DANMARKS TEKNISKE UNIVERSITET
IMM

15-10-2006

ADMINISTRATION AF E-MAILS

MARTIN LAURITZEN

IMM-B.ENG.-2006-71

Anerkendelser

1 Anerkendelser

Denne opgave blev skrevet i efteråret 2006, og fungere som afgangsprojekt fra DTU, et projekt til opfyldelse af en ønsket funktionalitet på min arbejdsplads og samtidig fungerer projektet ligeledes, som en del af min egen lyst til at løse et aktuelt problem.

Jeg vil derfor gerne takke min arbejdsplads, Blue Ocean Ventures Company, for at have givet mig plads og tid til at lave denne opgave, og samtidig, at have givet mig den problemstilling, jeg har forsøgt at løse.

DTU skal have tak for det frie valg, og den støtte og åbenhed min vejleder har givet mig med mit projekt.

Ligeledes skal mine nærmeste have tak, for at holde ud, mens jeg sad foran skærmen, og hjælpe mig med at teste systemet, samt komme med råd til udformning af systemets brugerflade.

Så vil jeg også gerne takke alle de mennesker, der stiller deres viden til rådighed via debatfora, newsgroups og andre informationskilder på Internettet. Det er altid godt at få en udefrakommendes vurdering af en problemstilling, samt få råd fra dem, der har mere erfaring end en selv.

- Martin Lauritzen

Abstract

2 Abstract

The following report will describe the development of a computer system, and act as the final project for Martin Lauritzen at DTU.

The project's intention is to create a system, which is able to handle emails in an organisation, in a way, that email will be available in a more specified way, than directly through a standard email client.
The system must act as the first hub in the process of managing incoming emails, and as the last hub, in dealing with outgoing emails. It is thought as a system, that will be used by organisations, that wish to integrate emails into their existing systems, so that all communication with customers via an organisations email address, are connected to the specific customer in an Customer Relations Management System.

The system must be independent from existing systems, so that it is as flexible as possible, and can be used under different circumstances. In that way, the system can be used by multiple existing systems, with a minimum of customization.

It is imperative that the system being developed has the functionality, which enables other systems to connect to it, in such a way that existing systems can search for e-mails connected to users defined there. Fr this communication, an interface must be developed, to work as the link between the systems. The specific solution to this problem will be found, based on the analyses of a company's existing systems.

To be able to access e-mails on the organisation e-mail server, a module has to be build, which enables the system to download e-mails, from one or more servers.

Main functionalities for the system are:

- A communication layer, to access emails on email server via the SMTP/POP3 protocol.
- A logical layer, which enables the system to process, sort and store email.
- An interface layer, for communication with existing systems.

The system will be developed using .net technology, with c# as main programming language. Microsoft Visual Studio is to be used as development environment.

The project lasts 10 weeks, which must cover analyses, design, implementation and documentation of the project.

The project is started in cooperation with Blue Ocean Ventures Company.

Ellsinore, the 15th of October 2006.

Resumé

3 Resumé

Følgende rapport redegøre for udviklingen af et computersystem, og fungere som afgangs projekt for studerende Martin Lauritzen (s022344).

Projektet skal bruges til at håndterer e-mails i en organisation, således at disse bliver tilgængelige på en mere brugbar måde, end via en normal e-mail klient.
Systemet, der skal fungere som første led i behandlingen af indkommende- og sidste led i udsendelsen af udgående e-mails, tænkes knyttet til en organisations kundeservice og gøre kundekontakt via e-mail mere integreret i organisationens eksisterende systemer, således at al kontakt, så vidt muligt, kan knyttes til eksisterende kunder i kunde/ordre systemet.

Ideen med dette er, at det skal overskueliggøre og integrere kommunikation med kunder i sammenhænge, hvor kunder skriver til og får svar fra et firmas kundeservice adresse (som fx info@firma.dk).

Der ønskes et system, der er uafhængigt at eksisterende systemer, da det ønskes brugt i flere sammenhænge. Således skal systemet kunne bruges i flere del organisationer, der hver bruger egne systemer.

