HOVEDPROSJEKT:

F@T

FORFATTER(E): ERIK MELLEM, RUNE HATLELID,
KRISTIAN G. ANDREASSEN OG
PÅL ERIK ENG

Dato: 23.mai 2002

Tittel:	F@T – Fastpris @ Team G.E.D.	Nr.: 11
	F@T – Fixed price @ Team G.E.D.	Dato: 23.05.02
Deltakere:	Erik Mellem	
	Rune Hatlelid	
	Pål Erik Eng	
	Kristian G. Andreassen	
Veileder:	Harald Liodden	
Oppdragsgiver:	Team G.E.D. A/S	
Kontaktperson:	Frode Larsen, Kim Sønes	
Stikkord (4 stk)	Fastpris, Java, PHP, UP	
Antall sider: 119	Antall bilag: 8	Tilgjengelighet (åpen/konfidensiell): Åpen

Kort beskrivelse av hovedprosjektet:

Hovedprosjektet hadde til formål å analysere, systemere, designe, kode og teste et system som skal forenkle arbeidet med å fastsette korrekt pris på ulike bilservicer. Hensikten var å finne en fastpris på en gitt service, med utgangspunkt i registreringsnummer og type service. Systemet skal være tilgjengelig for både privat kunder og verksteder som er underordnet Team G.E.D. A/S, og vil være tilgjengelig på web i separate sider for disse to ulike aktørene. En administrator del vil være ansvarlig for vedlikehold av systemet, herunder blant annet innlesing av data, endring av data og oppsett av database.

Den delen av systemet som er tilgjengelig på web er kodet i PHP. Enkel GUI er vektlagt, slik at web siden skal være brukervennlig. Kundedelen gir mulighet for å sende direkte forespørsel på service til de enkelte verkstedene, mens verksted delen vil inneholde en komplett liste over deler som er tilknyttet de ulike servicene.

Administrator programmet er kodet i Java. Dette er et program som kun skal foreligge hos Team G.E.D. A/S, der administreringen av systemet skal foregå. Det er også her fokusert på enkel GUI.

Systemet jobber mot en database som organiserer alle dataene. Databasen vil kunne inneholde rundt 1,5 millioner registreringsnummer, samt nødvendig informasjon om disse ettersom dataene blir innlest.

Forord

Prosjektet F@T ble påbegynt i januar 2002 av Pål Erik Eng, Rune Hatlelid, Erik Mellem og Kristian G. Andreassen. Problemområdet rundt F@T var allerede kjent fra et tidligere prosjekt, som tre av medlemmene på gruppen hadde gjennomført. Derfor ble valget av F@T ganske naturlig, da prosjektet ble lansert som hovedprosjekt ved Høgskolen i Gjøvik sin dataingeniørlinje.

F@T er et pilotprosjekt hos Team G.E.D A/S. Siden Team G.E.D A/S ikke er noen IT- bedrift, stilte dette ekstra krav til gruppemedlemmene i den praktiske gjennomføringen. Dette var enutfordring gruppen så på som meget interessant.

I gjennomføringen av prosjektet har gruppen hatt hjelp av flere personer og vil derfor takke Harald Liodden for god veiledning, Øyvind Kolloen for faglig assistanse innen Java og PHP og hovedprosjekt gruppen HEVN-J for et godt faglig og sosialt samarbeid. I tilegg vil gruppen takke Frode Larsen og Kim Sønes hos Team G.E.D A/S for et godt samarbeid.

Gjøvik den 23.05.2002

Pål Erik Eng	Erik Mellem
Rune Hatlelid	Kristian G Andreassen

1 Innholdsfortegnelse

1 Innledning 4

1.1 Problemområde 4

1.2 Målgruppe 4

1.3 Bakgrunn og kompetanse 4

1.4 Arbeidsform 5

1.5 Original kravspesifikasjon 5

1.6 Ord og uttrykk 6

2 Systemutvikling 7

2.1 Utviklingsmodell 7

2.1.1 Valg av utviklingsmodell 7

2.1.2 Om Unified Process 7

2.1.2.1 Om fasene i Unified Process 8

2.1.3 Om Unified Modelling Language 9

2.1.4 Forventning til gjennomføring 9

2.2 Valg av verktøy 9

2.3 Gjennomføring av systemutvikling 10

2.3.1 Inception 10

2.3.2 Elaboration 6

2.3.3 Construction 10

2.3.4 Transition 12

3 Design 13

3.1 Systemarkitektur 13

3.1.1 Web delen 13

3.1.2 Applikasjonen 14

3.1.3 Databasestruktur 15

3.2 Layout 17

3.3 Programmering av design F@T 17

3.4 Programmering av design på F@T- admin 20

3.4.1 Beskrivelse av de forskjellige komponentene i F@T- admin 21

3.4.1.1 Les inn filer 21

3.4.1.2 Rediger Servicer 22

3.4.1.3 Olje- og Timepris 24

3.4.1.4 Verksteder 25

3.4.1.5 Brukere 25

3.4.1.6 Feillogg 25

3.4.1.7 DBSetup 26

4 Implementering 27

4.1 Utviklingsmiljø 27
4.1.1 Serveroppsett 27

4.2 Valg av verktøy 27
4.2.1 Programmeringsspråk 27
4.2.1.1 F@T 27
4.2.1.2 F@T- admin 28
4.2.1.3 Database 28
4.2.2 Installasjonsprogram 29

4.3 Prinsipper ved koding 30
4.3.1 Kodingen av F@T 30
4.3.1.1 Generelt 30
4.3.1.2 Utskrift fra programmet 30
4.3.1.3 Navn på variable 31
4.3.1.4 Globale variable 31
4.3.1.5 Kommentering 31
4.3.1.6 Gjenbruk av kode 32
4.3.2 Koding av F@T- admin 32
4.3.2.1 Globale variable 32
4.3.2.2 Gjenbruk av kode 32
4.3.2.3 Navn på variable 32
4.3.2.4 Kommentarer 32
4.3.2.5 Prinsipper ved koding 33
4.3.2.6 Oppsummering 33

5 Kvalitetsikring 34

5.1 Sikre kvalitet på prosessen 34

5.2 Sikre kvalitet på produktet 34

6 Testing 35

7 Analyse av prosjektet 36

7.1 Prosessen 36

7.2 Produktet 36

7.3 Forbedringer/Videreutvikling 37

7.4 Gruppearbeidet 38

7.5 Oppdragsgiver 39

7.6 Veileder 39

8 Konklusjon 40

9 Referanser 41

10 Vedlegg 42

2 Innledning

2.1 Problemområde

BOSCH kjeden består av et mangfold ulike bilverksteder. For disse verkstedene er det viktig å minimere tiden de ansatte bruker på kontoret, slik at mer tid kan benyttes i verkstedet. En periodisk service på en bil innebærer at bestemte deler skal skiftes. De fleste verksteder har i dag problemer med å gi potensielle kunder den rette prisen ved en prisforespørsel på en slik service.

Det finnes i dag to måter man kan gi svar på en prisforespørsel. Den ene er å gjøre en mengde oppslag i diverse kataloger og lister. Dette innebærer som regel at det tar lang tid før kunden får svar på sin forespørsel. Noe som igjen kan føre til at kunden velger et annet verksted med kortere responstid. Et svar på en prisforespørsel kan også gjennomføres "på sparket", man kan antyde en pris med utgangspunkt i prisnivået man har benyttet for samme service på en tilsvarende bil. Denne metoden fører ofte til at man oppgir en pris som er langt fra det reelle, noe som enten kan resultere i at prisen blir for lav som videre vil gi problemer tilknyttet oppgjøret, eller en for høy pris som fører til at man mister kunder. Metoden vil derfor også gi rom for at ulike personer får forskjellige priser på eksakt samme service.

Dette har ført til at Team G.E.D. A/S ønsker i gi sine verksteder system som:

- Gir riktig pris.
- Gir en umiddelbar pris
- Gir samme pris til alle

2.2 Målgruppe

Systemer er fortrinnsvis utviklet for kundebehandlere på verksteder under BOSCH kjeden. Dette er ofte personer som ikke besitter utstrakt datakompetanse, kjennskapen til data begrenser seg da ofte til Windows grensesnitt, med mer eller mindre standard Microsoft programmer. Den gir også privatpersoner anledning til å undersøke servicepriser via Internett.

Selve rapporten er hovedsakelig utarbeidet for de som skal evaluere vårt prosjektarbeid. Den vil også kunne være av interesse for andre med relevant databakgrunn, ikke minst studenter som eventuelt kunne tenke seg å videreutvikle prosjektet vårt ved en senere anledning.

2.3 Bakgrunn og kompetanse

Prosjektgruppen består av 4 personer med noe ulik bakgrunn innen utdannelse.

Bakgrunn før HiG
Erik Mellem : Allmennfaglig bakgrunn
Rune Hatlelid : Yrkesskole elektro, fagskole
Kristian G. Andreassen : Allmennfaglig bakgrunn
Pål Erik Eng : Yrkesskole elektronikk, forkurs for ingeniør

Utdannelsen fra HiG er nokså lik, der alle er i ferd med å avslutte dataingeniør studiet. Erik har fordypet seg i drift av flerbrukersystemer, mens de øvrige har valgt spesialisering innen systemutvikling.

Gruppen er dermed sammensatt av personer med noe forskjellig kompetanse og bakgrunn innen ulike fagområder. Oppgavene har blitt fordelt mellom de ulike gruppemedlemmene på en fornuftig måte med hensyn til kompetanse og interesseområde. Det har derimot vært et godt samspill mellom medlemmene slik at alle til enhver tid har vært oppdatert på oppgavens fremdrift, og man har tilrettelagt for innspill fra de øvrige gruppemedlemmene.

2.4 Arbeidsform

De enkelte gruppemedlemmene har vært tildelt konkrete ansvarsområder som gruppeleder, sekretær, ansvarlig for utviklingsmiljøet og bedriftskontakt, dette for å sikre at alle oppgavene blir utført etter beste hensikt. Gruppelederen har det overordnede ansvaret for fremdriften, samt kontakten mellom HiG og prosjektgruppen. Sekretærens oppgaver er å føre loggbok og skrive referater fra møter med oppdragsgiver og veileder. Ansvarlig for utviklingsmiljøet skal drifte servere og fordele dataressurser. Bedriftskontakten har ansvaret for å opprettholde god kontakt med oppdragsgiveren og avtale møter ved behov.

2.5 Original kravspesifikasjon

I starten av prosjektet la oppdragsgiver frem en del overordnede krav om systemet skulle oppfylle. Kraven gikk på det meste ut på hva systemet skal gjøre, men det var også et krav at systemet skulle kjøres på en Windows plattform.
Krav til systemet :

- Gi den rette prisen på en bestemt service
- Gi samme pris for lik service for samme biltype
- Systemet skal til slutt vise verkstedet:
    - Servicens tidsforbruk
    - Servicens deleforbruk
    - Total pris
- Systemet skal inneholde en vedlikeholdsdel der en administrator enkelt kan vedlikeholde systemet.
- Gi muligheten for kunder å finne ut prisen på periodiske servicer via Internett.
- Mulighet for utvidelse av systemet mot andre typer service
- Generere deleliste av servicene

2.6 Ord og uttrykk

Aktør	En eller noe som opererer på systemet.
Applikasjon	Dataprogram
Echo	Kommando i PHP som kan brukes til å skrive HTML til skjerm.
Form	Skjema i HTML som bruker for å få sendt informasjon.
Foretningsregler	Regler for hvordan spesielle ting skal behandles
GUI	Grafisk Brukergrensesnitt, utforming av presentasjon, Bevisste valg som er blitt gjort for å utorme og presentere sidene for brukere.
Hidden attributt	Felt i et HTML skjema som holdes skjult for brukeren, men som blir sent med på lik linje som en synlig attributt.
Iterasjon	Kort periode. I dette prosjektet en til to uker.
Kravspesifikasjon	Detaljer beskrivelse av hva systemet skal inneholde av funksjonalitet, hva systemet skal løse og på hvilken måte.
MySQL	Et spesifikt databasesystem som støtter SQL
PHP	Programmeringsspråk for utvikling av dynamiske websider.
Patterns	Kjente problemstillinger ved ansvarsforholdet i mellom dataklasser som gir kjente løsninger.
Submit	Kommando som sender skjemaer i HTML.
SQL	Standard Query Language, standardisert språk for å kommunisere med en database.
Supplementary Specification	Kravspesifikasjon med krav som ikke dekkes av Use Case.
UML	Unified Modeling Language.
UP	Unified Process
Use Case	Historie som beskriver noe systemet skal utføre for en aktør
VM	Virtual Machine, et ”program” som Java kjører på.

3 Systemutvikling

3.1 Utviklingsmodell

3.1.1 Valg av utviklingsmodell

Unified Process, heretter kalt UP er utviklingsmodellen som benyttes i prosjektet. Fordelene med UP er at det er en iterativ utviklingsmodell, en iterasjon er en kort periode. Iterasjonene er til stor hjelp for å bevare framdriften gjennom hele prosjektet. Erfaring fra tidligere prosjekter viser at startfasen i slike prosjekter kan synes ueffektiv, da rammeverket for oppgaven ikke er fullstendig fastsatt og mye tid blir benyttet til ustrukturerte arbeid. Det synes derfor som en motivasjonsfaktor at UP setter kortsiktige mål, i prosessen mot å nå det endelige målet.

