DANMARKS TEKNISKE UNIVERSITET

Software til den internetbaserede handelsvirksomhed

Eksamensprojekt

Jesper Karl Sørensen

Maj 2005

IMM – DTU section for ComputerScience and Engineering

Abstract

Product sales via the internet is under heavy growth, and many new companies arise as a consequence of this growth. Research in this thesis analyzes the requirements to the software demands in these companies. I bring fort a detailed system analysis and a detailed UML based description of the system requirements.

Finally a system description and an implementation of the basic functionality in C# is presented. The solution uses a 3-tier model by using a presentation- , business- and datalayer. The communication between layers is done via a SOAP-webservice in the businesslayer.

The presented solution is not finished but there is an immense amount of work already done and further development should not be a problem.

Forord

Dette projekt er et eksamensprojekt udarbejdet hos DTU i Lyngby. Det er den afsluttende opgave på et uddannelsesforløb til civilingeniør med speciale i informatik. Projektet er sat til 35 ECTS-point hvilket svarer til 7 måneders arbejde.

Formålet med eksamensprojektet er selvstændigt at planlægge og udføre et større forskningsrelateret arbejde, hvor den lærte viden anvendes i praksis. Rapporten afsluttes med en tolkning, vurdering og konklusion af de opnåede resultater. Det hele munder ud i et mundtligt forsvar af projektets indhold og resultater.

Mine vejledere på DTU har været Jens Thyge Kristensen og Hans Bruun. De har været behjælpelige med råd og vejledning igennem hele processen. Jeg vil gerne sige tak til mine vejledere på DTU. Desuden vil jeg gerne takke min familie, der har støttet og hjulpet mig igennem hele processen.

Der er vedlagt en cdrom der indeholder: Kildekoden til systemet, rapporten i latex, pdf og postscript format samt browservenlig html dokumentation af den præsenterede løsning. Der er en vejledning til brugen cdrommen i appendiks. Alternativt kan indholdet af cdrommen findes på adressen http://www.bytewise.dk/projekt.zip.

Århus den 20. maj 2005

Jesper Sørensen
c961595

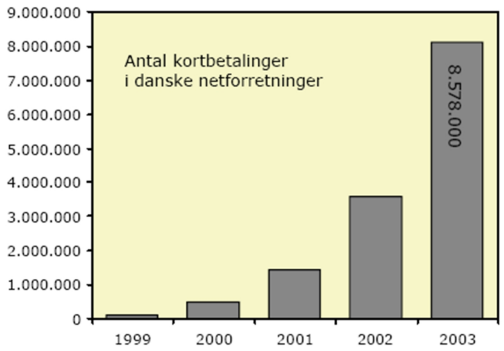
Indhold

KAPITEL 1

Indledning

Internethandlen er i kraftig vækst, antallet af dankort transaktioner via internettet fordobles, som det ses af figur 1.1, hvert år. Det samlede antal dankort transaktioner i 2004 oversteg 1 milliard hvoraf de 16 millioner havde relation til internethandel.

»Internethandel« defineres som en enhver handel, der indgås af 2 eller flere parter, hvor ordren og betalingen foregår elektronisk via internettet. Det vil oftest betyde, at ordren oprettes via en hjemmeside, men der er også tale om internethandel hvis en vare bestilles via en mobiltelefon, der har adgang til internettet.



Figur 1.1: Antallet af dankort betalinger på internettet i perioden 1999 - 2003. [Kilde: *www.pbs.dk - PBS præsentation 2004*].

På baggrund af dette potentiale skyder mange nystartede internetbaserede handelsvirksomheder op. Disse virksomheder står i en situation, hvor de har brug for software til at hjælpe dem i den daglige håndtering af hjemmeside, ordrer, kunder, lager, leverandører etc. Disse systemer kaldes i industrien og litteraturen for

henholdsvis ERP og CRM systemer. Nedenfor introduceres kort hvad begreberne dækker over samt hvilke ERP systemer, der er tilgængelige i Danmark.

CRM *Customer Relationship Management* beskriver processen med hele tiden at vedligeholde kontakten til kunderne og følge op på ordrer ved at få feedback på kunden, så hele kunde kontakten afspejler kundes ønsker. I den simpleste form er det et spørgsmål om at holde kontakten til kunden således, at kunden hele tiden er sikker på, at han får det han har bestilt og til tiden.

ERP *Enterprise Resource Planning* betyder oversat til dansk *virksomhedens ressource planlægning*, i praksis bruges begrebet om alle systemer, der varetager en eller flere af virksomhedens daglige opgaver. Hvis man kigger på de eksisterende systemer finder man ud af, at der ikke er ret mange systemer, der er specielt rettet mod internetbaserede handelsvirksomheder og de løsninger der eksisterer, er meget dyre løsninger der kræver tilpasning. I appendix C er der lavet en tabel med de kendte ERP systemer. Det ses af tabellen, at der kun er få systemer, der umiddelbart kan tages i brug uden at der skal involveres konsulenter. Det var ikke muligt at finde priser på alle produkterne, men de store producenter tager mere end 10000 kr. pr. bruger af systemet plus en årlig afgift. Der er derfor ikke mange systemer, der retter sig direkte mod de mindre internetbaserede handelsvirksomheder. Microsofts produkter ser umiddelbart meget interessante ud, da grundpakkerne er billige, men hvis man kigger nærmere på det viser det sig, at et samlet system hurtigt bliver dyrt.

1.1 Grundlæggende problem

Der findes ikke produkter på markedet p.t., der dækker alle de behov, der er for software løsninger til de mindre internetbaserede handelsvirksomheder. De største problemer er de høje startomkostninger, driftsomkostninger og manglede muligheder for at videreudvikle løsningerne i takt med virksomhedens vækst.

Det er derfor interessant for mange af dem, at udvikle deres egne it-løsninger helt fra bunden.

1.2 Afgrænsning

Der fokuseres på at lave en løsning, der henvender sig til de mindre internetbaserede handelsvirksomheder med 1-20 ansatte, der har brug for hjælp til den daglige administration. Der er primært tale om virksomheder, der handler med varer, der kan sendes med fragtmand eller postvæsnet.

Den præsenterede løsning henvender sig ikke direkte til andre typer af virksomheder, da der i så fald skal laves en større tilpasning for at de kan anvende det. Her tænkes specielt på produktionsvirksomheder, der har et behov for en stor integration af produktions- og salgsprocessen. I dette projekt fokuseres derfor ikke på produktionsprocesser.

Der fokuseres på beskrivelse af krav, godt grundlæggende design og mulighederne for videreudvikling af systemet.

1.3 Problemformulering

Der fokuseres på 2 områder:

- Det ønskes analyseret hvilken software, der er behov for i en mindre internetbaseret handelsvirksomhed. Herefter udvælges de begreber og arbejdsgange, der ønskes understøttet i virksomhedens samlede it-system.
- Der skal designes og implementeres et samlet it-system, der opfylder de i analysen fundne behov.

1.4 Metoder

Analysen er foretaget ved at kigge på eksisterende virksomheder, der driver forretning via internettet. Deres hjemmesider er blevet gennemgået for på den måde at identificere de logistiske udfordringer der er. Der er også foretaget uformelle interviews med en internetbaseret handelsvirksomhed, der er i opstartsfasen.

Begreberne er herefter undersøgt nærmere ved at søge i litteraturen og på internettet. Forfatteren har trukket på sine egne erfaringer fra driften af enkeltmandsfirmaet Bytewise[1] samt hundredvis af afsluttede internethandler.

Der er i eksamensprojektet fokuseret på at lave en færdig løsning, hvorfor der er lagt mange ressourcer i at undersøge og analysere tekniske problemstillinger og derefter implementere dem.

Det er et meget stort vidensområde og forfatteren har været afhængig af at sætte sig ind i regler omkring den daglige bogføring og hvilke krav Told og Skat stiller. Denne viden er hentet ved dels at læse litteratur om emnet bl.a *Bogføringsvejledningen* [6] og dels at læse vejledningerne på Told og Skats hjemmeside. Endvidere

[1] Bytewise startede som et konsulentfirma, der ydede bistand i forbindelse med udvikling af software, men er siden blevet udvidet med et handelsselskab, der primært handler Business to Business.

er der gennemlæst relevant materiale, bøger og hjemmesider, angående tekniske løsninger i softwaren.

1.5 Rapportens struktur

Kapitel 2 - Analyse
Analyse af internethandel som koncept og beskrivelse af begreber og tekniske termer, der knytter sig til domænet.

Kapitel 3 - Udviklingsmodeller og notation
Kort gennemgang af udviklingsmetoder samt valg af metode og notation.

Kapitel 4 - Kravspecifikation
De formelle krav opstilles og ud fra analysen udvælges aktørerene på systemet; derudfra opstilles Use Cases.

Kapitel 5 - Tidlig designmodel (Domænemodel)
Der laves en foreløbig skitse over systemts opbygning og de første tekniske krav opstilles.

Kapitel 6 - Designmodel
Den endelige design- og løsningsmodel opstilles og klasser beskrives.

Kapitel 7 - Implementering
Beskrivelse af overvejelser, problemstillinger og tekniske løsninger i forbindelse med implementationen af programmet. Diskussion af opnåede erfaringer.

Kapitel 8 - Teknisk test
Teknisk test af den opnåede løsning.

Kapitel 9 - Konklusion og persepktivering
Konklusion og forslag til primære fokusområder til videre arbejde med projektet.

1.6 Arbejdsforløb

Det arbejsforløb, der her vises, er lavet ved projektets afslutning og viser det faktiske forløb med tidsangivelser. Under hele processen er rapporten løbende blevet opdateret. Ligesom der undervejs har været en stor vidensindsamling, primært via hjemmesider.

Måned	Kommentar
1.	Indsamling af literatur og udfærdigelse af problemformulering. Opsætning af rapportskabelon.
2.	Gennemlæsning af litteratur og studie af tekniske muligheder i .NET.
3.	Afgrænsing og analyse af domænet. Afprøvning af tekniske løsningsmodeller.
4.	Kravspecifikation, definition af aktører og afgrænsning af funktionaliten og Use cases.
5.	Begynder på systemdesignet, definerer objekter. Opsætning af databaseserver og opsætning af udviklingsmiljøet.
6.	Systemdesignet videreudvikles og programmering påbegyndes. Yderligere afprøvning af tekniske løsningsmodeller.
7.	Programmering og tilpasning af systemdesignet således, at kode og design følger hinanden.
8.	Tilpasning af koden, lave kommentarer til koden, udtrække kommentarer. Opsætning og udskrift af koden. Afslutte rapporten.

1.7 Værktøjer

Programmeringen til dette projekt er udført i Visual Studio 2003, der er anvendt som editor, oversætter og debugger. Endvidere er det anvendt til at holde styr på alle elementerne i programmet. Visio er anvendt som UML værktøj.

Som webserver er anvendt en Internet Information Server v6.0 og som database er anvendt MS SQL server. Værktøjerne Enterprise Manager og Query Analyzer er anvendt til administration og oprettelse af tabeller i databasen.

Rapporten er skrevet i Latex formatet og er oversat til postscript- og pdffiler ved hjælp af MikTex og som Latex editor er anvendt WinEdt 5.4. Koden er oversat til Latex formatet ved brug af værktøjet lgrind. Og kodebeskrivelserne er skrevet ind i kildefilerne til koden og oversat til Latex ved brug af værktøjet NDoc 1.3.

Figurerne i projektet er tegnet i Visio og efterbehandlet i Adobe Photoshop. Billederne på hjemmesiden er lavet og efterbehandlet i Adobe Photoshop.

KAPITEL 2

Analyse

Dette kapitel indeholder en beskrivelse af den mindre internetbaserede handelsvirksomhed, herefter IHV. Der vil her være en beskrivelse af centrale arbejdsgange og begreber. Det forventes at alle arbejdsgange foretages 'i huset'. Der købes kun ekstern bistand til at servicere hjemmeside, dankortbetaling, og evt. database.

2.1 Internethandel

Selvom internethandelen ikke for alvor har taget fart endnu, er der efterhånden etableret de facto standarder, der knytter sig til det at lave salgshjemmesider. Det er funktioner, som brugerne forventer er til rådighed og undersøgelser har vist, at folk føler sig mere »trygge« ved det de kender. Derfor skal man, i forbindelse med præsentationen af sine varer, ikke forsøge at opfinde den dybe tallerken. Det kan, i denne forbindelse, betale sig at lytte til eksperterne. [7]. Jeg vil give en kort beskrivelse af den erfaring og viden jeg har indsamlet omkring emnet.

2.1.1 Virtuel indkøbskurv

Ved internethandel anvendes altid en virtuel indkøbskurv. Det er blevet standard og kunderne forventer at den er der og at den er overskuelig og at det er nemt at tilføje og fjerne emner fra den. Dette er altså en funktion, der skal gøres meget synlig og let tilgængelig på hjemmesiden.

2.1.2 Vare præsentation

Da det ved internethandel ikke er muligt at røre ved varerne, er det meget vigtigt, at der tages gode billeder og der skal laves fyldestgørende beskrivelser af varerne. Det er nødvendigt at brugerens oplevelse af hjemmesiden er professionel og uden bugs. Kunderne er ikke trygge ved at give sine kreditkort oplysninger ud, hvis det ikke virker som om teknologien bag hjemmesiden er 100% i orden.

Det er vigtigt at der ikke er for mange niveauer på hjemmesiden, alle informationer skal være let tilgængelige uden for mange unødige tastetryk. Derfor skal der fokuseres på, at der ikke præsenteres for mange produkter på en gang og at de er emne-opdelte. Mange brugere af interhandel forventer også at der er en søgefunktion tilrådighed.

2.1.3 Betaling

I forbindelse med en handel på internettet gennemføres betalingen elektronisk. Da der kun eksisterer relativt få udbydere af betalingsservices er der også her en standard måde at gøre tingene på. Det primære er at betalingen foretages som en selvstændig del af ordren og at der kun benyttes de indtastningsfelter brugeren er vant til. Der benyttes følgende betalingsmuligheder:

- Bank overførsel.
- Giro.
- Kreditkort.
- Diverse e-betalings løsninger (Valus, e-wire, eDankort, ...).
- Efterkrav (via postvæsnet).

2.1.4 Opdatering

Det er vigtigt at siden opdateres ofte, da det øger hjemmesidens troværdighed. Hvis der er nyheder på siden, skal disse opdateres mindst en gang om ugen og der skal kun skrives informationer, der kan have interesse for kunderne. Det er vigtigt at der jævnligt bliver opdateret med nye produkter på forsiden.

2.1.5 Søgemaskiner

For at kunne trække kunder til er det nødvendigt, at de kan finde hjemmesiden. Man kan vælge at reklamere via de traditionelle kanaler, men rigtigt mange mennesker benytter søgemaskinerne når de skal finde produkter. Det er derfor vigtigt

at man kommer blandt de 10 første links, når der søges på ord og produkter, der beskriver ens virksomhed.

Dette er en hel videnskab for sig selv, men man kan forøge sine chancer ved at gøre følgende 3 ting: Sørg for at hjemmesiden bliver opdateret. Sørg for at der linkes til hjemmesiden. Sørg for at din hjemmeside overholder standarderne[1]. Flere søgemaskiner tilbyder, at der linkes til ens hjemmeside når bestemte ord optræder i søgningen, dette koster penge.

2.2 Arbejdsområder

I den typiske internet virksomhed i dag er der mellem 1 og 20 ansatte. Disse ansatte varetager de arbejdsopgaver, der beskrives i det følgende afsnit.

2.2.1 Hjemmesiden

Der er en medarbejder, der tager sig af at opdatere firmaets hjemmeside med nye tilbud og nye varer. Det er derfor væsentligt, at det er simpelt at vise nye produkter på hjemmesiden.

2.2.2 Netværk

Firmaets computere og netværk skal altid være køreklare, da virksomheden ikke kan gennemføre handler hvis systemerne ikke kører. Det vigtigste er, at hjemmesiden fungerer hele tiden, da det ikke er muligt at tage imod ordrer hvis hjemmesiden er offline.

2.2.3 Salg

Salgsmedarbejderne tager sig af at godkende ordrer, der ligger i ordrebogen. En ordre, der er indgået via hjemmesiden, bliver ikke effektueret før den er godkendt. Det undersøges hvad der er bestilt og om det er på lager. Kunden får sendt en mail fra salgsmedarbejderen med en ordre-bekræftelse og et forventet leveringstidspunkt.

Salgsmedarbejderen tager sig også af at slette ordrer, hvis kunden annulerer en ordre.

[1] Overhold html standarderne (*www.w3.org*), så har søgemaskinerne nemmere ved at indeksere hjemmesiden og anvend beskrivende 'title' tags og 'alt' tags på billederne.

2.2.4 Lager

Lagermedarbejderne har 3 primære arbejdsopgaver.

1. Forsendelse - Når en ordre er sendt videre fra slagsafdelingen finder medarbejderen alle varerne på lageret. Herefter pakker han og vejer dem og påsætter en pakkelabel. Dernæst placeres pakken i et bur til senere afhentelse af pakkeposten.

2. Opdatere lageret ved indkøb - Når der ankommer ny indkøbte varer, er det lagermedarbejderens ansvar, at lageret bliver opdateret.

3. Returvarer - Det er lagermedarbejderens opgave, at sørge for at varen registreres og undersøges for skader. Dette kaldes herefter en RMA[2] sag.

2.2.5 Indkøb

Indkøberens opgave er at undersøge hvilke produkter, der efterspørges og sørge for, at disse produkter er på lager. Endvidere tager indkøberen sig af at undersøge, hvilke nye produkter, der har interesse. Dette baseres på hvilke nye produkter forhandleren har, samt hvilke produkter brugerne af hjemmesiden forespørger. Indkøberen har også ansvar for at bestille varer hjem, som ligger i ordrebogen, men ikke på lager.

2.2.6 Support

Supporterens opgave er at være behjælpelig ved fejl og problemer med at benytte firmaets hjemmeside. Endvidere skal supportmedarbejderen være behjælpelig i forbindelse med RMA og afhjælpning af kundens problemer med et produkt.

2.2.7 Reklame

Reklamemedarbejderens opgave er at promovere firmaets produkter. Dette gøres ved at følge op på feedback fra kunder samt analysere om der er nogle varer, der er svært tilgængelige på hjemmesiden.

[2]Return Merchandize Authorization: Beskriver den process det er at returnerer varer købt via internettet. Det er at foretrække at køberen opretter en RMA sag inden han returnerer varen således at lageret ved hvad de skal gøre med den.

2.2.8 Bogholderi

Bogholderen tager sig af den daglige bogføring, bogføringen beskrives i afsnittet »Bogføring«. Han afregner moms med Told og Skat. Momsen beskrives i afsnittet »Moms«.

Han optræder som kasserer og godkender alle ind og udbetalinger.

Når der skal laves årsregnskab er han revisoren behjælpelig med at finde de oplysninger og den dokumentation der er nødvendig for at regnskabet kan godkendes og dermed overholder Told og Skat's regler.

2.3 Bogføring

Den daglige bogføring har til formål at fastholde, bearbejde og dokumentere oplysninger om virksomhedens økonomiske hændelsforløb [6] og [5].

Ved hjælp af bogføringen beskrives altså de økonomiske konsekvenser af alle virksomhedens transaktioner samt de øvrige hændelser og forhold, der har påvirket virksomhedens økonomi. Dette kaldes i bogføringsloven for, at der skal eksistere et transaktion- og kontrolspor. Det skal være muligt at følge hvordan beløbene i regnskabet er opstået.

Alle momsregistrerede virksomheder er ved lov forpligtet til at bogføre. Der stilles dog ikke samme krav til store og små virksomheder: Et lille firma kan godt slippe lettere, da deres aktiviter umiddelbart kan overskues.

Inden jeg beskriver hvordan den faktiske bogføring foregår, defineres begreberne »bilag«, »konto«, »kontoplan« og »kassekladde«.

Bilag skal kunne bevise registreringernes rigtighed og gøre det muligt at konstatere, om transaktionerne registreres i den rigtige periode og med korrekte beløb. Hver eneste registrering af transaktioner og hændelser skal dokumenteres med et bilag. Bilag er ikke blot papirbilag, men enhver form for dokumentation, der er nødvendige for at opretholde transaktions- og kontrolsporet. Bilag nummereres fortløbende. Der er krav om hvordan bilag fysisk skal opbevares, det er tilladt at indskanne dem i en computer og destruere original bilaget, men der gælder for begge typer bilag, at der skal være »truffet passende foranstaltninger« for at materialet ikke bortkommer. Dette betyder i praksis, at man skal tage backup og sørge for at materialet er forsvarligt opbevaret.

Der bruges betegnelserne 'internt' bilag og 'original' bilag. Et original bilag er alle almindelige bilag: fakturaer, regninger, kreditnotaer, lønsedler etc. Et internt bilag er et bilag, der bliver oprettet i forbindelse med konkrete posteringer, hvor

der ikke findes et original bilag. Dette gøres for at dokumentere hvad der er posteret. Dette bruges bl.a. i forbindelse med rettelser.

Konto Er »et sted« til notering af ensartede oplysninger. Indeholder:

1. Kontoens nummer, jvf. kontoplanen.
2. Kontoens navn.
3. Det år kontoen gælder.
4. Bogføringsdato.
5. Beskrivelse af det bogførte.
6. Nummer på det bilag der er bogført efter.
7. Debet beløb.
8. Kredit beløb.
9. Kontoens saldo; beregnes efter hver postering.