Systemet skal bygges op, så det giver eksisterende systemer mulighed for, at forbinde til det udviklede system, og der igennem hente e-mails, på en sådan måde, at de kan tilknyttes eksisterende brugere.
Til denne kommunikation, skal der udvikles et interface, der kan fungere som bindeled mellem systemerne. Den specifikke løsning til dette interface vil blive fundet, ud fra en analyse af firmaets arbejdsgange og eksisterende systemers funktionaliteter.

Ligeledes skal der udvikles et modul, til kommunikation med en eller flere e-mail servere, således at det udviklede system kan hente og gemme e-mails fra en given e-mail server.

Hovedfunktionaliteterne i systemet vi være følgende:

- Kommunikations lag, til kommunikation med emailserver over SMTP/POP3 protokollen.
- Et logisk lag til sortering og lagring af e-mails.
- Et interface lag, til kommunikation med eksisterede systemer.

Systemet vil blive udviklet i og til .NET (version 1.1 eller muligvis 2.0) frameworket, med C# som programmeringssprog. Der vil blive brugt Visual Studio 2003/2005 som udviklingsplatform.

Systemet skal udvikles over 10 uger, hvilket dækker over både analyse, design og dokumentation for projektet, samt den indledende analyse af firmaets arbejdsprocesser og systemer, der skal til for at fastslå den optimale udformning af det ønskede system.

Projektet bliver udviklet i samarbejde med Blue Ocean Ventures Company.

Helsingør, 15. oktober 2006.

Indholdsfortegnelse

Kapitel 4

Introduktion

4 Introduktion

4.1 Blue Ocean Ventures Company

4.1.1 Introduktion til firmaet

Blue Ocean Ventures Company (BOVC) er et venturefirma, der beskæftiger sig med flere forskellige forretningsområder, men med en bindende strategi omkring brug af IT-løsninger i firmaets produkter. Hoved-ideen i firmaet er at bruge IT-løsninger til at forenkle arbejdsgange og omkostningesoptimere et produkt, samt at formidle produktet til forbrugere på en billigere og mere effektiv måde.
Firmaet startede i '98, som et 3 mands firma, og er i dag vokset til at have over 100 medarbejdere. Især inden for de seneste to år, er udviklingen gået stærkt.
BOVC har flere underfirmaer, der hver arbejder med deres eget forretningsområde. De tre større er:

- HumanConsult: Portal og kursusaktivitet inden for HR[1] og ledelse.
- Green Credit: Forbrugslån.
- FirstPrize: Obligationsspil, hovedsageligt i Sverige.

4.1.2 Udvikling

Efterhånden som firmaet er vokset, er flere systemer blevet taget i brug, og kompromiser er blevet taget. Mere og mere komplekse forretningsgange og krav om større effektivitet har ligeledes ført til, at diverse IT systemer er blevet udviklet eller købt, til at varetage forskellige forretningsområder.
Der foregår løbende rationaliseringer og udvidelser af disse systemer, hvilket medfører større eller mindre udviklingsprojekter, hovedsageligt inden for CRM[2] systemer og portaler.

HumanConsult udviklede sig hurtigt til at have brug for et salgssystem, der kunne styre salg og gensalg af abonnementer og kurser til kunderne. Der eksisterede allerede et CRM system, til varetagelse af eksisterende kunder, men det blev besluttet at investerer i et nyt og færdigt system, Microsoft Super Office, i stedet for at udvide det eksisterende, på grund af manglende udviklings kapacitet.
For kort tid siden, blev Microsoft Super Office dog skrottet, da to systemer medførte redundant data og der igennem større problemer med usynkroniseret kunde data. Et projekt omhandlende udvidelse af CRM systemet blev fuldført af undertegnede, og derved blev to funktionaliteter samlet i samme system.

4.1.3 Mangler og behov

HumanConsults CRM system er bygget til at styre kundedata, samt salg og portal aktivitet på kunderne. Det indeholder moduler til at overvåge alle aktiviteter der er foretaget på en kunde, samt aktiviteter en kunde har foretaget på portalen. Dette er en vigtig faktor i HumanConsult's forretningspolitik, da man ud fra disse data kan tegne en profil af en kunde, eller en gruppe af kunder, og derved tilbyde dem de funktionaliteter de højst sandsynligt finder interessant.
Samtidig sørger et yderst automatiseret salgssystem, for at forretningen holder omkostningerne nede.

