte som en slags kobling mellom pakkene.

Dette var også viktig da eksporteringsmodulen senere skulle kobles inn og integreres i administrasjonsmodulen. Interaksjonen mellom disse to modulene måtte være godt testet for ikke å få uforutsigbare eller feil resultat.

I denne testfasen oppdaget vi at krypteringsalgoritmen som vi i utgangspunktet hadde tenkt å ta i bruk for kryptering av passord, før disse ble lagret i databasen, ikke gav tilfredsstillende resultater. Dermed måtte denne erstattes. Valget falt senere på en MD5-hash som en erstatning for kryptering, da dette er en irreversibel, og dermed sikrere, måte å lagre uleselige versjoner av passordene. Etter ytterligere testing fant vi at denne algoritmen holdt mål.

5.1.3 Testing av moduler

I denne fasen ble modulene testet for generell bruk, hvor brukeren også i stor grad måtte fange opp eventuelle feil og mangler ved modulene. Hovedmålet her var å se til at modulene gjorde alt de skulle, det vil si i henhold til kravspesifikasjonen. Derfor var det i hovedsak mangler og endringer som kunne føre til mer intuitive moduler som ble oppdaget på dette stadiet. Interaksjonen med brukeren ble også satt på prøve i denne fasen hvor fokus ble satt på feil bruk av enkeltfunksjonalitet for å sikre robusthet. Småfeil som modulene kunne inneholde internt og i bakgrunn ble senere testet under parallelltestingsfasen.

5.1.4 Databasetesting

Da all informasjon lagres og ivaretas i databasen, var det viktig med databaserelatert testing. Dette for å sikre at integriteten ivaretas og at innsetting av nye data ikke har konsekvenser for eldre data som allerede finnes i databasen, og ikke minst at data som hentes ut av databasen er konsekvent med de data som ble lagt inn. Systemet måtte ha funksjonalitet for å gjenopprette integritet dersom denne skulle kompromitteres, for eksempel ved oppretting av avhengige felter og valg måtte systemet gjenskape integritet i henhold til de nye endringene.

Mange begrensninger og potensielle feilsituasjoner ble oppdaget i denne fasen som for eksempel lagring av større mengder tekst. For å kunne muliggjøre inntasting av lengre tekster fra brukeren, måtte felter i databasen baseres på et annet tekstformat enn det vi tidligere hadde implementert.

Nye relasjoner og tabeller oppstod som resultat av testing i denne fasen, hvor begrensninger eller bedre løsninger på problemer ble identifisert etter hvert som utviklingsarbeidet skred frem.

5.1.5 Parallelltesting

Her gjaldt det, ved å bruke en modul, å oppdage feil som eventuelt kunne oppstå ved bruk av en eller flere av de andre modulene. For eksempel vil det være enkelt å se feil i en mal under registreringsmodulen som hadde sitt opphav i administrasjonsmodulen, og omvendt. Siden modulene har såpass høy grad av interaksjon var det her de fleste feil ble oppdaget og luket ut. Enkelte feil som oppsto grunnet feil bruk fra brukeren side ble også oppdaget her og ofte måtte brukergrensesnittet restruktureres eller begrenses noe for å hindre brukeren i å begå samme feilen. Det var et viktig krav at de forskjellige modulene måtte ha en høy grad av samspill seg i mellom, og dette måtte skje på en kontrollert og harmonisk måte.

5.1.6 Brukertesting

Stort sett fant vi her feil i systemet som oppsto grunnet misbruk av funksjonalitet, spesielt i administrasjonsmodulen. Brukeren kunne ofte forsøke å utføre ting som modulen ikke var beregnet på å kunne håndtere. Dette førte til at endringer dukket opp underveis og at

kravspesifikasjonen konstant var under revidering, nærmest på en daglig basis da brukeren her oppdaget funksjonalitet som han ville ha med, eller være for uten. Imidlertid førte dette til en høy gard av forståelse for bruken av modulene hos brukeren, noe som igjen hjalp både oss og brukeren med å oppdage hvordan systemet egentlig skulle være. Med hensyn til brukeren var dette den viktigste fasen av testing da han her kunne oppdage og se klarere for seg sin egentlige interaksjon med systemet, og dermed oppdage hvilke feil som kunne oppstå som resultat av hans distinktive interaksjon med systemet. Dette var verdifull feedback for oss som utviklere da vårt mål hele tiden har vært å skape et produkt som er enkelt, intuitivt og som oppfyller brukerens krav til standard på en best mulig måte.

5.1.7 Sikkerhetstesting

Da faglærere ved første innlogging bruker samme brukernavn og passord som på skolens e-postserver og kan velge å forstette å benytte seg av samme passord på systemet, er det av høyeste prioritet at passordet ikke blir lagret i klartekst i systemets database. Passordet skal ikke være tilgjengelig for noen, inkludert administrator, databaseansvarlig og utviklere, da dette passordet er personlig og brukes på andre, mer sensitive systemer. For å kunne ivareta dette var krypteringsalgoritmer ikke egnet, da disse kan reverseres fra kryptert form til klartekst så fremt krypteringsnøkkelen er kjent, noe som i dette tilfellet rammer utviklere og administrator av systemet. Derimot er hash-algoritmer noe mer attraktive da disse er irreversible og krever ingen nøkkel. På denne måten kan passord hashes til et bestemt format som lagres i databasen.

Vi har brukt en vel utprøvd hash-algoritme, kalt MD5. Denne er i utstrakt bruk, og blant annet innebygd i flere databaseservere for hashing av passord. Denne algoritmen er såpass godt testet av folk med mer innsikt enn oss. Dermed var det intet poeng å videre teste MD5, unntatt å påse at man ikke kunne komme rundt våre innloggingsrutiner uten bruk av gyldig brukernavn og passord.

5.1.8 Plattformtesting

I utgangspunktet gikk vi ut i fra at Java var totalt plattformuavhengig. Under utviklingsarbeidet oppdaget vi at selv om programmene var kjørbare på tvers av diverse plattformer, var det allikevel visse forskjeller, spesielt med tanke på brukergrensesnitt. Den underliggende funksjonaliteten ble for øvrig behandlet identisk under forskjellige plattformer og operativsystemer. Siden det kun er utseende som påvirkes, og siden dette ikke har stor relevans for den daglige bruken av programmet, var ikke det å sikre identisk brukergrensesnitt mellom operativsystemer noen høy prioritet mot slutten av prosjektfasen.

