HOVEDPROSJEKT:

BilBooking
CarAdmin

FORFATTERE:

Marius Ensrud
Ketil Gjerde
Yngve Solberg

DATO:

19.5.2005

SAMMENDRAG AV HOVEDPROSJEKT

Tittel:	BilBooking – CarAdmin	Nr. : 03 Dato : 19.5.2005
Deltaker(e):	Marius Ensrud Ketil Gjerde Yngve Solberg	
Veileder(e):	Førsteamanuensis Vegar Johansen	
Oppdragsgiver:	Electric Time Car (ETC)	
Kontaktperson:	Dag L. Solhaug	
Stikkord (4 stk)	Outlook, Exchange, web, reservering	

Antall sider: 37	Antall bilag: 48	Tilgjengelighet (åpen/konfidensiell): Åpen

Kort beskrivelse av hovedprosjektet:

ETC CarAdmin er en biladministreringsløsning utviklet for å effektivisere og lette kostnadene ved den interne biladministrasjonen i kommuner og bedrifter.

Det var her to hovedoppdrag vi fikk av oppdragsgiver.

Det ene var å bruke Microsoft Outlook sin kalender mot Microsoft Exchange, som har mulighet for ressursreservering, til å gi brukere av CarAdmin-løsningen mulighet til å reservere biler. Oppdrag to, var å bygge opp en webside som kunne gi en kalender visning til bruker og også gi muligheten til å reservere biler.

Begge disse delene skulle kommunisere med samme database og også oppdatere hverandre, slik at reserveringer på web kunne vises i kalenderen i Outlook og omvendt.

Forord

BilBooking er utviklet av tre studenter på datalinjen ved Høgskolen i Gjøvik våren 2005. Prosjektoppgaven ble utformet i samarbeid med oppdragsgiver Electric Time Car (ETC) representert ved Dag L Solhaug, og har bestått i å videreutvikle et system for reservering av biler gjennom Outlook kalenderen og web.

Vi ønsker å takke følgende personer for deres bidrag til prosjektet

- Vegar Johansen, vår veileder, som under hele prosjektperioden har vært interessert og engasjert i oppgaven vår, og kommet med mange gode råd.

- Dag L Solhaug, vår oppdragsgiver, som også under hele prosjektperioden har vært interessert og engasjert. Dag har kommet med mange gode råd og vink til hvordan rapport og systemet skal utformes.

- Øyvind Johansen som er ansatt ved ETC, har gitt verdifulle innspill i forbindelse med detaljer når det gjelder koding.

KETIL GJERDE

YNGVE SOLBERG

MARIUS ENSRUD

Innhold

Figurer

Tabeller

1 Innledning

ETC CarAdmin er en biladministreringsløsning utviklet for å effektivisere og lette kostnadene ved den interne biladministrasjonen i kommuner og bedrifter. Et foretak som leier/leaser firmabiler, ønsker å fordele bruken over bilene best mulig, for best utnyttelse. Et eksempel er kilometerutjevning: CarAdmin sørger her for at alle biler innen samme mnd kjører tilnærmet likt, da lik kjørelengde på alle biler gir best biløkonomi. CarAdmin tar også hensyn til andre faktorer rundt den interne biladministrasjonen i den enkelte bedrift og kommune, som: booking, kjørebok, serviceoppfølging, oversikter +++.

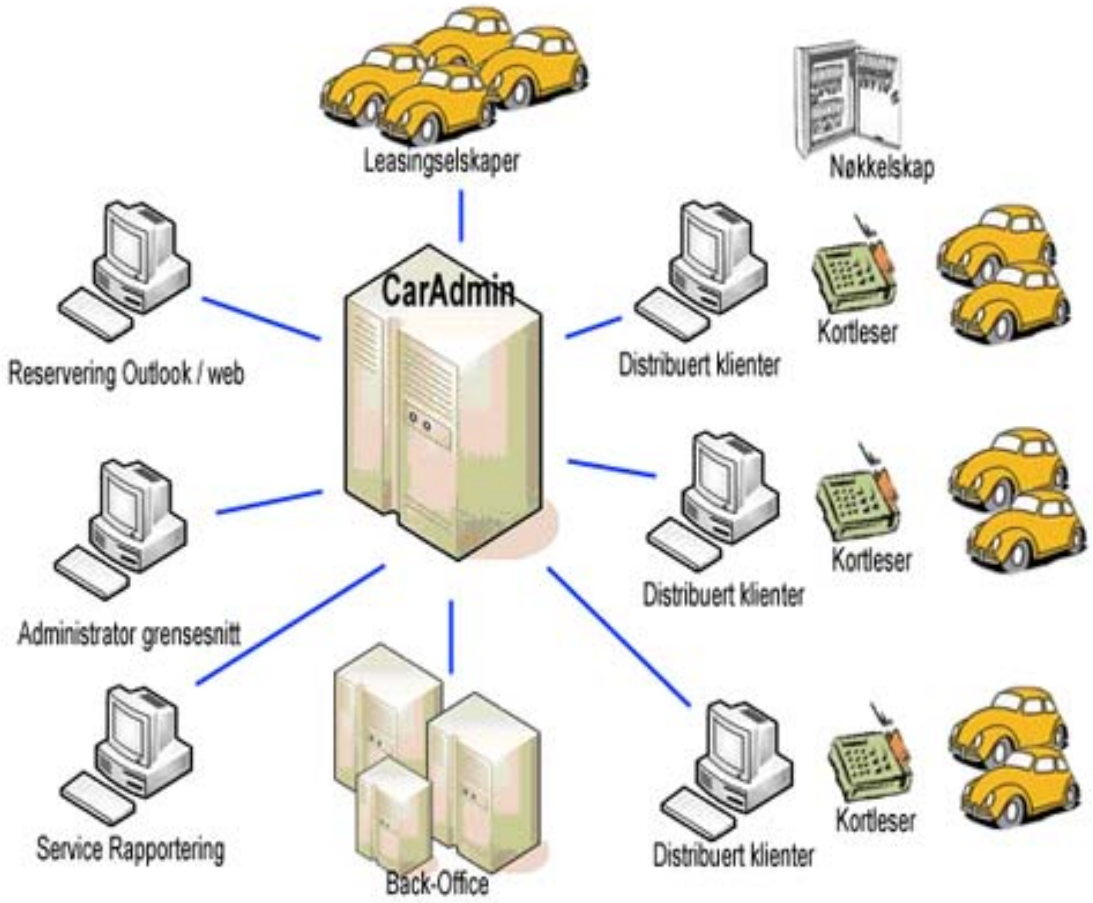
Vår oppgave er å lage to moduler for reservering av bilene: Modul basert på Outlook-kalender og en web-basert modul, som begge skal samhandle med CarAdmin sin adhock booking. Et viktig element er mulighet for bilgruppe reservering. Som et eksempel vil en bruker reservere en varebilgruppe via en av våre moduler. CarAdmin vil så hente ut (rekalkulere) den bilen som er best egnet, og denne bilen vil tildeles brukeren.

1.1 Oppgaven

Oppgaven vår er å lage to moduler for reservering av bilene i CarAdmin løsningen. Outlook med sin kalender, og en webside med tilsvarende brukergrensesnitt som kalenderen i Outlook. For å ha best utnyttelse av bilparken vil det være gruppereservering, men også mulighet for enkeltbilreservering. Ved gruppereservering vil det bli en bedre fordeling mellom bilene, så dette er en foretrukket reserveringsmetode.

1.1.1 Bakgrunn

Bakgrunnen til vår oppgave, henger sammen med hva som er gjort med CarAdmin fra før. ETC har allerede utviklet en adhoc modul (en terminal men kortleser) der bruker skal hente ut bilen, mens vår oppgave blir å se på reserveringen. Samhandlingen mellom disse to funksjonene har vært en forutsetning for våre reserveringsmoduler.



Figur 1-1- CarAdmin og dens omgivelser

1.1.2 Beskrivelse

En bruker, det vil si noen som har et ønske om å leie en bil for en periode, skal ha mulighet for å reservere en bil innenfor en periode, enten via Outlook kalender eller via et webgrensesnitt som han kan åpne i en nettleser, for eksempel Internet Explorer. Ved hjelp av disse metodene skal man få muligheten for å velge tidspunkt og biltype for sitt behov. Når brukeren skal hente bilen, går han til en distribuert klient med en kortleser som vil fortelle brukeren hvilken bil han skal ha.

1.2 Egen bakgrunn og kompetanse

Prosjektgruppen består av medlemmer fra studieretningen for dataingeniør på Høgskolen i Gjøvik, herunder drifts og programmeringslinjen. Gruppen har kompetanse både innen drift og oppsett av server, programutvikling i Java og C++, samt litt generell kompetanse innenfor systemutvikling.

1.3 Målgruppe for rapport

Målgruppen for rapporten vil være oppdragsgiver og sensor. Dette i på grunn av de faglige og tekniske sidene av prosjektet, men allmennheten vil også ha nytte av innledning og kravspesifikasjonen.

1.4 Organisering av rapporten

1. Kapittel 1 Innledning

Det innledende kapittelet med en overordnet beskrivelse av oppgaven. Kapittelet sier også noe om gruppens sammensetning og kompetanse. Videre beskriver kapittelet kort hva de andre kapitlene inneholder.