UPs evne til å tilpasse seg forandringer taler også til fordel for denne utviklingsmodellen. Siden verken prosjektgruppen eller oppdragsgiver har noen videre erfaring med dataprosjekter av slikt omfang, forventes en del tilpasninger i rammebetingelsene underveis i prosessen. Det forventes også endringer med bakgrunn i oppgavens noe uspesifiserte form, som byr på flere ubesvarte spørsmål vedrørende funksjonalitet. Gruppens kjennskap til verkstedsbransjen er forholdsvis liten, slik at en god dialog er nødvendig for å oppklare eventuelle misforståelser som høyst sannsynlig vil oppstå underveis i prosessen.

I tillegg til UP blir verktøyene Use Case og Unified Modelling Language (UML) benyttet som elementer i UP. Use Case benyttes for å angi hva systemet skal utføre, dette foregår slik at både oppdragsgiver og utviklere forstår innholdet i prosessen. UML benyttes for å modellere systemet på en objektorientert form. Det er også valgt å inkludere dokumenter og fremgangsmåte, da disse er spesielle for RUP.

Som et tilskudd til UP ble besluttet å ta med elementene parprogrammering og såkalte stå opp møter [1], fra eXtream Programming (XP). Stå opp møter vil si et raskt møte hver morgen angående dagens gjøremål. Disse elementene inkluderes for å teste ut hvordan de lar seg gjennomføre i et UP prosjekt.

3.1.2 Om Unified Process

UP er en iterativ utviklingsmodell, som er utviklet av svensken Ivar Jacobson i 1987. Han utviklet den objekt orienterte prosessen, som resulterte i boken, *The Unified Software Developement Process*, som han skrev i samarbeid med Grady Booch og James Rumbaugh. Modellen kombinerer de beste egenskapene til andre generelt aksepterte utviklings modeller, til en bedre og mer effektiv utviklingsmodell. Den krever ikke at det skal brukes metoder som er uegnet for det enkelte prosjekt. Modellen har blitt stadig optimalisert gjennom årene av forskjellige firmaer, spesielt av Rational, som har laget sine egne modeller som bygger på UP. Disse modellene blir kalt RUP og er lagt opp til at prosjektene kan benytte seg av seks såkalte *best practice* [3].

Ved å utvikle i samsvar med UP deles prosjektperioden inn i de fire hovedfasene inception, elaboration, construction og transition, hvor hver overgang markeres med en milepæl. I motsetning til for eksempel fossefallsmodellen drar man ikke hele prosjektet tilbake til forrige fase, hvis en del av prosjektet må gjøres om igjen. Ved å benytte seg av Use Case deles oppgaven opp i flere mindre moduler. Et Use Case er en historie som beskriver en handling en bruker vil at systemet skal utføre. Ved å dele systemet opp i Use Case vil endringer i betingelsene nødvendigvis ikke påvirke alle deler av systemet, slik at kun de berørte delene må gjøres om igjen. Hver fase er igjen delt inn i iterasjoner, som er korte perioder som tar for seg ett eller flere Use Case.

Et viktig element i UP er iterasjoner. Dette er korte perioder, helst på en til to uker. En iterasjon tar for seg noen utvalgte oppgaver som er forhånds estimert. UP presiserer at det viktigste med en iterasjon er overholdelse av tidsfristen. I stedet for å overskride tidsfristen fjernes oppgaver fra iterasjonen. Denne harde politikken på tidsfrister kalles *timeboxing*.

Utvelgelse av oppgaver til en iterasjon skjer gjennom en risikovurdering der de mest risikofylte oppgavene blir valgt først. Dette medfører ikke at de mest risikofylte oppgavene blir valgt slavisk, men det blir gjort et utvalg av oppgaver med både høy og lav risiko til en iterasjon er fylt opp med de ressursene som til enhver tid er tilgjengelige. Risikoen til en oppgave beregnes ut fra forhåndsbestemte kriterier. Poenget med risikovurdering er at de oppgavene som er mest kritiske for prosjektets suksess håndteres først. Det er dette som gjør UP til en risikoorientert utviklingsprosess.

3.1.2.1 Om fasene i Unified Process

I inception fasen setter man seg inn i problemområdet og undersøker dette, for å kunne kartlegge Use Case og andre krav til systemet. Dette vil skje gjennom nær dialog med oppdragsgiver, der begge parter utveksler ideer. Det gjennomføres ikke ytterligere undersøkelser utover det som er avgjørende for å kunne fortsette videre i prosjektet. Det er planlagt å bruke omtrent to uker på denne fasen.

Videre går prosjektet over i elaboration fasen. I denne fasen undersøkes Use Casene med høyest risiko ytterligere, systemets skjellet designes og de fleste krav identifiseres. Deretter velges utviklingsspråk og utviklingsverktøy testes. Avslutningsvis i denne fasen vil man oppnå en god oversikt, slik at den gjenstående tiden på prosjektet kan estimeres. Elaboration fasen vil vare i omtrent to uker.

Construction er den fasen hvor det er viet mest tid. Her bygges resten av systemet, man kan si at det legges kjøtt på skjellettet som opprettes i inception fasen. Under denne fasen utføres *alpha* testing, det vil si at man tester de delene man utvikler fortløpende i slutten av hver iterasjon. Klargjøring for *beta* testing er også en aktivitet i construction. Varigheten av denne fasen vil være omtrent to måneder.

Til slutt er det transition fasen. Dette er fasen der systemet skal overføres til oppdragsgiver. På dette tidspunktet er systemet nærmest ferdigstilt. Det

gjennomføres *beta* testing av systemet. Som resultat av testingen vil det eventuelt bli gitt tilbakemeldinger på forhold som må rettes eller finjusteres. I denne sammenhengen blir det gitt undervisning i bruk av systemet.

For en grafisk tidsfordeling av fasene se GANTT-skjema, Vedlegg A, som viser planlagt gjennomføring av hovedprosjektet.

3.1.3 Om Unified Modelling Language

Unified Modelling Language (UML) er en teknikk som oppretter et rammeverk for et datasystem. Som Objekt Management Group sier på sine egne nettsider [4] :

"Modellering er design av applikasjonen før koding."

Ved modellering fremkommer det objekter som håndterer systemets funksjoner på riktig måte. UML er et verktøy for å spesifisere, visualisere og modulere et datasystem. UML passer for å modellere systemer som støtter alle type programmeringsspråk som bygger på objekter og funksjoner.

3.1.4 Forventning til gjennomføring

Gruppens erfaring fra prosjekter av denne størrelsen er begrenset. Tidligere har medlemmene kun vært igjennom prosjekter med kortere varighet. Enkelte gruppemedlemmer har tidligere arbeidet med UP, men erfaringen er begrenset da dette har vært prosjekter av mindre omfang.

Det er ønskelig med jevnlig kontakt mellom oppdragsgiver og prosjektgruppen, for at de potensielle brukerne skal inkluderes i avgjørende beslutninger.

Det forventes at inception fasen bringer god innsikt til problemområdet og bidrar med kjennskap til ord og uttrykk innen verkstedsnæringen. Her skal de fleste Use Case spesifiseres, men nye Use Case eller features kan også komme til ettervert. Det vil også bli brukt noe tid til å sette opp et utviklingsmiljø i denne fasen.

Under elaboration fasen skal de viktigste Use Casene undersøkes og påbegynnes, og en databasestruktur skal etableres. Det vil også benyttes tid på valg og testing av utviklingsverktøy.

I construction fasen vil selve utviklingen gjennomføres, og her er det total fokus på produktet. I slutten av denne fasen vil databasestrukturen være endelig, og systemet klargjort for testing hos oppdragsgiver.

Transition fasen benyttes for å rette eventuelle feil og optimalisere systemet. Det vil også bli brukt noe tid for å ferdigstille prosjektrapporten.

3.2 Valg av verktøy

På grunn av begrensede økonomiske ressurser var det ikke mulig å anskaffe optimale hjelpeverktøy til UML modellering eller gjennomføring av UP. Markedet tilbyr enkelte kostnadsfrie alternativer men da disse ikke regnes som gode løsninger ble det besluttet å ikke benytte tid på utforskning og opplæring av disse. Dette førte til at det ble besluttet å gjøre jobben manuelt.

3.3 Gjennomføring av systemutvikling

3.3.1 Inception

Målet med denne fasen var å sette seg inn i problemområdet, samt identifisere de viktigste oppgavene til systemet. Det ble bestemt å utvikle systemet i to deler, en web- og en administratordel. Det ble avgjort at webdelen skulle utvikles i PHP, og derfor uten hjelp av UML. Dette fordi PHP ikke er et objektorientert programmeringsspråk. Men da det ble vedtatt at F@T- admin skulle utvikles i Java, ble det ytret ønsker om å prøve UML på denne.

Først ble oppgavene til systemet identifisert. Dette er oppgaver som systemet utfører for en aktør. Disse oppgavene kalles Use Case, og skal hjelp utviklerne med å strukturere arbeidet sitt. "Use Case beskriver hvordan en bruker, anvender systemet", som Alistair Cockburn beskriver det [5]. Det ble først vedtatt å skrive Use Caset på *brief format,* Eksempel 3-1, som kun er en kort historie om hva som skjer ved denne opperasjonen, dette ble gjort for å få bedre oversikt over oppgavene.

Finn fast pris : Brukeren av F@T ønsker å vite prisen på en bestemt service på en bestemt på en bestemt bil

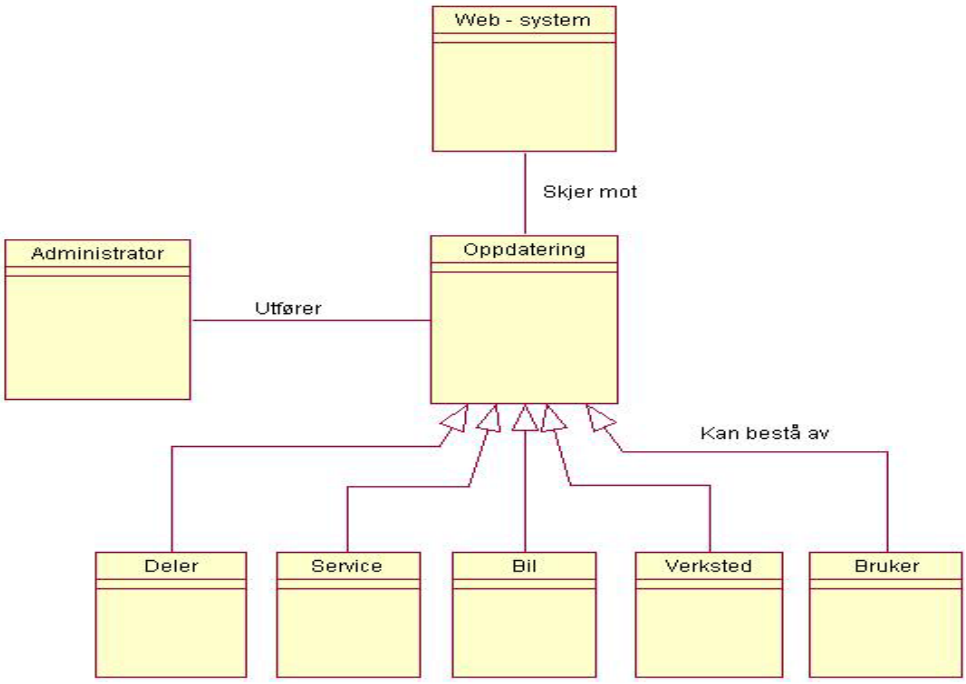
Eksempel 3-1 Use Caset Finn fast pris på brief format

Arbeidsmetoden som ble benyttet for å finne Use Case bestod av idémyldring og samtaler med oppdragsgiver. Use Case som kom fram i denne fasen :

- Legg inn verksted
- Rediger verksted
- Slett verksted
- Les inn filer
- Redigere service
- Slette service
- Legge inn bruker
- Endre bruker
- Slett bruker
- Redigere faste variable
- Finn fastpris
- Vis deleliste
- Bestille service.

I tillegg til Use Case fremkom det andre krav til systemet, som er beskrevet i Supplementary Specification se Vedlegg B . Her fremstilles krav om innlogging, GUI- design, databasedesign og forretningsregler.

For å kartlegge virkeligheten til administratorverktøyet ble det lagd et konseptuelt klassediagram, se Figur 3-1. Dette er et diagram som skal modellere virkeligheten.