Den eneste underliggende funksjonaliteten som ikke fungerte på UNIX/Linux, men som samtidig fungerte på Windows var autorisering av brukere ved hjelp av skolens e-postserver. Grunnen til dette var forskjellig implementasjon av linjeskift på de forskjellige plattformene. I UNIX/Linux markeres linjeskift med ”LF” (Line Feed), mens på Windows markeres den med ”CR/LF” (Carriage Return, Line Feed). Windows-baserte e-postservere, som blant annet skolen benytter seg av, forventer både LF og CR. Derfor fungerte det ikke med UNIX/Linux’ versjon av linjeskift. Vi løste dette ved å manuelt sende med LF og CR, i stedet for å stole på operativsystemets implementasjon av linjeskift.

For øvrig oppdaget vi at det kunne være relativt store forskjeller når det gjelder funksjonalitet og inkluderte klasser mellom gjeldene versjoner av Java VM, noe som gjorde det nødvendig å inkludere den Java VM-versjonen som ble brukt under utvikling i systemets installasjonsprogrammer.

6 Installasjon og konfigurasjon av SAFE

6.1 Systemkrav og forberedelser

For å kjøre SAFE-systemet kreves følgende:

- Microsoft SQL Server 2000
- Web-server med PHP v4.3 eller nyere
- Installasjonsfiler for de forskjellige modulene som inngår i SAFE

Uavhengig av valgt Web-server må den støttes av PHP[13]. PHP kan lastes ned fra `http://www.php.net/downloads.php` under overskriften ”Windows Binaries”. Hvis valgte Web-server er Microsoft IIS trengs følgende filer:

PHP 4.3.x zip package	(php-4.3.x-Win32.zip)
PHP 4.3.x installer	(php-4.3.x-installer.exe)

Hvis valgte Web-server er Apache, trengs kun den førstnevnte, *PHP 4.3.x zip package*.

6.2 Installasjon og oppsett av databaseserver

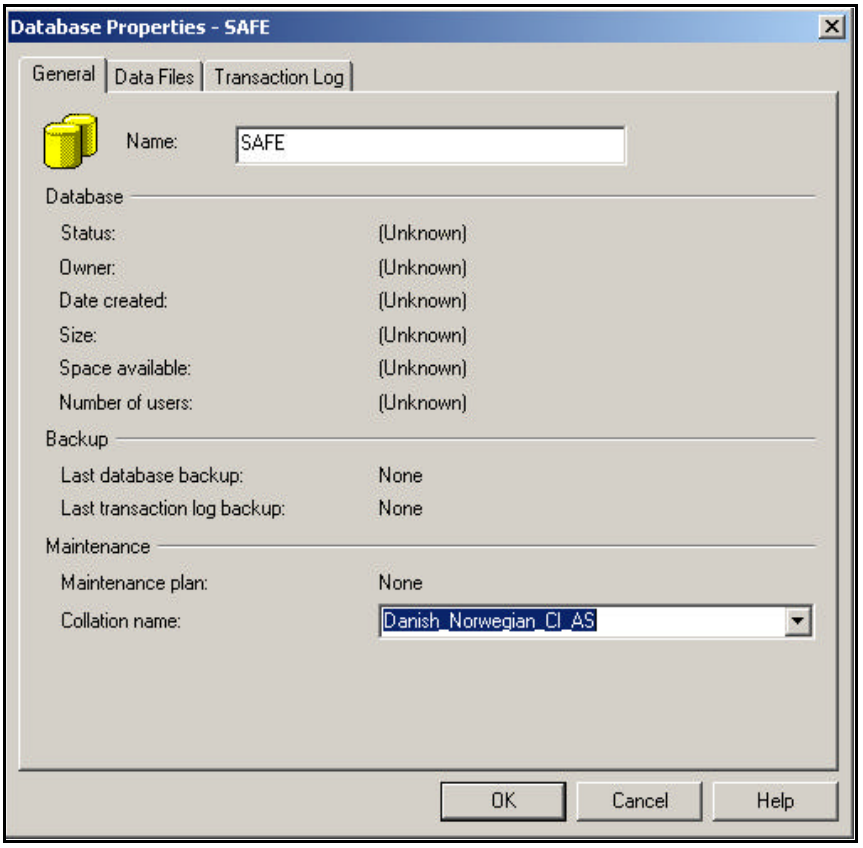
Installer Microsoft SQL Server 2000 på vanlig måte i henhold til Microsofts installasjonsveiledning. Husk å velge å installere både server og klientverktøy (”Server and Client Tools”).

6.2.1 Opprett database

Etter at installasjonen av databaseserveren er ferdig, start opp Microsoft SQL Server 2000 Enterprise Manager for å sette opp database og brukerkonto for SAFE-systemet. Slik går du fram for å opprette ny database:

1. Hvis det ikke finnes noen databaseservere under ”SQL Server Group” i treet i venstre panel, høyreklikk på ”SQL Server Group”, og velg ”New SQL Server registration”.
    a. Klikk ”Next”.
    b. Flytt maskinnavnet til maskinen som serveren ble installert på fra venstre til høyre liste ved å klikke ”Add”. Klikk ”Next”.
    c. Velg ”The SQL Server login information that was assigned to me by the system administrator [SQL Server Administration]”. Klikk ”Next”.
    d. Skriv inn det brukernavn og passord som ble valgt for systemadministrator under installasjon av Microsoft SQL Server 2000. Klikk ”Next”.
    e. Klikk ”Next”.
    f. Klikk ”Close”.
2. Ekspander treet til databaseserveren ved å klikke på ’+’ ved siden av maskinnavnet.
3. Høyreklikk på ”Databases” og velg ”New Database…”.
4. Under ”Name”, skriv inn navnet for SAFEs database. Bruk for eksempel ”SAFE”.
5. Under ”Collation Name”, velg ”Danish_Norwegian_CI_AS” (se figur 6-1).
6. Klikk ”OK”.

[13] Liste over støttede Web-servere finnes på http://www.php.net/manual/en/installation.php