2. Kapittel 2 Kravspesifisering

Kapittelet inneholder en oppsummering av kravspesifikasjonen for systemet. Den komplette kravspesifikasjonen er vedlagt i eget vedlegg: se vedlegg B.

3. Kapittel 3 Design

Dette kapittelet inneholder beskrivelser av systemarkitektur, datastruktur og databasestruktur

4. Kapittel 4 Implementering

Kapittelet beskriver valg av utviklingsverktøy.

5. Kapittel 5 Kvalitetssikring og testing

Her beskrives hva prosjektgruppen har utført for å ivareta den ønskede kvaliteten på systemet.

6. Kapittel 6 Diskusjon av resultater

Egen vurdering av prosjektet og prosjektgruppens arbeid. Inneholder en vurdering av resultatet målt opp mot kravspesifikasjonen og drøfting av avvik i forhold til kravspesifikasjonen, og utfordringer vi har møtt på.

7. Kapittel 7 Konklusjon

Oppsummering av prosjektet og de erfaringer vi har gjort oss.

8. Kapittel 8 Referanser

Oversikt over litteratur vi har benyttet underveis i prosjektet.

9. Kapittel 9 Vedlegg

Oversikt over alle vedleggene til prosjektrapporten.

2 Kravspesifikasjon

Dette kapittelet er et ekstrakt av vår kravspesifikasjon. For å lese hele kravspesifikasjonen se vedlegg B.

2.1 Innledning

Det er to hoveddeler vi har fått i oppgave å utvikle, en Outlook del og en web del, og at data synkroniseres mellom disse.

2.1.1 Outlook/Exchange

Figur 2-1- Skjermbilde av Outlook kalender

Her skal det utvikles en modul rundt kalender grensesnittet til Outlook, slik at reservering av en bilgruppe eller en enkelt bil skal være mulig, dette for å gjøre det enkelt for brukere og administratorer å benytte seg av systemet og å administrere det. Det er et mål å gjøre minst mulig endringer på MS Windows server og MS Exchange for å få det til å fungere.

Innholdet i de forskjellige kalenderne i Exchange må da også tilsvare det som ligger i databasen i CarAdmin løsningen. Det må også forhindres at det kan skje dobbeltreserveringer i noen form.

Vi har to typer reservering som kan benyttes i Outlook, enten reservering av en spesifikk bil, eller gruppereservering, som er reservering av en type bil, som for eksempel varebil.

2.1.2 Web

Med web skal det lages en side som vil gi en tilsvarende funksjonalitet som Outlook, med en kalendervisning og mulighet for å reservere. Det må i tillegg være en administrator mulighet, der administratorbrukeren vil få mulighet til å se kalenderen til andre brukere og også legge inn reserveringer for andre.

2.1.3 Rekalkulering

Ved nye/endringer i reserveringer eller for sent innlevert, ikke aktiviserte reserveringer og nye reserveringer skal løsningen rekalkulere hvilken bil som er booket til hvem slik at en hele tiden oppnår optimal utnyttelse av bilparken.

2.1.4 Påfølgende reserveringer

En person skal på flere møter etter hverandre og vedkommende trenger bil til alle møtene. Da skal bookingmodulen sørge for at vedkommende får samme bil på alle møter. Her kan det forekomme at bilene må byttes om (rekalkuleres) mellom de som har booket innen et tidsintervall, slik at ingen behøver å bytte bil fra det ene til det andre møtet.

Det kan være timers mellomrom mellom møtene, så ved innlevering i mellomtid trenger en ikke ta hensyn til at vedkommende skal ha samme bil. Dersom en bil ikke leveres inn mellom møtene bør det på forhånd være ryddet plass for det.

2.1.5 Single sign on

”Single sign on” brukes for å minimere pålogging på de forskjellige systemer. Og derfor et krav mot web, for å spare brukeren mot å logge seg inn på nytt. LDAP vil bli brukt som påloggingsmetode.

2.2 Kort om krav til prioritering

Med en Exchange server, og en database, må det inn en synkronisering mellom databasen og Exchange sin egen database. I denne sammenheng vil Exchange bli nedprioritert i motsetning til web, som vil ha en direkte kobling mot hoveddatabasen. En prioritering kommer inn i bilde

hvis flere brukere prøver å reservere samme bil. Siden synkroniseringen mot Exchange er en tidkrevende affære, må Outlook ha lavere prioritet en web.

2.3 Målgruppe for løsning

ETC har definert sin målgruppe mot foretak med firmabiler som har 3 eller flere brukere.

Prioriterte målgrupper for CarAdmin-løsningen er:

- Kommuner
- Lensmannsetater
- Private foretak med over 5 firmabiler

Felles for alle målgruppene er at de har bilparker som helt eller delvis benyttes sekvensielt av forskjellige brukere.

2.4 Kravtabell

Outlook/Exchange	Ansvarlig	Status	Merknad
Legge til reservering			
Legge til reservering i db	Ketil	✓	
Legge reservering til i kalender i Exchange	Ketil	✓	
Endre reservering			
Endre reservering i db	Ketil	✓	
Endre reservering i kalender i Exchange	Ketil	✓	
Slette reservering			
Slette reservering i db	Ketil	✓	
Slette reservering fra kalender i Exchange	Ketil	✓	
Legge til bruker ved førstegangsbruk	Ketil	✓	
Web			
Legge til reservering			
Legge til reservering i db	Ketil/Marius	✓	
Legge reservering til i kalender i Exchange	Ketil/Marius	✓	
Slette reservering			
Slette reservering i db	Ketil/Marius	✓	
Slette reservering til i kalender i Exchange	Ketil/Marius	✓	
Vise kalender	Marius	✓	
Administrator			
Legge til reservering	Marius	%	Se kap. 6.3.4
Vise kalender	Marius	✓	
Synkronisering	Marius	✓	
Rekalkulering	Ketil	%	Se kap. 6.3.12
Påfølgende reserveringer	Alle	✗	Se kap. 6.3.1
Single sign on	Alle	✗	Se kap. 6.3.3
Dokumentasjon	Alle	✓	
Rapport	Alle	✓	

Tabell 2-1 – Kravtabell

(✓ står for fullført, ✗ står for avvik og % står for delvis fullført)

3 Teknisk Design

3.1 Innledning

I starten hadde vi delt inn arbeidet i fem deler, Exchange Sync, Outlook reservering, web reservering, web administrasjon og LDAP (Lightweight Directory Access Protocol), alle steg i en inkrementell utviklingsmodell.

Av de opprinnelige fem inndelingene endte vi opp med tre inndelinger:

- SyncServer
- webside
- COM+ applikasjon.

Grunnen til denne forandringen er at vi ikke hadde noe erfaring med MS Exchange, MS Outlook og LDAP. Websiden ble også laget på en slik måte at administrasjonen ble så nært den første reserverings siden, at de gikk under samme punkt.

3.2 Opprinnelig inndeling

3.2.1 Exchange Sync

Utvikle en server applikasjon for å synkronisere data mellom database og Exchange. Dette er det som er kalt SyncServer under endelig inndeling.

Deloppgaver til Exchange Sync:

- Webdav
- Sockets
- mySQL

Se kapittel 3.3.1 for endelig inndeling av SyncServer.

3.2.2 Outlook reservering

Her skal det utvikles en COM+ applikasjon på Exchange serveren for ”overvåking” av kalender, også kalt COM+ applikasjon under endelig inndeling. Dette var det punktet som var mest utfordrende og nytt for oss. Mye tid gikk med på utvikling av denne delen. I hovedtrekk er dette en DLL-fil som lytter på kalenderen i MS Outlook/Exchange.

Deloppgaver til Outlook reservering:

- Automatisk slette reserveringer
- Automatisk godkjenne reserveringer
- Sende kommandoer til SyncServer

Se kapittel 3.3.3 for endelig inndeling av COM+ applikasjonen

3.2.3 Web reservering

Utvikle en webside, et alternativ for Outlook kalenderen.

Deloppgavene til websiden var:

- Ny reservering
- Slette sine reserveringer
- Kalender

Se kapittel 3.3.2 for endelig inndeling av websiden

3.2.4 Web administrering

Utvikle en webside for administrator. En administrator oversyrer andre brukere.

- Ny reservering, med valg av bruker
- Slette reservering, med valg av bruker
- Kalender, med valg av bruker

Se kapittel 3.3.2 for endelig inndeling av websiden

3.2.5 LDAP

Her skulle vi utvikle en modul for pålogging av web mot LDAP server. Dette ble bakt inn i websiden, siden dette ble en mye mindre oppgave enn vi først var klar over. I den opprinnelige planen var dette ment som en del av ”Singel Sign On”, noe vi senere gikk vekk ifra.

- Felles brukernavn/passord

Se kapittel 3.3.2 for endelig inndeling av websiden

3.3 Endelig inndeling

Den inndelingen vi endte opp med etter mer erfaring rundt de forskjellige temaene, var en tredelt inndeling med SyncServer, webside og COM+ applikasjon.

3.3.1 SyncServer

SyncServer er ikke mye forandret fra opprinnelig inndelig, men noen punkter ble lagt til underveis.

Hovedpunktene til SyncServer er:

- Webdav og kommunikasjon mot MS Exchange.
- Socket og lytting på kall fra COM+ applikasjonen.
- mySQL, legge til, endre eller slette data i CarAdmin sin database.