HumanConsult modtager ofte mange e-mails, og disse varetages i dag manuelt, gennem en standard mailklient. Forretningen har i dag en enkelt hovedadresse, info@humanconsult.dk, hvor alt e-mail trafik går igennem – dette, samt antallet af e-mails, gør at der i dag ikke er styr på e-mails sendt til og modtaget af kunder. Ofte er der tale om et længere forløb over en rum tid, hvor kunde og forretning sender flere e-mails frem og tilbage – hvilket ofte fører til unødigt tidsspilde og detektiv arbejde, for den enkelte medarbejder, når denne skal sætte sig ind i et eksisterende forløb.
Før skrotningen af Super Office, kunne man manuelt kategorisere e-mails i Super Office – hvilket var en acceptabel løsning. Efter at al funktionalitet er flyttet over i CRM systemer, er der dog ikke mulighed for denne manuelle løsning.

[1] Human Resources
[2] Customer Relations Management

Forretningen har behov for at få lavet et system, der automatisk kan hente e-mails ind i CRM systemet, således at disse indgår i den aktivitet der registreres på en kunde.

4.2 Projektet

4.2.1 Indledning

Projektet blev sat i værk for at optimere HumanConsult's forretningsgang og gøre det mere overskueligt at benytte e-mails som kommunikationsmiddel med den enkelte kunde.
Der har ikke tidligere været udviklet et lignende projekt i forretningen, men da der ønskes en megen specifik funktionalitet, der er direkte rettet mod eksisterende systemer, blev det besluttet at udvikle at system i forretningen, modsat at købe et eksisterende produkt.

4.2.2 Formål

Der ønskes udviklet et system til håndtering af e-mails, der skal fungere som første led i behandlingen af indkommende- og sidste led i behandlingen af udgående e-mails, der er knyttet til forretningens kundeservice.

Systemet skal gøre behandlingen af kundekontakt via e-mail mere integreret i forretningens eksisterende systemer, således at al kontakt, så vidt muligt, kan knyttes til eksisterende kunder i CRM systemet.
Ideen med dette er, at det skal overskueliggøre og integrere kommunikation med kunder i sammenhænge, hvor kunder skriver til og får svar fra et firmas kundeservice adresse (som fx info@humanconsult.dk).

Der forestilles et system, der henter e-mails fra en eller flere e-mail servere, gemmer disse på en hensigtsmæssig måde og stiller et interface til rådighed, hvorigennem andre systemer kan få adgang til disse e-mails. E-mails skal være kategoriseret og kunne sorteres og udvælges efter visse parametre, for at eksterne systemer, kan kunne importere disse korrekt i deres egne strukturer.

Systemet ønskes udviklet, så det kan fungere uafhængigt af eksisterende systemer. Således ønskes opnået et mere fleksibelt system, det kan genbruges i flere af BOVC's eksisterende systemer, eller af andre virksomheder med samme problemstillinger.

4.2.3 Omfang

Projektet består af to faser:

Første fase er en analyse af forretningens systemer, samt en analyse af hvorledes et nyt system kan bygges op, så det bedst muligt kan integreres med de eksisterende.
Hvorledes eksisterende systemer i dag fungerer, vil blive belyst således, at der opbygges et fundament, hvorpå en løsning kan findes. Samtidig vil det blive belyst, hvordan interaktion mellem e-mail server og projekt-systemet kan forgå. Disse analyser vil benytte sig af de teknologier, der egner sig bedst på området – som f.eks. Microsoft's .NET platform og POP3 protokollen. Disse valg vil blive omtalt i analyse afsnittet.

Den anden fase består i at udvikle et egentligt system, til at udføre det ønskede arbejde. Gennem denne udvikling vil det blive vist, at analysen fungerer i praksis, og det forventes at et demo system vil forelægge, således at egentlig funktionalitet vil kunne demonstreres.