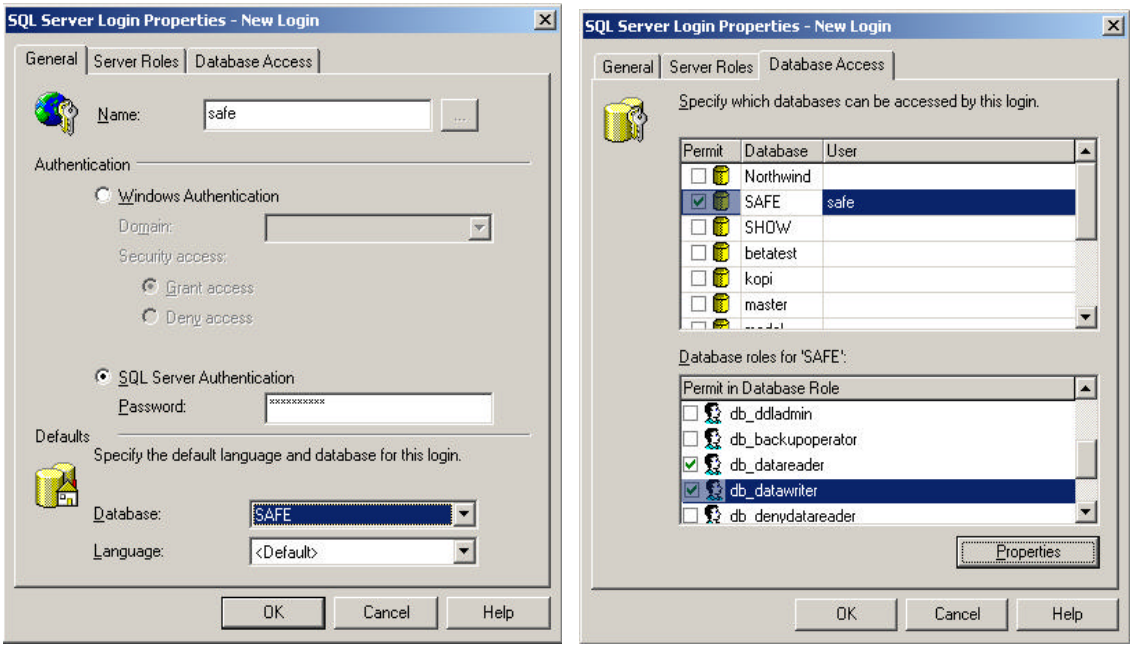


Figur 6-33 - Oppretting av database

6.2.2 Opprett brukerkonto

Etter å ha opprettet databasen må det settes opp en brukerkonto som SAFE skal benytte seg av ved oppkobling mot databasen.

1. I panelet til venstre, klikk på ”Security”.
2. Dobbeltklikk deretter på ”Logins” i panelet til høyre.
3. Høyreklikk så i panelet til høyre og velg ”New Login…”. Et skjermbilde som på figur 2, venstre, vises.
4. Skriv inn et brukernavn under ”Name”. Dette kan være hva som helst, for eksempel ”safe”.
5. Under ”Authentication”, velg ”SQL Server Authentication” og skriv inn et passord under ”Password”. Bruk helst en blanding av tall og store og små bokstaver.
6. Under ”Defaults”, sett ”Database” til den databasen du opprettet for SAFE. Dette er et utrolig viktig punkt som gjør at SAFE kan koble til den riktige databasen på databaseserveren.
7. Klikk deretter på fanen (”tab” på engelsk) ”Database Access” (se figur 3, høyre).
8. Kryss av ved databasen du opprettet for SAFE.
9. Under ”Database Roles”, kryss av for ”db_datareader” og ”db_datawriter” i tillegg til ”public”.
10. Klikk ”OK”.



Figur 6-34 – Oppretting av ny brukerkonto for SAFE

6.2.3 Opprett databasestruktur

Etter at brukeren er opprettet er tiden kommet til å lage databasens tabeller og strukturelle innhold. Dette gjøres ved å kjøre SQL-filen som ligger vedlagt på CD-en. Denne filen inneholder SQL-kommandoer for oppretting av tabeller og nødvendig data. For å utføre disse SQL-kommandoene må Microsoft SQL Query Analyzer benyttes. Dette er et program som blir installert sammen med Microsoft SQL Server 2000.

Fremgangsmåten er som følger:

1. Start opp Microsoft SQL Query Analyzer
2. Hvis ikke ”Connect to SQL Server”-vinduet automatisk kommer opp, klikk ”File” -> ”Connect”.
3. I ”Connect to SQL Server”-vinduet, velg ”SQL Server authentication” og skriv inn brukernavn og passord for databaseserverens systemadministrator (ble satt opp under installasjon av databaseserveren).
4. Hvis oppkoblingen mot databasen går bra vil det dukke opp et tomt vindu. Klikk ”Query” -> ”Change Database…”
5. Velg SAFEs database i listen, og klikk ”OK”.
6. Klikk ”File” -> ”Open” for å åpne filen med SQL-kommandoene.
7. Naviger til CD-ROM-stasjonen og katalogen \installasjonsfiler\database\.
8. Dobbeltklikk på ”databaseskript.sql”.
9. Innholdet i filen vil nå komme opp i det blanke vinduet. Klikk ”Query” -> ”Execute”.

Hvis alt går greit vil det komme opp en del linjer hvor det står ”(1 row(s) affected)”. Hvis du opplever at noe går galt, vennligst se til at du benytter en brukerkonto med rettigheter til å opprette nye tabeller i SAFEs database. Som nevnt over bør systemadministratorkontoen som ble satt opp under installering av databaseserveren benyttes.

6.3 Installasjon og oppsett av Web-server

SAFE kan brukes med mange forskjellige Web-servere, men det er et krav at PHP kan installeres på serveren. SAFE har blitt utprøvd på Apache v1.3, Apache v2.0 og Microsoft Internet Information Services v5.1.

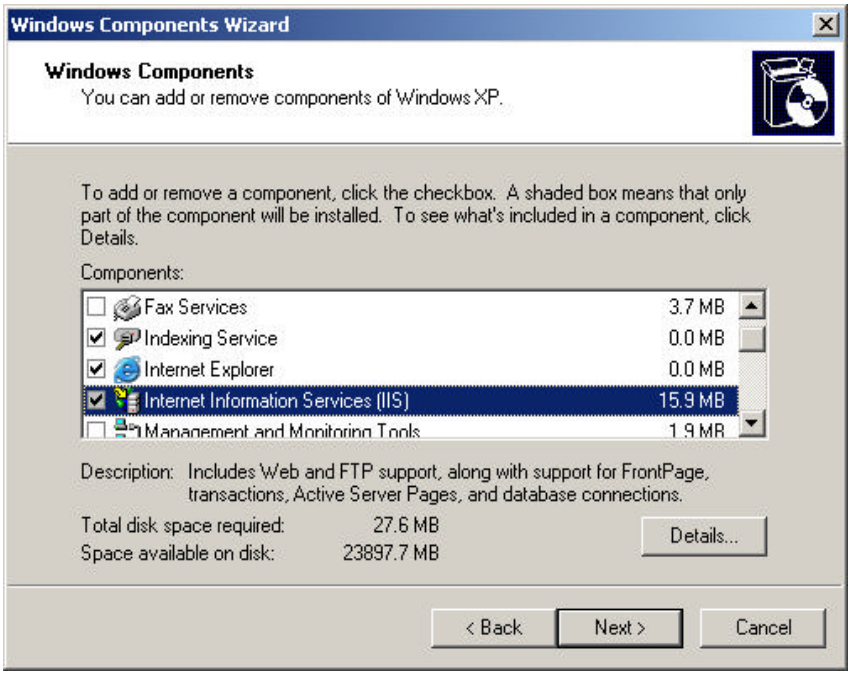
Installasjonsveiledningene under gjelder i hovedsak for Windows XP-baserte systemer, men installasjon på andre systemer avviker ikke stort.