I tillegg til hovedpunktene over, ble følgende punkter tatt inn:

- GUI
  - Hovedvindu med knapper og meny
  - TrayIkon med meny
  - Informasjonsboble
- Konfigurasjonsfil
- Logging mot konsoll og fil, denne også konfigurerbar.

3.3.2 Webside

Den største forandringen fra opprinnelig inndeling, er at administrator og brukerdelen har blitt satt inn i samme del/modul. Dette er fordi administratordelen ble mye mindre enn først antatt.

En egen innloggingsklasse ble laget, her med en metode som kunne bli kalt og som ga sann eller usann om bruker er en administrator eller en vanlig bruker. Dermed var det enkelt i gi riktige rettigheter, og vise riktig grafikk.

Figur 3-1- Administrators mulighet til å velge brukere på websiden

I tillegg ble lagdeling innført. Dette vil si at alle data ligger i Java beans (Java-filer som kalles i jsp-filene), og all HTML og grafikk i JSP.

Funksjonaliteten til websiden er:

- Legge til y reservering.
- Slette reservering.
- Vise reserveringer i en kalender.
- Administrator skal ha mulighet for legge til, slette og vise reserveringer for andre brukere.

I tillegg kom:

- Oppretting av en konfigurasjonsfil.
- Logging mot konsoll og fil, denne skal også være konfigurerbar.

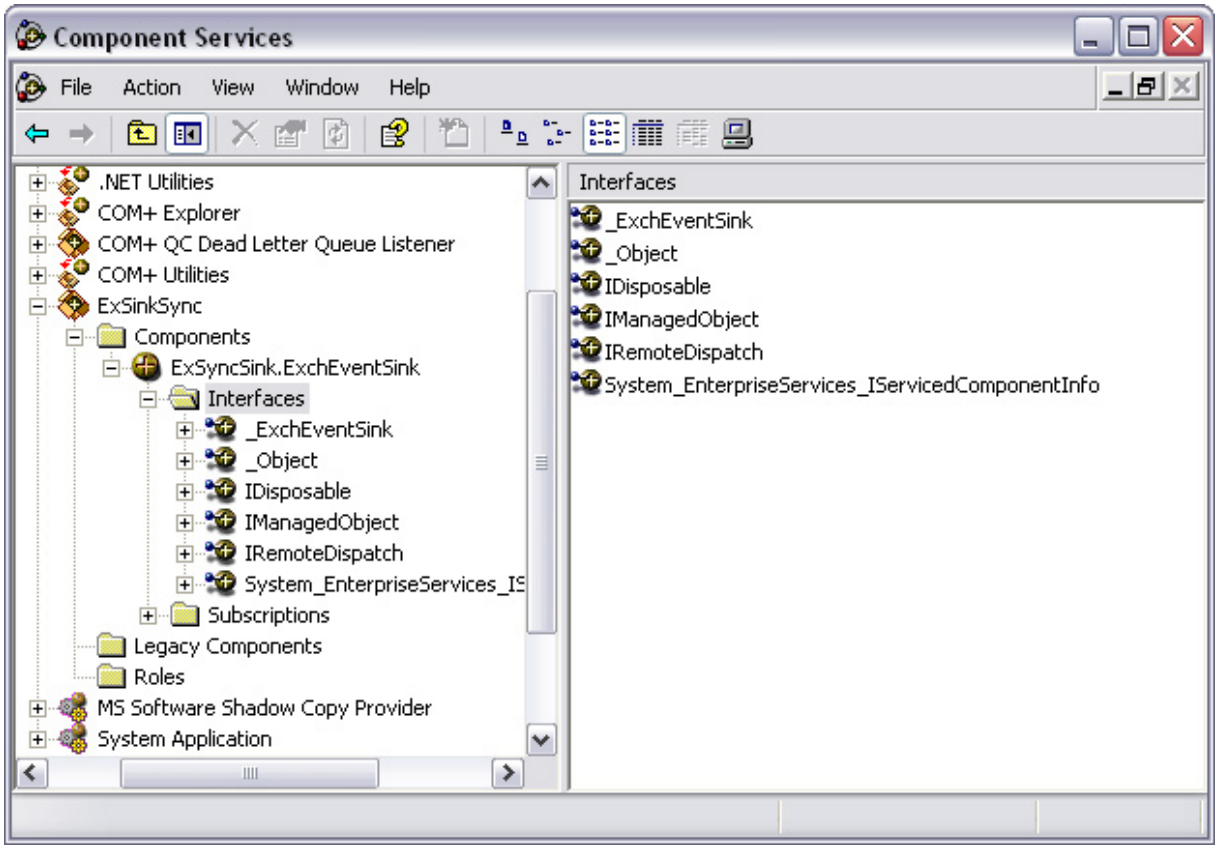
3.3.3 COM+ applikasjon

Denne delen skulle fange opp endringer i kalenderen til alle de bilene eller bilgruppene som det skal være mulighet til å reserveres på, og sende disse endringene videre til SyncServer for videre behandling av data.

Først var denne tenkt å overvåke alle kalenderne. Outlook skulle selv da stå for å automatisk godkjenne disse reserveringene, som beskrevet i "How to create a resource account" [32]. Vår applikasjon skulle bare sende data videre til SyncServer. "Developing Managed Event Sinks/Hooks for Exchange Server Store using C#" [33] ble brukt som en nyttig guide til å kode applikasjonen.

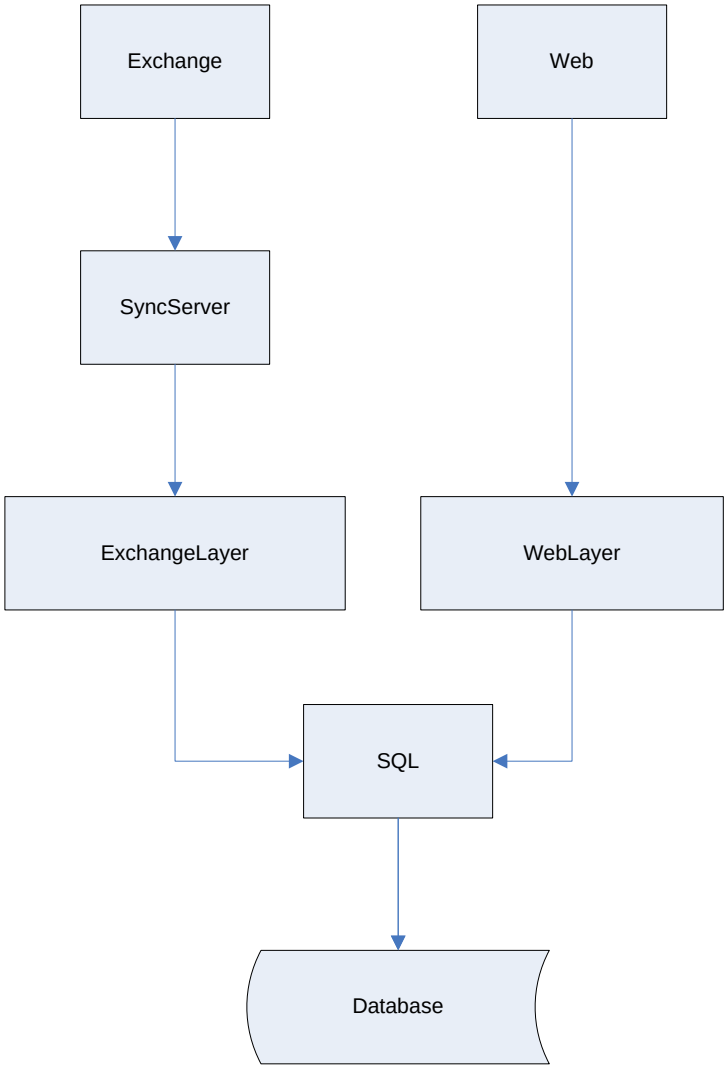
Etter hvert viste det seg at denne automatiske godkjenningen i Outlook ikke var god nok for vårt bruk da denne ikke fungerer når webdav brukes for å legge til nye reserveringer. Vi måtte da gjøre noen endringer for at vår COM+ applikasjon også skulle stå for dette. "Autoaccept Sink for Exchange" [34] ble brukt som et eksempel så vi kunne se hvordan dette kunne gjøres.

Vi endte opp med at vår applikasjon bare overvåker innboksen til den som er satt opp som en biladministrator. Denne kontoen vil være en konto som mottar alle reservinger eller avbrutte reserveringer i innboksen sin, og applikasjonen vår behandler disse automatisk, og sender informasjonen videre til vår SyncServer.