Figur 3-1 Domain model for F@T- admin

3.3.2 Elaboration

I denne fasen er poenget å fordype seg i de viktigste oppgavene, dette innebærer kun de oppgavene som er av essensiell betydning for å komme videre i arbeidet. Noe av det første som ble utført da vi fordypet oss i et Use Case var å skrive et utdypende Use Case, som kalles et *fully dressed* Use Case,Eksempel 3-2

Use Case UC1 : Legge inn verksted

Primær aktør : Administrator
Interessenter : Administrator : Ønsker å legge inn et nytt verksted
Webkunder av F@T : Ønsker et å kunne velge det nye verkstedet i F@T.
Verksted : Ønsker å ligge inne i F@T, slik at webkunder kan sende forespørsel om service til dem.
Pre betingelser :

- Systemet må ha kontakt med databasen.

Post betingelser :

- Verkstedet blir lagret i databasen, eller feilmelding ble vist.

Suksess scenarium :

1. Administratoren velger å legge inn et nytt verksted.
2. Systemet returnerer et blankt verkstedskjema.
3. Administratoren fyller inn informasjon og lagrer.
4. Systemet returnerer kvittering for lagringen.

Avvik :

3. Klarer ikke å lagre i databasen.
    1. Systemet viser feilmelding.

Eksempel 3-2 Eksempel på fully dressed Use Case

Denne fasen ble delt opp i to iterasjoner, på en uke hver. For å avgjøre oppgavene tilknyttet hver iterasjon, startet den enkelte iterasjon med en risikovurdering av oppgavene. Denne vurderingen avgjør hvilke oppgaver som skal løses i kommende iterasjon. Risikovurderingen er kun et hjelpemiddel for å velge oppgaver og bør derfor ikke følges slavisk.

Ved starten av første iterasjon ble alle oppgaver satt opp og vurdert mot hverandre ut i fra følgende kriterier med vekt, Eksempel 3-3.

	Vekt
AS : Arkitektur betydning	3
Risk : teknisk vanskelig, middels eller lett	2
Kritisk : Foretningsverdi	2

Eksempel 3-3 Vektfordeling til risikovurdering 1

Ut i fra dette oppsettet ble risikovurderingen satt opp, Eksempel 3-4.

Aktivitet	**Type**	AS	Risk	Kritisk	Sum
Legge inn verksted	UC	2	2	3	16
Rediger verksted	UC	1	1	3	11
Slette verksted	UC	1	1	3	11
Les inn filer	**UC**	**2**	**3**	**3**	**18**
Rediger service	UC	1	1	3	11
Slette service	UC	1	3	2	13
Legge inn bruker	UC	2	3	2	16
Endre bruker	UC	1	3	2	13
Slette bruker	UC	1	2	1	9
Redigere faste variable	UC	1	1	3	11
Finn fastpris	UC	3	1	3	17
Vise deleliste	UC	1	1	3	11
Bestille service	UC	1	2	2	11
Databasedesign	**Design**	**3**	**2**	**2**	**17**
GUI	**Design**	**0**	**2**	**3**	**10**

Eksempel 3-4 Risikovurdering 1

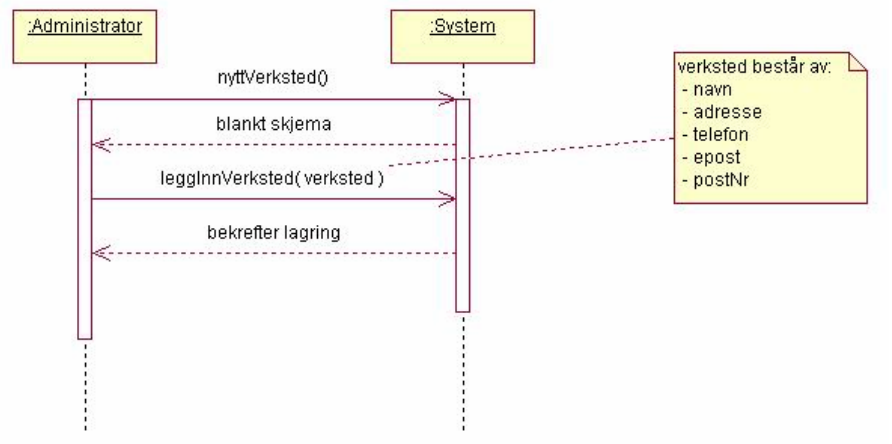
Ut fra denne vurderingen ble følgende oppgaver valgt:

- les inn filer
- databasedesign
- webdesignet.

Som vist i vurderingen er ikke GUI den aktiviteten som scorer høyest, men vi så den allikevel som et avgjørende element. Ved å behandle designet på web- delen tidlig kunne vi raskere få en tilbakemelding på arbeidet. Det var i den sammenheng viktig å finne et design som oppdragsgiver var tilfreds med.

Gruppen ble delt i to undergrupper, det ene paret jobbet med oppgaver tilknyttet web-delen, hvor det andre paret jobbet parallelt med oppgaver tilknyttet administrator-delen.

Use Casene som omhandler administratorverktøyet benyttet UML i undersøkelsene av problemområdet. For de valgte Use Casene ble det utviklet systemsekvensdiagram (SSD), som viser informasjonsutvekslingen mellom aktøren og systemet. Her representert ved SSD av Use Caset legg inn nytt verksted, Figur 3-2.



Figur 3-2 : SSD av Legg inn verksted

På slutten av iterasjonen kom det fram at det hadde blitt utviklet funksjoner som leste inn de nødvendige filene, et utkast til databasedesign og et grovt rammeverk på web utseende. Siden ingen avpunktene var ferdigstilt ble det i neste risikovurdering evaluert de gjenstående oppgavene på lik linje med de andre oppgavene.

En ny risikovurdering i starten av andre iterasjon i elaboration fasen avgjorde hvilke oppgaver som skulle utføres den neste uken. Vurderingen ble det valgt å sette mer fokus på det som var teknisk vanskelig, og mindre fokus på oppgaver med betydning for arkitekturen. Dette førte til at vektene for denne vurderingen ble fordelt som vist i Eksempel 3-5.

	Vekt
AS : Arkitektur betydning	2
Risk : teknisk vanskelig, middels eller lett	3
Kritisk : Foretningsverdi	2

Eksempel 3-5 Vektfordeling av risikovurdering 2

For den fullstendige risikovurderingen, Vedlegg B.

Ut ifra denne vurderingen ble det bestemt at det skulle jobbes med oppgavene

- Finn fastpris
- Pålogging
- Databasedesign
- GUI

som det skulle jobbes videre med. Enighet i gruppen førte til at det var riktig å fortsette med både GUI og databasedesign, fordi dette var oppgaver som det var

avgjørende å få på plass. I denne perioden ble det begynt å se på designet til administrator verktøyet.

Det ble foretrukket å la parene fortsette å jobbe med samme oppgaveområde fra forrige iterasjon. Dette var et bevisst valg, unødvendig tid skulle ikke i denne fasen gå bort for at personer måtte sette seg inn i et nytt problemområde.

I løpet av denne iterasjonen ble påloggingen og finn fastpris ferdigstilt, samt at databasedesignet var kommet så langt at hovedrammeverket var ferdig. GUI på webdelen var også nesten ferdig. For administrasjons- verktøvet ble det utviklet et forslag til GUI, men det viste seg at dette ikke var bra nok.

Siden de største hullene om hvordan systemet skulle se ut og virke hadde blitt tettet, samt at de viktigste tekniske spørsmålene var løst, var det klart for å forlate elaboration fasen.

3.3.3 Construction

I construction var målet å løse alle oppgavene som gjenstod. Til dette var det satt av ganske lang tid, men noe av tiden kom til å gå bort på grunn av et annet prosjekt etterfulgt av eksamen og påske. Dette var ting gruppen hadde beregnet fra starten av, slik at det ikke fikk konsekvenser for gjennomføringen. Fasen ble delt opp i fem iterasjoner, hver på ca. 2 uker, se Vedlegg A.

Før første iterasjon ble nok en risikovurdering utført, se Vedlegg B. Denne vurderingen utelot GUI og databasedesign, ettersom rammeverket for disse var klare. All jobbing på disse oppgavene i fremtiden vil skje i sammenhegn med andre oppgaver. Etter vurderingen ble det bestemt å sette i gang med oppgavene les inn filer, rediger service og vis deleliste. Etter jobbing med Use Casene F@T-admin ble det bestemt at det ikke var noe grunn å fortsette med UML modellering. Grunnen til dette var at vi hadde begynt å bruke Borland sin JBuilder. JBuilder genererte en rekke klasser på egenhånd, og siden vi på dette tidspunkt ikke hadde særlig kunnskaper om bygging av større applikasjoner i Java, ble beslutningen tatt om å avslutte UML og heller følge JBuilder sitt løp.

Også i denne iterasjonen jobbet gruppemedlemmene i par, der hvert par jobbet på hvert sitt område. Parenes sammensetting ble også forandret da noen ønsket å jobbe med det et annet problemområde, enn man hadde jobbet med tidligere.

For å forsikre seg om at Use Casene oppfylte de kravene som hadde blitt satt for de, ble det i starten av hver oppgave laget ett test case, som beskriver hvordan man skulle teste den ferdigstilte oppgaven. Eksempel på test case er her representert ved test case for Use Case legg inn verksteder, Eksempel 3-6.

TEST CASE 1 : LEGGE INN VERKSTED

Det må være mulighet til å legge inn nytt verksted. Det må vises ett blankt skjema som er klart til innfylling av data om verkstedet. Det skal legges inn informasjon om, Navn, Adresse, Postnr, Poststed, Telefon og e-post. Poststed skal komme automatisk fra postnr tabellen. Telefon nummer bør sjekkes om det er 8 siffer.
Test på at all informasjon blir lagret i databasen og at ingen bokstaver eller siffer blir borte i enden av setninger. Lengden på feltet i applikasjonen bør stå i forhold til lengden det er plass til i databasen.
Det bør også sjekkes at kvittering for lagring vises.
På slutten av denne iterasjonen ble det ikke foretatt noen risikovurdering, fordi det ikke hadde skjedd noe annet enn at de oppsatte oppgavene ble ferdigstilt.

Eksempel 3-6 Test Case 1 Legg inn verksted

Ved å utføre tester som ivaretok disse kravene kunne man konkludere med at Use Caset var ferdigstilt.

For andre iterasjon var det risikovurderingen som ble gjort før første som lå til grunn. Det var nå rediger service og legg inn verksted som sto for tur. På grunn av tilbakemeldinger på webdelen, ble det også satt av resurser til utbedring av denne.. Ressursene i denne iterasjonen vil bli redusert som følge av eksamen og påske. Under arbeidet og diskusjon med oppdragsgiver i denne iterasjonen kom det frem to nye Use Case for web- delen, og et nytt Use Case for administratordelen. Disse Use Casene var for web- delen skrive ut deleliste og liste opp verksteder, og for administrator delen vis feillogg.

Arbeidsoppgavene ble også her fordelt i to grupper, en for web- delen og en for administrator- delen. Sammensetningen av parene forble den samme for denne iterasjonen.

På grunn av de nye Use Casene var det nødvendig med en ny risikovurdering før neste iterasjon. For risikovurderingen, Vedlegg B.

Den tredje iterasjonen kom samtidig med påsken, slik at også denne ble litt redusert. Det ble bestemt å jobbe med bestille service, legg inn verksted, slett verksted, samt en del GUI design var på krevd på administrator delen. Også under denne iterasjonen kom det frem et par nye Use Case.

I denne iterasjonen ble parene oppløst, slik at det ble jobbet mer selvstendig. Grunnen til dette var at påsken førte til litt forskjellig oppmøtetid. Dette betydde ikke at gruppen ikke
samarbeidet hvis spørsmål dukket opp, men kun at gruppemedlemmene hver jobbet med hver sin oppgave.

Under arbeidet med administrasjonsverktøyet kom det frem at det var behov for funksjoner som kunne endre databaseoppkoblingen og sette opp en ny database. Dette gjorde at en ny risikovurdering måtte utføres

For fjerde iterasjon ble det avgjort å jobbe ekstra mye. På grunn av dette og risikovurderingen ble oppgavene rediger verksted, slette service, endre bruker, slette bruker, skriv ut deleliste og vis feillogg valgt.

På grunn av at et gruppemedlem skulle være borte store deler denne iterasjonen, ble et bestemt å sette en mann som hovedsakelig jobbet med web- delen og en mann til å jobbe med administrator- delen. Slik at resten av medlemmene kunne hjelpe til der det var behov. Grunnet mange oppgaver og reduserte resurser, ble det lange dager i denne fasen.

I femte og siste iterasjon i construction fasen ble de gjenstående oppgavene satt opp på listen over ting som skulle gjøres. Det ble også satt av resurser til klargjøring til en *beta* test. Denne jobben besto av å legge til et installasjonsprogram på administrasjons delen, lage evalueringsskjema, samt justere litt på GUI.

Siden det meste på web- delen var klart jobbet kun en person på de oppgavene som omfattet den, mens en mann jobbet med å forbrede beta testingen. De to gjenstående personene jobbet begge med oppgaver tilknyttet administrasjons- delen.

3.3.4 Transition

I denne fasen var det stort sett retting av feil og analysere tilbakemeldinger fra *beta* testingen som sto i fokus. Det ble også brukt en del tid på rapporten. Oppdragsgiver ga tilbakemelding om de ønsket en forandring på måten man bestilte service på. Denne forandringen gikk ut på at de ønsket at man måtte velge for hvilket fylke aktuelle verkstedet lå i. Dette førte til forandringer på flere andre områder, som blant annet databasen, innlegging av verksteder i administrator- delen og litt forandring på web- delen.