Hvis valget av Web-server falt på Microsoft IIS, les kapittel 6.3.1. Hvis valget falt på Apache, les kapittel 6.3.2.

6.3.1 Installasjon av Microsoft IIS med PHP

IIS installeres via kontrollpanelet i Windows.

1. Klikk Start -> Control Panel.
2. Klikk ”Add or Remove Programs”
3. Klikk “Add/Remove Windows Components” som finnes på den venstre vertikale knapperaden.
4. I vinduet som kommer opp, kryss av for ”Internet Information Services (ISS)” (se figur 6-3).
5. Klikk ”Next”, og følg instruksjonene Windows XP gir.

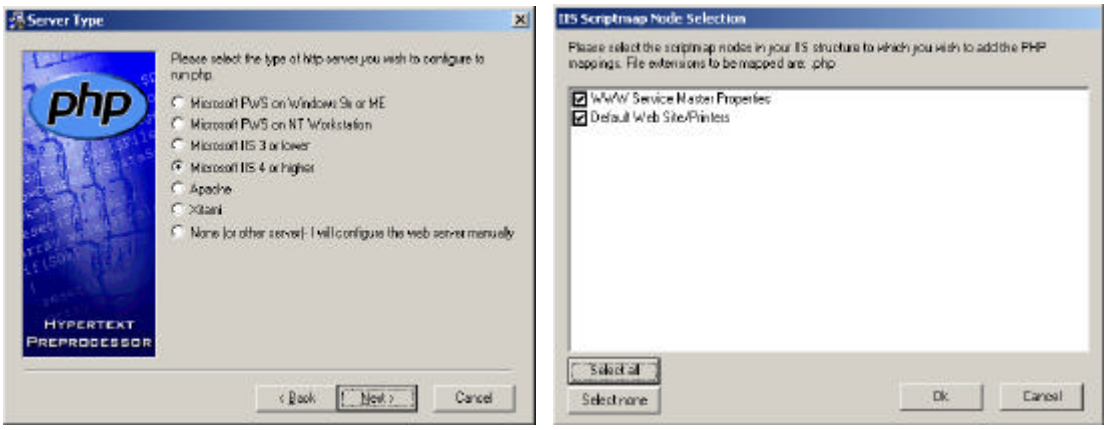


Figur 6-35 - Installasjon av IIS

Etter at Windows har gjort seg ferdig, er Microsoft IIS installert, og det er klart for å installere PHP.

1. Start den kjørbare installasjonsfilen for PHP (php-4.3.x-installer.exe).
2. Du vil bli presentert med velkomstbildet. Klikk ”Next”.
3. Les igjennom lisensen og klikk deretter ”I Agree” hvis du er enig.
4. Velg ”standard” installasjon, klikk ”Next”.

5. Velg katalog og klikk ”Next”.
6. Hvis Web-serveren skal brukes for å generere e-postmeldinger kan det fylles ut med e-postserver og avsenderadresse her. SAFE benytter seg ikke av denne funksjonaliteten, og det er derfor greit å bare klikke ”Next” her.
7. Neste skjermbilde håndterer valg av Web-server. Velg her ”Microsoft IIS 4 or higher” (se figur 6-4, venstre). Klikk “Next”.
8. Klikk “Next” igjen.
9. Et vindu kalt “IIS Scriptmap Node Selection” vil bli åpnet etter at PHP-filene er kopiert (se figur 6-4, høyre). Kryss av for ”WWW Service Master Properties” og eventuelle andre tilgjengelige valgmuligheter og klikk ”OK”.
10. Installasjonen av grunnleggende PHP-funksjonalitet er nå ferdig. Start maskinen på nytt.

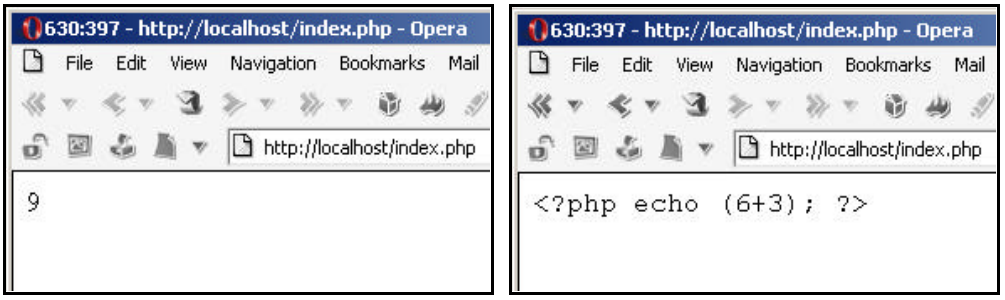


Figur 6-36 - Installasjon av PHP

For å teste om PHP fungerer kan det skrives et lite testskript. I C:\Inetput\wwwroot, opprett en fil kalt ”index.php”. Åpne filen i Notepad eller en lignende editor, og skriv inn for eksempel følgende testskript som skriver ut summen av 6 og 3:

```
<?php echo (6+3); ?>
```

Lagre filen. Åpne deretter en nettleser og naviger til http://localhost/index.php (bruk eventuelt maskinens IP-adresse i stedet for localhost). I nettleservinduet skal det nå kun stå **9** (se figur 6-5, venstre), altså summen av 6 og 3 som vi skrev i testskriptet. Hvis du i nettleseren ser det opprinnelige PHP-skriptet i klartekst (se figur 6-5, høyre), virker ikke PHP-implementasjonen på serveren. Hvis dette er tilfelle, vennligst gå igjennom installasjonsveiledningen og se til at intet er glemt.



Figur 6-37 - Resultat av testskript

Etter at PHP-implementasjonen er funnet i orden, er tiden inne for å installere Microsoft SQL Server-utvidelsen til PHP. Dette er beskrevet i kapittel 6.3.3.

6.3.2 Installasjon av Apache med PHP

Apache kan lastes ned fra `http://httpd.apache.org/download.cgi`. Installasjonsveiledningen beskrevet under går ut i fra at ”Win32 Binary (MSI Installer) – apache_2.0.xx-win32-x86-no_ssl.msi” er lastet ned. Dette er versjon 2 av Apache.

1. Installasjonen startes ved å dobbeltklikke på .msi-filen (apache_2.0.xx-win32-x86-no_ssl.msi).
2. Gå igjennom installasjonen som vanlig inntil du kommer til ”Server Information”. Her skrives inn informasjon angående serverens tilknytning. Det anbefales at ”Run as server for All Users” velges. Klikk ”Next”.
3. Velg ”Complete Install”. Følg deretter installasjonsveiviserens instruksjoner angående plassering av serverens filer, etc.