Projekt planlægning vil blive brugt som et redskab til at optimere forløbet, men vil ikke indgå som et studie i sig selv.

EMS Systemet vil så vidt muligt ikke komme til, at berøre eksisterende systemers funktionalitet, men integrationen med disse vil dog kræve, at der laves mindre udvidelser af de eksisterende systemer.

4.2.4 Begrænsninger

Den mest signifikante begrænsning på projektet, er den tidsbegrænsning eksamensprojekt perioden giver, nemlig 10 uger, samt det faktum, at understående er eneste udvikler på projektet. Disse faktorer gør dog ikke at projektet ikke vil kunne udføres, men sætter en begrænsning på omfanget og den endeligt udviklede funktionalitet.
Ønsker til EMS Systemet, vil blive implementeret på nødvendigheds basis, således at system kritiske funktionaliteter vil blive vægtet højere, end funktionaliteter der udelukkende er behændige.

Samarbejdsvirksomheden har været venlig til at stille medarbejdere til rådighed, således at det har været muligt at få dannet sig et overblik over behov og arbejdsgange i virksomheden. Dog begrænses denne analyse også tidsmæssigt, da medarbejdere ikke ubetinget har kunnet finde nødvendig tid til disse analyser. Dette gør, at en vis portion antagelser, må tages fra undertegnedes side – dette er dog holdt på et minimalt niveau, for at gøre slutproduktet så anvendeligt så muligt for virksomheden.

Begrænsninger i forhold til implementeret funktionalitet vil blive yderligere diskuteret i kapitel 6.

4.2.5 Akronymer og forkortelser

4.2.5.1 Navne

DTU:	Danmarks Tekniske Universitet
BOVC:	Blue Ocean Ventures Company
MS:	Microsoft
KTH:	Klaus Thiesen, vedjleder
MAH:	Morten Askholm, kontaktperson BOVC.
ITS:	IT-Support (Afdeling under BOVC).
HR:	Human Resources
CRM:	Customer Relations Management
W3C:	World Wide Web Consortium

4.2.5.2 Teknologier

VS.Net:	Visual Studio .Net 2003
POP3:	Post Office Protocol version 3
SMTP:	Simple Mail Transfer Protocol
IMAP:	Internet Message Access Protocol
SOAP:	Simple Object Access Protocol (version 1.1)
RUP:	Rational Unified Process
UML:	Unified Modelling Language

4.2.5.3 Andet

GUI:	Graphical User Interface (Grafisk brugerflade)
EMS:	Email Management System
EMS System:	Systemet, der ønskes udviklet i dette projekt.

4.2.6 Interesser og slutbrugere

Ethvert projekt vil have et antal interesserede; mennesker eller andre systemet, der har en interesse i projektet. Hvad enten det er dem, der i sidste ende bliver slutbrugere, ønsker en optimering af en afdelings arbejdsgang, eller er et system, der skal interagere med det udviklede. Disse kaldes stakeholders.
For at kunne forstå behovene hos de enkelte stakeholders, vil disse i det følgende afsnit blive defineret og beskrevet.

4.2.6.1 Marketingsafdeling og kundeservice

Ovenstående afdelinger er slutbrugere af systemet og ønsker at opnå en lettere arbejdsgang i deres afdelinger. Samtidig vil projektet kunne forhindre fejl og mistet data, således at disse afdelinger vil kunne give kunder et bedre indtryk, samt spare tid i den enkelte behandling.
Disse afdelinger vil blive inddraget, når projekt-systemet skal integreres med eksisterende systemer, således at der opnås en løsning, som fungerer bedst for afdelingens medarbejdere. Deres ønsker har også i en vis grad indflydelse på design af selve projekt-systemet.
Interesse området for denne gruppe er såvel den samlede effektivisering som systemet giver, men også de enkelte funktionaliteter i systemet. Derfor inddrages medarbejdere fra afdelingerne i design processen som fokusgruppe, for at finde en implementering der passer bedst.