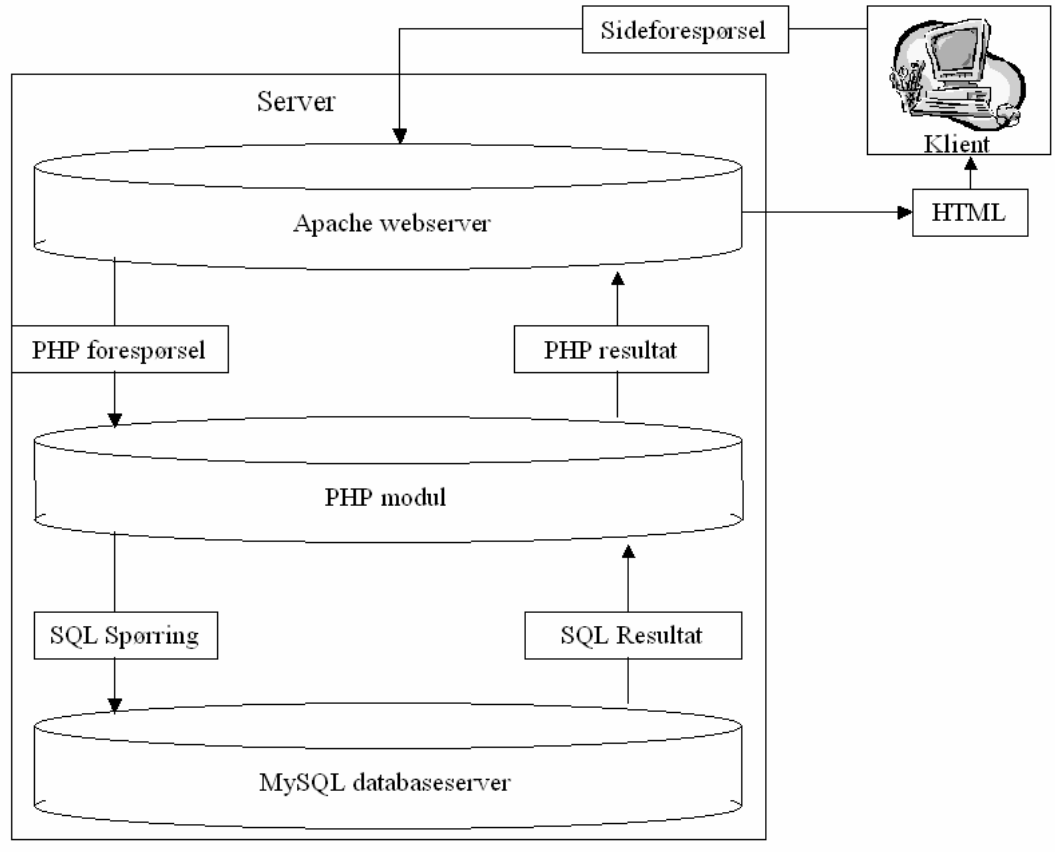
4 Design

4.1 Systemarkitektur

Systemet som skal designes skal brukes av flere verksteder som er spredd over et stort geografisk område, samt at kunder skal kunne benytte deler av systemet. Derfor er den mest hensiktsmessige måten med tanke på distribusjon å utvikle denne delen som en webløsning. Det var krav om at få utvalgte personer skulle ha mulighet til å administrere systemet, det er derfor valgt å utvikle en applikasjon som installeres lokalt hos Team G.E.D. A/S.

4.1.1 Web delen

Web delen er utviklet i PHP og HTML, og har en 3 lags arkitektur. Der databasen er representert som et eget databaselag, PHP serveren representert ved et applikasjonslag og nettleseren hos sluttbrukeren representert ved et presentasjonslag, Figur 4-1.



Figur 4-1 Viser 3 lags arkitektur

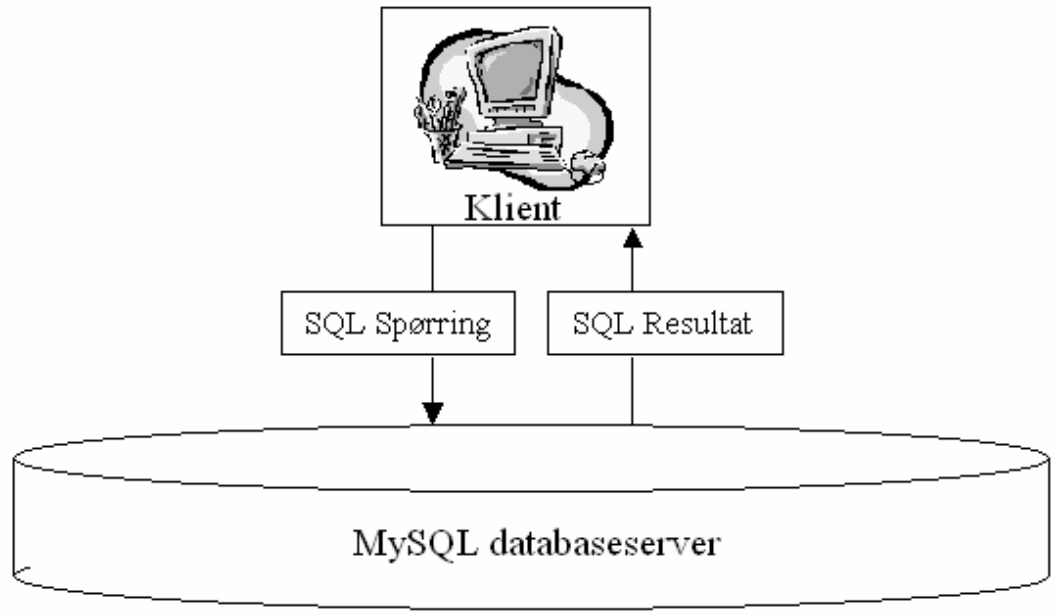
De tre lagene er som følger

1. Presentasjonslaget - Grensesnitt som bli vist mot nettleser
2. Hovedlaget - Kobling mot databasen
3. Lagringslaget - Databasen

I presentasjonslaget foregår all kommunikasjonen med brukeren, her vises resultatene fra forespørslene brukeren gjør i systemet. Systemet mottar forespørsler fra presentasjonslaget , som behandles i applikasjonslaget. I applikasjonslaget utføres all prosessering. Når applikasjons laget trenger data fra databasen, sendes en spørring til databaselaget som returnerer resultatet av spørringen tilbake til applikasjonslaget.

4.1.2 Applikasjonen

Administrator applikasjonen er utviklet i Java. Applikasjonen er delt opp i en to lags arkitektur, der presentasjonslaget og applikasjonslaget er et og samme lag, mens databaselaget opererer som et eget lag. Detter fører til det som kalles en tykk klient, som vist i Figur 4-2

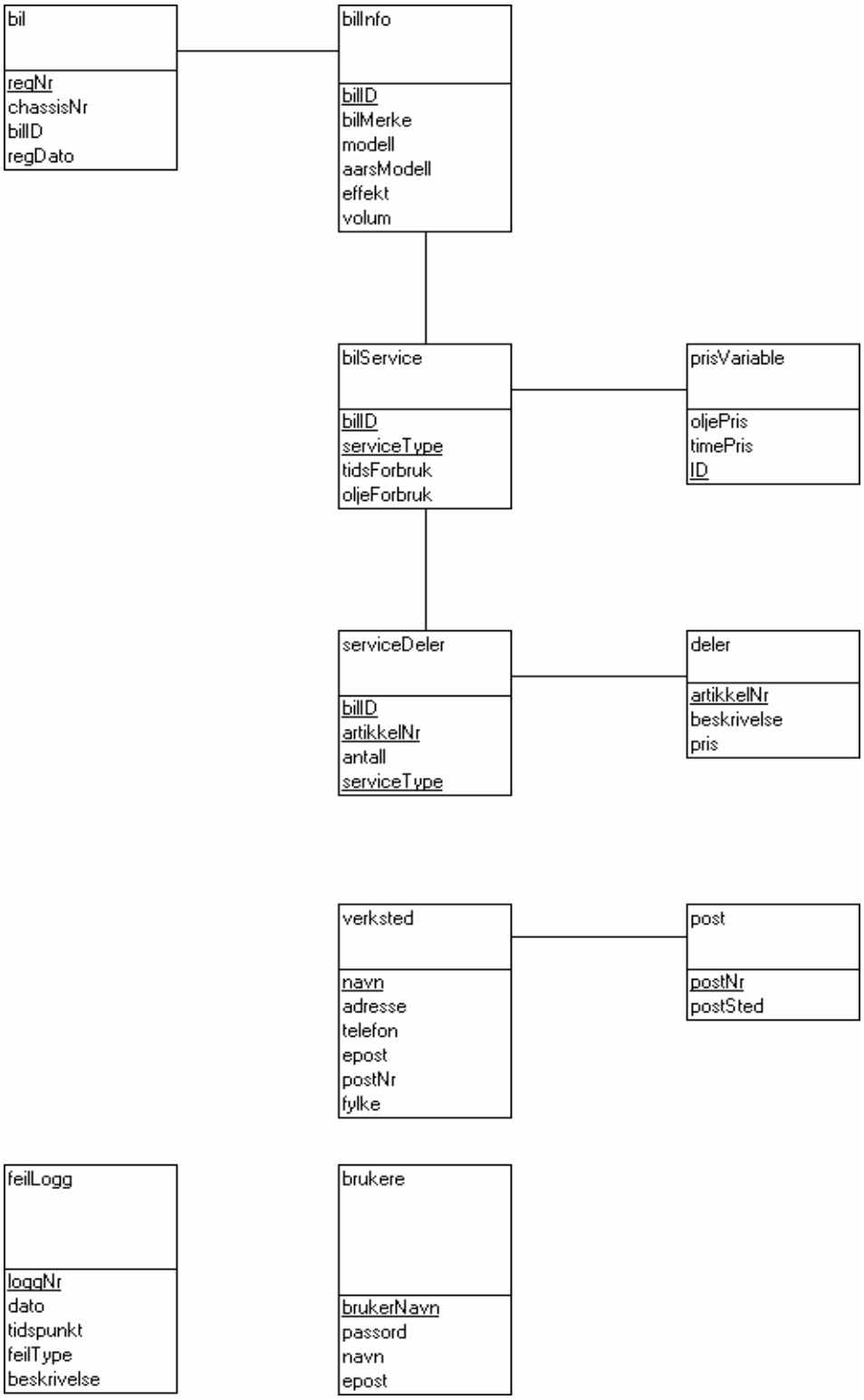


Figur 4-2 Viser 2 lags arkitektur

Applikasjonen kommuniserer direkte med databasen gjennom SQL spørringer.

4.1.3 Databasestruktur

Det ble lagt ned mye tid i å få til en god databasestruktur, fordi det ble sett på som en viktig suksessfaktor for oppgaven. Et bra design på databasen sørget for at søk og oppdateringer ville gå raskt. Dataene som lagres i databasen er en kombinasjon av data som er hentet fra andre systemet. Dette har ført til at en stor del av jobben med databasen har vært å finne ut hvilke data som skal være med, og ikke minst hvordan det skal leses inn. MySQL er ingen relasjonsdatabase, derfor er man avhengige av en attributt som er felles for flere av tabellene leses inn i flere steder. Det var også viktig å unngå redundans i databasen, slik at kun de attributtene som var helt nødvendige, for å knytte tabellene sammen, ble dobbeltlagret. For å oppnå dette er tabellene i databasen normalisert til tredje normalform. Figur 4-3 viser en oversikt over databasestrukturen, for en fullstendig oversikt Se Vedlegg G. Selv om MySQL ikke er en relasjons database er det valgt å ta med koblinger mellom tabellene for å vise sammenhengen mellom dem.


bil
regNr
chassisNr
billD
regDato
billnfo
billD
bilMerke
modell
aarsModell
effekt
volum
bilService
billD
serviceType
tidsForbruk
oljeForbruk
prisVariable
oljePris
timePris
ID
serviceDeler
billD
artikkelNr
antall
serviceType
deler
artikkelNr
beskrivelse
pris
verksted
navn
adresse
telefon
epost
postNr
fylke
post
postNr
postSted
feilLogg
loggNr
dato
tidspunkt
feilType
beskrivelse
brukere
brukerNavn
passord
navn
epost

Figur 4-3 Databasestruktur

4.2 Layout

Det er det første øyekastet som er avgjørende for hvordan en webside, og det er ut fra dette utgangspunktet det har blitt jobbet ved utvikling av F@T. Oppdragsgivers ønske var at F@T skulle være enkelt å bruke for de ansatte ved verkstedene. Disse vil ofte kun ha en gjennomsnittlig kjennskap til Internett og data generelt, slik at det var en faktor som måtte vektlegges når systemet skulle utvikles.