Apache er nå installert på maskinen. Under følger instruksjoner for å installere PHP på Apache.

Pakk ut zip-filen med PHP (php-4.3.x-Win32.zip) til en midlertidig katalog, som for eksempel C:\tempPHP\. C:\tempPHP\ brukes videre i beskrivelsen. Hvis zip-filen blir pakket ut til en annen katalog, erstatt alle referanser til C:\tempPHP\ med denne. Alle referanser til C:\Windows\system32\ kan være annerledes på systemet som Apache blir installert på (for eksempel hvis en eldre versjon av Windows NT enn Windows XP brukes vil katalogen være C:\WinNT\system32\).

Installasjon av PHP krever en del kopiering av filer:

1. Kopier C:\tempPHP\php-4.3.1-Win32\php4ts.dll til C:\Windows\system32\
2. Kopier C:\tempPHP\php-4.3.1-Win32\sapi\php4apache2.dll til C:\Windows\system32\

Deretter må konfigurasjonsfilen til Apache oppdateres for å implementere PHP.

1. Klikk ”Start” -> ”All Programs” -> ”Apache HTTP Server” -> ”Configure Apache Server” -> ”Edit the Apache httpd.conf Configuration File”
2. Finn listen med ”LoadModule”-direktiver (søk etter ”LoadModule” i filen). Legg til følgende *til slutt* i listen: ”LoadModule php4_module c:/windows/system32/php4apache.dll”. Dette vil føre til at Apache laster PHP neste gang den startes.
3. Søk etter ”AddType” til du finner en liste som inneholder ”AddType application/x-tar .tgz” og ”AddType image/x-icon .ico”. Legg til følgende i listen: ”AddType application/x-httpd-php .php”. Dette gjør at Apache kjenner igjen PHP-dokumenter og vet hva den skal gjøre med dem.
4. Når man først har konfigurasjonsfilen åpen, kan det lønne seg å samtidig sette rotkatalogen, dvs. katalogen som deles på serveren, til noe mer håndterlig enn standardinnstillingen. Søk etter alle referanser til ”C:/Program Files/Apache Group/Apache/htdocs” og bytt ut med for eksempel ”C:/htdocs”.
5. Lagre filen, og lukk Notepad.

Apache er nå satt opp til å bruke PHP. For å starte Apache på nytt klikk ”Start” -> ”All Programs” -> ”Apache HTTP Server” -> ”Control Apache Server” -> ”Restart”.

Etter at PHP-implementasjonen er funnet i orden, er tiden inne for å installere Microsoft SQL Server-utvidelsen til PHP. Dette er beskrevet i kapittel 6.3.3.

6.3.3 Installasjon av MS SQL-utvidelse for PHP

Hvis ikke det er gjort allerede, pakk ut zip-filen med PHP (php-4.3.x-Win32.zip) til en midlertidig katalog, som for eksempel C:\tempPHP\. C:\tempPHP\ brukes videre i beskrivelsen. Hvis zip-filen blir pakket ut til en annen katalog, erstatt alle referanser til C:\tempPHP\ med denne.

Åpne Windows Explorer (evt. Windows Utforsker) og naviger til **C:\tempPHP\php-4.3.x-Win32\extensions**. I denne katalogen vil det finnes en fil som heter **php_mssql.dll**. Denne filen står for kommunikasjonen mellom PHP og Microsoft SQL Server. For å gjøre filen tilgjengelig for PHP må den legges i C:\Windows\system32\, eventuelt C:\WinNT\system32\ avhengig av type operativsystem. Altså; kopier **php_mssql.dll** til **C:\Windows\system32**.

NB! I samme katalog som **php_mssql.dll** ligger også **php_msql.dll**. Se til at den rette filen kopieres (**php_mssql.dll**)!

Deretter må konfigurasjonsinnstillingene til PHP forandres. Disse finnes i filen **php.ini** som ligger i C:\Windows\ eller eventuelt C:\WinNT\. Åpne php.ini i Notepad og søk etter følgende tekststreng:

```
;extension=php_mssql.dll
```

Etter at linjen med tekststrengen er lokalisert, fjern semikolonet. Dette gjør at linjen avkommenteres slik at PHP laster DLL-filen når Web-serveren starter. Den resulterende linjen skal se slik ut:

```
extension=php_mssql.dll
```

Lagre filen og lukk Notepad. Du kan nå slette katalogen C:\tempPHP\. Start deretter maskinen på nytt.

6.4 Installasjon og oppsett av SAFE

6.4.1 Administrasjonsmodulen og registreringsmodulen

På vedlagt CD følger det med installasjonsprogrammer for de ulike SAFE-modulene på flere forskjellige plattformer. Det anbefales å kjøre de medfølgende installasjonsprogrammene på en Windows-basert sentral server. Administrasjonsmodulen og registreringsmodulen installeres da i hver sin katalog på serveren, og disse katalogene mappes opp hos de brukere som skal ha tilgang, slik at de kan starte programmene uten å måtte installere dem lokalt. Denne måten å sette opp og installere SAFE har mange fordeler, blant annet at oppdatering av programvaren vil være meget enkel i fremtiden. Dette detaljeres i kapittel 6.4.3. Utover fordeler med oppdatering, har man også den fordelen at brukerne ikke trenger å ha installert Java Virtual Machine lokalt. Programmet bruker automatisk den versjonen som følger med i SAFEs installasjonsprogrammer.

Det kan være hensiktsmessig å ikke installere administrasjonsmodulen i en delt katalog, da denne kun skal brukes av få personer og har mye spesiell funksjonalitet. Ulempen med dette er da at det vil bli mer arbeid å forandre konfigurasjonsinnstillinger og å oppdatere programvaren.

Installasjonsrutinene for administrasjonsmodulen og registreringsmodulen er meget enkle, da de benytter seg av et standardisert installasjonsprogram av typen InstallAnywhere NOW![14]. Alt man trenger å velge er hvilken katalog det skal installeres til, og hvor snarveier skal plasseres, hvis det i det hele tatt skal lages noen.

Det er viktig at etter installasjon, men før programmene tas i bruk, må SAFE Config (se kapittel 6.4.3.1) kjøres for å konfigurere de forskjellige modulene. Legg merke til at både registreringsmodulen og administrasjonsmodulen krever at presentasjonsmodulen er installert.

For å logge inn på administrasjonsmodulen første gang må brukernavn *admin*, med passord *admin* brukes. Denne brukernavn/passord kombinasjonen ligger som standardverdier i databasen når denne blir opprettet. Av åpenbare, sikkerhetsmessige grunner **må** det i administrasjonsmodulen opprettes en reell administrator før systemet kan tas i bruk. Dette kan gjøres i administrasjonsmodulens brukeradministrering.