Figur 3-2- Skjermbilde av COM+ applikasjon

3.4 Software arkitektur

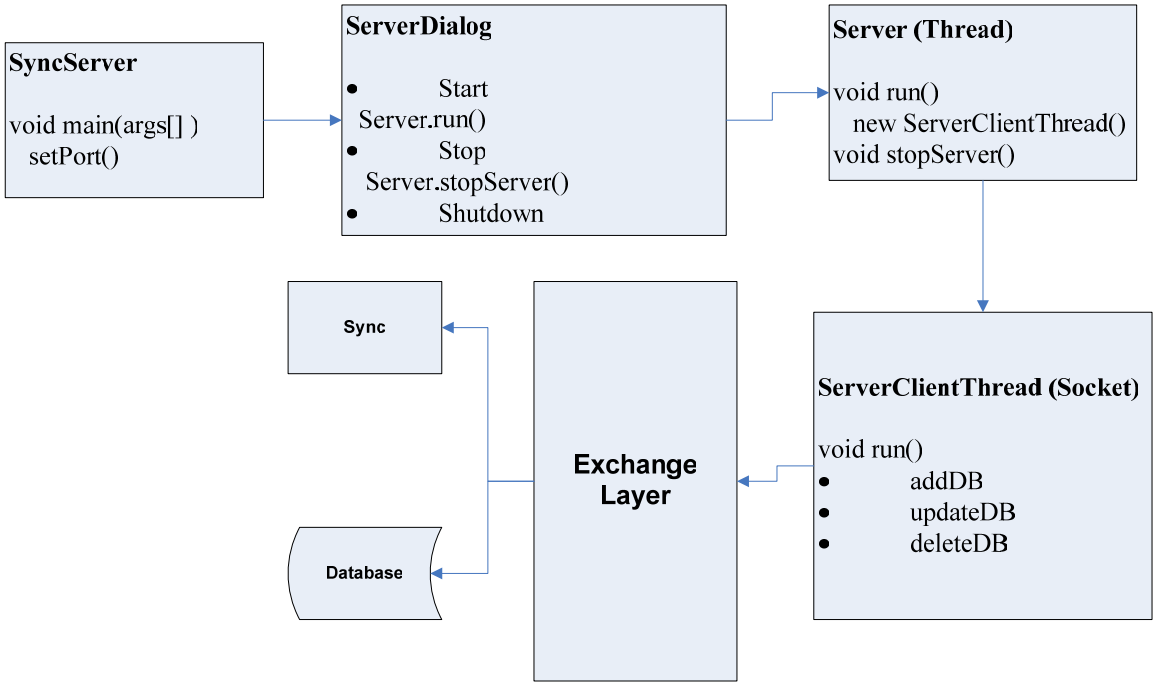


Figur 3-3- Overordna oversikt over arkitektur

3.4.1 SyncServer

SyncServer er det laget som står for å ta imot data fra Exchange og sørge for at Exchange og database inneholder likt og synkronisert innehold.

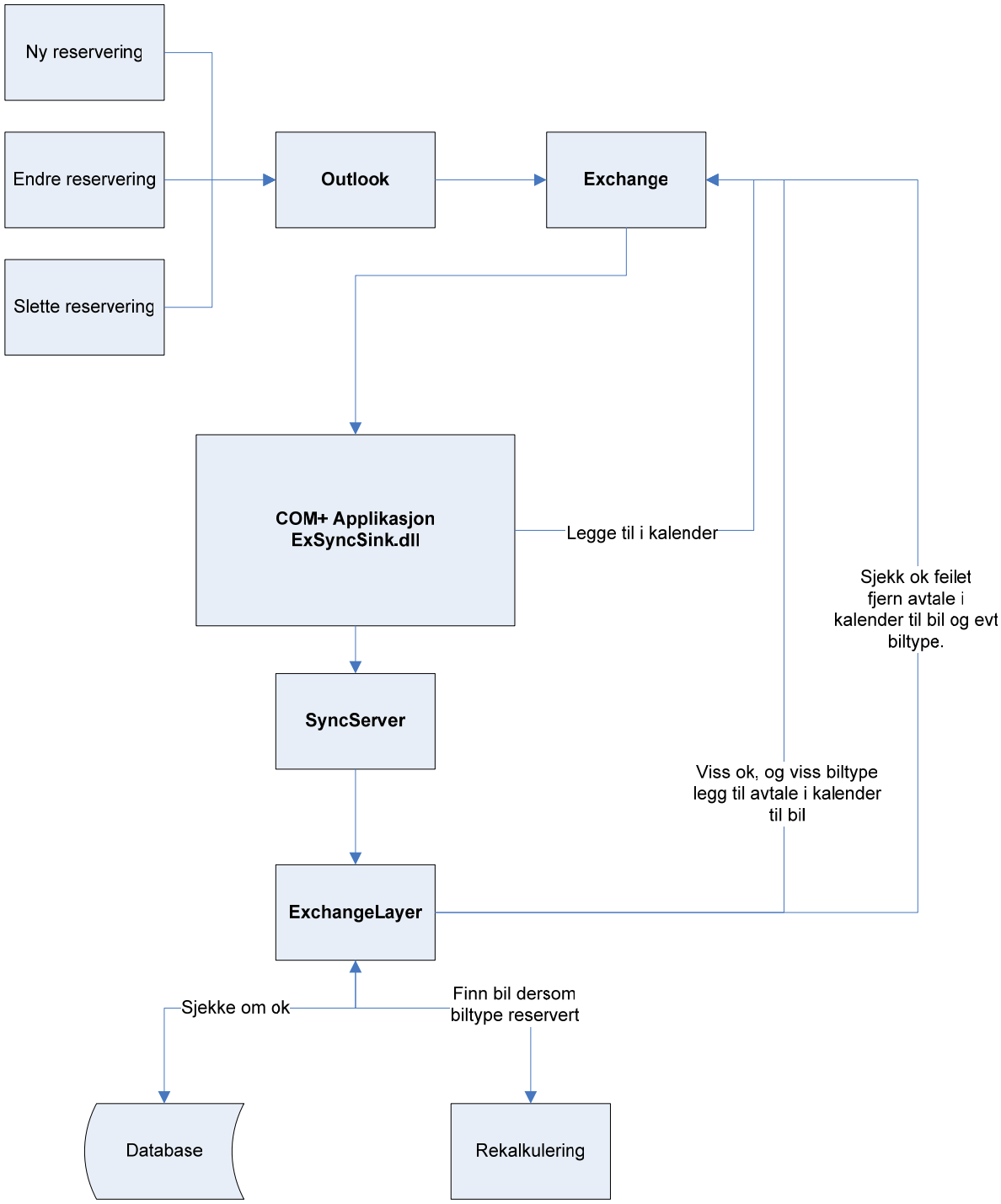
Figur 3-4 viser en oversikt over kjernen av synkroniseringsserveren. Dette er klasser som beskriver hvordan man starter opp serveren og hvilke ”socket-kall” man kan gi serveren.



Figur 3-4- Kjernen av SyncServer

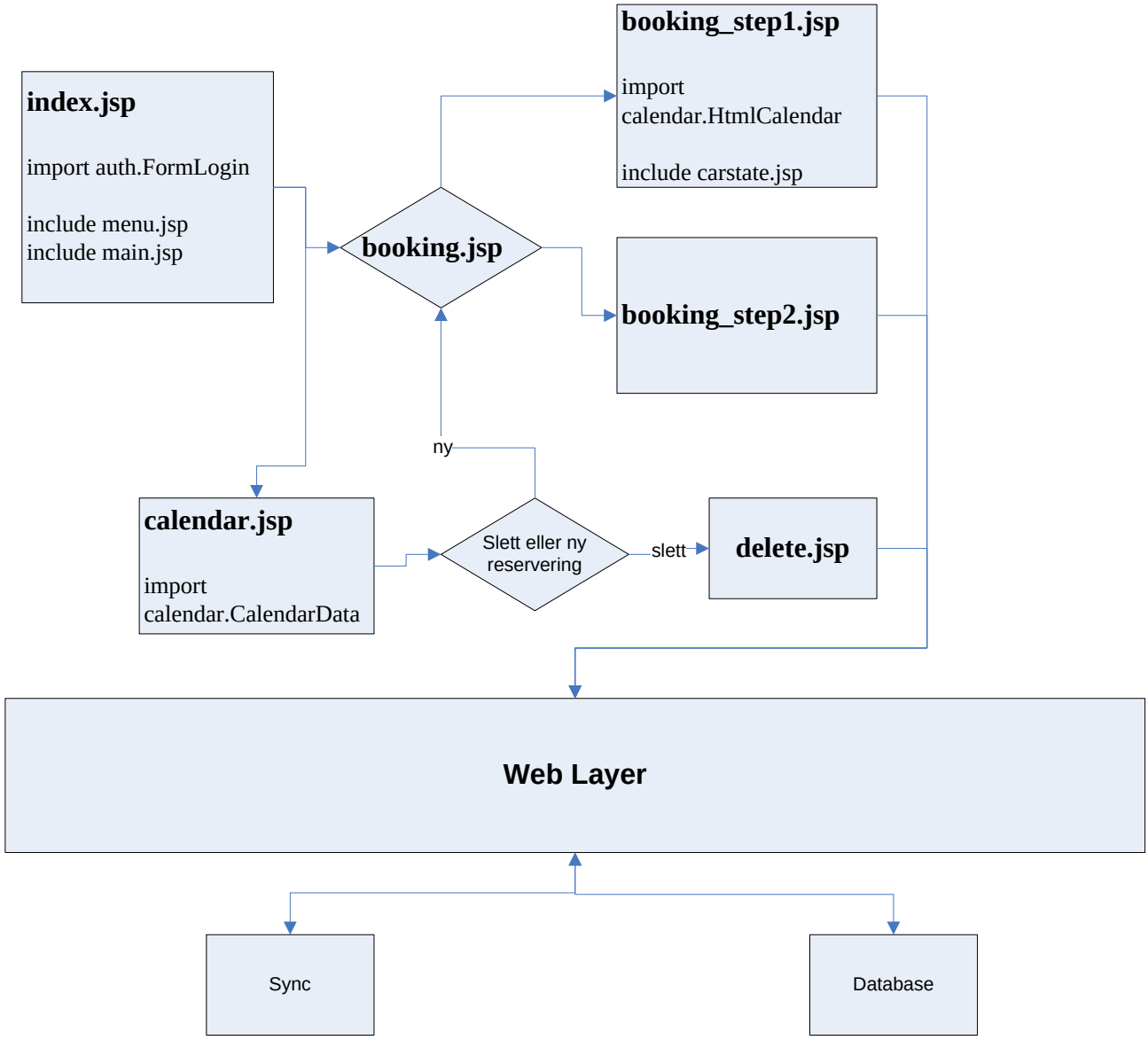
3.4.2 Outlook reservering (COM+ / SyncSink)

Figur 3-5 viser oversikt over hva som skjer ved en reservering i Outlook kalender.