4.2.6.2 Direktion

Direktionen i firmaet ønsker effektivisering og god kundepleje, for at opnå højst mulig profit og samtidig kunne give kunder et godt indtryk af firmaet – hvilket igen vil give bedre omtale, flere kunder og større omsætning.
Direktionens interesser ligger primært i den samlede effektivisering som projektet giver, og ikke i enkelt funktionaliteter, og vil derfor ikke være involveret som fokusgruppe.

4.2.6.3 IT-Support afdelingen

ITS' interesse i projektet er at få formindsket support opgaver, der kommer som følge af forsvunden data, samt andre problemer der opstår ved dagens håndtering af kundeplejen via e-mail.
Et integreret system vil kunne forenkle nogle af marketing og kundeservices arbejdsgange, hvilket bør give færre muligheder for fejl, og derved opgaver for ITS.
ITS er også ansvarlig for opsætning af software på servere og klienter, og derved giver det dem en interesse i EMS Systemets anvendelse og tilgængelighed, samt 'ease-of-use'.

4.2.6.4 Udviklingsafdelingen

Udviklingsafdelingen, og derved undertegnede, står for udviklingen af projektet. Det er dennes opgave at forstå alle aspekter i projektet, samle en viden omkring behov og analyserer eksisterende systemer, og på denne baggrund skabe et system, der opfylder disse behov.
Under analysen og indsamling af behov fra fokusgrupper, vil tekniske overvejelser ikke optræde, og et rent funktionelt design vil forsøges fundet. Herefter, i designfasen, kan tekniske løsninger for udførelsen af projektet diskuteres.

4.2.7 Antagelser

Det antages at samarbejdsfirmaet, BOVC, har e-mail server og eksisterende CRM systemer i brug. Dette gør at EMS Systemet, skal tilpasses til eksisterende systemer, som til en vis grad ligger fast i deres funktionaliteter og datastrukturer.
Platformen til afvikling af dette projekt skal være MS Server 2000/2003, da dette er den i firmaet anvendte server platform. Dette gør at systemet skal udvikles til denne platform.
Projektet bliver til i samarbejde med BOVC, men er fuldt ud et uafhængigt projekt, hvor til der ikke stilles egentlige krav fra BOVC. Dette giver en frihed i projektet, der dog ikke vil blive udnyttet, da det er ønsket at systemet skal tages i brug af BOVC på et senere tidspunkt.

4.2.8 Finansielle forhold

Dette projekt bliver udviklet som et afsluttende projekt fra DTU, og finansielle forhold som aflønning og egentlig pris, er meget begrænsede. Dette projekt bliver udført som et uafhængigt projekt fra BOVC, og derfor ikke lønnet. Det betyder samtidig, at der ikke er et krav til BOVC om at være behjælpelige med

ansattes tid – men en rimelig forståelse er dog opnået, i det BOVC har en interesse i at projektet udføres, og derved kan anvendes i deres forretning.

4.2.9 Projekt krav

Det er et krav at projektet følger gængse standarder, således at dets funktionalitet kan benyttes af flere forskellige systemer, og at det følger samarbejdsfirmaets politikker omkring nyudviklet software. BOVC benytter MS .Net platform til alle nyudviklede projekter, hvilket gør at dette projekt, også vil blive implementeret under denne platform.
For at opnå størst mulig anvendelses grad, vil interfacet EMS Systemet, bruge en standard givet af W3C eller MS. Det er endvidere et krav, at systemet skal kunne bruges af flere forskellige systemer, således at det ikke låses fast på et enkelt CRM system.

Kapitel 5

Planlægning

5 Planlægning

5.1 Proces beskrivelse

Dette projekt består af forskellige elementer. Umiddelbart er dette projekt et rent software projekt, med udvikling i hovedsædet. Men for at komme fra idé til design, vil det være nødvendigt at gå gennem flere faser først. Derfor vil der i det følgende afsnit, blive foretaget en analyse af projektets planlægning.
For at opnå en bedst mulig planlægning, og for ikke at begå samme fejl som andre, bruges der her tanker og idéer af tidligere udviklere.