6.4.2 Presentasjonsmodulen

Presentasjonsmodulen er en samling PHP-filer som ligger samlet i en zip-fil under katalogen \installasjonsfiler\presentasjonsmodul\ på medfølgende CD. For å installere presentasjonsmodulen trenger man kun å pakke ut disse filene, med katalogstruktur intakt, til en katalog som er delt av en Web-server som er i stand til å prosessere PHP-dokumenter. Se kapittel 6.3 for nærmere beskrivelse av hvordan en Web-server med PHP-støtte settes opp.

Også her er det viktig at SAFE Config kjøres (se kapittel 6.4.3.1) før modulen tas i bruk.

Når det gjelder bruksmetoder for presentasjonsmodulen, anbefales det at et fags presentasjon enten linkes til fra fagets side eller inkorporeres i et fags side ved hjelp av en såkalt ”internal frame”. Se HTML-spesifikasjonen for nærmere informasjon om hvordan internal frames skal brukes.

Her følger en liste over de forskjellige filer som er av nytte i presentasjonsmodulen:

!konfigurasjon.inc.php	- konfigurasjonsinstillinger
!vanligeFunksjoner.inc.php	- vanlige funksjoner som brukes av de andre filene
fagbeskrivelse.php	- for presentasjon av fagbeskrivelser
fagliste.php	- listeoversikt over fag ved HiG
se.php	- brukes av administrasjonsmodulen (se fagbeskrivelser)
sok.php	- søkemotor

[14] http://www.zerog.com/

6.4.3 Tilleggsapplikasjoner

6.4.3.1 SAFE Config

SAFE Config brukes til å konfigurere alle SAFEs moduler. Applikasjonen er i stand til å generere filen konfigurasjonsfilen (safecon.ini) som benyttes av samtlige moduler, og i tillegg Fagimportereren (se kapittel 6.4.3.2), for diverse konfigurasjonsinnstillinger, som for eksempel adresse til databaseserver og brukernavn og passord for denne.

SAFE Config kan installeres på en maskin, for eksempel hos en administrator som setter de riktige innstillingene og genererer konfigurasjonsfilen. Siden konfigurasjonsfilen er lik for alle modulene trengs den kun å lages en gang. Den kan deretter kopieres ut til der de ulike modulene er installert.

6.4.3.2 Fagimporterer

Fagimportereren er en applikasjon som i realiteten kun vil bli brukt en gang. Den er laget for å kunne importere fag fra en MSTAS-generert fil på et gitt format, for så å legge fagene inn i databasen brukt av SAFE. Applikasjonen trenger en konfigurasjonsfil generert av SAFE Config (se kapittel 6.4.3.1) for å operere korrekt.

Applikasjonen kan installeres hos en administrator som har nødvendige filer (MSTAS-data og konfigurasjonsinstillinger) tilgjengelig. Etter bruk kan applikasjonen avinstalleres ved å bruke kontrollpanelets ”Legg til/Fjern programmer”.

6.5 Oppdatering av SAFE

Hvis det noen gang blir aktuelt å gjøre oppdateringer i koden til en av modulene i SAFE slik at man trenger rekompilering, vil man ved å benytte seg av installasjonsmetoden som er skissert i kapittel 6.4.1 lett kunne oppdatere systemet kun ved å bytte ut en enkelt fil (.jar-filen som inneholder alle klassene) som ligger på den sentrale serveren. Alle brukere av systemet vil da kunne ta i bruk den nye versjonen uten å behøve noen form for oppdatering lokalt. Det holder også å erstatte konfigurasjonsfilen som ligger på serveren hvis noen innstillinger forandres.

7 Sluttdiskusjoner

7.1 Diskusjon og drøfting av valg og resultater

7.1.1 Valg av systemutviklingsmodell

Vi utførte i høst et systemutviklingsprosjekt hvor oppgaven var tilnærmet lik som for dette hovedprosjektet, hvor vi var pålagt å benytte RUP, Rational Unified Process. RUP er et meget dokumentasjonsorientert prosessrammeverk, og basert på våre erfaringer fra dette prosjektet følte vi at det ikke var riktig å bruke RUP i et prosjekt som dette hvor det er forholdsvis få involverte. Dette RUP-prosjektet var basert på en visjon hvor store deler av systemet, inkludert registreringsmodulen, var Web-basert, og vi kunne i liten grad benytte det vi fant der for videreføring i hovedprosjektet, utover det at vi fikk satt oss godt inn i den prosessen det er å produsere fagbeskrivelser og en komplett studiehåndbok hvert år.

I begynnelsen av hovedprosjektet så vi fort hvor lett det var å dele opp prosjektet i moduler. Av denne grunn valgte vi til å begynne med en inkrementell utviklingsmodell, hvor man spesifiserer kravene, designer og utvikler hver modul for seg. Man gjør seg altså ferdig med en modul før man begynner på den neste. Det viste seg å være noe naivt å tro at man kan utvikle en modul før den neste, dessuten var det vanskelig å fordele arbeidsoppgaver når vi jobbet kun med en modul av gangen.

Vi bestemte oss for å gå over til en litt mer fri utviklingsmodell, da vi innså at vi måtte utvikle registreringsmodul og administreringsmodul parallelt. Vi valgte å trekke inn mange elementer fra XP, eXtreme Programming, men baserte oss ikke totalt på denne modellen. Et av XPs hovedprinsipper, parprogrammering, har vi hatt stor suksess med. Vi har jobbet i par på de forskjellige modulene. En annen av XPs grunnregler som passet oss godt er ”on-site customer”, altså en oppdragsgiver som er konstant tilgjengelig for spørsmål om implementasjon og funksjonalitet. Vår veileder fungerte i dette prosjektet også på mange måter som oppdragsgiver, og vi hadde løpende kontakt under utviklingen av systemet hvor veileder kom med krav til endringer og ny funksjonalitet.

7.1.2 Resultat

Da vi gikk i gang med prosjektet var vi noe usikre på hvor mye vi ville rekke, og kravspesifikasjonen var på et tidlig stadium av prosjektet mer omfattende enn den etter hvert utviklet seg til å bli. Oppgaven omfattet i begynnelsen elektronisk lagring og generering av hele studiehåndboka, det vil si at i tillegg til fagbeskrivelser omfattet oppgaven også studieplaner og annen generell informasjon som studiehåndboka er bygget opp av. Etter drøftinger med oppdragsgiver og veileder, kom vi fram til at vi i første rekke skulle fokusere på håndtering av fagbeskrivelser. Her sto korrekt lagring, arkivering og presentasjon i fokus, og den nåværende kravspesifikasjonen reflekterer dette. Vi er fornøyd med at vi valgte å prioritere fagbeskrivelsene, i stedet for å angripe hele problemområdet. Vi tror at dette ville ført til at vi ikke ville ha rukket å lage en fullgod løsning som håndterte alle sider ved fagbeskrivelser, studieplaner og studiehåndboken generelt. Ved å fokusere på fagbeskrivelser har vi rukket å gjennomføre det vi satte oss fore, og vi har levert et godt produkt.