Faste oplysninger:

Nummer (1)	Navn (2)	Saldo (9)	Årstal (3)

Poster:

Dato (4)	Beskrivelse (5)	Bilag (6)	Debet (7)	Kredit (8)

Debet og kredit forholder sig som + og - til hinanden og når der posteres i kredit på en konto skal der modposteres i debet på en anden konto. En kontos saldo kan kun formindskes ved at der posteres i modsatte kolonne af det oprindelige beløb.

Kontoplan Det er påkrævet ved lov, at man udformer en kontoplan. Denne plan er en liste med konti. Kontiene er listet med et nummer og en kort beskrivelse. Konti deles op i balance- og resultatkonti.

Der er vist et eksempel på en kontoplan i Appendiks D.

Konteringsvejledning Det er påkrævet ved lov at man udformer en konteringsvejledning. Det betyder, at man skal lave en beskrivelse af hvordan der posteres på de enkelte konti. I appendiks E er et eksempel på en konteringsvejledning, den er lavet til kontoplanen i Appendiks D.

Kassekladde En kasse kladde er en liste af posteringer på en række konti. Inden disse indføres på de enkelte konti tjekkes det om debet beløbene og kredit beløbene er lige store.

Før bogføringen kan beskrives i detaljer er det nødvendigt med nogle supplerende oplysninger.

Aktiver (debet) = kapitalanvendelse (omkostning)= Hvad er pengene brugt på?
Passiver (kredit) = kapitalfremskaffelse (indtægt) = Hvor er pengene kommet fra?

Bogføringen foretages ved at stille disse spørgsmål [2]:

1. Hvad skal jeg bogføre:
   Et aktiv?
   Et passiv?
   En omkostning?
   En indtægt?
2. Forøges kontoen(+)?
   Formindskes kontoen(-)?

2.4 Begreber

I dette afsnit beskrives de introducerede begreber.

2.4.1 Varer

Der defineres 3 kategorier af varer:

- Serviceydelser.
- Digitale produkter. Produkter der kan sendes og modtages elektronisk.
- Traditionelle produkter.

2.4.2 Moms

En afgift der pålægges ved salg til private (salgsmoms). Den er for nuværende i Danmark 25%. Når der sælges til andre EU-lande skal der ikke pålægges moms. Man afregner så med Told og Skat, enten måneds-, kvartals- eller halvårsvis. Dette afgøres af virksomhedens størrelse og fastsættes af Told og Skat.

Ved indkøb i EU er der ingen moms, men virksomheden skal selv pålægge en særlig EU-moms, denne kan ligeledes fratrækkes ved salg.

Når momsen skal indbetales foregår det ved at bogholderen laver en opgørelse over kontiene »Købsmoms« (indkøb) og »Salgsmoms« (varesalg) og »Moms af EU-erhvervelser«. Købsmomsen er penge man har tilgode hos Told og Skat og kan modregnes i salgsmomsen, dette beløb kalder vi nettomoms. Nettomomsen indberettes til Told og Skat enten via en formular på Told og Skat's hjemmeside (tast-selv) eller via et girokort. Der findes en demo af tast-selv systemet på *http://momsdemo.toldskat.dk/moms-ind.htm*. Betalingen foregår via homebanking, giro eller PBS.

2.4.3 Ordre

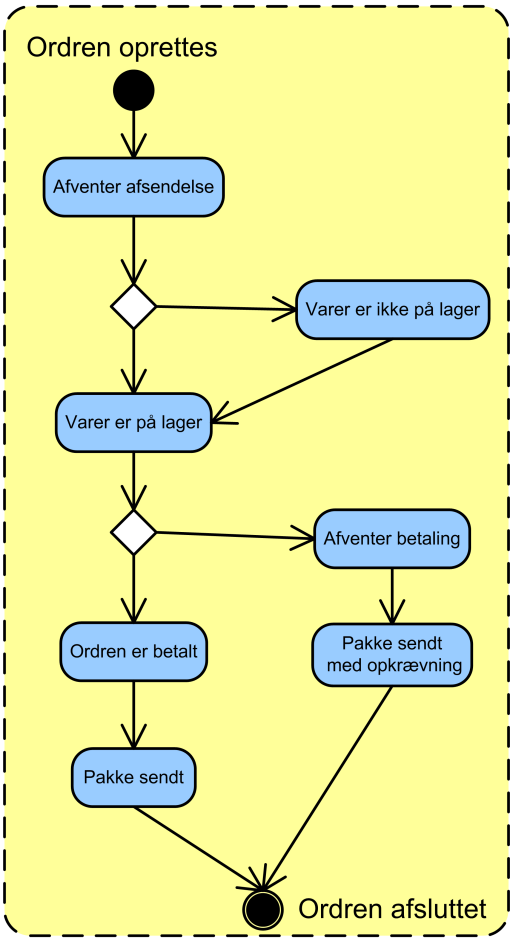
En ordre er en samlet bestillingsseddel. Indeholder i princippet de oplysninger der også forefindes på fakturaen.

En ordre oprettes på følgende måde:

Kunden ankommer på hjemmesiden og begynder at vælge varer og mængde. Disse tilføjes til en virtuel »indkøbskurv« og når kunden er færdig med at handle vises en oversigt over og samlet pris for købet. Kunden godkender og indtaster sine personlige oplysninger. Som minimum indtastes adressen og betalingsformen vælges.

Ordren sendes herefter til firmaet og en lagermedarbejder tjekker om alle varer er på lager, hvis alle varer er på lager sendes varerne samme dag. Hvis ikke alle varer er på lager undersøges leveringstiden og kunden informeres om leveringstiden via email.

I figur 2.1 vises de tilstande en ordre kan befinde sig i.



Figur 2.1: Tilstande for en ordre.

En ordre er ikke afsluttet før den er afsendt og betalt. Ordren åbnes igen ved RMA og lukkes igen ved afsluttet RMA.

2.4.4 Faktura

En faktura er en kvittering på en gennemført ordre. Der udstedes minimum 2 eksemplarer, en til sælger og en til køber. Told og Skat stiller krav til hvilke informationer virksomheder har pligt til at oplyse på deres fakturaer [kilde: Told & Skat.]:

- Fakturanummer, et unikt fortløbende nummer.
- Dato for udstedelsen.
- Sælgers navn og adresse samt SE/CVR-nummer.

- Købers navn og adresse.
- Varen eller ydelsens art, mængde (omfang) og pris
- Momsbeløbet
- Betalingsbetingelser

Derudover anføres ofte telefon, e-mail, bank- og/eller girokonto, disse er ikke lovpligtige.

Der er i 2005 indført en »e-faktura«, dette er en elektronisk faktura og det er ikke muligt at handle med offentlige institutioner med mindre man er i stand til at producere en elektronisk faktura. E-fakturaen er baseret på OIOXML standarden som beskrives længere fremme i dette kapitel.

2.4.5 Kreditnota og dekort

Udstedes når en kunde fortryder en ordre og sender varen retur, eller hvis der sker en reduktion af en vares pris efter der er udskrevet en faktura. Indeholder de samme oplysninger som en faktura men beløbene bogføres modsat.

2.4.6 Forsendelse

Leveringen af traditionelle produkter foregår via et fragt-firma. Ved mindre pakker anvendes oftest Post Danmark da de grundet deres størrelse og distributionsnet næsten altid er billigst.

Serviceydelser og digitale produkter sendes elektronisk som oftest via email eller download fra hjemmeside.

2.4.7 B2B

Der er tale om Business to Business når 2 virksomheder samhandler. Det kan foregå som en almindelig handel, men der er også muligt at det foregår via EDI.

2.4.8 Teknologier

EDI

Electronic Data Interchange. »Elektronisk overførsel af strukturerede data i aftalte meddelelsesstandarder fra et IT-system til et andet IT-system«. Det bygger

på proprietære protokoller og der er kommet en revision der hedder EDIFACT der er målrettet mod handelsvirksomheder og hvor protokollen er standariseret. Postvæsnet og Told og Skat tilbyder forskellige services via EDI. For mere info anbefales Dansk Standard's hjemmeside *http://www.ds.dk/1263*

OIOXML

Er en dansk XML specifikation baseret på den internationale standard UBL og er lavet for at lette virksomheders måde at »snakke« sammen elektronisk på.

En e-faktura skrevet i OIOXML indeholder de samme oplysninger som en almindelig faktura, plus et EAN nummer dette er et unikt nummer der identificerer modtageren af fakturaen så den kan finde vej gennem cyberspace. En e-faktura kan ikke sendes direkte til modtageren men sendes til modtagerns VANS »postkasse« igennem en VANS leverandør. Hvis man ikke har direkte adgang til en VANS leverandør tilbyder bl.a Danske Bank at sende en e-faktura videre, det koster 1 krone pr faktura. Her i overgangsfasen er der også firmaer der indskanner almindelige papir fakturaer til e-fakturaer som en sidste mulighed kan man emaile sin e-faktura til en VANS leverandør.

Der kan læses mere om OIOXML på Offentlig Information Online's hjemmeside *www.oio.dk*.

XBRL

XBRL står for eXtensible Business Reporting Language og er endnu en XML standard. Standarden er ved at blive indført men kan på nuværende tidspunkt ikke anvendes, det er meningen på sigt at større firmaers årsrapporter kan aflægges elektronisk i XBRL formatet. Kravene til en given finansiel rapportering udtrykkes gennem en XBRL-taxonomi, og selve rapporteringen med data i udtrykkes gennem XBRL instance dokumenter. [Kilde: http://www.eogs.dk/sw4224.asp]

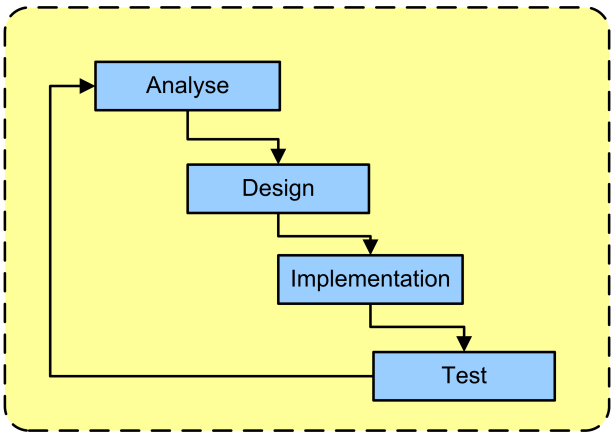
KAPITEL 3

Udviklingsmodeller og notation

Der findes lige så mange måder at lave software udvikling på som der findes programmører. Der udvikles hele tiden nye metoder og det kan ofte være svært at vælge en der passer præcis til et projekt. Jeg vil i dette kapitel fokusere på nogle generelle og veldokumenterede udviklingsprincipper.

3.1 Vandfaldsmodellen

Vandfaldsmodellen består af relativt få trin. De kan opsummeres til disse 4:



Figur 3.1: Viser den traditionelle udgave af vandfaldsmodellen

Det er en let, overskuelig proces, men det betyder ikke, at den ikke er kraftfuld.

Den har været benyttet i mange udviklingsprojekter i industrien. Modellen adskiller de forskellige trin skarpt og det er meningen at alle trin gennemføres i rækkefølge. Hvis man ikke laver en skarp adskillelse mellem trinene, er der ikke tale om traditionel vandfaldsmodel, men en iterativ udgave, som beskrives i afsnittet »Iterative modeller«.

Den skarpe adskillelse af analyse og implementering betyder, at det er nødvendigt at udarbejde en meget fyldestgørende kravspecifikation [8, s. 46]. Dette kan føre til forskellige problemer. Det er ofte nødvendigt at have en domæneekspert til rådighed, i stedet for at medarbejderne kan tilegne sig den fornødne viden løbende. Endvidere er det et stort problem, hvis man i test fasen opdager en fejl i det grundlæggende design, derpå må analysen laves om og man må igennem trinene igen. Det kan være katastrofalt, da fejlen jo ofte først fanges i den afsluttende fase og der ikke er meget tid tilbage til deadline.

Metoden har vist sig at føre til systemer, der er robuste og er forberedte på videreudvikling.

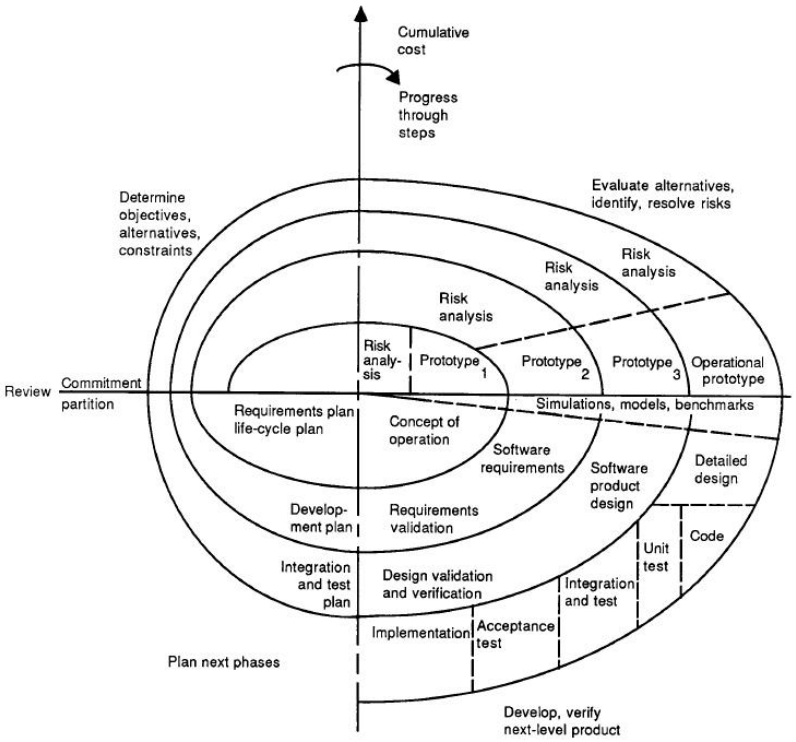
3.2 Iterative modeller

Iterativ-model, dækker over metoder hvor de enkelte trin i modellen gennemløbes flere gange. En iterativ model kan godt bygge på trinene i vandfaldsmodellen, men de gennemløbes så flere gange.

Forskellen på planlægningen ved traditionel vandfaldsmodel og iterativ vandfaldsmodel kan beskrives ved hjælp af følgende eksempel.

»Vi har 1 år til at færdiggøre projektet«
Ved den traditionelle vandfaldsmodel planlægge således: 3 måneder til hvert af de 4 trin. Ved iterativ vandfaldsmodel vil man afsætte sin tid således, at man kommer igennem hele vandfaldsmodellen fx. en gang om måneden. På denne måde får man testet sit systems grundlæggende funktionalitet igennem mange flere gange end ved den traditionelle metode og produktet udvikler sig løbende, fremfor at der først er 'testbar' kode tæt på deadline. Dette er især en fordel i meget omfattende projekter, da man på den måde tidligere opdager om der er fejl i det grundlæggende design og der er også større sandsynlighed for, at mindre bugs bliver fundet før man når ud i et produktionsmiljø.

Et andet konkret eksempel på en selvstændig iterativ-model er Boehms 'spiralmodel'. I spiralmodellen benyttes mange flere trin end i vandfaldsmodellen.



Figur 3.2: Boehms spiral model beskriver lang mere detaljeret kravene til projektforløbet [8]

3.3 Evolutionær udvikling

Benytter sig ikke af 'trin', som de 2 andre modeller. Der udarbejdes først i forløbet en abstrakt beskrivelse af systemets funktionalitet. Denne meget generelle beskrivelse danner herefter grundlag for et program udkast. Det tidlige program udkast modelleres så til den endelige form i et tæt samarbejde med brugerne af systemet. Dette finder sin anvendelse blandt andet i forbindelse med udvikling af mindre, specifikke værktøjer, der skal benyttes af en lille gruppe af personer. I dette tilfælde er der ikke behov for at bruge en model med mange trin, fordi kravene til programmets funktionalitet er overskuelige. Det egner sig derfor ikke til større projekter.

3.4 Formel udvikling

Til sidst skal kort nævnes 'Formel udvikling'. Ved denne model beskrives kravene til systemet i matematiske termer, der via matematiske formler reduceres og manipuleres. På den måde kan man, ved brug af matematiske beviser, dokumentere systemets funktionalitet. Dette er fx. meget anvendt i forbindelse med udvikling i

ML [1]. Det finder oftest sin anvendelse i forbindelse med udvikling af matematiske algoritmer. Det egner sig ikke til udvikling af brugerorienterede applikationer.

3.5 Notation

Når et program skal beskrives har man forskellige muligheder. Der findes rent tekstuelle repræsentationer, hvor man beskriver sit system ved hjælp af et struktureret sprog og der findes rent grafiske notationer, hvor systemet beskrives ved hjælp af diagrammer. Som oftest anvendes en kombination af de 2, da der er fordele ved begge. Det kan være meget enklere at beskrive en undtagelse ved at beskrive det med ord og det er meget enklere at beskrive sit programs klassestruktur ved at anvende et klassediagram.

Der er findes mange notationer, der kan bruges i forbindelse med objektorienteret software udvikling. Jeg vælger i rapporten at benytte UML[2] kombineret med tekstuelle forklaringer samt hjemmelavede diagrammer de steder hvor det vurderes, at det giver det bedste overblik. UML er en grafisk notation, hvor de tilgængelige diagrammer betyder, at man arbejder med simple grafiske elementer og på den måde får en meget kompakt struktur. De oftest anvendte diagrammer, i forbindelse med traditionel applikationsudvikling, er:

- Klassediagram
- Use-case diagram
- Aktivitetsdiagram
- Sekvensdiagram
- Tilstandsdiagram

[8, s. 456] og [4].

3.6 Valg af udviklingsmodel

I dette projekt anvendes objektorienteret udvikling. Der skal derfor vælges en metode, der understøtter den proces det er at identificere objekter og handlinger.

[1] Meta Language - et programmerings sprog, der oprindeligt blev udviklet til undervisningsbrug, men nu har fundet anvendelse i industrien. ML benyttes ofte til beskrivelse af formelle krav.

[2] Unified Modelling Language er en notation der er specielt velegnet i forbindelse med objekt orienteret udvikling - benyttes ofte sammen med The Unified Process, som er en metode udviklet af firmaet Rational Rose.

Overordnet benytter jeg en iterativ model, der er baseret på vandfaldsmodellen. De 5 trin jeg benytter er:

Kravspecifikation Use-cases identificeres og benyttes hvor det er praktisk og de resterende krav beskrives som formelle krav.

Domænemodel Tidlig skitse over systemets opbygning.

Designmodel De enkelte klassers funktionalitet beskrives i detaljer og i tilfælde hvor det er muligt skitseres GUI[3].

Implementering Valg af programmeringssprog og tekniske løsninger samt beskrivelse af den færdige løsning.

Teknisk test De enkelte elementer af programmet testes.

Processen beskrives dels vha af UML diagrammer, egne diagrammer og tekstuelle forklaringer.

[3]Graphical User Interface oversat til dansk betyder det »Det grafiske brugerinterface«

KAPITEL 4

Kravspecifikation

Først angives alle krav til systemet som tekst. Derefter laves en use case analyse af de arbejdsgange, der ønskes understøttet i det samlede it-system. Dette gøres ved at først finde aktørerne i systemet, derefter beskrives hvilke use cases der knytter sig til de enkelte aktører og til sidst beskrives den enkelte use case i detaljer.

4.1 Afgrænsning

4.2 Formelle krav

Tabel 4.1	
Generelt	**1**
1.1	Systemet skal kunne anvendes fra alle netværkstilsluttede maskiner
1.2	Systemet skal forberedes for udvidelser
Bogføring	**2**
2.1	Der skal være *en* kassekladde
2.2	Det skal være muligt at tilføje, rette og slette poster på kassekladden
2.3	Det skal være muligt at bogføre posterne på kassekladden
2.4	Det skal ikke være muligt at rette i bogførte poster på de enkelte konti, rettelser klares ved modposteringer
2.5	Der bør være mulighed for at der automatisk anvendes en modkonto således at det bogførte beløb modposteres på den konto
2.6	Der skal være *en* kontoplan
2.7	Det skal være muligt at tilføje, rette og slette konti på kontoplanen

Fortsættes på næste side

Tabel 4.1 – Fortsat fra forrige side	
2.8	Det skal det være muligt at angive om posteringer på kontoen inkludere købsmoms, salgsmoms, EU-moms eller ingen moms
2.9	Det skal kun være muligt at slette konti, der ikke har været benyttet i regnskabet i indeværende år
2.10	Det skal være muligt at lavet et momsregnskab baseret på en måned, et kvartal, halvt år eller et helt år
2.11	Det skal være muligt at oprette et bilag og det skal tildeles et fortløbende nummer
2.12	Det skal være muligt at knytte en kommentar til et bilag
Ordre 3	
3.1	Det skal være muligt at tilføje, rette og slette ordrer
3.2	Der skal være en liste med ordre der endnu ikke er afsluttet
3.3	Det skal være muligt at ændre en ordres status
3.4	Systemet skal selv oprette bilag i forbindelse med afslutningen af en ordre
3.5	Det skal være muligt at lave en faktura
3.6	Fakturaer oprettes automatisk som bilag
3.7	Det skal være muligt at oprette en kreditnota
3.8	Kreditnotaer oprettes automatisk som bilag
Administration 4	
4.1	Der skal oprettes en administrator konto ved installation af programmet
4.2	Det skal være muligt at tilføje, rette og slette brugere af systemet
Lager 5	
5.1	Fakturaen skal printes ud automatisk når en lagermedarbejder pakker en ordre
Hjemmesiden 6	
6.1	Det skal være muligt at tilføje et billede til en vare
6.2	Der skal være en indkøbskurv
6.3	Der skal være en menu
6.4	Det skal være muligt at tilføje og fjerne varer fra hjemmesiden
6.5	Det skal være muligt at køre med tilbud på en vare
6.6	Der skal være en søgefunktion på hjemmesiden

4.3 Use cases

Det første jeg gjorde var at identificere hvilke aktører der er på systemet. En aktør er enten en bruger af systemet eller en del af edb-systemet. Der defineres følgende aktører.