Som følger av UP vil ikke layouten være fullstendig før systemet er fullt utviklet. Men gjennom hele prosessen med å designe F@T er det lagt vekt på en del punkter :

- Enkelt og funksjonelt grensesnitt skal være lett å bruke selv for uerfarne brukere.
- Sidene skal være lette å sette seg inn i.
- Sidene skal være raske å bruke.
- Siden oppdragsgiver er i ferd med å innføre en firma profil for nettsidene sine vil vi prøve å benytte oss av farger som gjenspeiler dette.
- Det skal være enkelt å forandre på fargene på sidene. Gjøres ved at vi oppretter en egen config fil for farger. Her vil alle fargene vi bruker bli deklarert i variable som benyttes på de andre sidene.
- Det skal se likt ut i Internett Explorer, Netscape og Opera
- Plassering til elementene skal være lik uavhengig av nettleser.
- Lik bredde på alle sider.
- Ha et gjennomført likt design på alle sider, slik at brukere kjenner seg att.
- Skal bruke en skrift type som egner seg for web, og tilpasse skriftstørrelsen (relativ i forhold til nettleserens innstillinger) slik at leseligheten på skjermen blir best mulig.

4.3 Programmering av design F@T

I prosessen med å komme frem til den endelige layouten av sidene har hele gruppen deltatt og kommet med forslag og innspill, samt at oppdragsgiver har kommet med ønsker. Her har ønskene til oppdragsgiver vært de viktigste, så man har derfor prøvd å etterkomme disse så langt det har latt seg gjøre. Dette for at oppdragsgiver skulle få et produkt som de er fornøydd med, og dermed ønsker å ta i bruk.

F@T har delt opp skjermbildet i tre hoveddeler, Figur 4-4. Det er valgt å ha en hovedside bygd opp av tabeller som inkluderer undersider. Det ble også vurdert å benytte frames som et alternativ, men dette alternativet gikk man bort fra på grunn av frames sine mange ulemper.

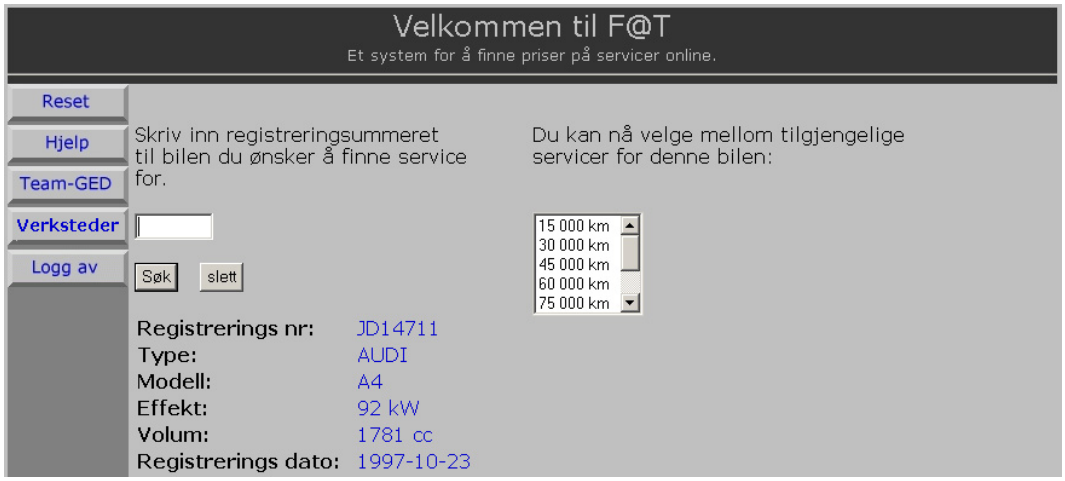
Figur 4-4 Startside

Grunndesignet er likt for både webkunder og verksteder, men innholdet vil variere. De tre hoveddelene er:

- Overskrift
  Vil variere ut fra hvilken side en befinner seg på. Disse skal fortelle brukeren hvor han/hun befinner seg samt å si litt om hvilke muligheter som finnes på siden.
- Meny del
  Brukes til å navigere seg mellom de forskjellige sidene. Innholdet på menyen vil variere etter hvilket brukernivå som benyttes.
- Hoveddelen
  Her blir søk etter biler og valg av servicer gjort. Hva som vises av informasjon om servicene varierer avhenging av om en er innlogget eller ikke

En av tingene som det er benyttet mye tid på i forbindelse med designet av F@T, er hvordan innholdet i hoveddelen skal bygges opp og hvordan informasjonen skal presenteres. Det ble bestemt at den beste løsningen ville være å vise søkemuligheter, valg og resultater på den samme siden. Det viste seg at dette ble litt innviklet å få til, siden filer som ble inkludert måtte inkluderes i en viss rekkefølge. På grunn av plasseringen av elementer ble det mange tabeller inni hverandre. Dette skapte problemer med tanke på at forskjellig nettlesere kan vise samme ting litt forskjellig.

Det ble derfor bestemt at hoveddelen ble delt i to, der den øverste delen er felles for alle brukere. Her utføres søk etter bil, valg av tilgjengelig service og opplisting av informasjon om den valgte bilen. Figur 4-5 viser hvordan den øverste delen av hoveddelen ble seendes ut.



Figur 4-5 Resultat av søk på registreringsnummer

Det ble besluttet å vie resten av hoveddelen til å vise informasjon om den aktuelle servicen. Denne delen vil variere for verksted bruker og privat bruker. Som verksted bruker vil en mer detaljert beskrivelse av servicen forekomme, Figur 4-6.

Figur 4-6 Detaljert beskrivelse av servicen

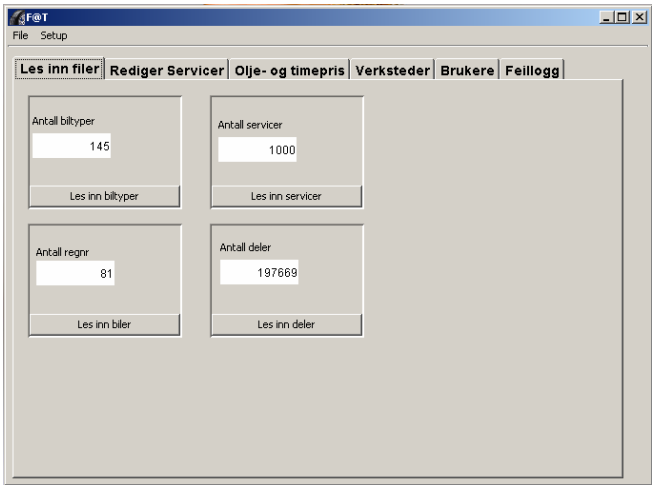
Oppdragsgiver ønsket ikke at privatkunder skulle få tilgang til detaljerte delelister, dette førte til at privatkunder får opp totalprisen for servicen. I stedet for den detaljerte delelisten vil disse brukerne få opp et skjema som kan fylles ut og sende til ønsket verksted, Figur 4-7.

Figur 4-7 Bestilling av service

4.4 Programmering av design på F@T- admin

For F@T- admin ble det laget et designutkast, Figur 4-8, som det skulle jobbes videre med. Dette designet var utviklet med et hoved vindu som igjen åpnet egne vinduer for hver funksjon, en såkalt SDI applikasjon. Det ble fort bestemt at dette designet ikke ville være det beste for brukervennligheten. Det ble derfor valgt å lage et klassisk design med ”tabbed-sheets”, Figur 4-9. Det kan sammenlignes med kartoteksystem der små flipper stikker opp, via disse kan det velges hvilken del man skal inn på. Kravet som ble satt til F@T- admin, var at det skulle være enkelt å bruke selv for en som ikke har brukt lignende programmer før.

Figur 4-8 Førsteutkast på GUI til F@T- admin



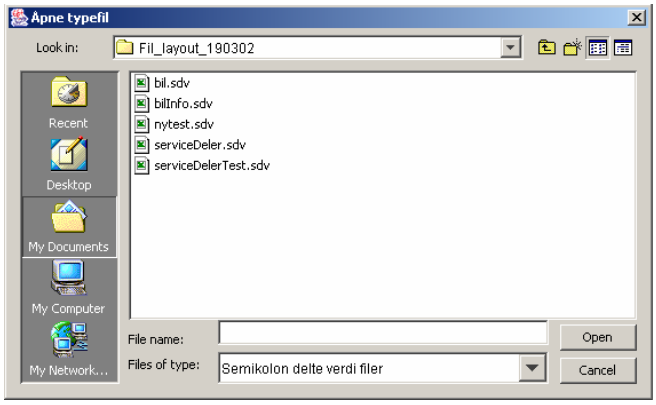
Figur 4-9 Endelig GUI på F@T- admin

4.4.1 Beskrivelse av de forskjellige komponentene i F@T- admin

4.4.1.1 Les inn filer

Dette kortet, Figur 4-9, består av fire felter hvor man vil til enhver tid ha oversikt over hvor mange deler, servicer og biler som ligger i databasen. Her kan man legge inn nye service og biler. Man kan også legge inn en helt ny deleliste. Grunnen til at dette kortet ligger som første element i F@T- admin er at det i hovedsak er dette F@T- admin blir brukt til, samt at man ser en indikasjon på statusen av systemet.
Ved å trykke på en av les inn knappene vil man få opp en filvelger, Figur 4-10, for å velge hvilken fil man skal lese inn. Klassen MittFileFilter.java, Eksempel 4-1, ble laget for å sette opp hva slags type fil man skal ha lov til å lese inn. Når en fil er valgt, vil det startes opp en tråd som sørger for innlesingen av filen og håndteringen

av statuslinjen som viser kontinuerlig hvor langt i innlesing av filen man har kommet. Disse filene er på et bestemt format, se Vedlegg D.



Figur 4-10 Filvelger

```
public class MittFileFilter extends FileFilter {
  public boolean accept(File f) {
    if (f.isDirectory()) {
      return true;
    }

    String extension = f.getName();
    if (extension != null) {
      if (extension.endsWith(".sdv")) {
        return true;
      } else {
        return false;
      }
    }
    return false;
  }
    // The description of this filter
  public String getDescription() {
    return "Semikolon delte verdi filer";
  }
}
```

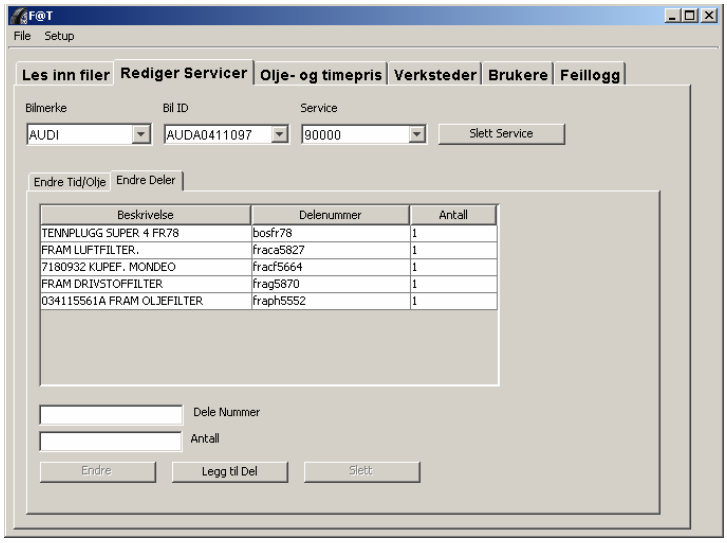
Eksempel 4-1 MittFileFilter.java

4.4.1.2 Rediger Servicer

Dette kortet sørger for at man kan endre på en bestemt service.
Første velges et bestemt bilmerke, slik at man får mulighet til å velge en spesifikk type av dette bilmerke, heretter kalt billID. Grunnen til dette er at etter hvert som databasen blir fylt opp med data kan det ligge flere hundre forskjellige bilID'r og det vil da være litt tungvint å skulle bla seg gjennom disse for å finne en bestemt bilID.

Når en bilID er valgt må det velges en service. Når denne er valgt vil det bli vist informasjon om servicen. Denne informasjonen er delt inn i to nye kort.

Underkortet Endre deler viser hvilke deler som skal være med på servicen, Figur 4-11, delene er lagt inn i en tabell. Det var en del problemer da denne tabellen ble laget. Det største problemet var muligheten å flytte på kolonnene. Dette medførte feil da det videre skulle hentets ut informasjon fra en enkelt rad i tabellen. For å løse dette problemet ble det undersøkt mye, og til slutt løste problemet seg med noen enkle triks, Eksempel 4-2. Ved å markere eller trykke på en rad så kommer informasjon om delenummer og antall i egne tekstfelt. Det kan også legges til eller fjernes deler fra en service. Når det legges til en del blir det kjørt en spørring mot databasen for å sjekke om delen virkelig finnes i databasens deleliste.