5.2 Rational Unified Process

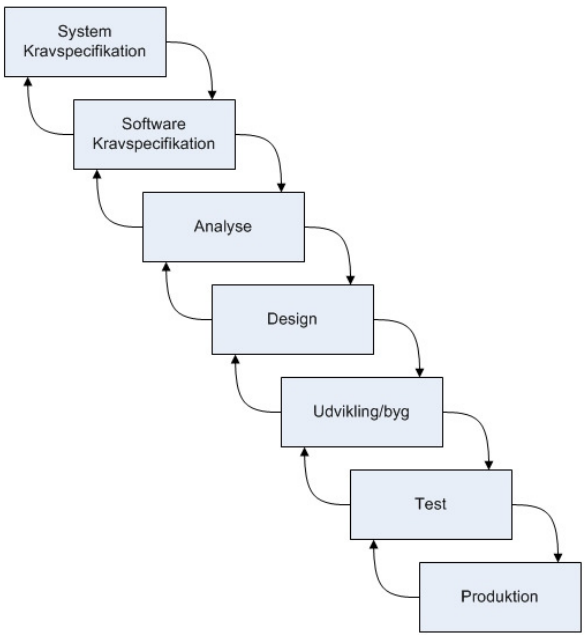
Rational Unified Process, RUP, er en metode til udvikling af software, der har været brugt i en del år. Der er flere bøger og publikationer beskrivende denne udviklingsmetode og det har samtidig været kursusmateriale på DTU i et eller flere kurser.
RUP beskriver metoder til planlægning af et udviklingsforløb, og kan bruges *as-is* eller med modifikationer. Der er meningen at RUP skal tilpasses et enkelt projekt, så der lægges vægt på de elementer, der er vigtige i det enkelte projekt, og ligegyldige elementer fjernes. Denne adaption, giver en god og fleksibel platform, som undertegnede, har haft gode erfaringer med.

5.2.1 Iterativ udvikling

Den aldrende vandfaldsmodel (figur 5.1) er stadig vidt brugt blandt udviklere. Denne beskriver en udviklingsmodel som en serie faser, der følger efter hinanden.

Disse faser er fastlagt, således at en ikke kan nås, før en anden er fuldt tilendebragt.
Dette giver et fastlåst forløb, der kan være meget ufleksibelt i udviklings øjemed.

Således gør denne model, at kun en fase, og derved kun en fases ressourcer, er i brug på en gang, hvilket gør at fejl begået i en tidligere fase, vil være svære eller umulige at rette, da denne fase er tilendebragt, og ressourcerne muligvis flyttet.
RUP der i mod, giver en iteration udviklingsmodel, der undgår disse fastlåste strukturer, og derved opnår en del fordele.