Aktør	Use case	Krav
Alle	Log ind på systemet	
Bogholder	Opret bilag Tilføj konto til kontoplan Fjern konto fra kontoplan Tilføj post til kassekladde Fjern post fra kassekladde Bogfør kassekladde Lav momsopgørelse Vis poster på en konto	
Hjemmeside	Opret ordre Vis indkøbskurv	
Administrator	Opret bruger Slet bruger Rediger bruger	
Webansvarlig		
Sælger	Opret ordre via applikation Afslut ordre	
Lagermedarbejder	Opret vare Slet vare Ret vare Pak en ordre Godkend ordre Slet ordre	

4.3.1 Log ind på systemet

Aktører:

- Alle

Beskrivelse:

- Brugeren ønsker at bruge systemet og er nødt til at give adgangsoplysninger til systemet.

Krav der skal opfyldes før use casen kan udføres:

- Brugeren skal være oprettet i systemet.

Krav for succes:

- Brugeren bliver godkendt og præsenteres for den for brugeren mest anvendte funktioner.

Primære flow:

1. Brugeren starter programmet.
2. Brugeren indtaster sit brugernavn og password.
3. Programmet kan nu anvendes.

Alternativer:

3A. Hvis brugernavn og password ikke kan findes i systemet, starter programmet ikke og pkt 2 køres igen.

Hyppighed:

- Meget ofte.

4.3.2 Opret bilag

Aktører:

- Bogholder.

Beskrivelse:

- Bogholderen ønsker at oprette bilag til brug som dokumentation i bogføringen.

Krav der skal opfyldes før use casen kan udføres:

- Use case »Log ind på systemet« skal være udført.

Krav for succes:

- Bilaget er oprettet i systemet og har fået tildelt et bilagsnummer.

Basis flow:

1. Vælger at oprette et nyt bilag.
2. Det er et original bilag.
3. Programmet viser et vindue.
4. Lægger original bilaget i skanneren og trykker scan.
5. Verificere at bilaget er læseligt.
6. Angiver en eventuel kommentar til bilaget.
7. Trykker godkend.
8. Programmet opretter bilaget og tildeler det et unikt fortløbende nummer.

Alternativt flow:

2A. Det er et internt bilag.

3A. Programmet viser et vindue.

4A. Angiver en eventuel kommentar til bilaget.

5A. Trykker godkend.

6A. Programmet opretter bilaget og tildeler det et unikt fortløbende nummer.

Hyppighed:

- Ofte.

4.3.3 Bogfør kassekladde

Aktører:

- Bogholder.

Beskrivelse:

- Når bogholderen har lavet et antal posteringer på kassekladden og ønsker disse posteringer indført i regnskabet på de respektive konti.

Krav der skal opfyldes før use casen kan udføres:

- Use case »Log ind på systemet« skal være udført.
- Beløbene i debet og kredit kolonnerne skal gå op.

Krav for succes:

- Posteringerne på kassekladden indføres på de respektive konti.

Basis flow:

1. Vælger at bogføre kassekladden.
2. Systemet undersøger om beløbene i debet og kredit går op.
3. Systemet fører de enkelte posteringer ind på de respektive konti.
4. Programmet nulstiller kassekladden.

Alternativt flow:

3A. Hvis beløbene ikke går op kommer der en besked »Beløbene på kassekladden går ikke op.«

4A. Brugeren retter posterne.

5A. Starter forfra med trin 1.

Hyppighed:

- Ofte.

4.3.4 Tilføj konto til kassekladde

Aktører:

- Bogholder.

Beskrivelse:

- Bogholderen ønsker at tilføje en ny konto på kontoplanen.

Krav der skal opfyldes før use casen kan udføres:

- Use case »Log ind på systemet« skal være udført.

Krav for succes:

- Kontoen oprettes med det angivne kontonummer og vises på kontoplanen.

Basis flow:

1. Vælger opret ny konto.
2. Indtaster kontonr, navn, momskode og angiver om det er en resultat- eller balancekonto.
3. Systemet undersøger om nummeret er ledigt.
4. Systemet opretter kontoen og sætter saldoen til 0.

Alternativt flow:

3A. Hvis nummeret ikke er ledigt skrives »Konto nummeret er ikke ledigt«.

3B Alle kontonumre er fortløbende og man kan ikke have balancekonti der har kontonumre der overlapper resultatkontisnumre.
$\boxed{\min(\text{balancekontonummer}) > \max(\text{resultatkontonummer}).}$

Hyppighed:

- Sjældent.

4.3.5 Fjern konto fra kontoplan

Aktører:

- Bogholder.

Beskrivelse:

- Bogholderen Bogholderen ønsker at slette en konto fra kontoplanen.

Krav der skal opfyldes før use casen kan udføres:

- Use case »Log ind på systemet« skal være udført.
- Kontoen må ikke være i brug for regnskabet i indeværende år.

Krav for succes:

- Kontoen slettes og den opdaterede kontoplan vises.
- Eller brugeren gøres opmærksom på at kontoen ikke kan slettes da kontoen er i brug i regnskabet.

Basis flow:

1. Vælger den konto i kontoplanen der ønskes slettet og vælger »Slet«.
2. Systemet verificere at man virkeligt ønsker at slette kontoen.
3. Brugeren trykker »OK«.
4. Systemet undersøger om kontoen har været i brug i regnskabet.
5. Systemet sletter kontoen.

Alternativt flow:

5A. Hvis kontoen er i brug i regnskabet, udskrives »Kontoen er sidst brugt i regnskabet den xx/xx/xxxx og kan derfor ikke slettes« og kontoen slettes ikke.

Hyppighed:

- Sjældent.

4.3.6 Tilføj post på kassekladde

Aktører:

- Bogholder.

Beskrivelse:

- Bogholderen ønsker at tilføje en post på kassekladden.

Krav der skal opfyldes før use casen kan udføres:

- Use case »Log ind på systemet« skal være udført.

Krav for succes:

- Posten vises på kassekladden og indeholder de nødvendige oplysninger.

Basis flow:

1. Vælger at tilføje en post på kassekladden.
2. Brugeren indtaster bilags nr, beskrivelse, kontonr, debet eller kredit beløb.
3. Systemet undersøger om alle felter er udfyldt.
4. Posten vises på kassekladden.

Alternativt flow:

2A. Hvis bilaget ikke eksisterer i systemet udføres use casen »Opret bilag«.

Hyppighed:

- Meget ofte.

4.3.7 Fjern post fra kassekladde

Aktører:

- Bogholder.

Beskrivelse:

- Bogholderen ønsker at slette en post fra kassekladden.

Krav der skal opfyldes før use casen kan udføres:

- Use case »Log ind på systemet« skal være udført.

Krav for succes:

- Posten slettes og den opdaterede kassekladde vises.

Basis flow:

1. Vælger posten i kassekladden og vælger »Slet«.
2. Systemet sletter posten.

Alternativt flow:

-

Hyppighed:

- Ofte.

4.3.8 Lav momsopgørelse

Aktører:

- Bogholder.

Beskrivelse:

- Bogholderen ønsker at opgøre momskontiene og beregne hvor meget virksomheden skylder Told og Skat.

Krav der skal opfyldes før use casen kan udføres:

- Use case »Log ind på systemet« skal være udført.

Krav for succes:

- Nettomomsen udregnes og kontiene salgsmoms, købsmoms og EU-moms nulstilles. Der laves en udskrift der viser hvilke beløb der skal indberettes.

Basis flow:

1. Vælger lav momsopgørelse (Det er i opsætningen af programmet angivet om virksomheden angiver moms hver måned, hvert kvartal eller hvert halve år).
2. Systemet beregner hvilken måned der afslutter momsperioden (Måned for sidste momsopgørelse + interval for angivelse)
3. Hvis der er gået minimum det interval der er angivet mellem nu og sidste angivelsesmåned flyttes beløbene fra Salgsmoms og Købsmoms over på kontoen Momsangivelse.
4. Systemet viser en liste med hvilke beløb der skal angives hos Told og Skat.

Alternativt flow:

4A. Programmet åbner automatisk https://www.tastselv.toldskat.dk/erhvervlogin/pinlogin.do (Det er Told og Skats hjemmeside til indberetning af moms)

5A. Systemet indsætter automatisk senummer og tastselvkode og trykker »Send«.

6A. Systemet indsætter de beløb der skal indberettes og trykker »OK«.

7A. Systemet tager en kopi af kvitteringen og opretter det som et bilag i systemet.

8A. Bogholderen åbner netbank systemet og betaler det skyldige beløb.

Hyppighed:

- Jævnligt.

4.3.9 Vis poster på en konto

Aktører:

- Bogholder.

Beskrivelse:

- Bogholderen ønsker at se hvad der er posteret på en enkelt konto.

Krav der skal opfyldes før use casen kan udføres:

- Use case »Log ind på systemet« skal være udført.
- Kontoplanen skal være synlig

Krav for succes:

- Posterne på kontoen vises.

Basis flow:

1. Finder kontoen på kontoplanen og trykker på den.
2. Systemet åbner et vindue der viser posterne der er på denne konto.

Alternativt flow:

-

Hyppighed:

- Ofte.

4.3.10 Opret vare

Aktører:

- Lagermedarbejder.

Beskrivelse:

- Der er ankommet en helt ny vare på lageret og varen skal oprettes i systemet.

Krav der skal opfyldes før use casen kan udføres:

- Use case »Log ind på systemet« skal være udført.

Krav for succes:

- Varen bliver oprettet med de rigtige parametre og antal.

Basis flow:

1. Hvis der ikke findes billeder af varen fotograferes varen først og billederne overføres til computeren i jpg format.
2. Vælger »Opret ny vare«.
3. Indtaster varenavn
4. Indtaster kort beskrivelse (en linie)
5. Indtaster lang salgsbeskrivelse (flere linier, vises på hjemmesiden og angives i html)
6. Tilføjer de billeder der er af produktet.
7. Tilføjer link til evt. brugsanvisninger.
8. Systemet opretter varen og det er nu muligt at oprette ordre hvor denne vare indgår.

Alternativt flow:

-

Hyppighed:

- Ofte.

4.3.11 Rediger vare

Aktører:

- Lagermedarbejder.

Beskrivelse:

- Der er en eller flere af attributterne på en vare der ønskes ændret. Fx beskrivelse indkøbspris, billeder.

Krav der skal opfyldes før use casen kan udføres:

- Use case »Log ind på systemet« skal være udført.
- Varen skal være oprettet i systemet.

Krav for succes:

- Varen bliver opdateret med de nye oplysninger.

Basis flow:

1. Søger efter varen. Indtaster enten del af navnet eller varenummeret.
2. Den eller de varer der matcher kriterierne vises og lagermedarbejderen vælger en af dem og trykker »Redigér«.
3. Systemet viser alle varens attributter i et vindue.
4. Tilføjer rettelser og afslutter med at trykke på »Gem«.
5. Systemet sørger for at varen bliver opdateret med de nye rettelser.

Alternativt flow:

4A Brugeren trykker »Cancel« og vinduet lukker uden at der foretages nogle ændringer.

Hyppighed:

- Jævnligt.

4.3.12 Pak en ordre

Aktører:

- Lagermedarbejder.

Beskrivelse:

- Der er en ventende ordre i systemet, ordren venter bare på afsendelse.

Krav der skal opfyldes før use casen kan udføres:

- Use case »Log ind på systemet« skal være udført.
- Alle varer skal være på lager.
- Ordren skal være betalt eller betales med efterkrav.
- Lagermedarbejderen laver et »reality check« for at finde ud af der er tale om en regulær ordre. (Så der ikke er tale om Anders And fra Andeby der har bestilt for 10000 kr.) kr.

Krav for succes:

- Varene pakkes i en kasse og der påføres en adresselabel og en pakkelabel.

Basis flow:

1. Vælger en ordre og vælger »Afsend ordre«.
2. Systemet viser et vindue hvor alle ordre oplysninger der er tilgængelige står. Navn og adresse på modtager samt varenummre og mængde på det bestilte. Det vises tydeligt at ingen af varerne er fundet på lageret endnu.
3. (ekstern) Vælger hvilken type emballage der er brug for.
4. Henter varerene efterhånden og vælger enten at indskanne deres stregkode aller at angive med musen at der er hentet en eller flere af en vare.
5. Systemet viser overfor brugeren hvad der mangler at blive pakket før ordren er komplet.

6. Når alle varerne er fundet og indtastet/indskannet gør systemet opmærksom på at det nu er tid til at få vejet pakken.
7. Vejer pakken og laver en pakkelabel.
8. Indtaster pakkens vejede vægt i systemet.
9. Trykker »OK«.
10. Systemet flytter ordren til listen med de afsendte ordre.

Alternativt flow:

10A Hvis kunden har valgt at betale med dankort, hæves ordre beløbet på kundens konto.

10B Hvis kunden har valgt efterkrav, føres ordren på listen over ordre med ventende betaling.

Hyppighed:

- Meget ofte (forhåbentlig).

4.3.13 Lav bestillingsseddel til leverandør

Aktører:

- Lagermedarbejder.

Beskrivelse:

- Der skal bestilles nye varer.

Krav der skal opfyldes før use casen kan udføres:

- Use case »Log ind på systemet« skal være udført.

Krav for succes:

- Varen bliver oprettet med de rigtige parametre og antal.

Basis flow:

1. Hvis der ikke findes billeder af varen fotograferes varen først og billederne overføres til computeren i jpg format.
2. Vælger »Opret ny vare«.
3. Indtaster varenavn
4. Indtaster kort beskrivelse (en linie)
5. Indtaster lang salgsbeskrivelse (flere linier, vises på hjemmesiden og angives i html)
6. Tilføjer de billeder der er af produktet.
7. Tilføjer link til evt. brugsanvisninger.
8. Systemet opretter varen og det er nu muligt at oprette ordre hvor denne vare indgår.

Alternativt flow:

-

Hyppighed:

- Ofte.

KAPITEL 5

Tidlig designmodel (Domænemodel)

Udfra forrige kapitels kravspecifikation udarbejdes nu de første tekniske krav til systemet. På baggrund af de konkrete krav til systemet fastsættes de rammer som resten af model opbygningen kommer til at foregå under.

Det er et krav at systemet kan anvendes af alle virksomhedens computere med net-adgang og det var fra starten af et krav at det skulle gøres så simpelt så muligt at tilføje ny funktionalitet til systemet. Derfor vælges overordnet en 3-lags model, der består af:

1. Præsentation (klient)
2. Forretningslogik (server)
3. Database

I forbindelse med valg af metode til kommunikation mellem lagene, er der nogle ting der er nødt til at blive fastlagt. Der er i dette tilfælde 2 oplagte muligheder for valg af kommunikation mellem klient og server: Remote Procedure Call (RPC) eller TCP/IP + egen protokol. I de følgende afsnit beskrives kort fordele og ulemper ved de 2 metoder.

RPC Fordelen ved RPC er at det bliver muligt at overføre objekter direkte mellem klienten og serveren det er derfor ikke nødvendigt at serialisere og de-serialisere (skrive/læse objekter til/fra en datastøm) objekterne selv. Der findes mange måder at lave RPC på, de fleste RPC implementioner er enten platformsafhængige eller afhængige af programmeringssprog eller afhængige af at der installeres en speciel server (proxy).

Egen protokol Fordelen ved at lave egen protokol er at det er muligt at lave sin dataoverførsel så kompakt som muligt. Dermed overføres der ikke mere data end højst nødvendigt. Dette gør dog programmet meget sværere at videreudvikle da det er nødvendigt at arbejde dybt i programmets logik.

I forbindelse med dette projekt anvendes en RPC implementation der bygger på SOAP. SOAP og andre RPC-implementationer beskrives nærmerer i Kaptitlet »Implementering«.

Kommunikationen mellem server og database afgøres af hvilken type database der anvendes. Databasen kunne i det simpleste tilfælde blot være flade tekstfiler der er umiddelbart tilgængelige for serveren. Denne metode vælges dog ikke til den løsning der præsenteres i denne rapport. Vi ved ikke hvor stort databehov den enkelte virksomhed har og dataene fra kravspecifikationen indikere meget kraftigt at det er fornuftigt at anvende et RDBMS[1]. Derfor foregår kommunikationen mellem server og database som SQL[2]-kommandoer der foretages via ODBC[3]-protokollen

Præsentaionslaget deles op i 2 hovedelementer, en applikation som er virksomhedens ansattes adgang til systemet og en hjemmeside som er kundernes primære adgang til systemet. Applikationen kan både laves som et selvstændigt kørende program eller som en hjemmeside i browseren. Hvis det laves som en hjemmeside er der en del begrænsinger på hvad der er tilrådighed af funktionalitet. Det er meget mere besværligt at tilgå tilsluttede komponenter som f.eks. en skanner der er påkrævet for at man kan indskanne bilag. Endvidere er det besværligt at lave en pæn GUI da alle komponenter skal designes. Ved en traditionel applikation er det meste af styresystemets GUI tilrådighed i udvikling af brugergrænsefladen. Der er også fornuft i at lade applikationen »ligne« andre programmer. Dette får brugeren til at føle sig tryg, da brugeren vil kunne genkende mange af komponenterne.

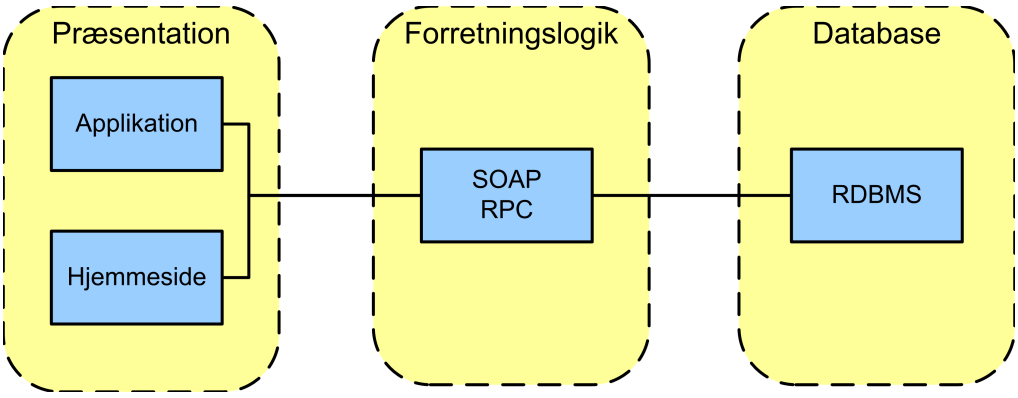
Derfor laves applikationen som et selvstændigt kørende program.

Systemet er skitseret i figur 5.1.

[1] Relational DataBase Management Systems

[2] Structered Query Language, er et sprog opfundet af Dr. Edgar Frank Codd, der mens han var ansat hos IBM var med til at opfinde relationelle databaser. SQL bruges til at lave strukturerede dataudtræk fra relationelle databaser.

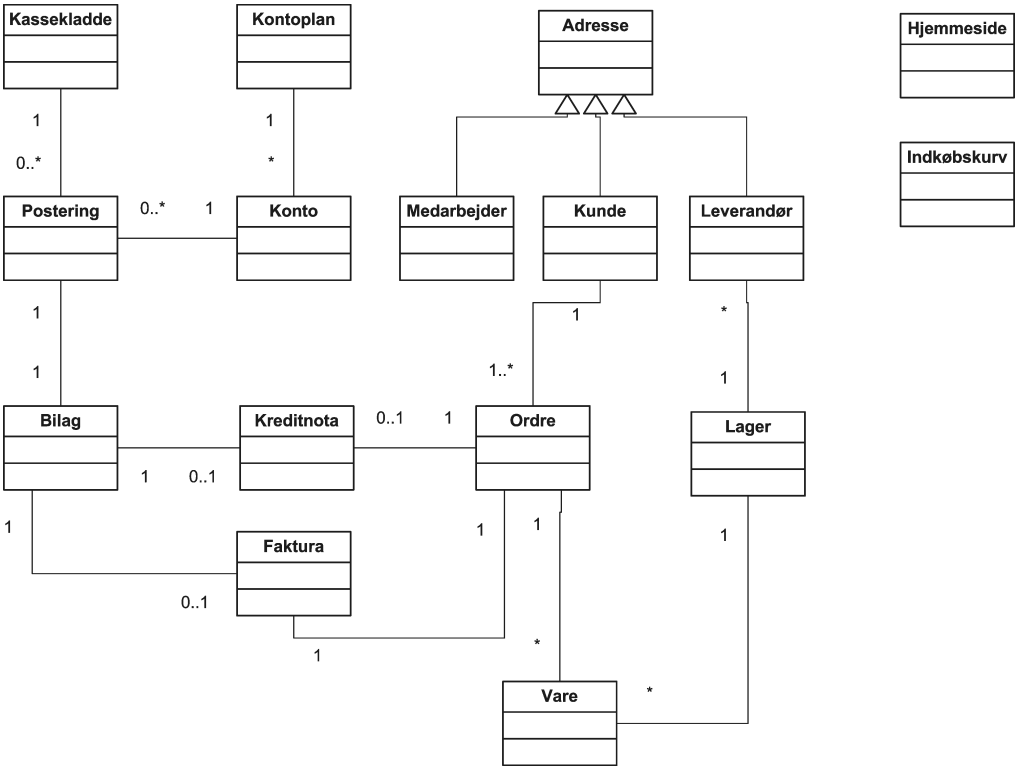
[3] Open DataBase Connectivity er en netværksprotokol til kommunikation med databaser.