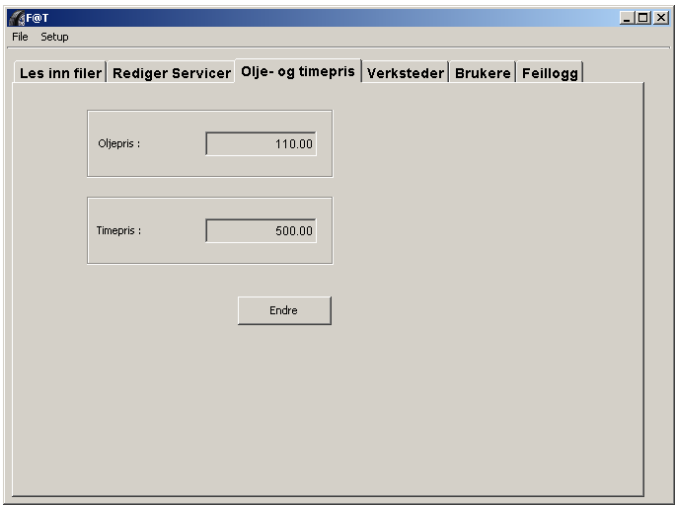
Figur 4-11 Rediger service

```
//Sørger at man ikke kan flytte på kolonner
kolonne.setReorderingAllowed(false);
//Sørger for at bredden på kolonnene blir satt
TableColumn column = new TableColumn();

column = tblTabell.getColumnModel().getColumn(0);
column.setMaxWidth(200);
column.setMinWidth(200);
column.setResizable(false);

column = tblTabell.getColumnModel().getColumn(1);
column.setMaxWidth(150);
column.setMinWidth(150);
column.setResizable(false);
```

Eksempel 4-2 Kode for å holde orden kolonner i tabell

4.4.1.3 Olje- og Timepris

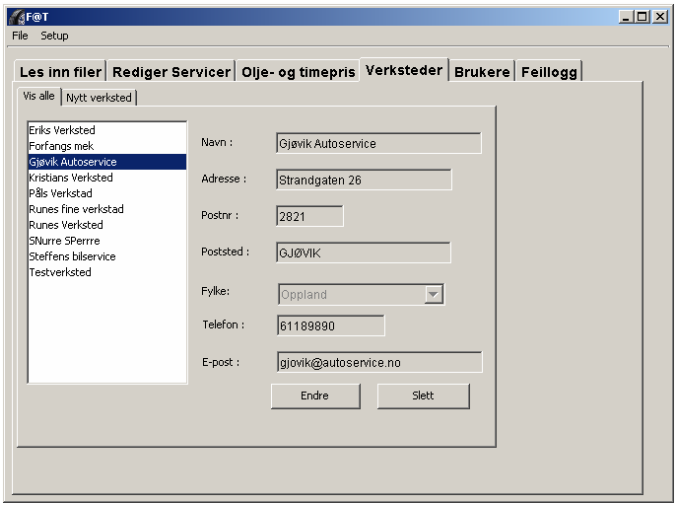
Dette kortet viser timepris og oljepris pr liter, Figur 4-12. Disse variablene er felles for alle verkstedene som er knyttet opp mot F@T- systemet, så hvis variable endres, vil det innvirke på alle servicer som ligger i systemet. Endringer skjer ved å skrive inn nye priser og lagre disse i databasen.



Figur 4-12 Olje- og timepris

4.4.1.4 Verksteder

Dette kortet viser en oversikt over verkstedene som ligger i systemet, Figur 4-13. Ved å markere et verksted så får man opp informasjon om verkstedet som adresse telefon og e-postadresse. Man kan endre informasjonen om et verksted hvis det skulle vise seg at det for eksempel har fått et nytt telefonnummer eller har flyttet.
Det tillates ikke å endre navnet på verkstedet da dette er unikt, hvis dette skal endres må en legge inn det nye verkstedet og deretter slette det gamle.
Når man legger inn et nytt verksted vil det automatisk bli søkt i databasen etter riktig poststed til postnummeret.



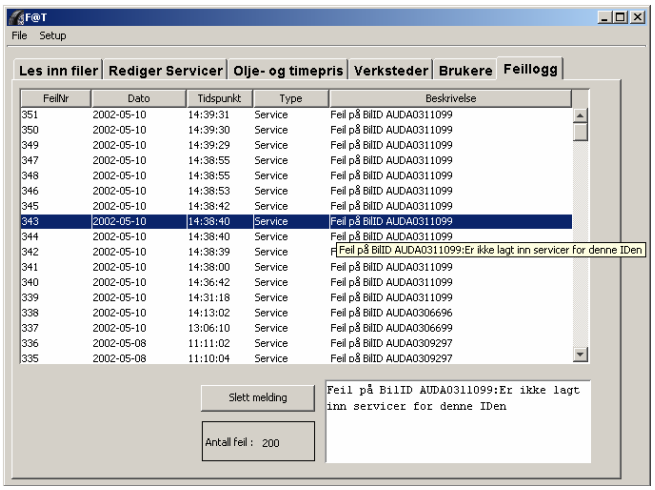
Figur 4-13 Verksted

4.4.1.5 Brukere

Dette kortet viser en oversikt over alle brukerne på systemet. De brukerne som er lagt inn her er de som kan logge seg på F@T- verksted. Her kan man endre informasjon om en bruker blant annet passord og e-post adresse. Man kan ikke endre brukernavnet da det er unikt. Hvis man vil endre brukernavnet så må man legge inn en helt ny bruker med det ønskede brukernavnet og slette den gamle brukeren. Ellers er designet på denne lik som Verksteder

4.4.1.6 Feillogg

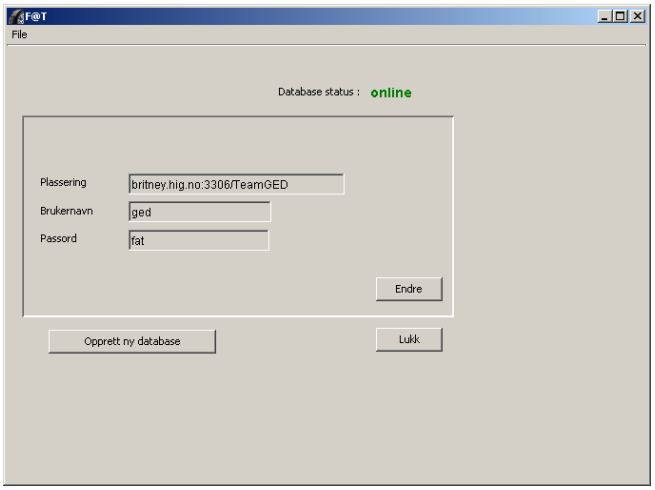
Dette kortet viser feilmeldingene som er generert via F@T, Figur 4-14. Feilmeldingene blir vist i en tabell. Ved å markere en feilmelding vil man få opp en beskrivene tekst over hva som er skjedd. Et eksempel på en feilmelding kan være ”*Feil på BilID AUDA0311099:Er ikke lagt inn servicer for denne IDen*”. Det vil si at det ikke er lagt inn noen servicer på denne biltypen. For å rette opp i denne feilen så kan man gå til les inn filer og lese inn filen som har servicene til denne bilen.



Figur 4-14 Feillogg

4.4.1.7 DBSetup

Denne delen er ikke en del av kortene men man kommer til denne delen via menyen på toppen i applikasjonen.
Her er det mulig å se hvilken database F@T- admin er knyttet opp mot, Figur 4-15. Man vil også kunne se om man får kontakt med databasen eller ikke. Det kan også endres hvilken database man skal koble seg opp mot.
For å opprette tabeller på en ny database kan man bruke *Opprett ny database* knappen. Det vil da bli lest fra filen *TeamGED.sql , Vedlegg E* ,som inneholder et script med oppsett av alle tabellen. Filen blir da lest inn og blir sendt som SQL spørringer til databasen.



Figur 4-15 Databasesetup

5 Implementering

5.1 Utviklingsmiljø

Straks etter at gruppen hadde fått tildelt grupperom flyttet de sitt eget datautstyr inn og opprettet et utviklingsmiljø. Alle arbeidsstasjonene kjørte Windows operativsystem. Mens utviklingsmiljøet inneholdt også en Linux-server som opererte som fil- og webserver.

5.1.1 Serveroppsett

Siden det skulle bli bedrevet utvikling mot web ble det bestemt at det skulle settes opp en egen webserver, dette for å unngå å være avhengig av skolens server. Da vil arbeidet kunne fortsette uavhengig av skolens server.

Det ble valgt en Apache webserver, da denne går for å være en meget god webserver og ikke minst for at den er gratis. Versjonen av Apache som ble installert var v1.3.22 . Også en SQL-server måtte installeres, her ble det valgt MySQL. Hovedgrunnen for at MySQL v 3.23.48 ble valgt, var at dette var nyeste stabile versjon av MySQL. For å få kjørt PHP så måtte det installeres en PHP-modul til Apache-serveren. Det ble også her installerte den siste stabile versjonen av PHP, versjon 4.1.1. Siden ingen i gruppen kjørte egne Linux maskiner så fant ble det valgt å installere SAMBA. SAMBA gjør det mulig å implementere Windows sine protokoller på Unix, dermed kan en Unix maskin tilby tjenester til Windows maskiner. Dette førte til et vanlig Windows-grensesnitt ved tilkobling av Linux maskinen fra en Windows maskin.

5.2 Valg av verktøy

5.2.1 Programmeringsspråk

Et av de første valgene som ble tatt var å finne ut hvilket språk de forskjellige delene av systemet skulle programmeres i. System vil bestå av en web del og et administrasjonsverktøy som opererer mot databasen til systemet. Dette blir forklart i underpunktene.

5.2.1.1 F@T

Det er meningen at web delen av systemet i dette tilfellet skal ligge på serveren til et firma, Visual Web Norge, som tilbyr sine kunder ulike alternativ:

- ASP mot Access databaser
- PHP mot MySQL database

Dette satte klare føringer på valg av utviklingsspråk. Som resultat av dette ble det bestemt at det skulle utvikles i PHP kontra ASP. Det var flere grunner for dette men de viktigste var:

- PHP ble benyttet av gruppen i et annet prosjekt her ved HiG, slik at det ville bli lettere å drive med de to prosjektene parallelt.
- Deler av gruppen hadde litt kjennskap til språket fra før.
- Ingen i gruppen hadde noen særlig erfaring med Access databaser. I utgangspunktet var Interbase førstevalget for database, men siden det ikke var noe valg, ble MySQL valgt, da denne databasen ikke er så ulik Interbase med tanke på SQL.

I webdelen er det også naturlig å ta i bruk JavaScript til vindushåndtering, ulike sjekker, fokusering og lignende funksjoner.

All kode for web delen er skrevet i tekst editoren EditPlus. Det eneste kravet som ble sett i forhold til editoren var at den hadde syntaks uthevning og linjenummerering. EditPlus fungert bra til sitt formål og i tillegg til den ovenfor nevnte funksjonaliteten har den også en del ferdigproduserte funksjoner for generering av skall, som brukes til standard innstillinger i HTML/PHP, som for eksempel tabeller og script. Programmet Dreamweaver 4 fra Macromedia ble også benyttet for å bygge opp strukturen til sidene. Dette ble gjort for å lette jobben med å sette opp tabellene og plasseringen av disse, da det var tabeller som var inne i tabeller igjen. Knappene på sidene ble laget i programmet Image Broadway, som er et program spesielt for dette.

5.2.1.2 F@T- admin

Her stod det mellom Java og Delphi (Pascal), men Delphi ble aldri et reelt alternativ på grunn av prisen på lisens. Delphi var tidligere benyttet i et fag medlemmene av gruppen hadde gjennomført. Det var også et ønske om å utvikle en applikasjon som skulle være uavhengig av operativsystem. Dermed endte vi opp med å utvikle applikasjonen i Java. Gruppen har også gjennomført et fag innen Java tidligere, men faget dekket ikke utvikling av større applikasjoner. Derfor så gruppen det som en fordel å lære seg mer Java. Java er et programmeringsspråk som er populært for tiden samtidig som det kanskje er et mer fremtidsretta verktøy enn Delphi.

Borland JBuilder 6 var verktøyet som ble valgt for å lage F@T- admin HiG hadde lisens som kunne brukes. Ved å bruke et slikt verktøy kan man raskt kunne sette opp design på sidene ved å benytte seg av såkalt *drag&drop* teknikk. Det vil si at man plasserer elementer der de skal være, dermed slipper man å tenke mer på hvordan de skal tegnes opp i applikasjonen. En annen fordel med et slikt verktøy er at man hele tiden har god oversikt over koden ved hjelp av *syntax highlightning*. Det er også en innebygd debugger-funksjon som gjør at verktøyet sier i fra hvor i koden det er noe feil.

5.2.1.3 Database

Systemet som ble laget var ment som et prøveforsøk for Team G.E.D. A/S. De skulle teste det ut, for på sikt prøve å innføre det blant verkstedene i BOSCH kjeden. Derfor

var det ikke satt av penger til for eksempel dyre databasemotorer. Dette førte til at man baserte seg på en gratis databasemotor. Det ble undersøkt rundt om hvilke databasemotorer som var gratis, og som samtidig var best egnet for formålet. De to alternativene det endte opp med var MySQL og Interbase 6.0.