Figur 3-5- Oversikt over COM+ applikasjonen

3.4.3 Web reservering



Figur 3-6- Kjernen av websiden

3.4.4 Eksternt innhentet Javakode / Javabibliotek

For å ha en enklere mulighet for videre utvidelser og oppfølging, valgte vi å bruke bibliotek som var ferdig utviklet.

- *commons-httpclient.jar [25]*
  Denne ble brukt til kommunikasjon med basis pålogging og kryptert NTLM (NT Lan Manager) metoder mot Exchange serveren.

- *jakarta-slide-webdavlib-2.1.jar [26]*
  Dette er et bibliotek som inneholder mange nyttige metoder for webdav. Ble brukt under kommunikasjonen fra SyncServer til Exchange.

- *jdic.jar*
  JDesktop **I**ntegration **C**omponents, ble brukt som utvidelse av GUI, der trayicon med meny og informasjonsboble ble brukt.

- *jdom-1.0.ja*
  Dette ble brukt til xml håndteringen rundt webdav. Er et enkelt verktøy for lesing, endring og skriving av xml dokumenter.

- *log4j-1.2.9.jar [24]*
  Dette er et solid loggeverktøy for Java. Muligheter for logging mot fil og console (skjermbilde), samt store konfigurasjonsmuligheter.

- *mysql-connector-java-3.1.7-bin.jar [3]*
  Ble brukt for å kommunisere mot mySQL database.

- *xml-im-exporter1.1.jar*
  Dette biblioteket ble brukt for xml håndteringen rundt webdav

3.5 Konfigurasjonsfiler

For å gjøre lagring av server adresser, passord, brukere og lignende, så enkelt som mulig, har vi en konfigurasjonsfil pr. modul. (SyncServer, Web og COM+). Web og SyncServer vil bruke samme konfigurasjonsfil, fordi de jobber mot samme MySQL server og fordi det vil være naturlig å kjøre disse på samme server.

Denne filen inneholder innstillinger rundt tilkobling mot MySQL, Exchange og reserveringsregler. Se vedlegg C for et eksempel på hvordan denne filen kan se ut.

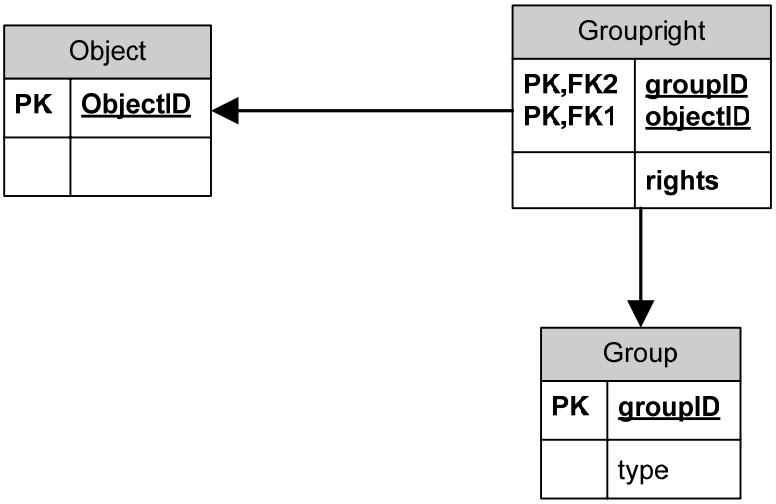
COM+ modulen har en tilsvarende konfigurasjonsfil, men da med innstillinger mot SyncServer og logging. Se vedlegg D for et eksempel på hvordan denne filen kan se ut.

3.6 Database struktur

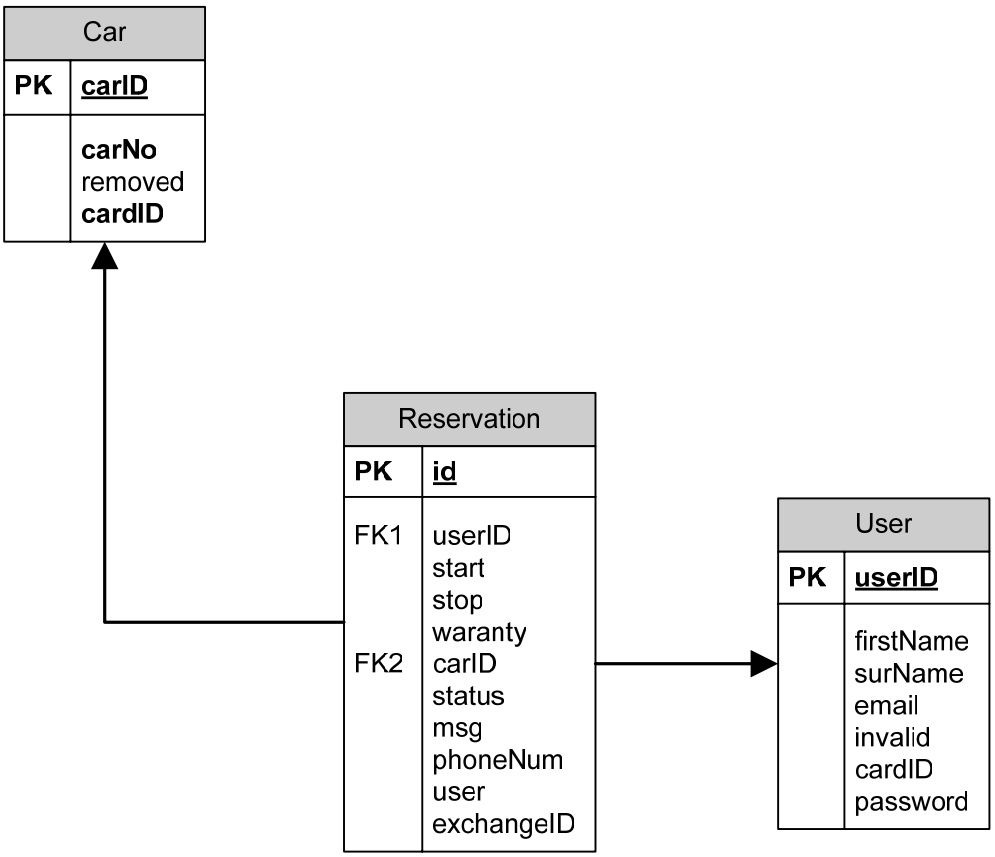
MySQL server brukes mot web og SyncServer. Dette er et gratis databaseserverprogram som er mye brukt mot web. MySQL var vi godt kjent med fra tidligere erfaringer med webprogrammering, samt at ETC allerede brukte den til sin adhoc applikasjon, slik at dette det et naturlig valg.

Fordi adhoc applikasjonen fra ETC var mer eller mindre klar før vi begynte, har vi bare gjort mindre forandringer i den allerede ferdige datastrukturen. Disse forandringene var nødvendige for å tilpassningen mot Exchange og SyncServer (”exchangeID” i tabell ”Reservation” på Figur 3-8). Det ble også utvidet med noen felter ved pålogging mot web som et alternativ til LDAP (”password” i tabell ”User” på Figur 3-8).

Se vedlegg F for en dump av MySQL server.



Figur 3-7- Oversikt over tabeller med bil og gruppe



Figur 3-8- Oversikt over tabeller med reservasjon

3.7 SyncServer Socket struktur

For kommunikasjon fra Exchange til SyncServer, har vi mellom COM+ applikasjonen og SyncServer, en ukomprimert socket connection. Dette ble gjort for å få en enkel behandling av dataene kommer inn, men også fordi en komprimering/kryptering ikke var nødvendig med så små datamengder.

Dette er input som SyncServer tar:

/deleteDB <kalender> <exchangeID >

/addDB < kalender > <exchangeID > <start> <slutt> <bruker>

/updateDB <sqlID> <gruppe> <exchangeID > <start> <slutt> <bruker>

Se vedlegg G for nærmere beskrivelse av kallene.

4 Implementering

4.1 Valg av verktøy

For å gjennomføre et systemutviklingsprosjekt er det behov for forskjellige typer verktøy. Man trenger en kompilator, programvare for versjonsstyring, figurtegning, generering av hjelpefiler og dokumentasjon.