Figur 5.1: Hovedkomponenterne i systemet og hvordan det deles op i de 3 lag.

5.1 Tidlig UML model

Systemets nuværende opbygning skitseres som et klasseskelet hvor kun klassenavnet er angivet.



Figur 5.2: Systemets centrale objekter og relationer

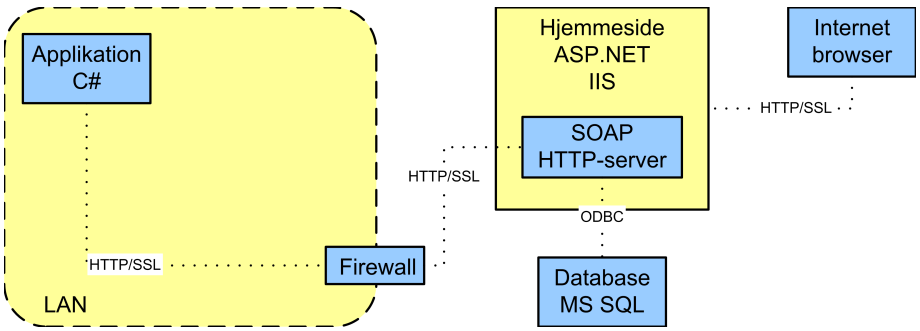
KAPITEL 6

Designmodel

I dette kapitel præsenteres den løsning, der er udviklet. Løsningen er skabelonen til den færdige løsning og beskriver alle klasser, både data-klasser, GUI-klasser, kommunikations-klasser, programmets opbygning og til sidst beskrives hjemmesiden.

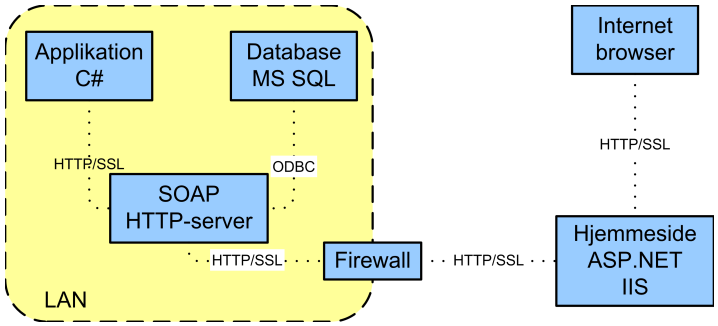
Først vises et overblik over hvordan systemet logisk er bygget op. Da programmet er bygget op af komponenter der kan sættes sammen på forskellige måder, er der 2 måder jeg vil fokusere på og de er afhængige af hvilke computere, der er tilrådighed og om man selv hoster SOAP-serveren eller det gøres af en hjemmeside udbyder.

I figur 6.1 er databasen, hjemmesiden og SOAP komponenten placeret hos en hjemmesideudbyder. Det betyder, at det er muligt for en hacker at kommunikere med SOAP-komponenten og vi er nødt til at lave nogle passende foranstaltninger, for at kun verificerede brugere har adgang.



Figur 6.1: Den usikre model.

I figur 6.2 er databasen og SOAP komponenten placeret på en eller flere maskiner indenfor virksomhedens firewall og eventuelle hackere skal først igennem firewallen før de kan få adgang til disse. Igen skal der foretages foranstaltninger der gør, at kun verificerede brugere har adgang, men der er kommet et ekstra lag: firewallen. Det ville være fornuftigt kun at give adgang til komponenterne på den webserver, der servicerer hjemmesiden.

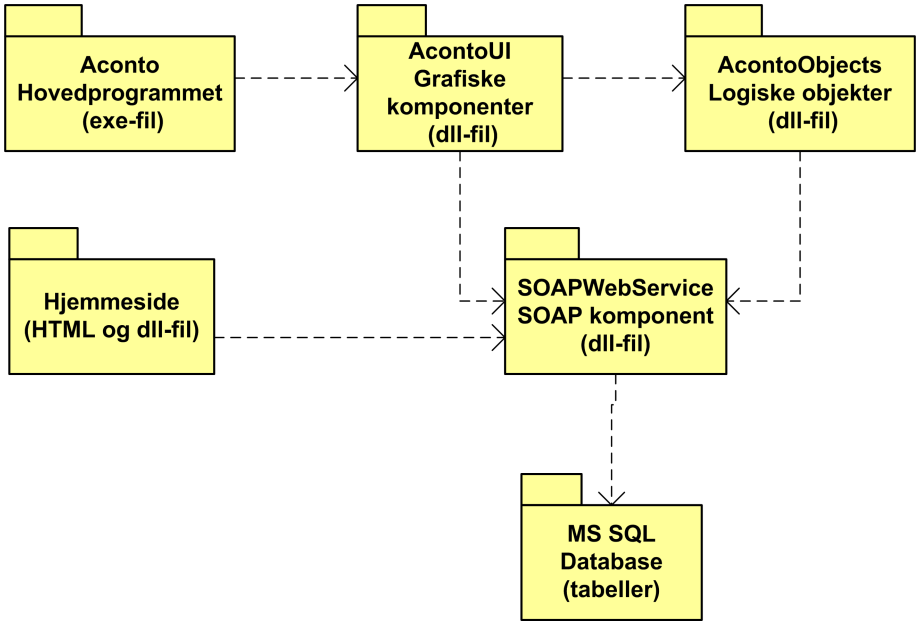


Figur 6.2: Den sikre model.

Projektet er udviklet under begge setup og det er meget simpelt at sætte op, hvis den påkrævede software er installeret. Se kapitlet Implementering.

6.1 Komponenter

Jeg vil kort beskrive hvad de enkelte logiske komponenter, som systemet er opbygget af, indeholder. Komponenter kaldes i UML termer for packages eller pakker. Komponenterne og deres relationer er vist i figur 6.3.



Figur 6.3: Systemets logiske opbygning i komponenter

6.1.1 Aconto

Dette er hovedprogrammet og det er den del, som virksomhedens ansatte kommer til at anvende dagligt. Aconto indeholder også den grafiske opbygning af hovedprogrammet. Det er bare en tom skal, hvor så den faktiske funktionalitet hentes fra de specialiserede GUI klasser i AcontoUI. Det er gjort sådan, at designet af programmet kan ændres uden at man skal ændre alle de grafiske elementer.

6.1.2 AcontoUI

Indeholder de specialiserede grafiske komponenter. Det er alle indtastningformularer og grafiske views af data fra databasen. Kort sagt al intelligent GUI ligger i denne komponent.

6.1.3 AcontoObjects

Indeholder alle de logiske objekter, der kort blev beskrevet i forrige kapitel. Denne komponent eksisterer både i hovedprogrammet og på SOAPserveren og på denne måde kan de objekter vi ønsker, udveksles mellem hovedprogrammet og SOAPserveren.

6.1.4 SOAPWebService

Denne komponent residerer på en webserver og AcontoUI indeholder en reference til den. Det er det logiske lag eller forretningslaget mellem hovedprogrammet og databasen. Her er defineret de funktioner, der er nødvendige for at kunne kommmunikere med databasen og lave brugervalidering.

6.1.5 Hjemmesiden

Dette er ikke en komponent i programmeringsmæssig forstand, men den udgør en logisk del af programmet. Den består af en dynamisk hjemmeside, der bygger på HTML og ASP. Indeholder også en dll-fil, hvor logikken bag hjemmesiden ligger: bl.a. indkøbsvogn og reference til SOAPserveren. Der kommunikeres direkte med SOAP komponenten.

6.1.6 Database

Dette udgør datalageret i programmet. Der er tale om en relationel database. Der er mange muligheder med den valgte database, men vi anvender den kun til at oprette og udtrække data fra tabeller.

6.2 Beskrivelse af klasser

I dette afsnit beskrives alle de klasser de forskellige komponenter indeholder. Der vil kun være en kort beskrivelse af klassernes funktionalitet og ingen detaljeret beskrivelse af metoder eller attributter. Disse beskrivelser er skrevet ind i kildekoden i et XML format og trækkes ud af koden igen som en samlet XML fil, der så formatteres med værktøjet NDoc. De detaljerede beskrivelser kan læses i Bilag I.

6.2.1 Aconto

Aconto Hovedprogrammet opbygget som en GUI skabelon med en hovedmenu.

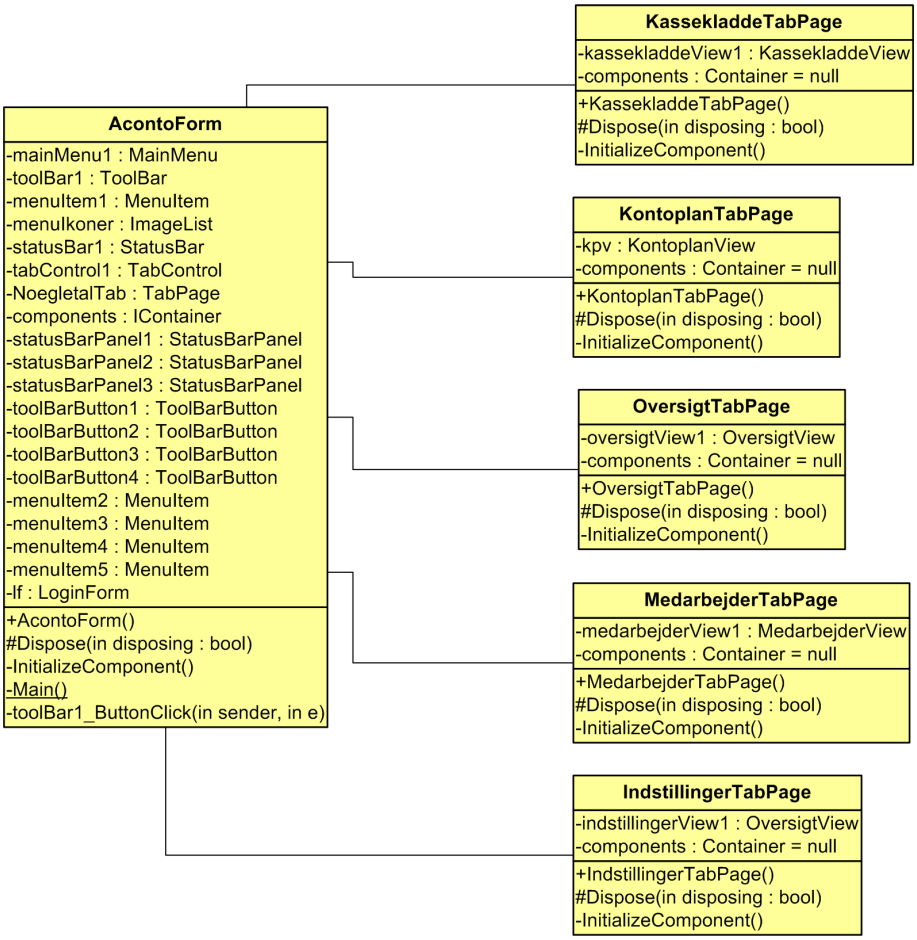
KassekladdeTabPage Udgør en logisk TabPage der indsættes i Aconto klassen TabControl, dens udseende og indhold findes i AcontoUI.KassekladdeView.

KontoPlanTabPage Udgør en logisk TabPage der indsættes i Aconto klassen TabControl, dens udseende og indhold findes i AcontoUI.KontoPlanview.

MedarbejderTabPage Udgør en logisk TabPage der indsættes i Aconto klassen TabControl, dens udseende og indhold findes i AcontoUI.MedarbejderView.

OversigtTabPage Udgør en logisk TabPage der indsættes i Aconto klassen TabControl, dens udseende og indhold findes i AcontoUI.OversigtView

IndstillingerTabPage Udgør en logisk TabPage der indsættes i Aconto klassen TabControl, dens udseende og indhold findes i AcontoUI.IndstillingerView.



Figur 6.4: Viser UML diagrammet over klasserne i Aconto komponenten.

6.2.2 AcontoUI

Alle klasserne i AcontoUI kommunikerer med SOAPWebService komponenten , dvs al data hentes/skrives til/fra SOAPWebService.

AdministratorView Udgør en grafisk komponent, herefter usercontrol. Der vises en et opdelt vindue hvor der er et TreeView til venstre og en vindue til højre der viser hhv. indstillinger, medarbejderer og log.

FakturaView Udgør en usercontrol, den er opbygget omkring en HTML kontrol således at den kan vise HTML formateret output. Denne bruges når man ønsker en grafisk udskrift af en faktura. den understøtter også udprintning.

IndstillingerView Udgør en Usercontrol, den er opbygget omkring et ListView (tabelform) og viser en oversigt over alle medarbejdere, der er tilknyttet en contextmenu (højreklik), der anvendes nås man øsnker at redigere indstillingerne i systemet. listen.

KassekladdeView Udgør en usercontrol, den er opbygget omkring et DataSet og viser hvilke posteringer der venter at blive indført i regnskabet.

KontoplanView Udgør en usercontrol, den er opbygget som et ListView (tabelform) og viser en grafisk præsentation af dels kontoplanen samt saldi for de enkelte konti. Der er tilknyttet en contextmenu (højreklik) der anvendes ved opdatering af kontoplanen.

LoginForm Udgør en formular, den er opbygget med en lille grafisk animation (se RainDrop længere nede). Indeholder to indtastningsfelter 'Brugernavn' og 'Password'. Animationen betyder at formularen også benyttes som »splashscreen«.

MedarbejderView Udgør en Usercontrol, den er opbygget omkring et ListView (tabelform) og viser en oversigt over alle medarbejdere, der er tilknyttet en contextmenu (højreklik), der anvendes nås man øsnker at redigere medarbejder.

Mediator Denne klasse benytter Singleton pattern således at den kan tilgås direkte fra alle andre klasser. Her ligges dels funktioner der skal bruges i hele systemet fx createPassword der beregner vores password hash. Den indeholder ligeledes referencen til vores SOAPWebService således at den kan tilgås overalt i programmet ved at skrive Mediator.Instance().logik.<navn på SOAP funktion>.

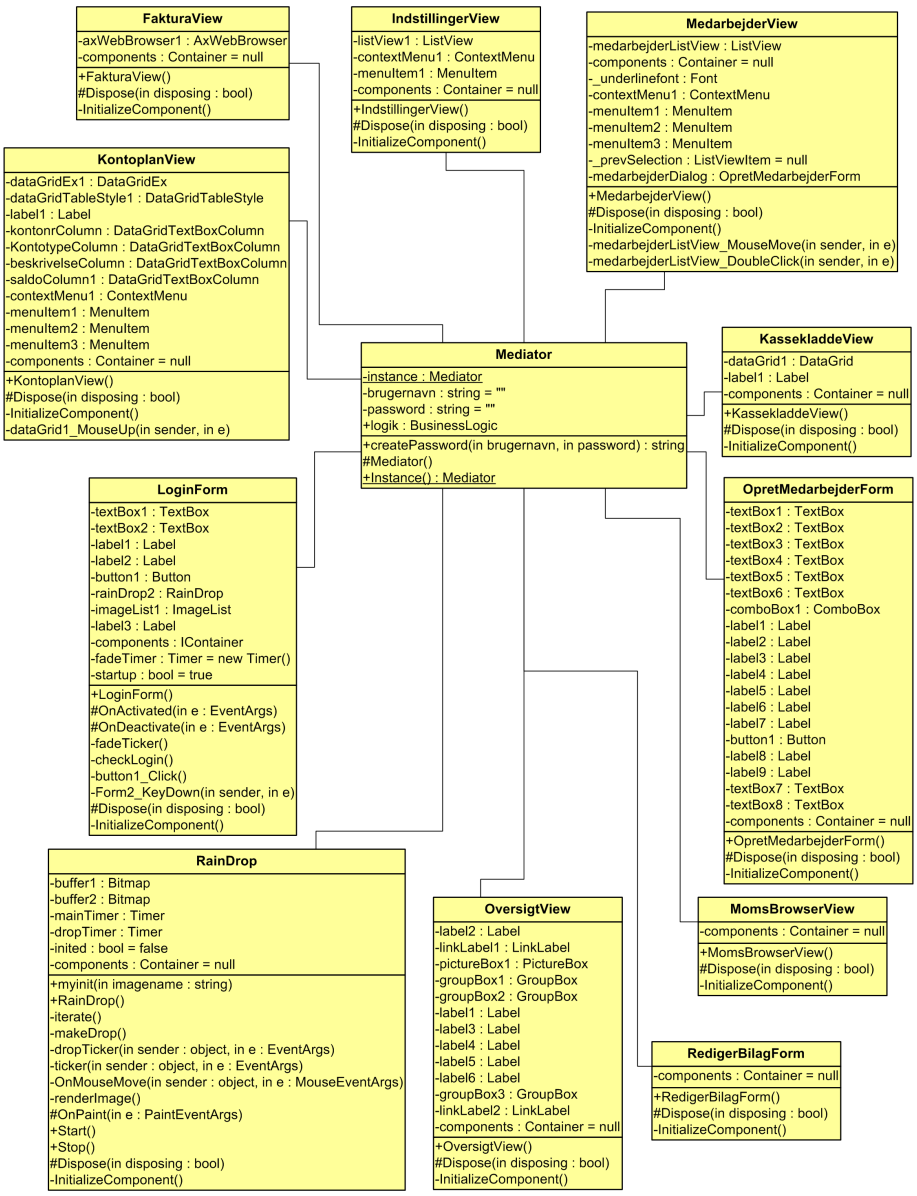
MomsBrowserView Udgør en usercontrol, den er opbygget omkring en WebBrowser komponent (IE 6.0 ActiveX) og benyttes til automatisk at indberette moms til Told og Skat via der hjemmeside. Det kan gøres da det er muligt via WebBrowser komponenten automatisk at indtaste de fornødne oplysninger.

OversigtView Udgør en usercontrol, denne kontrol er den første TabPage der vises efter login, viser status for systemet: Ventende ordrer, lagerstatus, og omsætning fordelt på dag, uge, måned, år.

RainDrop Udgør en usercontrol, denne kontrol laver en simuleret regn effekt oven på et billede. Bruges til at lave en animation af Aconto logoet.

RedigerBilagForm Udgør en formular til at indtaste et nyt bilag.

RedigerMedarbejderForm Udgør en formular til at indtaste/rette en medarbejders oplysninger.



Figur 6.5: Viser UML diagrammet over klasserne i AcontoUI komponenten.

6.2.3 AcontoObjects

Disse objekter sendes frem og tilbage imellem AcontoUI og SOAPWebService.

Adresse Objektklasse, er den interne repræsentation af en adresse.

Bilag Objektklasse, er den interne repræsentation af et bilag

Faktura Objektklasse, er den interne repræsentation af en faktura. Fakturaen henter sine oplysninger fra Ordre klassen.

KasseKladde Objektklasse, er den interne repræsentation af kassekladden. Det er en wrapper omkring et DataSet objekt da det er nemmere at overføre til SOAP komponenten.

Konto Objektklasse, er den interne repræsentation af en konto.

KontoPlan Objektklasse, er den interne repræsentation af kassekladden. Det er en wrapper omkring et DataSet objekt da det er nemmere at overføre til SOAP komponenten.

KreditNota Objektklasse, er den interne repræsentation af en kreditnota, den får sine oplysninger fra Ordre / Faktura.

Kunde Objektklasse, er den interne repræsentation af kunde.

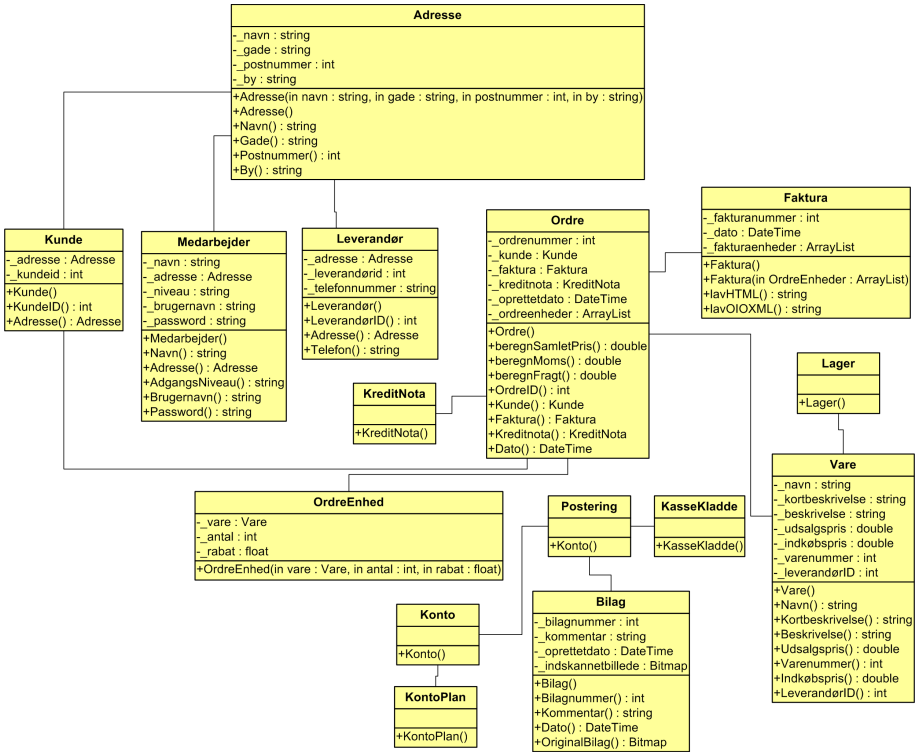
Lager Objektklasse, er den interne repræsentation af lageret består af varer + mængde.