Det ble diskutert å benytte Interbase, da denne var benyttet en del fra før, med stort sett positive erfaringer. Firmaet Visual Web Norge som leverer Internett tjenester til oppdragsgiver kjører kun MySQL og Access databaser. De ble kontaktet om mulighetene for å legge inn Interbase v 6.0. Denne versjonen av Interbase er etter gruppens forståelse gratis, noe som også ble bekreftet av veileder som hadde vært i kontakt med Borland Norge angående samme problemstilling ved innlegging av denne versjonen ved HiG. Leverandøren av hjemmesidene til oppdragsgiver var ikke så sikker på dette og ville heller drifte MySQL. Dermed falt valget på MySQL. Firmaet Zonavi, som har stor kompetanse innen databaser, ble også kontaktet, de konkluderte en MySQL ville være en bra database for prosjektet..

For å modellere databasen brukte ble programmet Modelator 4.0 benyttet. Fra dette programmet kan en også generert script som kan brukes til å opprette databasen.

Databasen ble realisert ved hjelp av programmet phpMyAdmin-2.2.3. Dette programmet lar en administrere MySQL via nettet. Mer informasjon om programmet kan finnes på den offisielle hjemmesiden for programmet [7]:

5.2.2 Installasjonsprogram

For å gjøre det enkelt å installere F@T- admin ble det valgt å bruke et installasjonsprogram. For å lage et installasjonsprogram finnes det mange muligheter, men de fleste av programmene som lager installasjonsprogrammer er lisensiert og koster da penger. *InstallAnywhere now* [6] er et gratisprogram som kan lastes ned via Internett. Dette ble valgt da gruppen hadde fått innspill fra andre utviklere om at dette skulle være et bra program. Det er relativt enkelt på lage ny installer, man starter opp programmet og følger bruksanvisning på skjermen. InstallAnywhere gjør det mulig å legge med en VM slik at de som ikke har denne installert installerer en lokal VM som brukes av applikasjonen(F@T- admin). Med installasjonsprogrammet kan man sette opp hvilke systemer det skal lages installasjonsfiler for, men siden systemet i denne omgang skal kun brukes på Windows plattformer ble det laget en installer for denne. Installasjonsprogrammet til F@T- admin ble laget i 2 versjoner, en med og en uten VM.

5.3 Prinsipper ved koding

5.3.1 Kodingen av F@T

5.3.1.1 Generelt

Det er forsøkt å holde en gjennomført struktur på kodingen, noe som stort sett har lykkes bra, men det har blitt noen forskjeller på overgangene mellom HTML og PHP. Varierende programmeringsteknikker blant utviklerne har ført til at kodens utseende har blitt noe forskjellig. Dette fordi noen foretrekker å skrive HTML koden som ren HTML kode som vist i Eksempel 5-1, eller via echo kommandoen i PHP som vist i Eksempel 5-2. Noen
plasser har denne overgangen gått naturlig da det i disse situasjonene har vært vanskelig å drive å veksle mellom de to språkene.

```
<form method='post' action='printvennlig.phtml'>
<input type='hidden' name='service_type' value='<?php echo "$service_type" ?>'>
<input type='hidden' name='idnr' value='<?php echo "$idnr" ?>'>
<input type='hidden' name='regnr' value='<?php echo "$regnr" ?>'>
<input type='submit' value='Utskrifts vennlig side'>
</form>
```

Eksempel 5-1 Koding ved å gå inn og ut av PHP modus

```
echo "<form method='get' action='$PHP_SELF'>";
echo "<input type='hidden' name='regnr' value='$regnr'>";
echo "<input type='hidden' name='btn_search' value='$btn_search'>";

echo "<br>";
echo "<select name='service_type' size=5  onChange='javascript:this.form.submit()'>\n";
do{
   $tmp = $bil_service[serviceType];
   $tmp1 = ltrim($tmp, "\0x00");                    //fjerner overflødige nuller.

   if(is_numeric($tmp[1])) {                        //Sjekker om det er et tall.
      $tmp1 =  number_format($tmp1, 0, '', ' ');    //Setter inn 1000 skille tegn.
      $km = $tmp1.' km ';                           //Legger til km etter tall.
   }else $km = $tmp1;

   echo "<option value='$bil_service[serviceType]'> $km</option>";
}while ($bil_service = $database->hentRad ($res_bil_service));
echo "</select> \n </form> \n";
```

Eksempel 5-2 Koding av HTML ved hjelp av echo funksjonen i PHP

5.3.1.2 Utskrift fra programmet

Siden det kan være vanskelig å få til gode utskrifter fra web sider ble det laget en egen utskriftsvennlig side. Denne siden inneholder opplysningene som er relevante å skrive ut fra systemet.. Det ble gjort bevisste valg med tanke på skrifttype som egner seg for å skrive ut.

5.3.1.3 Navn på variable

I motsetning til F@T- admin er det ikke innført noen standard for navnsetting av variablene som er brukt. Den eneste regelen for variablene har vært at de skal være selvforklarende, slik at det skal være greit å forstå hva de brukes til.

5.3.1.4 Globale variable

Globale variabler er realisert gjennom bruk av sesjons variable i PHP. Disse tas i bruk der det er behov for at variabler skal være tilgjengelig fra flere sider. Ved behov for å ta med attributter til nye sider, har det blitt benyttet skjulte (hidden) attributter, Eksempel 5-2, når en sender (submit) et skjema (form) i HTML. Den viktigste globale variabelen som er benyttet er den som holder styr på om brukere er innlogget eller ikke.

5.3.1.5 Kommentering

Koden er ikke kommentert i særlig grad bortsett fra en del små kommentarer. Mesteparten av kommentarene som er tatt med, er for å hjelpe programmererne med å holde oversikten under kodeprosessen. Det er ingen detaljerte beskrivelser av hvordan ting skjer, bare korte kommentarer om hva funksjoner gjør, Eksempel 5-3. I de tilfellene der det har vært nødvendig vil det være mer detaljert kommentering. Så lenge det ikke blir gjort spesielle ting, var det ikke behov for detaljert kommentering, da det stort sett er datakyndige personer som studerer kildekoden.

```
//Legger en feilmelding inn i databasen
function logg_feil($melding, $type){    //Får medsent feil beskrivelse og type feil.
   include_once ("db_include.phtml");
   $database=new DB();                  //Kobler til databasen

   $today = getdate();                  //Henter dato og klokkeslett
   $maned = $today['mon'];
   $dag = $today['mday'];
   $aar = $today['year'];
   $sekund=$today['seconds'];
   $minutt=$today['minutes'];
   $time=$today['hours'];

   $klokke="$time:$minutt:$sekund";
   $dato="$aar-$maned-$dag";

   $sql_melding="INSERT INTO feilLogg(dato,tidspunkt,feilType,beskrivelse)";
   $sql_melding = $sql_melding."VALUES ('$dato','$klokke','$type','$melding')";
   $res_melding = $database->utforSql($sql_melding);    //Legger inn feilen
}
```

Eksempel 5-3 Kommentering av PHP kode

5.3.1.6 Gjenbruk av kode

Funksjoner som brukes ofte er plassert på en egen fil, slik at man slipper å skrive disse opp igjen hver gang de brukes. Databasekommunikasjon er også lagt på en egen fil, som tar seg av alt fra oppkobling til å kjøre SQL kommandoer. En av fordelene ved å bruke egne funksjoner ved databasekommunikasjonen er at man ikke kan gi de ulike kommandoene en forklarende tekst, slik at det blir greiere å lese hva som skjer i koden.

5.3.2 Koding av F@T- admin

5.3.2.1 Globale variable

I systemet er det bevist ikke brukt globale variable, slik at et objekt ikke kan jobbe direkte på andre objekters variable. I stedet for globale variable har vi overført data til funksjoner med hjelp av parametere.

5.3.2.2 Gjenbruk av kode

For å unngå å skrive kode opp igjen er det benyttet funksjoner der det har vært nødvendig. Dette fører til at hvis noen må rettes opp trenger man kun å gjøre dette på ett sted slik at man unngår unødvendig mye tid med å lokalisere alle stedene feilen skjer og rette opp disse.

5.3.2.3 Navn på variable

En standard navnsetting ble bestemt. Ved innføringen av en slik standard blir det enklere å finne tilbake til hva slags variabel det er og hva den brukes til. Alle variabler begynner med en trebokstavsforkortelse for å kunne identifisere hva slags type variabel det er, for å så bli etterfulgt av et forklarende navn. For eksempel skal et tekstfelt for telefonnummer bli hetende tfdTelefon. Hvis en variabel navn består av flere ord benyttes det stor forbokstav i hvert ord, for eksempel tfdDenneHarEtLangtNavn.

5.3.2.4 Kommentarer

En strukturert måte å kommentere koden på var noe som ble jobbet med helt fra starten. Utover i prosjektperioden har dette sklidd litt ut, men i etterkant har koden blitt gjennomgått og kommentert på Javadoc standarden, Eksempel 5-4. Javadoc er en måte å kommentere på slik at det automatisk blir generert dokumentasjon av koden.

```
/**
 * Sørger for INSERT; UPDATE OG DELETE mot databasen
 * @param update streng med SQL spørringen
 * @return true/false alt etter om spørringen gikk bra eller ikke
 */
 public boolean updateSQL( String update ){
 Statement SQLuttrykk;
 boolean ok = false;
   try {
   SQLuttrykk = connection.createStatement ();
     SQLuttrykk.executeUpdate( update );
     ok = true;
   }catch(SQLException se) {
     System.err.println("Feil under update :"+se);
   }
   return ok;
 }
```

Eksempel 5-4 Kommentering av Java kode på Javadoc standarden

5.3.2.5 Prinsipper ved koding

For å kunne sette seg inn i ny kildekode på lettest mulig måte ble man enige om standarder for hvordan man skal skrive funksjoner og andre kodesnutter. Standarden sier
at man skal bruke ett tabulatorhopp for hver bolk i koden, dette gjør at koden blir lettere å lese da alt som ligger inne i en bolk starter på samme tabulatorhopp.

5.3.2.6 Oppsummering

Koden er ikke optimalisert, da lavt kunnskapsnivå i starten har ført til at man kom litt skjevt ut. Det er naturlig og rett at koden blir bedre i løpet av prosjektet. Med lengre tid kunne all dårlig kode blitt omstrukturert.

6 Kvalitetsikring

Ved å kvalitetsikre hele prosessen og ikke bare produktet, er det lettere å oppnå de krav som stilles til produktet. Som igjen fører til at oppdragsgiver blir fornøyd. Det skilles mellom planlagt kvalitet, produsert kvalitet og opplevd kvalitet. Planlagt kvalitet er hvordan prosessen på forhånd kvalitetsikres. Produsert kvalitet er hvordan produktet kvalitetsikret undervis. Mens opplevd kvalitet er hvordan produktet er i overensstemmelse med oppdragsgivers krav.

6.1 Sikre kvalitet på prosessen

For å kvalitetsikre prosessen ble utviklingsmodellen UP benyttet. Modellen setter fokus på å avdekke og forstå problemområdet. Samt ved å benytte Use Case teknikker kan kundens krav beskrives og identifisere. For å sikre at disse kravene ble fylt, ble resultatene kontinuerlig presentert oppdragsgiver.

Det ble gjennomført jevnlige statusmøter for å sikre fremdriften i prosessen. Fremdriften ble også sikret ved at UP er en såkalt iterativ utviklingsmodell, som krever et resultat ved slutten av hver korte periode.

6.2 Sikre kvalitet på produktet

Kvalitetsikringen av et produkt er sammensetningen av kvalitetsarbeid fur under og etter koding. Kvalitetsikringen av produktet har skjedd ved testing. Ved å benytte Use Case og test case sikres det at de enkelte oppgavene i systemet tilfredstiller de enkelte krav, mens testing hos kunden avgjøre om kundens behov er dekket.

Ved å utvikle programmet i moduler legges alt til rette for videreutvikling. Da det enkelt kan legge til og eventuelt fjernes moduler.

Med tanke på vedlikehold av produktet av det blitt kommentert alle funksjoner, slik at det tydelig går frem hva som skjer i funksjonen. Innholdet i funksjonene er ikke kommentert så lenge det ikke er gjort noe spesielt.

7 Testing

Testing har stått i fokus hele tiden. Da gruppen bestemte mye av hvordan ting burde se ut og hva slags funksjonalitet det burde ha, har vi selv stått for testing igjennom det meste av hele prosessen. Gruppen har gitt tilbakemeldinger til oppdragsgiver på ting som burde endres, det har vært mindre ting som for eksempel oppsettet av filer som de skal lage.

Det var ikke før mot slutten det ble gjort mer omfattende testing av hele systemet. Det ble levert ut en beta versjon 1.0 den 29.april. Etter dette kom det tilbakemeldinger om hvordan hele systemet fungerte. Det har hele tiden blitt foretatt testing innad i prosjektgruppen, men det er viktig at oppdragsgiver gir tilbakemeldinger på hvordan systemet fungerer. Både GUI og funksjonalitet ble testet. Det meste på systemet fungerer optimalt, men tilbakemeldinger trengs for å finne ut hvor sårbart ting kan være. Det gjelder da spesielt innlesinger av filer som oppdragsgiver lager. Det må også lages rutiner på hvordan ting skal gjøres og i hvilken rekkefølge.