4.2 Programmeringsspråk

4.2.1 SyncServer

Fra oppdragsgiver var det gitt at vi skulle bruke Java i utviklingen av det programmet som skulle stå for synkroniseringen i løsningen vår (SyncServer), dette på grunn av andre ferdigutviklede moduler i CarAdmin. Med tidligere erfaring med Java var ikke dette noe problem for oss. Java har også den fordelen at den er plattformuavhengig.

4.2.2 Web

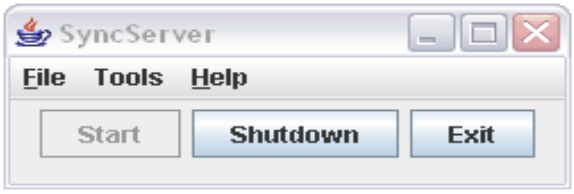
Med Java i SyncServer var det gitt fra oppdragsgiver at JSP skulle brukes i utviklingen av websiden.

4.2.3 COM+

Når det gjaldt COM+ applikasjonen hadde vi et par å velge mellom, nemlig C#.NET og VB.NET. Alle programmeringsspråk utviklet av Microsoft. VB hadde vi lite erfaring med, så vi landet på C#. Dette er et programmeringsspråk som er likt Java, som gav oss en fordel siden dette vi var kjent med dette. En annen fordel var at mye av referansekoden vi fant på nettet (Google / MSDN) også var utviklet i C#.NET. Prosessen med å sette seg inn i det nye utviklingsmiljøet gikk overraskende fort, og allerede etter få dager var vi i stand til å starte utviklingen for fullt.

4.3 GUI SyncServer

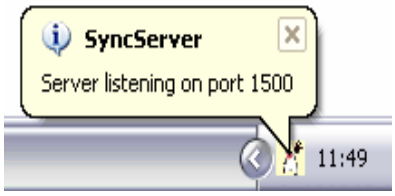
SyncServeren har en enkel betjening med funksjonene Start, Stop og Shutdown. I tillegg vil det være tilgjengelig ”alternativer” fra menyen, der de samme innstillingene fra konfigurasjonsfilen også vil være enkelt tilgjengelig for de som skal drifte systemet i etterkant.



Figur 4-1- GUI av SyncServer (hovedvindu)

Trayikon ble implementert slikt at hovedvindu kan minimeres fra oppgavelinjen, men også mulighet til å tas opp igjen fra ikonet. En meny, ved høyreklikking på ikonet, gjør at serveren blir enda enklere å betjene.

Figur 4-2- GUI av SyncServer (TrayIkon med meny)



Figur 4-3- GUI av SyncServer (TrayIkon med informasjonsboble)

4.4 Beskrivelse av programverktøy under utviklingen

Eclipse [1]
Brukt til å skrive og kompilere kildekode i Java, og generering av Javadoc dokumentasjon. Eclipse av implementer CVS klient, som vi brukte til versjonsstyring.

MyEclipse [2]
Er en plugin til Eclipse. Brukt får skrive og håndtere JSP/Web-filer samt Java kode til brukt under websidene.

MySQL [3]
Database som vi og ETC bruker til datalagring.

Microsoft Project [8]
Dette programmet ble brukt til å lage Ganttskjemaer.

Microsoft Visual Studio .NET 2003 [12]
Brukt til å skrive og kompilere kildekode i C#.NET, og generering av XML-dokumentasjon.

Microsoft Windows XP [5]
Operativsystemet som applikasjonene er utviklinget på.

Microsoft Server 2003 (Enterprise) [6]
Trengte en Microsoft Server får å kjære Exchange. Valgte derfor en 180-dagers trial versjon for å slippe kostnadene med lisenser.

Microsoft Word [7]
Brukt til rapportskriving.

Microsoft Exchange 2003 (Enterprise) [10]
Trengte Exchange server til testing, og sikring av produktet. 180-dagers trial versjon for å slippe kostnadene med lisenser.

PuTTy Secure Copy Client [17]
Ble brukt som test verktøy av socket-connection med SyncServeren.

Ethereal [18]
Brukt til pakkesniffing av webdav- og socket-headere mellom SyncServer og Exchange.

Mistaya [19]
Er en WebDav-explorer. Brukt til sikring av dataflyt og utforsking av webdav.

Adobe Photoshop [16]
Brukt til utvikling av grafikk i websidene.

Softerra LDAP Browser 2.5.3 [35]
Verktøy brukt til å utforske LDAP på en Windows server maskin.

MySQL Front [30]
Brukt for å få en litt enklere frontend mot mySQL databasen.

4.5 Utviklingsmiljø

Selv om det er fullt mulig å skrive både Java, JSP og C#.NET-kode i en enkel teksteditor, som for eksempel notepad, og manuelt kompilere kildekoden med kommandolinjekompilatoren javac og csc som er fritt tilgjengelig fra Microsoft [5] og Sun [21], er ikke dette den mest effektive arbeidsmåten. I de fleste tilfeller vil man kunne jobbe mye mer effektivt i et IDE.

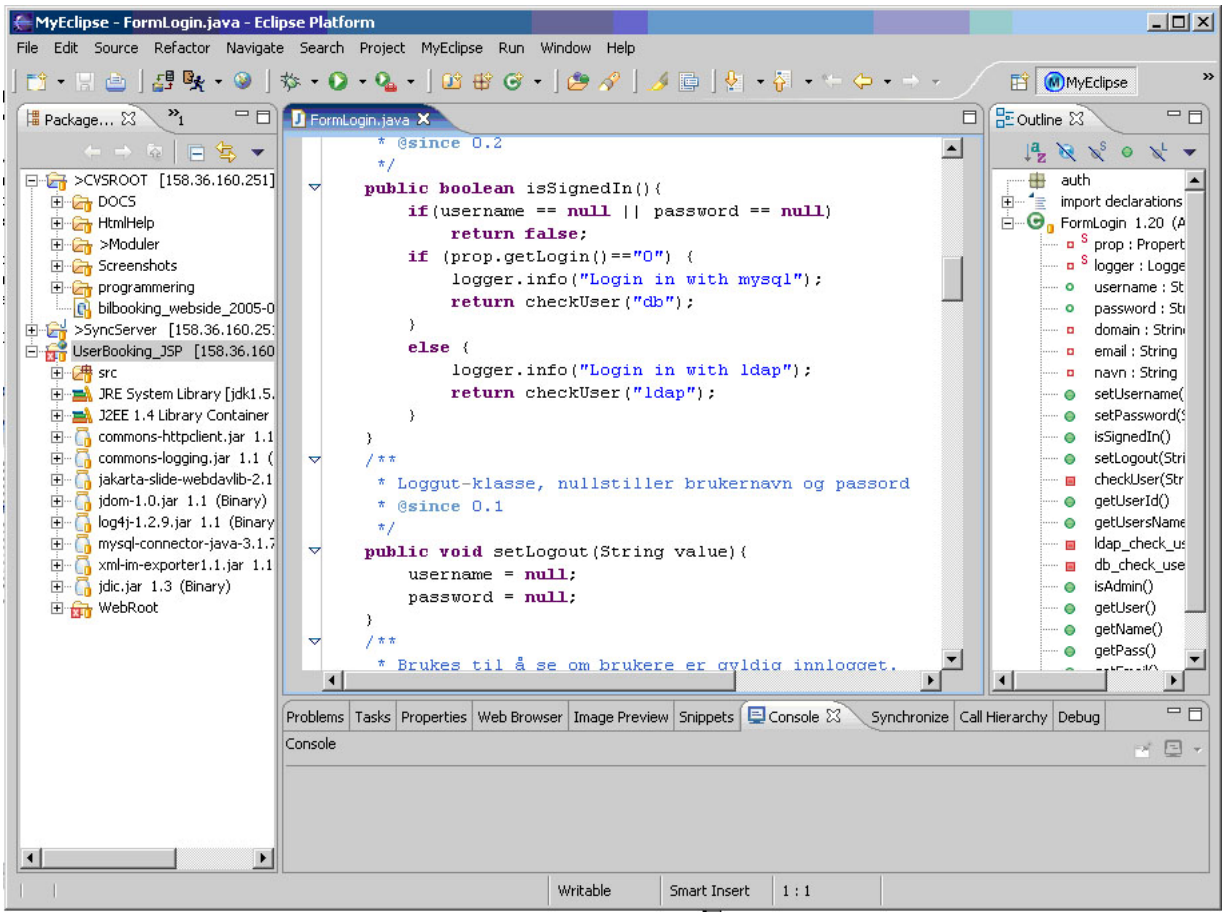
4.5.1 IDE

I en IDE man kan skrive og kompilere fra det samme grafiske brukergrensesnittet, og også gjerne har mer avanserte funksjoner for debugging osv.