Leverandør Objektklasse, er den interne repræsentation af en leverandør.

Medarbejder Objektklasse, er den interne repræsentation af en medarbejder.

Ordre Objektklasse, er den interne repræsentation af en ordre.

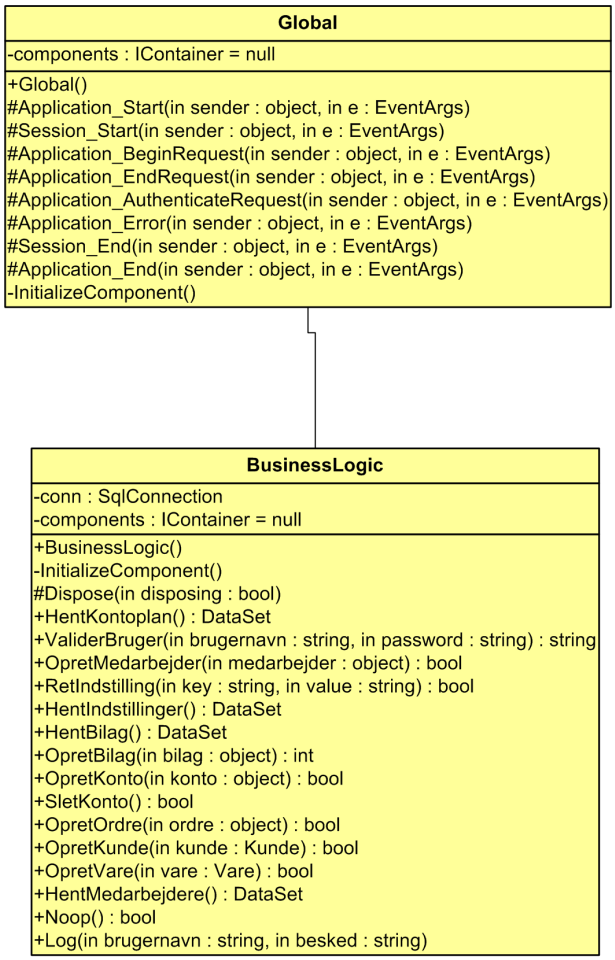
Vare Objektklasse, er den interne repræsentation af en vare.



Figur 6.6: Viser UML diagrammet over klasserne i AcontoObjects komponenten.

6.2.4 SOAPWebService

BusinessLogic WebService klasse, indeholder alle de metoder der kan kaldes fra hovedprogrammet(den viste klasse er ikke færdig).



Figur 6.7: Viser UML diagrammet over klasserne i SOAPWebService komponenten.

6.2.5 Database

Der er oprettet tabeller i databasen svarende til det som er implementeret færdigt, derudover er der bl.a en tabel med alle postnumre og byer i danmark. Herunder vises hvorledes tabellerne oprettes med SQL forespørgelser. Det vise hvilke attributter de enkelte tabeller indeholder.

Adresser

```
CREATE TABLE [dbo].[abc_Adresser] (
    [adresseid] [int] IDENTITY (1, 1) NOT NULL ,
    [navn] [varchar] (50) COLLATE Danish_Norwegian_CI_AS NULL ,
    [gade] [varbinary] (50) NULL ,
    [postnummer] [smallint] NULL ,
    [by] [varchar] (50) COLLATE Danish_Norwegian_CI_AS NULL
) ON [PRIMARY]
```

Indstillinger

```
CREATE TABLE [dbo].[abc_Indstillinger] (
    [key] [varchar] (50) COLLATE Danish_Norwegian_CI_AS NOT NULL ,
    [type] [varchar] (10) COLLATE Danish_Norwegian_CI_AS NOT NULL ,
    [værdi] [varchar] (128) COLLATE Danish_Norwegian_CI_AS NULL
) ON [PRIMARY]
```

Kontoplan

```
CREATE TABLE [dbo].[abc_Kontoplan] (
    [kontonr] [bigint] NOT NULL ,
    [beskrivelse] [varchar] (128) COLLATE Danish_Norwegian_CI_AS NOT NULL ,
    [kontotype] [char] (6) COLLATE Danish_Norwegian_CI_AS NOT NULL ,
    [saldotype] [char] (6) COLLATE Danish_Norwegian_CI_AS NOT NULL ,
    [saldo] [money] NOT NULL
) ON [PRIMARY]
```

Kunder

```
CREATE TABLE [dbo].[abc_Kunder] (
    [kundeid] [int] IDENTITY (1, 1) NOT NULL ,
    [adresseid] [int] NOT NULL
) ON [PRIMARY]
```

Log

```
CREATE TABLE [dbo].[abc_Log] (
    [brugernavn] [varchar] (50) COLLATE Danish_Norwegian_CI_AS NULL ,
    [besked] [text] COLLATE Danish_Norwegian_CI_AS NULL ,
    [dato] [datetime] NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
```

Medarbejdere

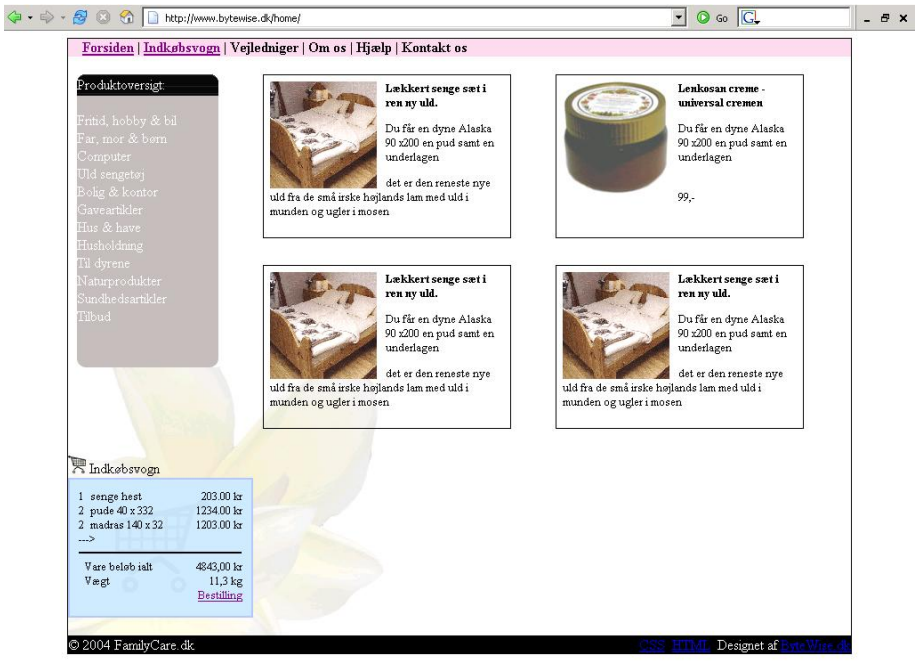
```
CREATE TABLE [dbo].[abc_Medarbejdere] (
    [medarbejderid] [int] IDENTITY (1, 1) NOT NULL ,
    [adresseid] [int] NULL ,
    [brugernavn] [varchar] (50) COLLATE Danish_Norwegian_CI_AS NOT NULL ,
    [password] [varchar] (50) COLLATE Danish_Norwegian_CI_AS NOT NULL ,
    [niveau] [varchar] (50) COLLATE Danish_Norwegian_CI_AS NULL
) ON [PRIMARY]
```

Postnumre

```
CREATE TABLE [dbo].[abc_Postnumre] (
    [postnummer] [int] NOT NULL ,
    [by] [nvarchar] (255) COLLATE Danish_Norwegian_CI_AS NULL
) ON [PRIMARY] GO
```

6.2.6 Hjemmeside

Der er lavet en skabelon til hjemmesiden, koden til den findes på cdrommen. Der vises en skærmbillede af hjemmesiden fra browseren. Det viser den logiske opbygning med indkøbskurv, menu og tilbud på forsiden. Siden er opbygget af moduler og layout er baseret på Cascading StyleSheets dette gør det nemmere at rette i sidens layout.



Figur 6.8: Viser hjemmesidens logiske og grafiske opbygning.

KAPITEL 7

Implementering

I dette afsnit beskrives de teknologier, der er anvendt og de erfaringer, der er gjort under udviklingen af systemet.

7.1 Softwarekrav

For at kunne benytte dette projekt stilles der en række krav til hvilken software, der skal være til rådighed.

Visual Studio 2003 Udviklingsværktøj.

Microsoft Internet Information Server Webserver eller hjemmeside server. Der skal være installeret understøttelse for SSL, ASP og .Net].

Microsoft SQL Server Database server. De nødvendige tabeller skal være oprettet, læs afsnittet "Opsætning af systemet". SE HER

Windows 2000 eller Windows XP Styresystem.

.Net runtime environment version 1.1 Platform.

7.2 C#

Både applikationen og den dynamiske del af hjemmesiden er implementeret i sproget C#. Sproget er udviklet af Microsoft© som en del af deres .NET platform. Sproget har mange af det dets styrker tilfælles med sprog som Java og C++:

Garbage collection[1], stort klasse bibliotek, syntax, polymorphisme, exceptions, etc.

C# afvikles via et runtime layer, dette fungerer på overfladen ligesom Javas virtual machine, men er på mange måder under overfladen forskellig. Bl.a er man med .NET platformen uafhængig af hvilket programmeringssprog man benytter, da der er lavet oversættere til de fleste sprog. Dette kan der læses meget mere om på nettet eller i [1, ch. 1] og [9].

Der er indenfor de sidste 2 år udviklet et runtime layer til unix arkitekturer, hvorfor systemet muligvis kunne gøres platformsuafhængigt. Dette projekt bærer navnet Mono. Det er undersøgt om det var muligt at afvikle programmet problemfrit både via Mono's og Microsoft's runtime layer. Mono projektet er stadig under udvikling og ikke alle klasser/services er tilgængelige og det viste sig desværre, at det blev for omfattende at ændre dette projekts kode til Mono's (endnu) begrænsede API.

7.3 Kommunikation

Der anvendes en SOAP komponent mellem programmet og databasen. SOAP bygger på XML og er en protokol, der er uafhængig af hvilken transportprotokol der anvendes og benyttes oftest som et lag ovenpå de gængse internetprotokoller. Denne rapport beskæftiger sig kun med HTTP som transport protokol, man kunne også have valgt at benytte SMTP eller FTP. Grunden til at der benyttes standard protokoller til transport er, at de fleste af internettets routere og firewalls har god understøttelse af dem.

Der er forskellige alternativer til SOAP metoden bl.a CORBA eller COM+. Begge er metoder til at lave applikations uafhængige distribuerede komponenter. CORBA og COM+ anvender deres egne (transport) protokoller og er derfor afhængige af, at de har kontrol med deres adgang til internettet, hvilket i dag er ret sjældent, da det udgør en stor sikkerhedsrisiko. Med risiko menes, at det ikke er fornuftigt at have åbnet for flere porte i firewallen end der anvendes. Derfor er man er nødt til enten, at tilføje nye regler til sin egen firewall eller bede sin internet udbyder om at åbne for specifikke porte.

7.4 SOAP

Det er nemmest at forklare hvordan SOAP fungerer ved at give et eksempel. Jeg har taget udgangspunkt i den funktion, der hedder ValiderBruger. Denne metode kaldes hver gang applikationen startes, for at undersøge om en bruger

[1] Findes ikke i standart C++ men det er også muligt at bruge Garbage Collection i C++

eksisterer i systemet. Hvis brugeren eksisterer, sendes brugerens adgangsniveau tilbage, ellers sendes en tom streng.

SOAP komponenten modtager denne HTTP forespørgsel. Som det fremgår er der tale om en standard HTTP POST forespørgsel (som normalt bruges når der sendes informationer fra en HTML form i browseren tilbage til serveren). Det interessante starter hvor XML dokumentet begynder, dette kaldes i SOAP regi for en envelope (konvolut). Først defineres hvilke XML namespaces og XML schemaer det overførte XML dokument overholder. Dernæst angives hvilken funktion der ønskes kaldet, som i dette tilfælde er `ValiderBruger`. Til sidst angives parametrene til funktionen.

```
POST /SOAPWebservice/Service1.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: 461
SOAPAction: "http://testsetup/SOAPWebService/ValiderBruger"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ValiderBruger xmlns="http://testsetup/SOAPWebService/">
      <brugernavn>Administrator</brugernavn>
      <password>mqIyX7FpVjN61qKKBLe6nHBdNtnMOFDPrchTZp04AP8=</password>
    </ValiderBruger>
  </soap:Body>
</soap:Envelope>
```

Da funktionen eksisterer og jeg har afgivet en korrekt forespørgsel svarer SOAP komponenten tilbage med:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 422

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ValiderBrugerResponse xmlns="http://testsetup/SOAPWebService/">
      <ValiderBrugerResult>Administrator</ValiderBrugerResult>
    </ValiderBrugerResponse>
  </soap:Body>
</soap:Envelope>
```

Som det ses er der tale om et standard HTTP svar og `HTTP/1.1 200 OK` indikerer, at SOAP komponenten har forstået min forespørgsel. Igen angives hvilke

namespaces og schemaer dokumentet overholder og til sidst kommer svaret fra funktionskaldet, der fortæller at brugeren er medlem af gruppen af administratorer. [Kilde: [3] og www.w3.org]

I C# bruges klassen `WebService` til at lave sin SOAP komponent med. Denne installeres som en komponent på IIS serveren og det gøres automatisk ved hjælp af en wizard, som Visual Studio stiller til rådighed. Det smarte ved at bruge WebService klassen er, at det ikke er nødvendigt selv at generere forespørgslerne og XML konvolutten. Som standard eksisterer kun simple typer, men da det er muligt at sende sine egne typer, er der ingen begrænsninger på hvor komplekse objekter der kan overføres. Dette klares også automatisk af WebService klassen.

Når man ønsker at anvende en webservice i sin klient, stiller Microsoft en speciel oversætter tilrådighed. Denne genererer en klasse, som man umiddelbart kan anvende i sit klientprogram. Klassen sørger automatisk for at skabe forbindelse til serveren og udføre kaldet og konvertere XML svaret til et C# objekt.

7.5 Sikkerhed

SOAP har ikke indbygget nogen form for brugervalidering eller kryptering, dette klares ved at tilføje brugerinfo i SOAP protokollens header. Disse informationer valideres inden dataoverførsel startes. Dette betyder blot at der tilføjes 2 ekstra tags til XML konvolutten:

```
<soap:Header>
  <AuthHeader xmlns="https://bytewise.dk/SoapDataAPI/">
    <Brugernavn>string</Brugernavn>
    <Password>string</Password>
  </AuthHeader>
</soap:Header>
```

For at kunne anvende systemet, er det nødvendigt at man er oprettet som bruger. Dette gøres af administratoren, der har fået sin konto oprettet under installationen af programmet. Da passwordet bliver sendt med hver gang man foretager et kald til SOAP serveren, sendes dette ikke i klar tekst. Der beregnes derfor en hash værdi af passwordet, dette lagres så i databasen. Der findes mange forskellige algoritmer til hashing af data. Specielt skal nævnes MD5, SHA-1, SHA-256 algoritmerne. MD5 beregner en 512 bit lang repræsentation af input data. Der er fundet svagheder i denne algoritme, hvor det er muligt matematisk at beregne 2 input, der giver samme hash værdi [10], dette kaldes en kollision. De har beregnet, at der teoretisk kræves 2^{40} operationer for at finde en kollision. SHA-1 algoritmen beregner en 160 bit lang repræsentation af input data. Der er af de samme mennesker ligeledes fundet svagheder i denne algoritme; de har beregnet,

at der her kræves 2^{69} operationer for at finde en kollision. Artikel omkring svagheden i SHA-1 er ikke offentliggjort endnu men Bruce Schneier har læst artiklen og verificeret resultaterne. SHA-256 beregner en 256 bit lang repræsentation af input og der er endnu ikke fundet nogle svagheder i denne, men nogle mener at det er et spørgsmål om tid. På baggrund af svaghederne i MD5 og SHA-1 algoritmerne vælges SHA-256. Som en ekstra foranstaltning indskydes brugernavnet i passwordet inden det hashes, da man på denne måde ikke kan, ved at kigge i `Medarbejdere` tabellen, identificere brugere, der benytter samme password.

Hashet beregnes i C# på følgende måde.

```
public String createPassword(String brugernavn, String password) {
  SHA256 sha = new SHA256Managed();
  byte[] result = sha.ComputeHash(
    (new System.Text.UnicodeEncoding()).
      GetBytes(password+brugernavn));
  return(Convert.ToBase64String(result));
}
```

SOAP sender som sagt al data i klar tekst og dette udgør en sikkerhedsrisiko. Problemet består i at en eventuel medlytter, en såkladt 'sniffer', nemt kunne læse brugernavnet og det krypterede password og på den måde udføre *uautoriserede* kald til SOAP komponenten. Det kan løses ved at kald til SOAP [3] komponenten udføres via en krypteret SSL forbindelse. SSL er understøttet på de fleste webservere.

I forbindelse databasen er der også også forskellige sikkerhedsmodeller. Man kunne forstille sig, at man lagde de virkelige følsomme oplysninger i en database som ligger indenfor firmaets firewall og kun lagde de oplysninger, der er nødvendige for hjemmesiden i en database, der er direkte adgang til fra internettet. Den løsning, der præsenteres i dette projekt, anvender ikke denne løsning, alle tabellerne er placeret i den samme database.

7.6 Hjemmesiden

Hjemmesiden er skrevet i HTML og den dynamiske del er skrevet i ASP.NET, hvor C# er anvendt som programmeringssprog. Der er 2 måder at udvikle en hjemmeside på når man snakker .NET. Microsoft har lavet et klassebibliotek de kalder `Webform`, hvor man bygger sin hjemmeside op som klasser og så generer WebFormen automatisk og dynamisk den HTML kode, der vises i browseren. Denne metode er ikke anvendt i dette projekt, da jeg ikke ønsker at låse brugeren fast på ASP.NET som platform for hjemmesiden. I stedet er anvendt en metode hvor HTML koden skrives separat og der laves mindre funktioner i C# som kaldes når det er nødvendigt. På denne måde kan hele hjemmeside delen skrives om til fx PHP på en eftermiddag.

Det er nødvendigt at holde styr på hvad kunden har i sin indkøbskurv, dette gøres ved hjælp af et SESSION objekt, der stilles tilrådighed på webserveren. Dette objekt eksisterer så længe kunden er inde på siden og kan bruges til at gemme fx indkøbskurvens status. Som standard fungerer en SESSION ved at der placeres en unik cookie[2] på kundens computer, denne cookie udløber når kunden enten er logget ud eller har været inaktiv for længe. Hjemmesidens C# funktioner har fuldt adgang til SOAP komponenten og den benytter bl.a. `OpretOrdre` funktionen.

7.7 Erfaringer

Der er brugt meget tid på at udforske mulighederne i de standard komponenter, der stilles tilrådighed på .NET platformen. Specielt er der gået meget tid med at undersøge og afprøve GUI-kontrollerne `PropertyGrid` og `DataGrid` og objektklasserne `DataSet` og `DataAdapter`.

`PropertyGrid` benyttes til at sætte attributter på sine objekter. Den generer selv indtastningsfelter, der svarer til de attributter et objekt indeholder. Men efter at have arbejdet en del med den opstod det problem, at man ikke selv kunne bestemme rækkefølgen af attributterne. De kunne enten komme i alfabetisk orden eller i vilkårlig rækkefølge. Dette var ikke hensigtsmæssigt fx i forbindelse indtastning af adresser. Problemet kunne have været løst ved at nedarve fra klassen og udvide klassen med en rækkefølge egenskab, men da det ville tage uforholdsmæssigt lang tid at sætte sig ind i dette, blev denne idé ikke anvendt og det var nødvendigt at redesigne nogle af GUI komponenterne.

`DataGrid` er en GUI kontrol til at vise og redigere tabeldata. Den indeholder et væld af funktionaliteter, men der kan som standard kun anvendes tekst eller boolske input. For at anvende 'dropdown'-menuer etc., er det nødvendigt at man laver sin egen søjle klasse. Det er omfattende arbejde, men der findes eksempler på nettet og det kunne rimeligt nemt implementeres. Problemet opstod da det blev forsøgt at lave individuelle skrifttyper for rækker i tabellerne. Det viste sig at være et omfattende stykke arbejde, hvor man hele tiden kæmpede imod at kontrollen ikke er bygget til at blive udvidet og arbejdet med dette blev droppet, da jeg vurderede, at det ikke var muligt at opnå et tilfredsstillende resultat indenfor rimelig tid.

`DataGrid` kædes sammen med en datakilde. Denne datakilde kan være et `DataSet`, som er en repræsentation af tabeldata og kan indeholde flere tabeller på en gang. Der tilknytter sig et væld af funktionalitet og kontrollen holder bl.a. styr på hvilke rækker, der er opdateret og kan, hvis den anvendes sammen med en `DataAdapter`, selv generere SQL-UPDATE, -INSERT og -SELECT statements. Der opstod problemer med disse forsøg da det blev klart, at hver tabel skulle have en DataAdapter tilknyttet og det vurderedes, at dette ville ødelægge idéen om at det skulle

[2] http://www.webopedia.com/TERM/c/cookie.html

være enkelt at tilføje funktionalitet til systemet.

Derfor er erfaringen fra forfatteren, at det ofte kan betale sig at lave sine egne grafiske komponenter, såfremt at den eksisterende komponent ikke umiddelbart stiller den ønskede funktionalitet til rådighed. Dette forhindrer ikke at man på et senere tidspunkt extender den originale kontrol, men gør at man ikke bliver sinket unødigt i udviklingsprocessen.

Webservices Arbejdet med Webservices på .NET platformen har været en fornøjelse. De værktøjer der bliver stillet til rådighed i Visual Studio .Net 2003 gør det til en fornøjelse at arbejde med Webservices, faktisk er det ikke nødvendigt at kende teorien bag SOAP for at anvende denne klasse.

MSSQL database Denne database indeholder rigtig mange funktioner og systemet kunne nemt laves om, så noget af arbejdet blev fjernet fra SOAP komponenten og flyttet over på database serveren. Det er muligt at lave *Stored Procedures*, som giver en mulighed for at definere sine egne funktioner. Hermed kan man lave meget avancerede SQL forespørgsler med temporære tabeller etc. Det er meget anvendeligt hvis man skal bruge data fra flere tabeller på en gang i en anden forespørgsel. Der er også en *Scheduler* således, at databasen selv kan foretage maintenance: eksempelvis slette log-filer eller lave backup. Der er også mulighed at lave såkaldte *Triggers*, hvor databasen kan holde øje med værdier i databasen og så sende en email hvis der er opstået problemer.

KAPITEL 8

Teknisk test

Det samlede system er ikke fuldt ud implementeret, dette skyldes som sagt i starten at det er et kæmpe domæne med mange logistiske og vidensmæssige udfordringer. Der præsenteres her en gennemgang af hvordan testforløbet skal være den dag produktionsversionen af dette system er færdigt.

De enkelte elementer er blevet testet undervejs for at verificere at de fungerer som ønsket. Derfor skal der gennemføres en samlet test af systemet hvor man gennemgår alle Use Cases. Ved hver test beskrives den handling der testes og det forventede resultat. I testene vises skærmbilleder der viser testens forløb og opdateringer i databasen.

Jeg gennemgår kun Use Case nr. 1. »Log ind« for at vise testforløbet.

8.1 Use Case nr. 1

Denne test undersøger kontakt med SOAP komponenten og brugervalidering ved at forsøge at logge ind på systemet som en medarbejder.