Det har hele tiden blitt testet opp mot databasen og det har da ført til at det ikke bare er selve applikasjonen som har blitt testet, men også hvordan databasen fungerer. Dette har da igjen ført til at databasen optimaliseres etter hvert. Databasen ligger i dag på en ikke alt for rask server, noe som gjør at alt det vi foretar oss her ikke går like raskt unna. Fordelen med dette er at det som fungerer greit her, vil kunne gå raskere når de setter systemet i drift.

Det ble fortatt en brukertesting hos Team G.E.D. A/S, testingen som ble foretatt på beta ver1.0. var svært positiv. Det ble delt ut evalueringsskjemaer, Vedlegg F, slik at oppdragsgiver kunne gi en tilbakemelding via disse. Det ble kjørt testing både av F@T- kunde, F@T- verksted og F@T- admin. Når det gjelder F@T- admin fungerte dette slik som oppdragsgiver hadde tenkt seg. Oppdragsgiver kom med mange gode tilbakemeldinger, noen av de tingene oppdragsgiver hadde å utsette på produktet var ting som allerede hadde blitt oppdaget og dermed rettet opp. En av tingene som oppdragsgiver hadde oppdaget var at det tar litt tid å starte selve F@T- admin. Dette ble endret på og det er nå blitt lagt inn et oppstartsvindu som viser at programmet er ferd med å starte. Når det gjelder F@T både for kunde og verksted var det også her mange positive tilbakemeldinger. Oppdragsgiver hadde ønsker om noen kosmetiske endringer blant annet å legge på Team G.E.D. A/S logo og få med litt informasjon på hensiktsmessige steder.

8 Analyse av prosjektet

8.1 Prosessen

Vi føler at UP har passet dette prosjektet bra. Spesielt fordi UP tillater at man tilføyer nye krav underveis. Siden vår erfaring innen UP var begrensede, kunne vi vært flinkere til å identifisere krav i starten, men på en annen side tillater UP mer urutinerte deltagere enn for eksempel fossefallsmetoden.

Grunntanken var å benytte UML, men ett stort avvik fra planen gjorde at vi gikk bort fra dette. Med tanke på resultatet var dette en beslutning som betydde at vi mistet litt struktur i koden, samt god trening i UML. Men resultatet sett fra brukerens synspunkt ville sannsynlig ikke blitt særlig annerledes.

Samarbeidet innad i gruppen har fungert meget bra. Grunnen til dette har nok vært at alle medlemmene kjente hverandre fra før, og viste hva de andre stod for. Arbeidsmengden har naturlig nok variert litt fra deltager til deltager. Det var heller ikke noe mål for denne gruppen å jobbe like mye, men å levere et godt prosjekt. Gruppen har også samarbeidet med en annen hovedprosjektgruppe på mange konkrete spørsmål. Siden begge gruppene har jobbet på samme rom, har det utviklet seg et godt miljø både faglig og sosialt. Det har vært veldig naturlig å konfrontere den personen med mest kunnskap innen det gjeldende feltet når men sitter med et problem.

Vi føler oss godt fornøyd med gjennomføringen. Ting man kan ta med seg til senere UP prosjekter er at transition fasen med fordel kunne vært litt lengre, slik at man kunne ha sluppet flere testversjoner. Vi erfarte også gode ting som kan være greie å ta med seg. Sånn som vårt forsøk på å innføre parprogrammering som vi til tider fikk til å fungere bra og den strenge politikken på *timeboxing* for iterasjonene. Muligheten til å diskutere spørsmål med personer som ikke deltar på prosjektet var positiv.

8.2 Produktet

Gruppen er godt fornøyd med det produktet ble produsert. Spesielt var det bra med positive tilbakemeldinger fra oppdragsgiver som er veldig fornøyd med det systemet de har fått.

Det har blitt utarbeidet en ferdig versjon som kan settes i drift når det overleveres til oppdragsgiver. Det eneste som gjenstår for at systemet skal fungere i full skala, er at de filene som leses inn i databasen må ferdigstilles. For databasen inneholder per i dag for lite data til å ha noen særlig hensikt. Men disse filene er det oppdragsgiver selv som er ansvarlig for å få laget.

Systemet ble et produkt som har potensial til å bli brukt i den daglige driften, og vil lette måten verkstedene finner prisene på.

8.3 Forbedringer/Videreutvikling

Det finnes flere muligheter for videreutvikling av systemet. Det ble vurdert å knytte systemet opp mot eksisterende ordre- og bookingsystem, men etter å ha diskutert disse med oppdragsgiver ble vi enige om å droppe dem. For oppdragsgiver var det viktigst å få med den funksjonaliteten som systemet tilslutt innehar.

Andre muligheter som ble vurdert var fakturering og muligheten for å implementere et ordresystem. Da oppdragsgiver allerede hadde systemer som tok seg av dette, var ikke dette prioritert i denne omgangen. Det er reelle utvidelser av systemet vårt. Men disse ser vi som svært reelle utvidninger av systemet vårt, da enten som integrerte moduler i systemet vi har laget eller knyttet mot eksisterendes systemer.

I motsetning til slik systemet er i dag, der ønsket verksted velges ut fra en select boks, hadde det vært muligheter å velge verksted ut fra et kart. Slik at det hadde vært lettere å orientere seg om hvor verkstedene ligger rent geografisk. Dette forslaget kom fra oppdragsgiver, men dette ble utelatt da det kom sent i prosjektet. Vi vurderte det til at det ville tatt for lang tid å implementere dette dersom dette skulle bli gjort på en tilfredstillende måte.

Av annen funksjonalitet som ble vurdert var muligheten for å knytte systemet opp mot et ordre system for deler. Det vil si at det foretas en sjekk på antallet deler som finnes på lageret i det systemet får en forespørsel om en service. Og finnes det da ikke tilstrekkelig med deler får en mulighet for å kunne bestille disse direkte fra systemet.

Det er også tatt høyde for en sjekk på om biler snart skal inn på EU-kontroll. Dersom det bestilles en service på en bil som skal inn til kontroll innen et gitt tidsintervall vil det gis en melding om dette. Her gjenstår det bare å implementere dette i kode delen for F@T.

Tilretteleggingen av de filene som leses inn i databasen er per i dag en tung prosess som med fordel kunne vært gjort lettere. I dag blir disse laget manuelt hos oppdragsgiver. Den tyngste biten vil da være prosessen med å lage filene første gangen alle biltyper skal være med, etter at den jobben er gjort vil eventuelle fremtidige endringer egentlig være en grei affære. Problemet er at opplysningene kommer fra ulike databaser og det er ikke mulig å få hentet ut fra alle disse automatisk. Det vil si at en del av de dataene vi bruker kommer fra ulike system som oppdragsgiver bruker i dag. Dersom det hadde vært mulighet for å hente disse direkte fra de ulike databasene ville det vært en fordel. Dette er noe som kan være aktuelt i fremtiden, men dette vil jo avhenge av om det er mulig å inngå avtaler med de som sitter på disse databasene.

8.4 Gruppearbeidet

Da vi startet på selve oppgaven i januar 2002 var det noe start problemer, vi hadde ikke fått tildelt et grupperom, men etter intens lobby virksomhet i skolen korridorer så fikk vi

et rom i kjelleren på skolen. Her var det sparsomt med kontorutstyr og det var heller ingen nettverkskontakter. Gjennom kontakter en av gruppens medlemmer hadde fikk vi tak i et stort bord som gjorde at vi alle kunne opprette et bra utviklingsmiljø. Etter at nettverkskontaktene ble installert her også følte vi at vi hadde vært svært heldige med rommet vårt. Uten dette grupperommet hadde prosjektet vanskelig latt seg gjennomføre da vi har vært avhengige av egne servere og PC-er, og ett tett samarbeid innad i gruppen

Samarbeidet i gruppen har fungert bra. Vi har hatt et uformelt miljø, så rollen til gruppeleder har ikke vært så stor som den kanskje ville vært hvis dersom hadde kjent hverandre så godt fra før, gruppeleder har vist ansvar og hatt et overordnet perspektiv på alle ting slik at prosjektet vårt ikke skled ut.

Det er i forprosjektrapporten nevnt at det skulle utføres ukentlige møter innad i gruppen. Siden vi har sittet såpass tett oppå hverandre uteble nødvendigheten i disse møtene.

Det ble lagt sterk vekt på å ha en jevn fordelig av arbeidsmengde gjennom hele prosjektperioden. Eneste faktor som har spilt negativt inn her er undervisning i skolefag med eksamener og prosjektarbeid rett før påske som har krevd mye tid. Dette har vi kompensert med høyere arbeidsmengde på prosjektet etter påske.

Gruppemedlemmene har hatt anledning til å disponere sin egen tid, men med ansvar for at tildelte oppgaver blir utført innen angitt tid. Høy selvdisiplin og godt samarbeid mellom medlemmene har sørget for at denne ordningen har fungert meget bra.

Arbeidet med prosjektet har gitt gruppen erfaring med å utvikle et produkt for en arbeidsgiver. Erfaringer har vist forskjell på å gjøre systemutvikling i teori og praksis. Det er ikke lett å planlegge i store detaljer hele tiden, til dette har gruppen for liten erfaring fra lignende prosjekter.

Vi har prøvd å holde en oversikt over hvor mange timer vi har brukt på prosjektet. Det er ikke enkelt å føre et nøyaktig timeantall på prosjektet da det ikke har vært fast arbeidstid. Til sammen er det gått med i overkant av 2000 timer på prosjektet:

- Pål 650 timer
- Rune 600 timer
- Kristian 450 timer
- Erik 600 timer

Dette er mer enn det som er påkrevd for hovedprosjekt ved HiG, men er ganske naturlig fordi prosjektet engasjerer mye og medlemmene er ute etter et bra resultat.

8.5 Oppdragsgiver

Våre oppdragsgivere har vært Kim Sønes og Frode Larsen hos Team G.E.D. A/S. Samarbeidet med de har fungert på en utmerket måte og de har alltid stilt opp hvis det skulle være noe. De satte oss tidlig inn på sporet og fikk forklart godt hva de ville ha. Selv om de har kommet med retningslinjer for systemet har de gitt oss frie tøyler når det gjelder utformingen av systemet. De har kun forklart oss hva de vil ha og det var opp til oss å stå for den tekniske løsningen. Mesteparten av kommunikasjonen har vært på møter, men kommunikasjonen har også foregått via e-post og telefon.

De har også vært våre test personer hos Team G.E.D. A/S. Det har vært meget nyttig siden det er de som skal sitte med F@T- admin når systemet settes i drift.

8.6 Veileder

Ved HiG har vi har hatt Harald Liodden som rådgiver under prosjektet. Vi synes at det har fungert tilfredstillende, og vi har hatt jevnlige møter med han omtrent hver 14 dag såfremt det ikke har vært noe spesielt. Siden han sitter med kompetanse innen databaser så var det naturlig av oss å søke litt hjelp der i begynnelsen av prosjektet med tanke på database designen. Selv om han ikke har noen erfaring med verken Java eller PHP, har han lang erfaring innen utvikling av data applikasjoner. Slik at han har også kunnet bidra med utforming av systemet, og da spesielt med tanke på hvordan grensesnittet burde se ut.

9 Konklusjon

Prosjektgruppen har utviklet en fungerende applikasjon for Team G.E.D. A/S. Resultatet av oppgaven har overgått gruppens forventinger til hvordan dette kunne løses, da ingen av oss hadde jobbet med et så stort prosjekt før. Det har vært nyttig å arbeide med en ekstern arbeidsgiver fordi vi har fått en følelse hvordan det fungerer i næringslivet.

Hovedtyngden i oppgaven var å analysere, systemere, designe, kode og teste et system som skulle forenkle arbeidet med å finne pris på servicer. Vi er meget fornøyd med at vi har klart å overholde de tidsfirster vi har satt og at vi har fått levert et system som nå skal ut i drift.

Prosjektoppgaven resulterte i et produkt som oppdragsgiver og forhåpentligvis skolen vil kunne dra nytte av senere. For vår egen del er vi meget fornøyd med hva vi har prestert og kommet fram til.

10 Referanser

1. K Beck & M. Fowler, Planning Extreame Programming, Addison Wesley, USA , 2000
2. http://www.extremeprogramming.org/rules/standupmeeting.html 29.03.2002 Don Wells
3. Rational, Rational Unified Process: Best Practices for Software Development Teams, URL, feb 2002, http://www.rational.com/products/whitepapers/100420.jsp
4. Objekt Management Group, 2002. Introduction to OMG's Unified Modeling Language http://www.omg.org/gettingstarted/what_is_uml.htm
5. A. Cookburn. Structuring Use Cases with Goals. URL, 2002. http://members.aol.com/acockburn/papers/usecases.htm
6. InstallAnywhere, www.zeroG.com, 23.04.2002
7. phpMyAdmin, http://phpwizard.net/projects/phpMyAdmin/, 27.01.02