4.5.2 Våre valg av IDE

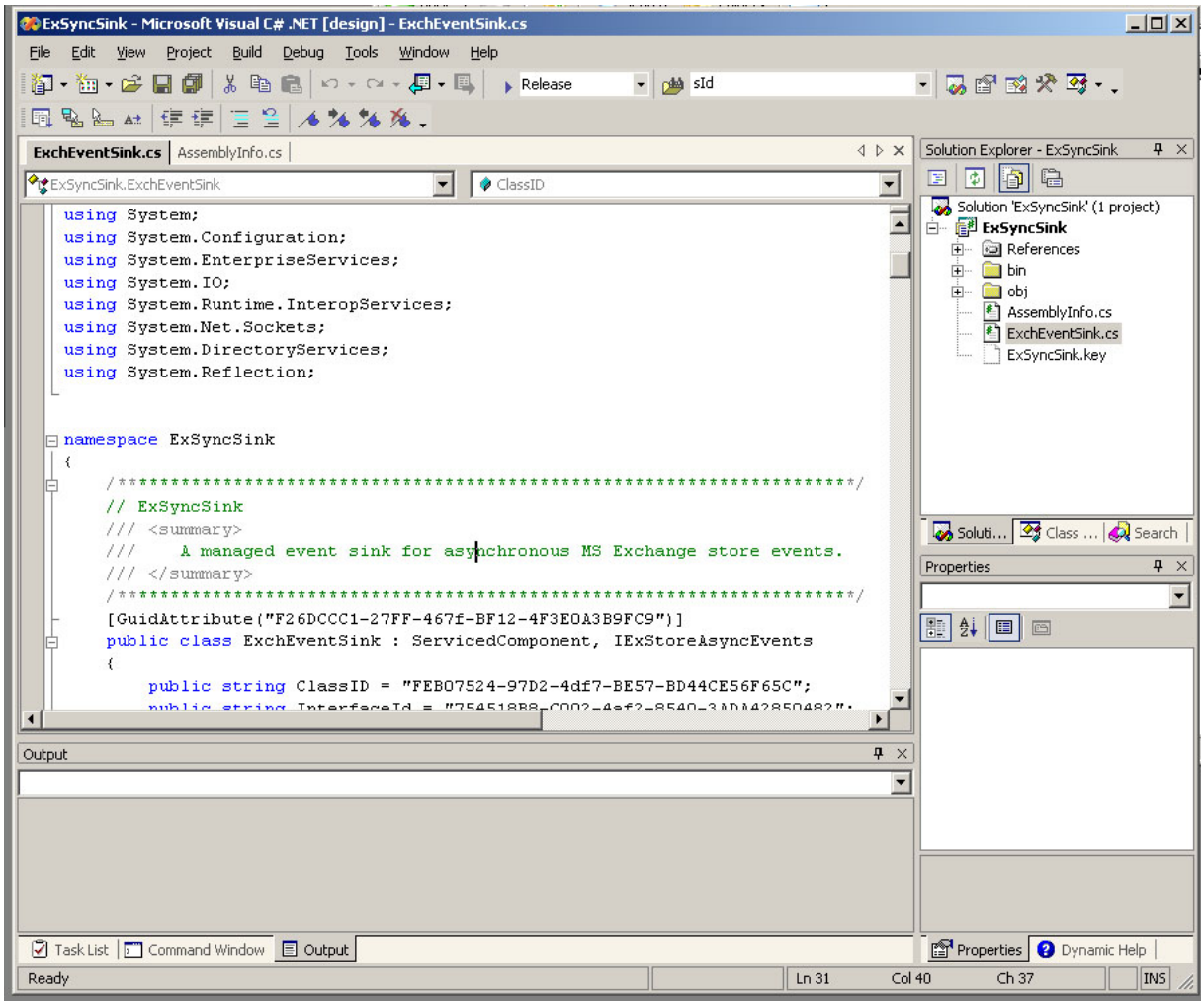
Eclipse og Visual Studio .NET 2003 ble våre valg, siden det er skreddersydd for utvikling av Java og .NET-applikasjoner. For øvrig er Visual Studio .NET 2003 det eneste utviklingsmiljøet med støtte for C#.NET som skolen har lisens på. Eclipse er et gratis verktøy som også ETC anbefalte oss å bruke.

4.5.2.1 Eclipse



Figur 4-4- Skjermbilde av MyEclipse

4.5.2.2 Microsoft Visual Studio.NET 2003



Figur 4-5- Skjermbilde av Visual Studio

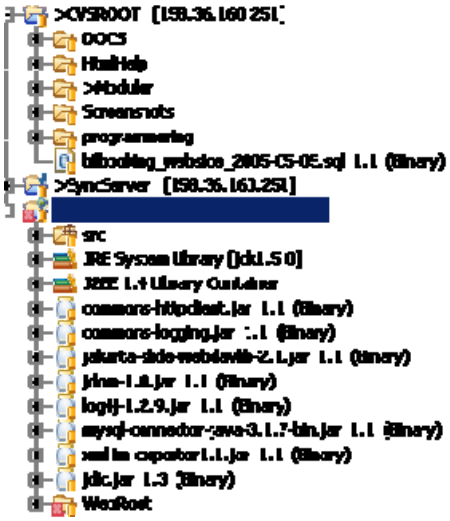
5 Kvalitetssikring og testing

Former for versjonsstyring, backup, oppdragsgivermøter og møter gruppedeltakere imellom har vært viktig for oss i forbindelse med å kvalitetssikre dette prosjektet. Oppdragsgiver har også alltid vært tilgjengelig om det har vært spørsmål eller uklarheter som har dukket opp underveis.

5.1 Versjonsstyring

Manglende versjonsstyring fører lett til problemer ved større prosjekter. For å forhindre dette, kan det være greit å bruke et versjonsstyringssystem.

Siden vi valgte å bruke Eclipse som utviklingsverktøy, ble denne også brukt for versjonsstyring da denne har en integrert CVS klient som har vist seg å fungere helt fint for vår bruk. Alle prosjektdeltakere har kunnet redigere i filene uten at ting har blitt overskrevet og gått tapt.



Figur 5-1 - Skjermbilde av CVS vindu i Eclipse

5.2 Backup

Backup har med jevne mellomrom blitt kjørt for å forhindre tap av data ved en eventuell ulykke, og også for å gi mulighet til å hoppe tilbake om noe skulle bli gjort feil. Det har blitt gjort av CVS server til oppdragsgiver, men også manuelt av oss mot en egen ekstern CVS server hver mandag.

Alle prosjektdeltakere har da hatt en oppdatert kopi av alle filer på sin maskin hver dag, samt at det også har vært en kopi på en CVS server hos oppdragsgiver, og hver mandag har også en egen ekstern CVS server blitt oppdatert med en kopi.

5.3 Testing

Under testingen ble White-box testing brukt. Denne testmetoden bygger på at bruker kjenner til kildekode og oppbyggingen rundt applikasjonen. Brukeren kan da finne hvor feilen oppstår, og kanskje også vite hva som skal til for å rette de opp.

White-box testingen ble utført av oss. Testingen har blitt utført fortløpende etter hver del som blir utviklet. Feil som oppstod, ble rettet opp med en gang. Implementeringer ble også grundig testet. Her ble feil notert, og utbedret med en gang.

6 Diskusjon av resultater

6.1 Resultater

6.1.1 Outlook

Det resultatet vi har oppnådd her er vi meget fornøyd med. Vi har en vellykket synkronisering mellom Exchange og CarAdmin sin database, der data blir overført riktig når nye ting legges til i kalender i Outlook. Se vedlegg I for skjermbilder av data i Outlook kalender.

Vi endte opp med at vår COM+ applikasjon godtar alle reserveringer og prøver å synkronisere ved hjelp at SyncServer. Akkurat denne biten må nok gjøres slik, da det var eneste måten vi fant dokumentert som fungerte for vår situasjon.

Det som nok kunne endres her var at COM+ applikasjonen sjekker mot Exchange for overlappinger i reserveringer. Som den er nå så overlater den det til SyncServer å sjekke mot databasen, og så ordner den opp i situasjonen alt etter som det er overlappende reserveringer eller ikke. Dersom Exchange og SyncServer ikke ligger på samme maskin vil vår måte å løse dette på skape litt mer nettverkstrafikk enn hva som er nødvendig.

6.1.2 Web

Vi har også fått et meget pent web grensesnitt som inneholder samme funksjonalitet som Outlook sin kalender i løsningen. Her er det mulig å legge til, slette, eller se eksisterende reserveringer. Se vedlegg H for skjermbilder av web grensesnitt.

6.2 Utfordringer

6.2.1 Webdav

Utfordringen her var å finne god dokumentasjon om webdav og Exchange. Best dokumentasjon fant vi her på MSDN [13] og Sun forum [20]. Vi måtte her til med en pakke sniffer for å bedre se hva som ble sendt og mottatt av pakker, slik at vi fikk feilsøkt bedre. Vi satte da opp Ethereal [18] for å få gjort det.

6.2.2 Påfølgende reserveringer i Outlook

Påfølgende reservering eller ”recurring event” er mulig å lage i Outlook sin kalender. Det vil si at du kan lage en avtale som gjentar seg selv over en viss tidsperiode. Denne muligheten er blitt satt i COM+ applikasjonen at den skal automatisk avvises på alle biler og biltyper. Dette grunnet at de parametere som ligger til grunn for at det er en ”recurring event”, ikke blir

behandlet i løsningen vår og dermed ville medført at kalender i Outlook ville vært ulik det en ser i kalender på web. Etter samtale med oppdragsgiver viste det seg at det heller ikke var ønskelig at bruker kunne lage en enkelt avtale og så ha reservert en bil hver onsdag i et år fremover.