```
select *  from abc_Medarbejdere
```

	medarbejderid	adresseid	brugernavn	password	niveau
1	1	NULL	Administrator	mqIyX7FpVjN61qKKBLe6nHBdNtn...	Administrator

Figur 8.1: Indholdet af abc_Medarbejdere tabellen.

8.1.1 Log ind test 1

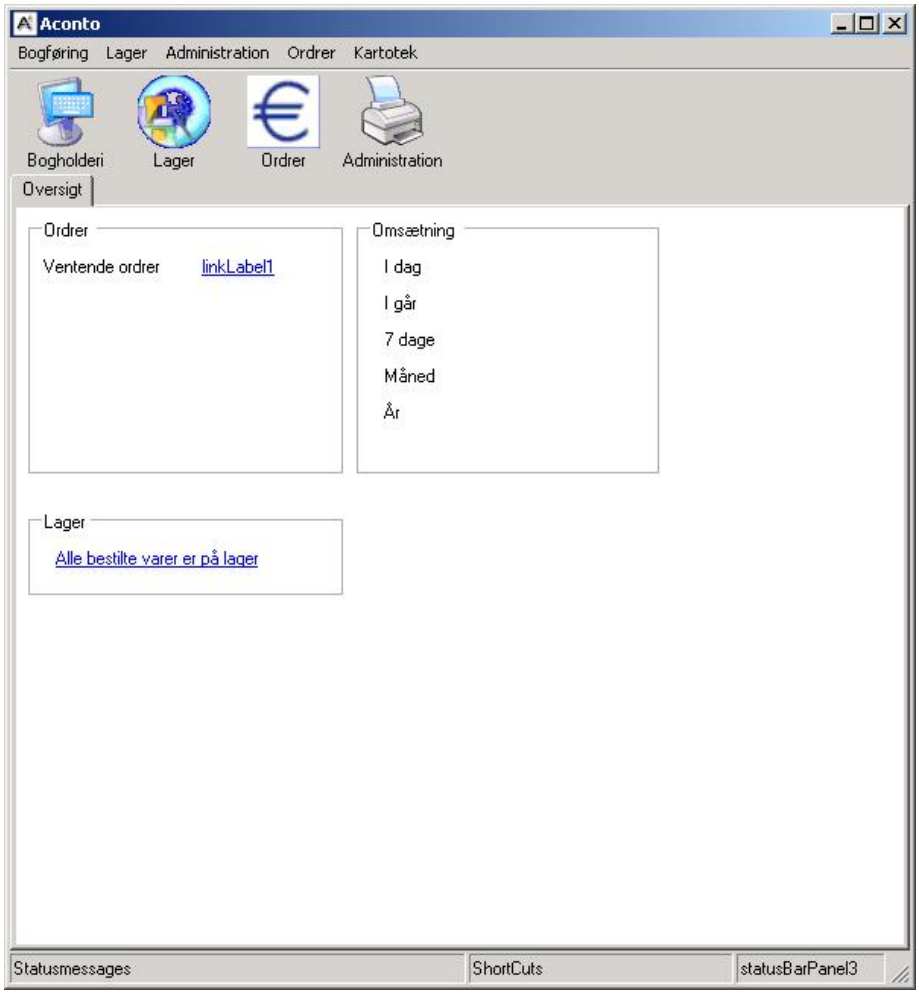
Handling	Forventet resultat	Reaktion
Fra en pc med Windows 2000 og .NET 1.1 installeret startes Aconto.exe	Programmet starter op og der vises en login formular (figur 8.2) vises.	OK

Figur 8.2: Login formularen.

8.1.2 Log ind test 2

Handling	Forventet resultat	Reaktion
Et gyldigt brugernavn og password (figur 8.3 for en medarbejder indtastes (hhv. 'Administrator' og 'volapyk') og der trykkes på 'Log ind' knappen	Formularen skriver '*' når passwordet indtastes. Programmet starter op og der vises en oversigt over status for systemet (figure 8.4.	OK

Figur 8.3: Login som Administrator.



Figur 8.4: Hovedprogrammet, viser ved opstart altid status/oversigt vinduet.

8.1.3 Log ind test 3

Handling	Forventet resultat	Reaktion
Et ikke gyldigt brugernavn og password indtaste (hhv 'Anders' og '1234')	Systemet afviser login forsøget, skriver en fejlmeddelse 'Prøv igen' pg er klar til ny indtastning (figur 8.5).	OK

Figur 8.5: Login som Anders. Brugernavnet og password afvises

KAPITEL 9

Konklusion og perspektivering

9.1 Konklusion

Der er lavet en behovsanalyse af hvilke krav der stilles til et stykke software, der skal dække den daglige administration i en mindre internetbaseret handelsvirksomhed. De eksisterende systemer blev kort gennemgået i denne process.

Der er opstillet en kravspecifikation og Use Cases til de dele af softwaren, der er implementeret. Desuden er der præsenteret en samlet software implementation i C# der, på grund af den objektorienterede opbygning, er nem at tilføje ny funktionalitet.

Den præsenterede løsning er baseret på en 3-trins model og inkluderer et præsentationslag, et forretningslag og et datalag. Denne struktur er gennemarbejdet og programmets overordnede struktur ligger fast. Der er lavet en detaljeret analyse af de tekniske krav og overvejelser i forbindelse med udviklingen af softwaren.

Systemet har ikke gennemgået en samlet teknisk test da softwaren ikke er klar til at komme ud i et produktionsmiljø.

9.2 Perspektivering

Brugergrænsefladen Systemets grundlæggende design er på plads, men der skal arbejdes videre med den grafiske brugergrænseflade. Der skal herunder tilføres flere fejl beskeder således at brugere, der ikke har anvendt systemet før, hele tiden føler at de har kontrol med hvad som foregår.

Hjemmesiden Det kunne være interessant at systemet benyttede et mindre *Content Management System* system, sådan at der kan foretages mindre rettelser af hjemmesiden uden at man skal rette direkte i den bagvedliggende kode. Herunder skulle der fokuseres på at lave et modul til at lave specielle tilbud på hjemmesiden.

CRM I stedet for at anvende et stand-alone mail-program skal denne del integreres i systemet således, at man får et nemt overblik over korrespondancen med den enkelte kunde.

Generelt Der skal tilføjes flere funktioner, der letter den daglige brug af systemet. Et centralt emne er at tilføje funktionalitet i forbindelse med bogføringen således, at der kan designes skabeloner til mest anvendte arbejdsgange. Dette kunne være formularer til at lave kørselsregnskab, lønregnskab og repræsentationsregnskab. Dette ville gøre at bogholderen ikke skulle huske, i hovedet, hvilke konti der skal benyttes ved disse posteringer. Dette ville føre til færre fejlposteringer og gøre fejlhåndteringen bedre.

Der findes i .NET en funktionalitet der minder om WebStart mekanismen i Java. Det vil være interessant at distribuere applikationen via denne metode, da det så er meget nemt at opdatere softwaren. Microsoft kalder denne funktionalitet for »Smart Client« og beskriver den i detaljer på:
http://msdn.microsoft.com/netframework/using/building/windows/analystreports/smartclient.aspx

Litteratur

[1] Derek Beyer. *C# COM+ Programming.* M&T BOOKS, 1 th. edition, 2001.

[2] Finn Blomhøj. *Introduktion til erhvervsøkonomi.* Merko Gyldendal Uddannelse, first edition, 2001.

[3] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. Securing soap e-services. *International Journal of Information Security*, 2002. ISSN 16155262.

[4] Martin Fowler. *UML Distilled.* Addison-Wesley, third edition, 2004.

[5] Bruno Kjær og Kurt Mellemgaard Lone Elgkær. *Grundlæggende Regnskabslære.* ØKNOM, 2. udg. edition, 1998.

[6] Arbejdsgruppe nedsat af Erhvervs-og Selskabsstyrelsen. Bogføringsvejledningen. Juni 1999. Formålet med vejledningen er først og fremmest at supplere bogføringsloven, der blev vedtaget den 17. december 1998. Den kan findes på www.eogs.dk.

[7] Jakob Nielsen. *Designing Web Usability: The Practice of Simplicity.* New Riders Publishing, Thousand Oaks, CA, USA, 1999.

[8] Ian Sommerville. *Software Engineering.* Addison Wesley, 6 th. edition, 2001.

[9] Andrew Troelsen. *C# and the .NET Platform.* A! Apress, 2002.

[10] Xiaoyun Wang, Dengguo Feng, Xuejia Lai, and Hongbo Yu1. Collisions for hash functions md4, md5, haval-128 and ripemd. 2004. Kan findes på http://eprint.iacr.org/2004/199.pdf.

APPENDIKS A

Relevante links

www.erhverv.toldskat.dk Told og Skat's hjemmeside der henvender sig til firmaer, her kan alle regler og krav til firmaer findes.

www.eogs.dk Erhvervs og Selskabsstyrelsens hjemmeside. her findes mange supplerende oplysninger til virksomheder og også informationer om EDI, OIOXML, XUBL.

www.erpbasen.dk Oversigt over ERP systemer, oprettet april 2005.

ndoc.sourceforge.net Open Source dokumentations værktøj til .net platformen. Kan lave dokumentationen som bl.a. html og latex.

www.oio.dk *Offentlig Information Online*. Indeholder oplysninger om de tekniske krav til E-faktura. bl.a findes OIOXML specifikationen her.

www.msdn.com *Microsoft Developer Network* Her findes beskrivelser af API'er, kodeeksempler og beskrivelser af tekniske problemstillinger.

www.post.dk Post Danmarks hjemmeside, her findes postnumre og informationer om fragtomkostninger, EDI etc.

www.uml.org Siden beskriver den seneste UML standard, siden varetages af OMG som er det standardiserings organ der varetager UML standarden.

www.w3.org *World Wide Web Consortium* varetager mange af de åbne standarder bl.a HTML, XML og SOAP.

APPENDIKS B

Ordforklaring

CRM *Customer Relationship Management* Begreb der beskriver processen med hele tiden at vedligeholde kontakten til kunderne og følge op på ordre ved at få feedback på kunden så hele kundekontakten afspejler kundes ønsker.

COM+ *Component Object Model+* Arkitektur der benyttes når man vil lave distribuerede komponenter.

CORBA *Common Object Request Broker Architecture* Arkitektur der benyttes når man vil lave distribuerede komponenter. Der er udviklet CORBA moduler til de fleste programmeringssprog. Javas RMI er opbygget på samme måde som CORBA.

ERP *Enterprise Resource Management* Begreb der dækker at samle al viden og resourcer i et samlet system. Det skulle gerne give mere effektivitet og større overskud.

GUI *Graphical User Interface* Grafisk brugergrænseflade.

HTML *HyperText Markup Language* Markup sprog der benyttes ved formattering af hjemmesider. Der er kommet en pendant XHTML som har samme funktionalitet som HTML, men er baseret på XML.

RMA *Return Merchandise Authorization* Begreb der anvendes når en varer sendes retur fra en kunde oprettes en RMA sag. Det skal helst foregå ved at kunden opretter en RMA sag og påføre et RMA nummer til den sendte vare.

SOAP *Simple Object Access Protocol* XML baseret protokol der kan benyttes oven på de eksisterende protokoller. Benyttes i distribuerede systemer.

SSL *Secure Socket Layer* Den gængse standart for krypterede forbindelser via HTTP-protokollen.

UBL *Universal Business Language*

UML *Unified Modelling Language* Overvejende grafiske notationer til beskrivelse af software.

VANS *Value Added Network Services* Et netværk der varetages af en række store firmaer, det muliggør sikke kommunikation virksomheder og institutioner imellem.

XBRL *eXtensible Business Reporting Language* Et XML baseret sprog der på sigt skal benyttes når man begynder at kunne aflægge sit årsregnskab elektronisk.

XML *eXtensible Markup Language*

APPENDIKS C

Eksisterende ERP systemer

Oplysningerne er hentet fra hjemmesiden *www.erpbasen.dk*, hvor alle systemerne er listet, nogle af dem er beskrevet meget detaljeret og andre er kun beskrevet med navn og producent. Derfor er der tomme pladser i tabellen.

IH søjlen angiver om produktet er forberedt for internethandel.
KA søjlen angiver om der påkræves konsulent arbejde.

Navn	**Producent**	**IH**	**K**A
Admiral for Windows	Admiral Software A/S	nej	nej
Aspect/4	EDB Gruppen A/S	ja	ja
ASW	IBS Danmark A/S	nej	ja
e-conomic	E-conomic ApS	nej	nej
Guide	Guideix A/S	ja	ja
HansaWorld	HansaWorld	?	?
IFS Applications	IFS AB	nej	ja
iScala	Strongline A/S	ja	ja
JD Edwards	Peoplesoft	nej	ja
KeyBalance	Atomic Software A/S	nej	ja
Maconomy	Maconomy A/S	nej	ja
MaconomyOne	Maconomy A/S	ja	ja
Mapics XA	Mapics Solutions A/S	nej	ja
MBS Axapta	Microsoft Business Solutions ApS	ja	ja
MBS C5	Microsoft Business Solutions ApS	ja	ja
MBS C5 Light	Microsoft Business Solutions ApS	ja	ja
MBS Navision	Microsoft Business Solutions ApS	ja	ja
MBS XAL	Microsoft Business Solutions ApS	ja	ja

Fortsættes på næste side

– Fortsat fra forrige side			
Movex	Intentia A/S	ja	ja
MySAP All-in-One	SAP AG	ja	ja
MySAP Business Suite	SAP AG	?	?
Octopus	Paaup Hansen ApS	ja	ja
OfficeFinancials	Lykke Gruppen ApS	ja	nej
Opus2010	Elite Software ApS	nej	ja
Oracle Applications	Oracle Danmark ApS	?	?
Oracle E-business Suite SE	Oracle Danmark ApS	ja	ja
Peoplesoft Enterprise	Peoplesoft	?	?
Peoplesoft EnterpriseOne	Peoplesoft	?	?
Revilution	Revilution A/S	ja	nej
SAP Business One	SAP Danmark A/S	ja	ja
SSA Baan	SSA Global A/S	ja	ja
Stellar Office	Allerup edb AS	?	?
SummaSummarum	Stones Software	nej	nej
Visma Business	Visma Software A/S	ja	ja
Winfinans	Winfinans aps	ja	ja

APPENDIKS D

Kontoplan

Kontoplan for Resultatopgørelsen	
DEBET (aktiv)	**KREDIT (passiv)**
2 Vareforbrug	**1 Nettoomsætning**
2100 Vareforbrug	1100 Varesalg, indland
	1200 Varesalg, EU
3 Andre eksterne omkostninger	1300 Varesalg, øvrig eksport
3100 Salgsfremmende omkostninger	
3200 Lokaleomkostninger	**7 Finansielle indtægter**
3300 Tab på tilgodehavender	7100 Renteindtægter
3400 Kassedifferencer	7200 Kontantrabat fra leverandører
3500 Bilers driftsomkostninger	7400 Indtægter fra værdipapirer
3600 Salgsfragt	7500 Indtægter fra omsætningsværdipapirer
3900 Øvrige omkostninger	7600 Valutakursgevinster
4 Personale omkostninger	**9 Skat og ekstraordinære poster**
4100 Lønafregning	9200 Ekstraordinære indtægter
4200 ATP-bidrag	
4300 AM- og SP-bidrag	
5 Afskrivninger	
5100 Afskrivning på udviklingsomkostninger	
5200 Afskrivning på goodwill	
5300 Afskrivning på grunde og bygninger	
5400 Afskrivning på tekniske anlæg	
5500 Afskrivning på biler	
5600 Afskrivning på inventar	
8 Financielle omkostninger	
8110 Renteomkostninger	
8120 Prioritetsrenter	
8200 Kontantrabat til kunder	
8600 Valutakurstab	
9 Skat og ekstraordinære poster	
9100 Selskabsskat	
9300 Ekstraordinære omkosninger	

Kontoplan for Balancen	
AKTIVER (debet)	**PASSIVER (kredit)**
11 Anlægsaktiver	**13 Egenkapital**
111 Immaterielle	13110 Aktie-/anparts-/kontokapital
11110 Udviklingsomkostninger	13120 Privatforbrug (kun i pers virks.)
11111 Akkum. afskr.på udviklingsomkostninger	13210 Overkurs ved emmision
11120 Goodwill	13310 Opskrivningshenlæggelser
11121 Akkum. afskr. på goodwill	13410 Reserver
	13510 Overført resultat
112 Materielle anlægsaktiver	
11210 Grunde og bygninger	**14 Hensættelser**
11211 Akkum. afskr. på tekniske grunde og bygninger	14100 Hensættelse til udskudt skat
11220 Tekniske anlæg og maskiner	14200 Hensættelse til pensioner
11221 Akkum. afskr. på tekniske anlæg og maskiner	14300 Andre hensættelser
11230 Biler	**15 Gæld**
11231 Akkum. afskr. på biler	**151 Langfristet**
11240 Inventar	15110 Prioritetslån
11241 Akkum. afskr. på inventar	15120 Valutalån
	15130 Andre langfristede lån
113 Finansielle anlægsaktiver	
11310 Kapitalandele i datterselskaber	**152-9 Kortfristet**
11320 Tilgodehavender hos datterselskaber	15210 Kassekredit
11330 Andre anlægsværdipapirer	15250 Modtagne forudbetalinger fra kunder
	15270 Skyldig arbejdsløn
12 Omsætningsaktiver	15310 Varekreditorer/ gæld til leverandører
121 Varebeholdninger	15320 Omkostningskreditorer
12110 Varelager	15400 Skyldig selskabsskat
	15510 Skyldigt ATP-bidrag
122 Tilgodehavender	15520 Skyldigt AM og SP bidrag
12210 Varedebitorer / tilgodehavende hos kunder	15530 Skyldigt A-skat
12211 Forventet tab på varedebitorer	15610 Købsmoms
12220 Tilgodehavender hos datterselskaber	15615 Energiafgift
12230 Andre tilgodehavender	15620 Salgsmoms
12240 Periodeafgrænsningsposter	15625 Moms af EU-erhvervelser.
	15630 Momsafregning
123 Omsætningsværdipapirer	15640 Skyldig told og importmoms
12310 Omsætningsværdipapirer	15730 Andre kreditorer
	15810 Periodeafgrænsingsposter
124 Likvide beholdninger	15910 Skyldigt udbytte for regnskabsåret
12410 Kasse	15915 Skyldigt udbytte for tidligere år
12420 Bank	
12430 Giro	**Afslutningskonti**
	21000 Resultat opgørelse
	22000 Balance

[Kilde: [2, s. 44 og 45]]

APPENDIKS E

Konteringsvejledning

Nummer	Navn	Debet	Kredit	Saldo
1100-1300	Varesalg	Dekorter, returvarer, returemballage, mv.	Kontantsalg, Kreditsalg	Kredit: Nettoomsætning
2100	Vareforbrug	Indkøbspriser for solgte varer (overført fra varelager kontoen)		Debet: Periodens vareforbrug
3100	Salgsfremmende omkoskninger	Markedsføringsomkostninger, fx salgsbrochurer, reklamefilm, annoncer og udstillinger	Dekorter, Returnerede varer	Debet: Periodens nettoforburg
3200	Lokaleomkostninger	Husleje, varme, belysning, rengøring, reparation og vedligeholdelse	Dekorter	Debet: Periodens nettoforbrug
3300	Tab på tilgodehavender	Tab og hensættelser til imødegåelse af tab		Debet: Periodens tab på kunder
3400	Kassedifferencer	Manglende penge i kassen	For mange penge i kassen	Debet eller Kredit: Periodens difference
3500	Bilers driftsomkostninger	Vægtafgift, forsikringer, brændstof, reparation og vedligeholdelse	Dekorter	Debet: Periodens nettoforbrug
3600	Salgsfragt	Fragt og kørsel af solgte varer	Dekorter	Debet: Periodens nettoforbrug

Fortsættes på næste side

– Fortsat fra forrige side				
Nummer	**Navn**	**Debet**	**Kredit**	**Saldo**
3900	Øvrige omkostninger	Driftsomkostninger der ikke kan henføres til en af de øvrige konti, fx. Emballage, forsikringer, revision, telefon, fax, internet, kontorartikler og faglige kontingenter	Dekorter, Returnerede varer	Debet: Periodens nettoforbrug
4100	Lønafregning	Udbetalt skattepligtig løn til medarbejderne (bruttoløn - ATP og AM bidrag)		Debet: Periodens lønomkostninger
4200	ATP-bidrag	Virksomhedens og medarbejdernes skattefrie pensionsbidrag.		Debet: Periodens ATP-bidrag
4300	AM-/SP-bidrag	Medarbejdernes arbejdsmarkeds- og særligt pensionsbidrag		Debet: Periodens tilbageholdte bidrag til AM og SP.
5100-5600	Afskrivninger	Årets afskrivninger på anlægsaktiver - i hovedgrupper		Debet: Periodens anlægsafskrivninger
7100	Renteindtægter		Rente af indestående i pengeinstitutter og tilgodehavender hos kunder	Kredit: Periodens renteindtægter
7200	Kontantrabat fra leverandører		Opnåede kontantrabatter ved rettidig betaling	Kredit: Periodens kontantrabatter
7400	Indtægter fra værdipapirer		Udbytte og renter fra aværdipapirer	Kredit: Periodens renter og udbytte fra værdipapirer
7500	Valutakursgevinster		Kursgevisnter på udenlandske mellemværende	Kredit: Periodens valutakursgevinst
8110	Renteomkostninger	Rente og provision af kassekreditter og lån i pengeinstitutter (ekskl. Prioritetslån)		Debet: Periodens renteomkostninger
8120	Prioritetsrenter	Rente af lån mod pant i fast ejendom		Debet: Periodens prioritetsrenteomkostninger
8200	Kontantrabat til kunder	Ydede kontantrabatter, bl.a. ved rettidig indbetaling		Debet: Periodens kontantrabat til kunder
8600	Valutakurstab	Kurstab på udenlandske mellemværender		Debet: Periodens kurstab

Fortsættes på næste side

– Fortsat fra forrige side				
Nummer	**Navn**	**Debet**	**Kredit**	**Saldo**
9100	Selskabsskat	Beregnet skat af årets overskud før skat		Debet: Periodens skat
9200	Ekstraordinære indtægter		Engangsindtægter, fx.forsikringserstatning, fortjeneste ved salg af fast ejendom	Kredit: Periodens ekstraordinære indtægter
9300	Ekstraordinære omkosninger	Engangsudgifter, fx. Tab ved salg af fast ejendom		Debet: Periodens ekstraordinære udgifter
11110	Udviklingsomk.	Købspris + opstillingsomkostninger (=anskaffelsesprisen)	Salgspris for brugte aktiver. Tilbageførte afskrivninger. Tab ved salg	Debet: Anskaffelsespris (eller kostpris) for de aktiver, der anvendes
11120	Goodwill			
11210	Grunde og bygninger			
11220	Tekniske anlæg og maskiner			
11230	Biler			
11240	Inventar			
11111	Akkum. afskr.på udviklingsomkostninger	Tilbageførte akkumulered afskrivninger på anlæg	Periodens afskrivninger	Kredit: Akkumulerede afskrivninger på anlæg der anvendes
11121	Akkum. afskr. på goodwill			
11211	Akkum. afskr. på tekniske grunde og bygninger			
11221	Akkum. afskr. på tekninske anlæg og maskiner			
11231	Akkum. afskr. på biler			
11241	Akkum. afskr. på inventar			
11310	Kapitalandele i datterselskaber	Beholdning af aktier og anparter i datterselskaber (vurderngsprincip angives)	Salg af aktier og anparter i datterselskaber (faktiske salgspriser)	Debet: Værdien af investernger i datterselskaber
11320	Tilgodehavender hos datterselskaber	Lån til datterselskabet	Afdrag på lån fra datterselskabet	Debet: Nettolån til datterselskaber
11330	Andre anlægsværdipapirer	Køb af anlægsværdipapirer, aktier og anparter (vurderngsprincip angives)	Salg afanlægsværdipapirer (Faktisk salgspris)	Debet: Beholdning af anlægsværdipapirer

Fortsættes på næste side

– Fortsat fra forrige side				
Nummer	**Navn**	**Debet**	**Kredit**	**Saldo**
12110	Varelager	Indkøbspris iflg. Faktura. Hjemtagelsesomkostninger	Dekorter og returvarer. Udleveret til privatforbrug.Periodevis overførsel af vareforbrug	Debet:Varelagerets værdi ved periodens afslutning
12210	Varedebitorer / tilgodehavende hos kunder	Faktureret varesalg. Renter aftilgodehavender	Indbetalinger fra kunder. Kontantrabatter. Dekorter og returvarer	Debet: Tilgodehavender hos kunder
12211	Forventet tab på varedebitorer	Tilbageført forventet tab på debitorer (kunder) eller konstaterede tab	Beløb hensat til imødegåelse af tab på debitorer	Debet: Forventet tab på debitorer
12220	Tilgodehavender hos datterselskaber	Salg til datterselskaber	Betaling fra datterselskaber. Dekorter og returvarer	Debet: Tilgodehavende hos datterselskaber
12230	Andre tilgodehavender	Andre tilgodehavender end varetilgodehavender	Afdrag på tilgodehavender	Debet: Nettotilgodehavende
12240	Periodeafgrænsningsposter	Beholdning/forudbetalte beløb, der vedrører en senere regnskabsperiode		Debet: Nettobeoldninger / forudbetalinger ved periodens udgang
12310	Omsætningsværdipapirer	Køb af kortfirstede værdipapirer	Salg af kortfristede værdipapirer	Debet: Beholdning af kortfristede værdipapirer (handelsværdi)
12410	Kasse	Indebetalinger. Kassedifferencer: for meget i kassen	Udbetalinger. Kassedifferencer: for lidt i kassen	Debet: Kassebeholdning
12420	Bank	Indbetalinger (kontant, checks, overførelser). Rente af bankindestående	Hævet v. Checks, dankort og kontant	Debet: Bankindestående
12430	Giro	Indsat, rentetilsrkivning og overførte indbetalinger	Hævet, overførte udbetalinger, gebyrer og blanketter	Debet: Giroindestående
13110	Kapitalkonto	Overført fra privatforbrug og evt. underskud	Kapitalindskud, periodens overskud	Kredit: Nettoegenkapital ved periodens slutning
13110 (alternativ)	Aktiekapital / Anpartskapital	Tilbagebetaling af aktie- eller anpartskapital	Pålydende værid af aktie- eller anpartskapital	Kredit: Egenkapitalandel: Aktie- eller anpartskapital ved periodens afslutning
13120	Privatforbrug (kun i pers virks.)	Kontant privatforbrug. Varer udleveret til indehaver - incl. Moms. B-skat for indehaveren	Saldo overført til kapitalkonto	Ingen

Fortsættes på næste side

– Fortsat fra forrige side				
Nummer	**Navn**	**Debet**	**Kredit**	**Saldo**
14100	Hensættelse til udskudt skat	Reduktioner	Hensatte beløb til udskudt skat	Kredit: Hensættelser ved periodens afslutning
14200	Hensættelse til pensioner	Løbende udbetalinger	Hensatte beløb	Kredit: Nettohensættelser ved periodens afslutning
14300	Andre hensættelser	Udbetalinger	Hensættelser	Kredit: Nettohensættelser ved periodens afslutning
15110	Prioritetslån	Afdrag	Eksisterende og nye lån mod pant i fast ejendom	Kredit: Nettogæld ved periodens afslutning
15120	Valutalån	Afdrag. Kursgevinst på afdrag/restgæld	Eksisterende og nye lån i fremmed valuta. Kurstab på afdrag/restgæld	Kredit: Nettogæld ved periodens afslutning
15130	Andre langfristede lån	Afdrag	Optagne lån	Kredit: Nettogæld ved periodens afslutning
15210	Kassekredit	Indsat	Hævet. Rente og provision	Kredit: Kredit: Nettogæld ved periodens afslutning
15250	Modtagne forudbetalinger fra kunder	Leverancer	Modtagne forudbetalinger	Kredit: Nettoforudbetalinger fra kunder. (Overføres til 15810 periodeafgrænsninger)
15270	Skyldig arbejdsløn			
15310	Varekreditorer	Betaling til leverandørerne. Kreditnotaer fra leverandørerne over returvarer, emballage og dekorter.	Fakturaer på kreditkøb. Renter af leverandørgæld	Kredit: Nettogæld ved periodens afslutning
15320	Omkostningskreditorer	Betaling til leverandørerne. Kreditnotaer fra leverandørerne over returvarer, emballage og dekorter.	Fakturaer på kreditkøb. Renter af leverandørgæld	Kredit: Nettogæld ved periodens afslutning
15400	Skyldig selskabsskat	Betalt skat	Hensat skyldig skat	Kredit: Skattegæld ved periodens afslutning
15510	Skyldigt ATP-bidrag	Betalt ATP-bidrag	Tilbageholdt ATP-bidrag fra medarbejderne. Arbejdsgivers ATP-bidrag	Kredit: Nettogæld til ATP ved periodens afslutning
15520	Skyldigt AM og SP bidrag	Betalt SP- og AM-bidrag	Tilbagehodlte SP- og AM-bidrag fra medarbjderne	Kredit: Nettogæld til AM og SP ved periodens afslutning
15530	Skyldigt A-skat	Betalt A-skat for medarbejderne	Tilbageholdt A-skat hos medarbejderne	Kredit: Nettogæld A-skat ved periodens afslutning

Fortsættes på næste side

– Fortsat fra forrige side				
Nummer	**Navn**	**Debet**	**Kredit**	**Saldo**
15610	Købsmoms	Moms af købsfakturaer, fx. Varer fast anlæg og omkostninger	Moms iflg kreditnota: returvarer og dekorter. Saldooverførsel til momsafregning	Kredit: Saldoen overføres periodevis til konto for Momsafregning
15615	Energiafgift	Betalt energiafgist	Tilbageholdt energiafgift	Kredit: Skyldig energiafgift
15620	Salgsmoms	Moms iflg. Kreditnota: returvarer og dekorter. Saldooverførsel til momsafregning	Moms af salgsfakturaer: varer og fast anlæg. Moms af varer udleveret til indehaverens privatforbrug	Debet Saldoen overføres periodevis til konto for Momsafregning
15625	Moms af EU-erhvervelser.		Eu-købsmoms af varekb i EU-lande	Kredit: Netto EU-moms
15630	Momsafregning	Overført fra købsmoms. Betaling af forfalden momsgæld	Overført fra Salgsmoms. Godtgørelse fra Told og Skat af tilgodehavende moms	Debet eller kredit: Nettotilgodehavende eller gæld til Told og Skat
15640	Skyldig told og importmoms	Betalt til Told og Skat	Pålagt told og importmoms ved import	Kredit: Nettogæld ved periodens afslutning
15730	Andre kreditorer	Afdrag på gæld	Fakturaer på køb af anlægsaktiver og omkostningsvarer	Kredit: Nettogæld periodens afslutning
15810	Periodeafgrænsningsposter		Saldo fra konto 15250 (Forudbetalinger fra kunder). Øvrige periodeafgrænsninger til passiv	Kredit: For meget indtægtsført ved periodens slutning
15910	Skyldigt udbytte for regnskabsåret	Betalt udbytte	Erklæret udbytte	Kredit: Skyldigt udbytte for regnskabsåret
15915	Skyldigt udbytte for tidligere år	Betalt udbytte for tidligerer år	Overført restudbytte fra tidligere år	Kredit: Skyldigt udbytte fra tidligere år. Der vil normalt oprettes en konto for hvert regnskabsår, idet udbytte efter 5 år kan overføres til egenkapitalen
21000	Resultat opgørelse			
22000	Balance			

[Kilde: [2]]

BILAG I

Detaljeret kodebeskrivelse

Dette bilag er genereret udfra XML kommentarer i koden. På den måde bliver alle klasser, funktioner, og attributter beskrevet et sted. XML strukturerene trækkes ud med værktøjet NDoc som laver Latex dokumentet. *http://ndoc.sourceforge.net/*

Contents

Chapter 1

Namespace Aconto

Namespace Contents *Page*

Interfaces

Classes

1.1 Interfaces

1.2 Classes

1.2.1 Class **AcontoForm**

Summary description for Form1.

Declaration

public class AcontoForm : Form

Constructors

- *.ctor*

  `public` **AcontoForm**`( )`

  Opretter hovedvinduet til programmet er en tom skal med menubar, toolbar og tabbedpages. Det er kun oversigten der vises ved login de andre funktioner tilgs ved at bruge toolbaren.

Methods

- *Dispose*

  `protected` **void Dispose**`( )`

  Clean up any resources being used.

  - **Parameters**
    - `disposing` -

Extended Information

- Assembly: `Aconto`

1.2.2 Class **KassekladdeTabPage**

Opretter den TabPage der indeholder KasseKladdeView, det er en tom skal hvor funktionalitet ligger i KasseKladdeView.

Declaration

public class KassekladdeTabPage
: TabPage

Constructors

- *.ctor*

  `public` **KassekladdeTabPage**`( )`

  Opretter den tabpage der viser kassekladden.

Methods

- *Dispose*

  `protected` **void Dispose**`( )`

  Clean up any resources being used.

  - **Parameters**
    - `disposing` -

Extended Information

- Assembly: `Aconto`

1.2.3 Class **KontoplanTabPage**

Opretter den TabPage der indeholder KontoPlanViewView, det er en tom skal hvor funktionalitet ligger i KontoPlanView.

Declaration