Det ferdige systemet implementerer alt som kravspesifikasjonen detaljerer, i tillegg til en del funksjonalitet som verken vi eller veileder og oppdragsgiver hadde avdekket da vi skrev denne. Vi kan med sikkerhet si at vi kom i mål.

Mye av den nye funksjonaliteten som ble innført underveis ble avdekket under implementasjon av de allerede fastsatte kravene. Vi så da hvilke muligheter som lå i systemet for å gjøre det til et mer effektivt saksbehandlingsverktøy for fagbeskrivelser. Et eksempel på slik funksjonalitet er hvordan registreringsmodulen merker mangelfulle fagbeskrivelser hvorpå administrasjonsmodulen viser disse på en slik måte at administrator kan få en god oversikt over mangelfulle fagbeskrivelser og eventuelt varsle ansvarlige faglærere.

Systemet slik det framstår i dag er det et utrolig dynamisk og tilpasningsdyktig system for oppsett og registrering av fagbeskrivelser, men også andre situasjoner der man ønsker en korrekt registrering av data ved hjelp av en mal. Med andre ord, man kan med kun kosmetiske endringer i administrasjonsmodulen og muligens noen nye feltklasser i de andre modulene, lage maler for helt andre ting enn fagbeskrivelser, for eksempel spørreundersøkelser.

Vi føler vi har oppnådd gode resultater under dette prosjektet. Vi har laget et produkt som vi er godt fornøyd med, og som vi har tro på at er kvalitetsmessig godt nok til daglig bruk ved Høgskolen i Gjøvik. Etter å ha demonstrert systemet for studieadministrasjonen ved skolen fikk vi gode tilbakemeldinger, og det er snakk om å ta i bruk SAFE allerede kommende sommer, noe som vi tar som en indikasjon på at også andre er fornøyd med hva prosjektet har resultert i.

7.2 Kritikk av oppgaven

Oppgaven var gitt med navnet ”Elektronisk studiehåndbok – produksjon, bruk og arkivering”. Etter vår mening var denne alt for vid, spesielt med tanke på at det var forventet at systemet skulle kunne produsere både en elektronisk versjon og en papirversjon av hele studiehåndboka, komplett med fagbeskrivelser, studieplaner og alt annet som studiehåndboka innebefatter. Da skolen er i en overgangsfase til den nye Kvalitetsreformen, i tillegg til at det er snakk om et Innlandsuniversitet i samarbeid med Høgskolen i Lillehammer og Høgskolen i Hedmark, er det også store forandringer på gang når det gjelder både fagbeskrivelser og studieplaner. Slik det ser ut nå, ville det vært umulig å lage et system for studieplaner som skolen fortsatt kunne bruke om et års tid.

Bortsett fra fokuset på studiehåndboka har oppgaven vært god, med nok av utfordringer for oss å ta tak i.

7.3 Videre arbeid

En mulig utvidelse av systemet er å implementere støtte for registrering og generering av studieplaner og all annen informasjon som man forventer å finne i studiehåndboka, for eksportering både til Web og filformater. Herfra er ikke veien lang til et komplett saksbehandlingsverktøy for hele prosessen med å sette sammen den komplette studiehåndboka både elektronisk og i papirformat.

Videre er vårt system også designet med tanke på modularitet, slik at man enkelt kan legge til nye typer felter om dette er ønskelig fra skolens side.

7.4 Evaluering av eget arbeid

7.4.1 Innledning

Da gruppens medlemmer allerede hadde jobbet med andre prosjekter sammen, og derfor hadde kjennskap til den enkeltes kunnskaper, ferdigheter og begrensninger, ble arbeidet med inndeling av oppgaver, ansvar og organisering en forholdsvis kort prosess som senere viste seg meget tilfredsstillende og effektivt.

7.4.2 Organisering av gruppen

Det har innad i gruppen vært lett å ha en flat organisasjonsstruktur hvor alle har vært med på å bestemme og velge arbeidsoppgaver. Ingen har vært en klar sjefsperson, men gruppens leder har i hovedsak fungert som kontaktperson og gruppens ansikt utad. De gangene gruppen har stagnert og diskutert forskjellige valg som måtte tas, har gruppens leder hatt det endelige ansvaret for å ta de avgjørelser som var nødvendige før for mye tid hadde gått tapt på uproduktiv diskusjon rundt disse valgene. Andre roller som spesielt har falt på gruppens leder har vært å passe på kontinuerlig fremgang og at tidsrammene har blitt overholdt.

Alle gruppemedlemmene har hatt et felles ansvar for prosjektets fremgang, og alle viktige avgjørelser ble diskutert samlet.

7.4.3 Arbeidsfordeling

Da vi fulgte prinsippet med parprogrammering var det ikke vanskelig å fordele arbeidsmengden innad i gruppen. To gruppemedlemmer hadde hovedansvaret for administrasjonsmodulen, mens de to andre hadde hovedansvaret for registreringsmodulen, eksporteringsmodulen og presentasjonsmodulen. Med denne fordelingen ble arbeidsmengden på hvert gruppemedlem forholdsvis lik, og alle visste grovt sett hva sine arbeidsoppgaver var til enhver tid. Vi er godt fornøyd med denne arbeidsfordelingen.

Andre oppgaver utover utvikling av applikasjonen, slik som skriving av statusrapporter, møtereferater og lignende ble gjort av den eller de som hadde tid og anledning.

7.4.4 Prosjekt som arbeidsform

Prosjekter er en arbeidsform som blir hyppig brukt i næringslivet, og i så henseende har hovedprosjektet vært god og verdifull erfaring som det første skikkelige prosjektet vi har gjennomført som studenter.

Det å jobbe sammen i et prosjekt gjør at man lærer mye om sine medarbeidere, og man lærer også at det er viktig å ha en god tone prosjektmedlemmene i mellom. Uten en god tone prosjektmedlemmene i mellom kunne raskt situasjoner oppstå som hadde låst hele fremgangen i prosjektet, og det å nå et endelig vellykket mål kunne blitt vanskelig.

Når man arbeider sammen i et prosjekt er det naturlig at de forskjellige prosjektdeltakerne har forskjellig kompetanse, kunnskaper og bakgrunn fra før prosjektets start. Det er da viktig at prosjektgruppen som helhet kartlegger hver deltakers spesialfelt eller interesseområde slik at gruppens brede kompetanse blir videreført til systemet, som deretter vil utvikles på best mulig måte. Vi føler at hver prosjektdeltaker har bidratt med sine spesielle kompetanser og kvaliteter, noe som har vært med på å heve kvaliteten på sluttproduktet.