6.2.3 Rettigheter i Exchange

Utfordringen her lå i at vi har forskjellige brukere i Exchange. De forskjellige kontoene vi har er:

- Vanlig bruker.
- Konto til en bil.
- Konto til en bilgruppe.
- Konto til biladministrator.

Den konto som er biladministrator må ha tilgang til kalenderen til biler og til bilgrupper, noe som er greit å ordne. Brukerkontoen til administrator på Windows maskinen må også ha tilgang til disse tre epostkontotypene i Exchange.

Det som var en utfordring er at ingen av disse har tilgang til kalender til en vanlig bruker, det har bare brukeren selv. Så vi måtte her ha en form for innlogging på websiden vår slik at vi kunne bruke brukernavnet og passordet til brukeren for å opprette reserveringer i Outlook-kalenderen til brukeren via webdav-protokollen. Dette ble også en av grunne til at vi fikk et avvik på ”Single sign on”. Se mer om dette i kapittel 6.3.3.

6.3 Avvik

Vi endte opp med følgende avvik pr dags dato i forhold til kravspesifikasjon og i forhold til vår egen plan over løsningen.

6.3.1 Påfølgende reserveringer

Denne påfølgende reserveringen, er etter definisjonen som ligger i kravspesifikasjonen. Den skiller seg en del fra den påfølgende reservasjonen som er definert i Outlook. Her vil det bety at dersom en bruker legger inn flere reserveringer etter hverandre, skal han slippe å bytte bil mellom hver. Denne delen skal ligge i rekalkuleringsfunksjonen i løsningen vår, men dette er ikke implementert på grunn av tidsnød.

Løsningen vil her være å kjøre en sjekk i databasen om bruker har en gruppereservering i et gitt tidsintervall før eller etter den nye reserveringen som er lagt til. Om han har det skal han tildeles denne bilen om den er ledig. Er den ikke ledig (for eksempel reservert av bruker2) så må det sjekkes om bruker2 kan flyttes til en annen bil, eller om bruker1 kan få begge sine reserveringer flyttet over til en annen bil som har alle reserveringer ledig.

6.3.2 Rekalkulering

Funksjonaliteten her var egentlig tenkt mye mer omfattende enn den er i dagens løsning. Det den gjør nå er å ta imot biltype og tidsperspektiv som en parameter og hente alle ledige biler fra denne biltypen inn i en vektor. Denne vektoren blir så gått igjennom og delt opp i to etter hvor mye kilometer bilen er kjørt i forhold til hva den burde ha vært. Denne utvelgelsen er basert på javakode som allerede eksisterer i ETC CarAdmin-løsningen. Det den skulle gjøre i tillegg er å ta høyde for biler som er skadet, eller på verksted og også påfølgende reservasjoner som nevnt i kapittel 6.3.1.

Skader på bil og om bilen er på verksted skal ligge i en egen tabell i CarAdmin-løsningen, så det vil her være å kjøre en sjekk mot denne ved reservering, og så utelukke biler som ligger der.

6.3.3 Single sign on

Denne biten ble nedprioritert tidlig etter et forsøk på å finne god dokumentasjon på dette, og senere da sett vekk ifra da vi fant ut at dette ble vanskelig å utføre i vår løsning.

Problemet vårt her er at selv om en bruker er logget på Windows og Outlook er det ingen mulighet for oss på vår webside å få verifisert at en bruker som åpner siden faktisk er innlogget og at innloggingen er riktig. Der er vel og merke en mulighet for å hente ut brukernavn som en bruker er innlogget som i Internet Explorer, men denne ble sett på som litt for usikker og dessuten ikke passende siden den bare fungerer for en nettleser.

Vi kom også borti en del av denne problematikken da det viste seg at vi ville få behov for brukernavn og passord til bruker for å få redigert i kalender til bruker slik at nye reserveringer automatisk legges til der. Når vi endte opp med at bruker må skrive inn brukernavn og passord i begynnelsen av websiden, har vi mulighet til å benytte oss av disse parametrene.

6.3.4 Administrator reservering

Mot slutten viste det seg at det skulle dukke opp et problem angående rettigheter, når det gjelder reservering som en eventuell administrator gjør for andre brukere. Problemet her er at skal en administrator få tilgang til å legge inn noe i Exchange kalenderen til bruker han reserverer for, må han ha tilgang til e-postboksen til bruker. Dette medfører en økning i administrative operasjoner en må utføre for å få løsningen til å fungere, og det kan også ha følger angående personvern.

Dette medfører at bruker ikke vil kunne se denne reserveringen automatisk i sin Outlook kalender, men den vil dukke opp på web og i kalender til bil. Bruker kan imidlertid få en tilbakemelding på e-post om at en administrator har lagt inn denne reserveringen for en. Pr i dag er det ikke innlagt noen slik tilbakemelding.

6.3.5 Feilmeldinger og tilbakemelding til bruker

Normalt er det ønskelig å gi bruker en best og mest mulig korrekt tilbakemelding om han gjør feil eller om hva han har gjort i en gitt situasjon; som for eksempel at bruker har prøvd å reservere en bil som allerede har en reservasjon i det gitte tidspunkt, eller at det ikke eksisterer ledige biler av den biltypen bruker prøver å reservere.

Dette har vi måttet nedprioritere grunnet tidsnød i vårt prosjekt. I denne versjonen har de fleste funksjoner som kan feile bare en sann eller usann, noe som medfører at det bare er en generell feilmelding på ting som feiler.

Det eksisterer nok flere måter å løse dette på. For eksempel å returnere en ”int” i alle funksjonene. Dersom den er ”0” så gikk alt i orden, men på alt annet enn ”0” er det en feilmelding. Denne feilmeldingen kan en så hente ut ifra en egen klasse som tar ”int” som parameter og returnerer en ”string” med aktuell feilmelding som igjen kan vises til bruker på web, eller da sendes på e-post til bruker i Outlook.

7 Konklusjon

Vi har gjennom et halvt års prosjektarbeid hatt mulighet for å følge vårt prosjekt fra en tanke om hvordan det skulle være, til en ferdig, fungerende løsning. I denne perioden har vi tilegnet oss en del nye kunnskaper, og dette har vært en meget lærerik prosess. Først og fremst har det vært lærerikt når det gjelder programmering. Vir har også lært noe om systemutvikling og problemer knyttet til det å utvikle ny kode til en eksisterende løsning.

Programmeringsmessig har vi fått frisket opp og tilegnet oss nye kunnskap om Java og JSP, og vi har også lært mye om C# og Visual Studio .NET, som vi ikke hadde erfaringer med fra før.

Til slutt så vil vi si at vi er ganske godt fornøyd med prosjektet vårt, til tross for noen utfordringer underveis, og det at vi fikk våre nye moduler til å spille sammen mot en eksisterende løsning sin database. Vi håper at oppdragsgiver får bruk for det vi har gjort, både det vi har programmert og den dokumentasjon om problemstillingene vi har produsert i løpet av prosessen.

8 Referanser

[1] Eclipse. http://www.eclipse.org/

[2] MyEclipse. http://www.myeclipse.com/

[3] MySQL. http://www.mysql.org/

[4] phpMyAdmin. http://www.phpmyadmin.net/

[5] Microsoft Windows XP. http://www.microsoft.com/windowsxp/

[6] Microsoft Server 2003. http://www.microsoft.com/windowsserver2003/

[7] Microsoft Word. http://www.microsoft.com/office/word/

[8] Microsoft Project. http://www.microsoft.com/office/project/

[9] Microsoft Outlook. http://www.microsoft.com/office/outlook/

[10] Microsoft Exchange. http://www.microsoft.com/exchange/

[11] Microsoft Internet Explorer. http://www.microsoft.com/ie/

[12] Microsoft Visual Studio .NET 2003. http://msdn.microsoft.com/vstudio/

[13] Microsoft MSDN. http://msdn.microsoft.com/

[14] Mozilla FireFox. http://www.mozilla.com/

[15] Opera. http://www.opera.no/

[16] Adobe Photoshop http://www.adobe.com/products/photoshop/

[17] PuTTy. http://www.chiark.greenend.org.uk/~sgtatham/putty/

[18] Ethereal. http://www.ethereal.com/

[19] Mistaya. http://www.infinitec.de/en/software/mistaya.aspx

[20] Sun forum. http://forum.java.sun.com/

[21] Sun. http://www.sun.com/

[22] Java API. http://java.sun.com/j2se/1.5.0/docs/api/

[23] Apache Jakarta. http://jakarta.apache.org/

[24] Log4j. http://logging.apache.org/

[25] Jakarta Commons. http://jakarta.apache.org/commons/

[26] Jakarta Slide. http://jakarta.apache.org/slide/

[27] Sync4j. http://www.sync4j.com/

[28] JDic. https://jdic.dev.java.net/

[29] JDom. http://www.jdom.org/

[30] Mysql Front. http://www.mysqlfront.de/

[31] Google. http://www.google.com/

[32] “How to create a resource account”.
http://support.microsoft.com/default.aspx?scid=kb;en-us;181988

[33] “Developing Managed Event Sinks/Hooks for Exchange Server Store using C#”.
http://www.codeproject.com/csharp/CsManagedEventSinksHooks.asp

[34] “Autoaccept Sink for Exchange”. http://autoaccept-sink.sourceforge.net/

[35] Softerra LDAP Browser. http://www.softerra.com/