```
public class KontoplanTabPage
  : TabPage
```

Constructors

- *.ctor*

  `public` **KontoplanTabPage**`( )`

  Opretter en TabPage der viser kontoplanen.

Methods

- *Dispose*

  `protected` **void Dispose**`( )`

  Clean up any resources being used.

  - **Parameters**
    - `disposing` -

Extended Information

- Assembly: `Aconto`

1.2.4 Class **MedarbejderTabPage**

Summary description for AdministratorTabPage.

Declaration

public class MedarbejderTabPage : TabPage

Constructors

- *.ctor*

  `public` **MedarbejderTabPage**`( )`

  Opretter en TabPage der viser medarbejdere.

Methods

- *Dispose*

  `protected` **void Dispose**`( )`

  Clean up any resources being used.

  – **Parameters**

  * `disposing` -

Extended Information

- Assembly: `Aconto`

1.2.5 Class **OversigtTabPage**

Summary description for OversigtTabPage.

Declaration

```
public class OversigtTabPage
  : TabPage
```

Constructors

- *.ctor*

  `public` **OversigtTabPage**`( )`

  Initializes a new instance of the class.

Methods

- *Dispose*

  `protected` **void Dispose**`( )`

  Clean up any resources being used.

  – **Parameters**

    * `disposing` -

Extended Information

- Assembly: `Aconto`

Chapter 2

Namespace AcontoObjects

Namespace Contents *Page*

Interfaces

Classes

Vare 30
Reprsenterer en vare benyttes.

2.1 Interfaces

2.2 Classes

2.2.1 Class **Adresse**

Summary description for Adresse.

Declaration

public class Adresse
: Object

Properties

- *By*

  `public` **string By** { `get; set;` }

  Stter/henter navnet p den by der knytter sig til postnummeret, der er en fortegnelse over alle byer og postnumre i databasen.

- *Gade*

  `public` **string Gade** { `get; set;` }

  Stter/henter den gade der knytter sig til adressen, det hele skal med ex. 'Orla Lehmanns All 7, 5. -16'

- *Navn*

  `public` **string Navn** { `get; set;` }

  Stter/henter navnet der knytter sig til adressen, kan bde vre person eller firmanavn

- *Postnummer*

  `public` **int Postnummer** { `get; set;` }

  Stter/henter postnummeret p adressen

Constructors

- *.ctor*

  `public` **Adresse**`( )`

  Opretter en ny adresse med de givne oplysninger

  - **Parameters**
    - `navn` -
    - `gade` -
    - `postnummer` -
    - `by` -

- *.ctor*

  `public` **Adresse**`( )`

  Opretter en ny adresse, adresser oprettes i en tabel i databasen for at samle oplysningerne

Extended Information

- Assembly: `AcontoObjects`

2.2.2 Class **Bilag**

Summary description for Bilag.

Declaration

public class Bilag **:** Object

Properties

- *Bilagnummer*

  `public` **int Bilagnummer** { `get; set;` }
  Stter/henter nummeret p bilaget.

- *Dato*

  `public` **System.DateTime Dato** { `get; set;` }
  Stter/henter datoen for bilagets oprettelse.

- *Kommentar*

  `public` **string Kommentar** { `get; set;` }
  Stter/henter en evt kommentar til bilaget.

- *OriginalBilag*

  `public` **System.Drawing.Bitmap OriginalBilag** { `get; set;` }
  Stter/henter det indskannede original bilag (billede) der er knyttet til bilaget.

Constructors

- *.ctor*

  `public` **Bilag**`( )`
  Opretter et nyt bilag

Extended Information

- Assembly: `AcontoObjects`

2.2.3 Class **Faktura**

En faktura oprettes nr en ordre effektueres, alle data findes i ordren.

Declaration

public class Faktura : Object

Constructors

- *.ctor*

  `public` **Faktura**`( )`

  Initializes a new instance of the class.

- *.ctor*

  `public` **Faktura**`( )`

  - **Parameters**
    - `OrdreEnheder` -

Methods

- *lavHTML*

  `public` **string lavHTML**`( )`

  Returnerer HTML udgaven af fakturaen, denne indsttes s i browserkomponenten i FakturaView. Det er alts denne der skal ndres hvis man nsker at ndre fakturaens udseende. Skal kompineres med XSLT for at man kan ndre layoutet eksternt.

- *lavOIOXML*

  `public` **string lavOIOXML**`( )`

  Returnerer en OIOXML udgave af fakturaen er ikke implementeret og laver en exception hvis den kaldes

Extended Information

- Assembly: `AcontoObjects`

2.2.4 Class **KasseKladde**

Opbjekt til at vedligholde kassekladden er en tom skal fornuvrende.

Declaration

```
public class KasseKladde
  : Object
```

Constructors

- *.ctor*

  `public` **KasseKladde**`( )`

  Initializes a new instance of the class.

Extended Information

- Assembly: `AcontoObjects`

2.2.5 Class **Konto**

Summary description for Konto.

Declaration

public class Konto : Object

Constructors

- *.ctor*

  `public` **Konto**`( )`

  Initializes a new instance of the class.

Extended Information

- Assembly: `AcontoObjects`

2.2.6 Class **KontoPlan**

Summary description for KontoPlan.

Declaration

```
public class KontoPlan
  : Object
```

Constructors

- *.ctor*

  `public` **KontoPlan**`( )`

  Initializes a new instance of the class.

Extended Information

- Assembly: `AcontoObjects`

2.2.7 Class **KreditNota**

Summary description for KreditNota.

Declaration