Den største fordelen med prosjekt som arbeidsform er kanskje at man innad i gruppen kan velge sine egne arbeidstider. Arbeidsdagen har som hovedregel vært hver hverdag fra 09.00 til 16.00, men gruppemedlemmene kan seg i mellom avtale andre tider uten at det påvirker utviklingens framdrift. Dette har vist seg å fungere bra igjennom hele prosjektperioden. Hver prosjektdeltaker har tatt ansvar for at nok tid har blitt avsatt til prosjektet slik at målet har blitt oppnådd.

7.4.5 Subjektiv opplevelse av hovedprosjektet

Det har vært verdifullt for oss å jobbe i en slik prosjektform som hovedprosjektet er. Vi har gjennom prosjektet fått oppleve mange av de situasjoner som kan oppstå når vi begynner å jobbe i forskjellige bedrifter. Vi måtte planlegge, se fremover, og jobbe mot en absolutt avsatt dato. Dette satte krav til oss hele tiden, og aspekter som tidspress, arbeidsfordelig, og tidsbruk har vært faktorer som vi har kommet borti og har måttet takle. Dette tar vi med oss som positive erfaringer videre ut i arbeidslivet.

En annen ting som var spesielt i dette halve året med hovedprosjekt er at skoledagen har vært annerledes. I stedet for å tilbringe lange dager i forskjellige forelesningssaler har vi sittet på et eget tildelt grupperom fra HiG og styrt arbeidsdagen våres selv. Dette har fungert meget bra og vært en viktig trivselsfaktor for gruppen.

Samarbeidet med veileder og oppdragsgiver har fungert meget bra. I startfasen av hovedprosjektet hadde vi mest kontakt med oppdragsgiver for å få på plass hva som var ønsket resultat av dette hovedprosjektet. Etter hvert som vi kom i gang med prosjektet gikk vi over til å ha mer kontakt med veileder. Siden veileder kommer til å være fremtidig bruker av dette systemet og har mye kunnskap om fagbeskrivelser, var det naturlig at han fikk en større rolle i systemutviklingsfasen enn bare veileder. Videre gikk derfor veileder over til å være mer enn bare veileder, men også en representant for oppdragsgiver. Derfor har han i stor grad vært med på å utforme systemet. Denne rollen som veileder har spilt for oss har fungert veldig bra, og bidratt til ytterligere bedring av kvaliteten til det endelige produktet.

Samarbeidet oss imellom på gruppa har fungert veldig bra. Alle har bidratt til å lage en god stemning og en god tone innad i gruppen. Dette har vært veldig viktig da vi har sett hverandre veldig mye i denne prosjektperioden. Erfaringsmessig er dette også en god erfaring for oss å ta med videre inn i arbeidslivet siden man her vil over lengre perioder arbeide sammen med de samme menneskene. Altså har vi lært at et godt samarbeid er viktig i en prosjektgruppe for å komme frem til et godt resultat.

Alt i alt kan vi si at arbeidet med hovedprosjektet har vært en krevende men lærerik prosess. Til tider har det nesten føltes ut som arbeidsmengden har vært uoverkommelig, og vi har hele tiden følt at vi har måttet jobbe veldig mye for å få et ferdig og godt resultat. Dette har ført til at vi har jobbet mye utover det som opprinnelig var planlagt, så det har blitt mange sene kvelder med prosjektjobbing, brus og pizza. Tilgjengjeld har vi tilegnet oss mye bred kunnskap, som det å sette opp gode arbeidsplaner, til det å få gode rutiner rundt implementering og programutvikling.

8 Konklusjon

Det er blitt sagt at smidige eller tilpasningsdyktige utviklingsmodeller som for eksempel extreme programming er lite seriøst og vil i de fleste tilfeller feile. Vi mener å ha bevist en gang for alle at også slike utviklingsmodeller kan føre til tilfredsstillende produkter som ivaretar brukernes krav til grafisk brukergrensesnitt og funksjonalitet, men samtidig også oppfyller krav til robusthet. Selv om vi har benyttet oss av en meget fri form for XP, har resultatet blitt et produkt som svarer til brukernes og ikke minst våre forventninger.

Oppdragsgiver ville ha et system som kunne håndtere fagbeskrivelser på HiG og som gjorde disse lett tilgjenglig på Web. Det var et viktig krav at systemet i stor grad skulle tvinge brukere som registrerer fagbeskrivelser til å fylle ut fagbeskrivelser på en konsekvent måte. Under utviklingsarbeidet kom det fram punkter og funksjonalitet som ikke var påtenkt, men som ville vesentlig forbedre brukervennligheten av systemet og som ville lette arbeidet med oppretting, redigering og administrering av fagbeskrivelser. Da vi, som tidligere nevnt, benyttet oss av en ”agile” utviklingsmodell, og fordi gruppen generelt hadde en åpen holdning til endringer som oppsto underveis, ble alle finesser implementert i systemet. Dette førte til mange ekstra arbeidstimer og noe restrukturering av fremgangsplanene, men da det var et høyt prioritert mål for gruppen å utvikle et produkt som tilfredsstiller arbeidsgiver på en best mulig måte, var det aldri snakk om å sløyfe tilleggsfunksjonalitet som kom til under utviklingsarbeidet. For eksempel under administrasjonsmodulen finnes det funksjonalitet for oppsett av avhengige valg, redigering av fritekst i fagbeskrivelser og visning av mangler i fagbeskrivelser. Alle disse er tilleggsfunksjonalitet og finesser som først ble oppdaget under utviklingsarbeidet og som dermed ble implementert uten å finnes i kravspesifikasjonen. Alle disse er funksjonalitet som er meget komplekse og som krevde mye arbeid og implementasjon. Også hele eksporteringsmodulen ble lenge vurdert sløyfet fra arbeidsgivers side, men denne ble altså av likevel implementert. Slike endringer som kom underveis har uten tvil satt sine spor i fremdriftsplanene, da disse er radikalt endret. Men effektive restruktureringer av planer har ført til forbedrede og mer tilfredsstillende resultater. Her skal det også sies at vår fremgangsmåte ikke ville fungert, dersom ikke gruppens medlemmer hadde vært tilpasningsdyktige med tanke på omlegging og restrukturering av sine private planer og liv for øvrig. Gjensidig respekt, omtanke og tillit innad i gruppen gjorde det meget enkelt å jobbe med dette prosjektet da ingen personlige konflikter og uenigheter oppsto under utviklingsarbeidet, noe som er relativt vanlig der mennesker jobber såpass tett sammen som vi har gjort under dette prosjektet. Ikke bare kunnskaper tar vi med oss videre i livet, men også uvurderlige minner.