```
public class KreditNota
  : Object
```

Constructors

- *.ctor*

  `public` **KreditNota**`( )`

  Initializes a new instance of the class.

Extended Information

- Assembly: `AcontoObjects`

2.2.8 Class **Kunde**

Summary description for Kunde.

Declaration

public class Kunde : Object

Properties

- *Adresse*

  `public` **AcontoObjects.Adresse Adresse** { `get; set;` }

  Stter/henter adresen p kunden.

- *KundeID*

  `public` **int KundeID** { `get; set;` }

  Stter/henter kundens ID dette genereres af SOAP komponenten fra databasen.

Constructors

- *.ctor*

  `public` **Kunde**`( )`

  Opretter en ny kunde.

  - **Parameters**
    - `kundeid` -
    - `navn` -
    - `adresse` -
    - `postnummer` -
    - `by` -

- *.ctor*

  `public` **Kunde**`( )`

  Opretter en ny kunde.

Extended Information

- Assembly: `AcontoObjects`

2.2.9 Class **Lager**

Lageret inedholder en liste med varer og tilhrende mngder. P sigt tilfjes der funktioner til at holde styr p lagerets flow(er ikke lavet).

Declaration

public class Lager
: Object

Constructors

- *.ctor*

  `public` **Lager**`( )`

  Opretter et objekt der indeholder alle lagerets varer.

Extended Information

- Assembly: `AcontoObjects`

2.2.10 Class **Leverandr**

Leverandr

Declaration

public class Leverandr
: Object

Properties

- *Adresse*

  `public` **AcontoObjects.Adresse Adresse** { `get; set;` }
  Stter/henter adressen p leverandren.

- *LeverandrID*

  `public` **int LeverandrID** { `get; set;` }
  Stter/henter leverandrens id.

- *Telefon*

  `public` **string Telefon** { `get; set;` }
  Stter/henter telefonnummeret p leverandren.

Constructors

- *.ctor*

  `public` **Leverandr**`( )`
  Initializes a new instance of the class.

Extended Information

- Assembly: `AcontoObjects`

2.2.11 Class **Medarbejder**

Summary description for Medarbejder.

Declaration

public class Medarbejder **:** Object

Properties

- *AdgangsNiveau*

  `public` **string AdgangsNiveau** { `get; set;` }
  Stter/henter medarbejderens AdgangsNiveau.

- *Adresse*

  `public` **AcontoObjects.Adresse Adresse** { `get; set;` }
  Stter/henter medarbejderens Adresse.

- *Brugernavn*

  `public` **string Brugernavn** { `get; set;` }
  Stter/henter medarbejderens Brugernavn.

- *Navn*

  `public` **string Navn** { `get; set;` }
  Stter/henter medarbejderens Navn.

- *Password*

  `public` **string Password** { `get; set;` }
  Stter/henter medarbejderens Password.

Constructors

- *.ctor*

  `public` **Medarbejder**`( )`

  Initializes a new instance of the class.

Extended Information

- Assembly: `AcontoObjects`

2.2.12 Class **Ordre**

Denne klasse reprsenterer en ordre

Declaration

public class Ordre : Object

Properties

- *Dato*

  `public` **System.DateTime Dato** { `get; set;` }
  Stter/henter datoen for ordrens oprettelse.

- *Faktura*

  `public` **AcontoObjects.Faktura Faktura** { `get; set;` }
  Stter/henter den faktura der er knyttet til ordren.

- *Kreditnota*

  `public` **AcontoObjects.KreditNota Kreditnota** { `get; set;` }
  Stter/henter den kreditnota der er knyttet til ordren.

- *Kunde*

  `public` **AcontoObjects.Kunde Kunde** { `get; set;` }
  Stter/henter den kunde ordren er oprettet af.

- *OrdreID*

  `public` **int OrdreID** { `get; set;` }
  Stter/henter Ordrenummeret p ordren.

Constructors

- *.ctor*

  `public` **Ordre**`( )`

  Opretter en ny ordre.

Methods

- *beregnFragt*

  `public double` **beregnFragt**`( )`

  Beregner de samlede fragt omkostninger, disse data findes ikke endnu i systemet og skal tilfjes.

- *beregnMoms*

  `public double` **beregnMoms**`( )`

  Beregner momsen for en ordre. Satsen hentes i databasen.

- *beregnSamletPris*

  `public double` **beregnSamletPris**`( )`

  Beregner en total saldo for ordren inklusiv moms og fragt.

Extended Information

- Assembly: `AcontoObjects`

2.2.13 Class **OrdreEnhed**

Declaration

public class OrdreEnhed
: Object

Constructors

- *.ctor*

  `public` **OrdreEnhed**`( )`

  - **Parameters**
    - `vare` -
    - `antal` -
    - `rabat` -

Extended Information

- Assembly: `AcontoObjects`

2.2.14 Class **Vare**

Reprsenterer en vare benyttes.

Declaration

public class Vare : Object

Properties

- *Beskrivelse*

  `public` **string Beskrivelse** { `get; set;` }

  Stter/henter varens detaljerede beskrivelse, den skal vre formateret med HTML tags da denne beskrivelse bliver indsat p hjemmesiden.

- *Indkbspris*

  `public` **double Indkbspris** { `get; set;` }

  Stter/henter indkbsprisen p en vare.

- *Kortbeskrivelse*

  `public` **string Kortbeskrivelse** { `get; set;` }

  Stter/henter varens korte beskrivelse, den der vises p hjemme siden og i vare oversigten

- *LeverandrID*

  `public` **int LeverandrID** { `get; set;` }

  Stter/henter identifikationen p leverandren af den pgldende vare.

- *Navn*

  `public` **string Navn** { `get; set;` }

  Stter/henter Navnet p en vare.

- *Udsalgspris*

  `public` **double Udsalgspris** { `get; set;` }
  Stter/henter den pris p varen som kunden ser.

- *Varenummer*

  `public` **int Varenummer** { `get; set;` }
  Stter/henter varens varenummer.

Constructors

- *.ctor*

  `public` **Vare**`( )`
  Opretter en ny vare.

Extended Information

- Assembly: `AcontoObjects`

Chapter 3

Namespace AcontoUI

Namespace Contents *Page*

Interfaces

Classes

3.1 Interfaces

3.2 Classes

3.2.1 Class **DataGridEx**

Summary description for DataGridEx.

Declaration

```
public class DataGridEx
  : DataGrid
```

Constructors

- *.ctor*

  `public` **DataGridEx**`( )`

  Initializes a new instance of the class.

Methods

- *Dispose*

  `protected` **void Dispose**`( )`

  Clean up any resources being used.

  – **Parameters**

  * `disposing` -

- *OnMouseDown*

  `protected` **void OnMouseDown**`( )`

  – **Parameters**

  * `e` -

Extended Information

- Assembly: `AcontoUI`

3.2.2 Class **FakturaView**

Summary description for FakturaView.

Declaration

```
public class FakturaView
  : UserControl
```

Constructors

- *.ctor*

  `public` **FakturaView**`( )`

  Initializes a new instance of the class.

Methods

- *Dispose*

  `protected` **void Dispose**`( )`

  Clean up any resources being used.

  - **Parameters**
    - `disposing` -

Extended Information

- Assembly: `AcontoUI`

3.2.3 Class **IndstillingerView**

Summary description for IndstillingerView.

Declaration

```
public class IndstillingerView
  : UserControl
```

Constructors

- *.ctor*

  `public` **IndstillingerView**`( )`

  Initializes a new instance of the class.

Methods

- *Dispose*

  `protected` **void Dispose**`( )`

  Clean up any resources being used.

  - **Parameters**
    - `disposing` -

Extended Information

- Assembly: `AcontoUI`

3.2.4 Class **KassekladdeView**

Summary description for KassekladdeView.

Declaration

```
public class KassekladdeView
  : UserControl
```

Constructors

- *.ctor*

  `public` **KassekladdeView**`( )`

  Initializes a new instance of the class.

Methods

- *Dispose*

  `protected` **void Dispose**`( )`

  Clean up any resources being used.

  - **Parameters**
    - `disposing` -

Extended Information

- Assembly: `AcontoUI`

3.2.5 Class **KontoplanView**

Summary description for KontoplanView.

Declaration

```
public class KontoplanView
  : UserControl
```

Constructors

- *.ctor*

  `public` **KontoplanView**`( )`

  Initializes a new instance of the class.

Methods

- *Dispose*

  `protected` **void Dispose**`( )`

  Clean up any resources being used.

  - **Parameters**
    - `disposing` -

Extended Information

- Assembly: `AcontoUI`

3.2.6 Class **LoginForm**

log ind

Declaration

public class LoginForm : Form

Constructors

- *.ctor*

  `public` **LoginForm**`( )`
  Operttet

Methods

- *Dispose*

  `protected` **void Dispose**`( )`
  Clean up any resources being used.
  - **Parameters**
    * `disposing` -

- *OnActivated*

  `protected` **void OnActivated**`( )`
  Kaldes nr formen aktiveres
  - **Parameters**
    * `e` -

- *OnDeactivate*

  `protected` **void OnDeactivate**`( )`
  Summary description for Form2.
  - **Parameters**
    * `e` -

Extended Information

- Assembly: `AcontoUI`

3.2.7 Class **MedarbejderView**

Summary description for MedarbejderView.

Declaration

```
public class MedarbejderView
  : UserControl
```

Constructors

- *.ctor*

  `public` **MedarbejderView**`( )`

  Initializes a new instance of the class.

Methods

- *Dispose*

  `protected` **void Dispose**`( )`

  Clean up any resources being used.

  - **Parameters**
    - `disposing` -

Extended Information

- Assembly: `AcontoUI`

3.2.8 Class **Mediator**

Mediatorklassen bruges til: 1) at kommunikerer mellem programmet og det logiske lag 2) holde styr p status for programmet, bl.a login informationer

Declaration

public class Mediator **:** Object

Fields

- *logik*

  `public` **AcontoUI.SOAP.BusinessLogic logik**

Constructors

- *.ctor*

  `protected` **Mediator**`( )`

  Initializes a new instance of the class.

Methods

- *createPassword*

  `public` **string createPassword**`( )`

  - **Parameters**
    - `brugernavn` -
    - `password` -

- *Instance*

  `public` **AcontoUI.Mediator Instance**`( )`

Extended Information

- Assembly: `AcontoUI`

3.2.9 Class **MomsBrowserView**

Summary description for MomsBrowserView.

Declaration

```
public class MomsBrowserView
  : UserControl
```

Constructors

- *.ctor*

  `public` **MomsBrowserView**`( )`

  Initializes a new instance of the class.

Methods

- *Dispose*

  `protected` **void Dispose**`( )`

  Clean up any resources being used.

  – **Parameters**

    * `disposing` -

Extended Information

- Assembly: `AcontoUI`

3.2.10 Class **OpretMedarbejderForm**

Summary description for OpretMedarbejderForm.

Declaration

```
public class OpretMedarbejderForm
  : Form
```

Constructors

- *.ctor*

  `public` **OpretMedarbejderForm**`( )`

  Initializes a new instance of the class.

Methods

- *Dispose*

  `protected` **void Dispose**`( )`

  Clean up any resources being used.

  – **Parameters**

    * `disposing` -

Extended Information

- Assembly: `AcontoUI`

3.2.11 Class **OversigtView**

Summary description for Oversigt.

Declaration

public class OversigtView : UserControl

Constructors

- *.ctor*

  `public` **OversigtView**`( )`

  Initializes a new instance of the class.

Methods

- *Dispose*

  `protected` **void Dispose**`( )`

  Clean up any resources being used.

  – **Parameters**

    ∗ `disposing` -

Extended Information

- Assembly: `AcontoUI`

3.2.12 Class **RainDrop**

Dette er en rent grafisk kontrol der bruges til at lave et animeret logo, den bruges p loginformen i stedet for at lave en decideret splashscreen

Declaration

public class RainDrop : UserControl

Constructors

- *.ctor*

  `public` **RainDrop**`( )`

  Opretter animationskontrollen

Methods

- *Dispose*

  `protected` **void Dispose**`( )`

  Clean up any resources being used.

  - **Parameters**
    - `disposing` -

- *myinit*

  `public` **void myinit**`( )`

  Kaldes efter komponenten er oprettet for at stte navnet p den billede fil der indeholder logoet

  - **Parameters**
    - `imagename` - filnavn p det billede der bruges til animationen

- *OnPaint*

  `protected` **void OnPaint**`( )`

Kaldes nr kontrollen skal gentegnes, vi overstyrer den indbyggede tegne funktion

– **Parameters**

* `e` -

- *Start*

  `public` **void Start**`( )`

  Kaldes nr man snker animationen skal starte

- *Stop*

  `public` **void Stop**`( )`

  Kaldes nr man nsker at animationen skal stoppe/pause

Extended Information

- Assembly: `AcontoUI`

3.2.13 Class **RedigerBilagForm**

Summary description for RedigerBilagForm.

Declaration

```
public class RedigerBilagForm
  : UserControl
```

Constructors

- *.ctor*

  `public` **RedigerBilagForm**`( )`

  Initializes a new instance of the class.

Methods

- *Dispose*

  `protected` **void Dispose**`( )`

  Clean up any resources being used.

  – **Parameters**

    * `disposing` -

Extended Information

- Assembly: `AcontoUI`

Chapter 4

Namespace AcontoUI.SOAP

Namespace Contents *Page*

Interfaces

Classes

4.1 Interfaces

4.2 Classes

4.2.1 Class **Adresse**

Declaration

public class Adresse **:** Object

Fields

- *By*

  `public` **string By**

  - **Usage**
    *

- *Gade*

  `public` **string Gade**

  - **Usage**
    *

- *Navn*

  `public` **string Navn**

  - **Usage**
    *

- *Postnummer*

  `public` **int Postnummer**

  - **Usage**
    *

Constructors

- *.ctor*

  `public` **Adresse**`( )`

  Initializes a new instance of the class.

Extended Information

- Assembly: `AcontoUI`

4.2.2 Class **BusinessLogic**

Declaration

```
public class BusinessLogic
  : SoapHttpClientProtocol
```

Constructors

- *.ctor*

  `public` **BusinessLogic**`( )`

  Initializes a new instance of the class.

  - **Usage**
    *

Methods

- *BeginHentKontoplan*

  `public` **System.IAsyncResult BeginHentKontoplan**`( )`

  - **Usage**
    *
  - **Parameters**
    * `callback` -
    * `asyncState` -

- *BeginHentMedarbejdere*

  `public` **System.IAsyncResult BeginHentMedarbejdere**`( )`

  - **Usage**
    *
  - **Parameters**
    * `callback` -
    * `asyncState` -

- *BeginNoop*

  `public` **System.IAsyncResult BeginNoop**`( )`

  - **Usage**
    *
  - **Parameters**
    * `callback` -
    * `asyncState` -

- *BeginOpretBruger*

  `public` **System.IAsyncResult BeginOpretBruger**`( )`

  - **Usage**
    *
  - **Parameters**
    * `callback` -
    * `asyncState` -

- *BeginOpretKunde*

  `public` **System.IAsyncResult BeginOpretKunde**`( )`

  - **Usage**
    *
  - **Parameters**
    * `kunde` -
    * `callback` -
    * `asyncState` -

- *BeginOpretVare*

  `public` **System.IAsyncResult BeginOpretVare**`( )`

  - **Usage**
    *
  - **Parameters**
    * `vare` -
    * `callback` -
    * `asyncState` -

- *BeginValiderBruger*

  `public` **System.IAsyncResult BeginValiderBruger**`( )`

  - **Usage**
    *
  - **Parameters**
    * `brugernavn` -
    * `password` -
    * `callback` -
    * `asyncState` -

- *EndHentKontoplan*

  `public` **System.Data.DataSet EndHentKontoplan**`( )`

  - **Usage**
    *
  - **Parameters**
    * `asyncResult` -

- *EndHentMedarbejdere*

  `public` **System.Data.DataSet EndHentMedarbejdere**`( )`

  - **Usage**
    *
  - **Parameters**
    * `asyncResult` -

- *EndNoop*

  `public` **bool EndNoop**`( )`

  - **Usage**
    *
  - **Parameters**
    * `asyncResult` -

- *EndOpretBruger*

  `public` **bool EndOpretBruger**`( )`

  - **Usage**
    *
  - **Parameters**
    * `asyncResult` -

- *EndOpretKunde*

  `public` **bool EndOpretKunde**`( )`

  - **Usage**
    *
  - **Parameters**
    * `asyncResult` -

- *EndOpretVare*

  `public` **bool EndOpretVare**`( )`

  - **Usage**
    *
  - **Parameters**
    * `asyncResult` -

- *EndValiderBruger*

  `public` **string EndValiderBruger**`( )`

  - **Usage**
    *
  - **Parameters**
    * `asyncResult` -

- *HentKontoplan*

  `public` **System.Data.DataSet HentKontoplan**`( )`

  - **Usage**
    *

- *HentMedarbejdere*

  `public` **System.Data.DataSet HentMedarbejdere**`( )`

  - **Usage**
    *

- *Noop*

  `public` **bool Noop**`( )`

  - **Usage**
    *

- *OpretBruger*

  `public` **bool OpretBruger**`( )`

  - **Usage**
    *

- *OpretKunde*

  `public` **bool OpretKunde**`( )`

  - **Usage**
    *
  - **Parameters**
    * `kunde` -

- *OpretVare*

  `public` **bool OpretVare**`( )`

  - **Usage**
    *
  - **Parameters**
    * `vare` -

- *ValiderBruger*

  `public` **string ValiderBruger**`( )`

  - **Usage**
    *
  - **Parameters**
    * `brugernavn` -
    * `password` -

Extended Information

- Assembly: `AcontoUI`

4.2.3 Class **Kunde**

Declaration

public class Kunde **:** Object

Fields

- *Adresse*

  `public` **AcontoUI.SOAP.Adresse Adresse**

  - **Usage**
    *

- *KundeID*

  `public` **int KundeID**

  - **Usage**
    *

Constructors

- *.ctor*

  `public` **Kunde**`( )`
  Initializes a new instance of the class.

Extended Information

- Assembly: `AcontoUI`

4.2.4 Class **Vare**

Declaration

public class Vare
: Object

Fields

- *Beskrivelse*

  `public` **string Beskrivelse**

  - **Usage**
    *

- *Indkbspris*

  `public` **double Indkbspris**

  - **Usage**
    *

- *Kortbeskrivelse*

  `public` **string Kortbeskrivelse**

  - **Usage**
    *

- *LeverandrID*

  `public` **int LeverandrID**

  - **Usage**
    *

- *Navn*

  `public` **string Navn**

  - **Usage**
    *

- *Udsalgspris*

  `public` **double Udsalgspris**

  - **Usage**
    *

- *Varenummer*

  `public` **int Varenummer**

  - **Usage**
    *

Constructors

- *.ctor*

  `public` **Vare**`( )`
  Initializes a new instance of the class.

Extended Information

- Assembly: `AcontoUI`

Chapter 5

Namespace SOAPWebservice

Namespace Contents *Page*

Interfaces

Classes

5.1 Interfaces

5.2 Classes

5.2.1 Class **BusinessLogic**

Dette er hele forretningslaget, det kunne splittes op i mere specialiserede services.

Declaration

public class BusinessLogic **:** WebService

Constructors

- *.ctor*

  `public` **BusinessLogic**`( )`

  Opretter SOAP komponenten/webservicen

Methods

- *Dispose*

  `protected` **void Dispose**`( )`

  Clean up any resources being used.

  – **Parameters**

  ∗ `disposing` -

- *HentBilag*

  `public` **System.Data.DataSet HentBilag**`( )`

- *HentIndstillinger*

  `public` **System.Data.DataSet HentIndstillinger**`( )`

- *HentKontoplan*

  `public` **System.Data.DataSet HentKontoplan**`( )`

  Henter hele kontoplanen fra databasen

- *HentMedarbejdere*

  `public` **System.Data.DataSet HentMedarbejdere**`( )`

- *Log*

  `public` **void Log**`( )`

  Intern metode i webservicen der bruges til at fre logfil i databasen sledes man kan se alle transaktioner

  – **Parameters**
    * `brugernavn` - Hvem opretter denne logbesked
    * `besked` - Selve beskeden

- *Noop*

  `public` **bool Noop**`( )`

  Denne metode er et no operations kald til webservices, men hvis den kaldes nr applikationen startes startes database forbindelsen automatisk op og webservicen bliver initialiseret herefter krer servicen markant hurtigere

- *OpretBilag*

  `public` **int OpretBilag**`( )`

  – **Parameters**
    * `bilag` -

- *OpretKonto*

  `public` **bool OpretKonto**`( )`

  – **Parameters**
    * `konto` -

- *OpretKunde*

  `public` **bool OpretKunde**`( )`

  Opretter en ny kunde i databasen

  - **Parameters**
    - `kunde` - Et Kunde object klassen findes i AcontoObjects

- *OpretMedarbejder*

  `public` **bool OpretMedarbejder**`( )`

  - **Parameters**
    - `medarbejder` -

- *OpretOrdre*

  `public` **bool OpretOrdre**`( )`

  - **Parameters**
    - `ordre` -

- *OpretVare*

  `public` **bool OpretVare**`( )`

  Opretter en ny vare p lageret

  - **Parameters**
    - `vare` - Et Vare objekt klassen findes i AcontoObjects

- *RetIndstilling*

  `public` **bool RetIndstilling**`( )`

  - **Parameters**
    - `key` -
    - `value` -

- *SletKonto*

  `public` **bool SletKonto**`( )`

- *ValiderBruger*

  `public` **string ValiderBruger**`( )`

  Undersger om brugeren er oprettet i systemet og har adgang til systemet

  – **Parameters**

  * `brugernavn` - Brugernavnet
  * `password` - Det krypterede password (se rapport "Implementering")

Extended Information

- Assembly: `SOAPWebservice`

5.2.2 Class **Global**

Summary description for Global.

Declaration

public class Global
: HttpApplication

Constructors

- *.ctor*

  `public` **Global**`( )`

  Initializes a new instance of the class.

Methods

- *Application_AuthenticateRequest*

  `protected` **void** **Application_AuthenticateRequest**`( )`

  – **Parameters**
  * `sender` -
  * `e` -

- *Application_BeginRequest*

  `protected` **void** **Application_BeginRequest**`( )`

  – **Parameters**
  * `sender` -
  * `e` -

- *Application_End*

  `protected` **void** **Application_End**`( )`

  – **Parameters**

        * sender -
        * e -

- *Application_EndRequest*

  `protected` **void Application_EndRequest**`( )`

  - **Parameters**
    * sender -
    * e -

- *Application_Error*

  `protected` **void Application_Error**`( )`

  - **Parameters**
    * sender -
    * e -

- *Application_Start*

  `protected` **void Application_Start**`( )`

  - **Parameters**
    * sender -
    * e -

- *Session_End*

  `protected` **void Session_End**`( )`

  - **Parameters**
    * sender -
    * e -

- *Session_Start*

  `protected` **void Session_Start**`( )`

  - **Parameters**
    * sender -
    * e -

Extended Information

- Assembly: `SOAPWebservice`

BILAG II

Kildekode

Er lavet med lgrind og placeret i en bilagsrapport.