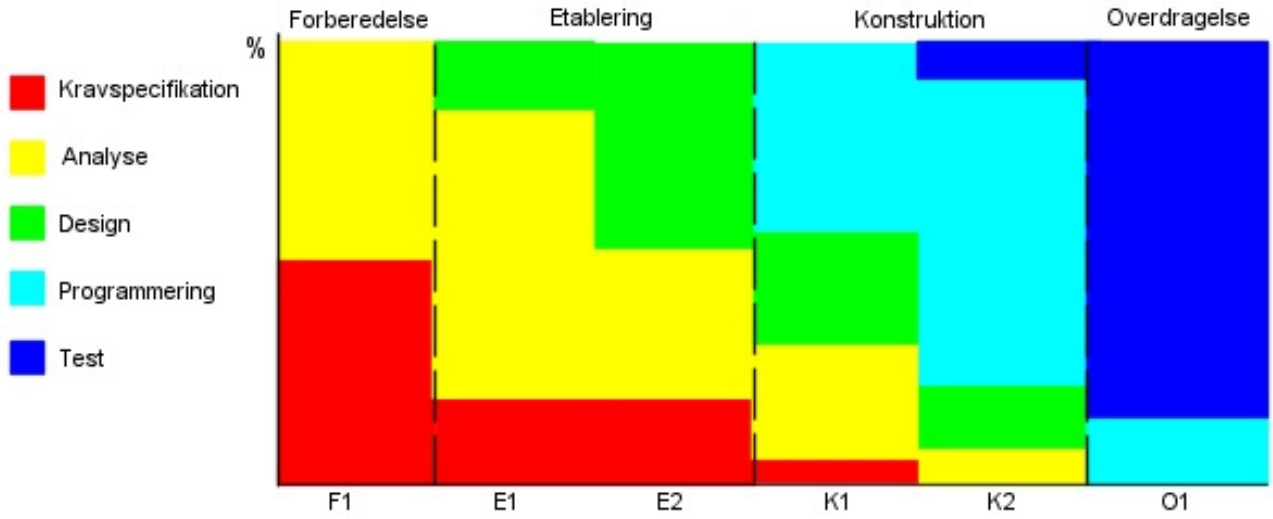


Figur 5.1: Vandfaldmodel

Under et projekt som modnes og udvikles under projektforløbet, kan det ikke undgås at der f.eks. opstår nye krav til projektet, og derved skal System Kravspecifikationen tages op til revurdering.
Dette gøres ved at dele projektet op i iterationer, der hver gennemløber de samme faser, men bygger på hvad den foregående iteration har opnået.
Således kan der nås en procesopdelt struktur, der giver mulighed for ændringer og opdateringer af tidlige faser, som projektet skrider frem.

RUP foreslår at der ved projektets begyndelse laves et estimat for hvor megen tid der bruges på enkelte faser i de enkelte iterationer. Et såkaldt *workload* diagram.

Faserne kan blive tildelt forskellige ressourcer afhængig af hvor kritiske de er for en iteration. Et eksempel kan ses på Figur 5.2 her under.



Figur 5.2

5.2.2 Software modeller

For at kunne få et overblik over den udviklede software, er den oftest nødvendigt at modellere softwaren visuelt. Til dette formål giver RUP et sæt værktøjer, under betegnelsen UML.
Dette 'sprog' giver muligheder for visualisering, specificering, konstruering og dokumentering af elementer i et software system. Det giver en standard for at lave *blueprints* af et system, og understøtter en lang række modeller, fra konceptionelle idéer, forretnings processor og egentlige klassediagrammer for forskellige programmeringssprog.
De forskellige modeller kan beskrive samme element i et software system, men fra forskellige vinkler (f.eks. struktur og forløb) – og der vil derfor være noget overlap af information.

Systemer er oftest bygget således at der er slutbrugere eller andre systemer, der skal interagere med systemet, og således er der behov for nogle specifikationer eller modeller for hvorledes denne interaktion skal foregå.
Dette kan beskrives i *Use-Cases* der er understøttet af RUP. I disse beskrives et handlingsforløb mellem en bruger og systemet, hvilke handlinger der forventes, men også handlinger der ikke umiddelbart forventes, som system fejl og brugerfejl.
I dette projekt, er der dog ikke mange scenarier hvor slutbrugere har den direkte kontakt med systemet, og *Use-Cases* vil derfor ikke fylde så meget i denne rapport.

5.3 Implementering

For at styre projektet, er det nødvendigt at opstille nogle mål og definere ønskede faser og iterationer. Her benyttes RUP's modeller, som beskrevet tidligere.

5.3.1 Milepæle

Milepæle beskriver tidspunkter i projektet, hvor visse elementer ønskes færdiggjort. Disse elementer kan stadig ændres efter tidspunktet som følge af den iterative proces, men hovedlinierne skal være trukket, når en milepæl accepteres som nået.
Her under er en liste over milepæle sat i dette projekt:

Dato	Aktivitet
07-08-06	Overordnet kravspecifikation færdiggjort
21-08-06	Hovedfunktionalitet implementeret og testet
01-09-06	Interface implementeret og testet
14-09-06	GUI implementeret og testet
21-09-06	E-mail kommunikations komponent implementeret og testet
21-09-06	Prototype færdigudviklet
30-09-06	Testscripts godkendes
15-10-06	Projekt afleveres

5.3.2 Iterationer

Projektet er blevet opdelt i følgende Iterationer.

1. Hoved funktionalitet (2 uger)
    a. Implementering af kommunikation med SMPT/POP3 server
        i. Sende e-mails
        ii. Modtage e-mails
        iii. Der bruges 3-parts komponent til kommunikation
    b. Opbevaring af data
    c. Sortering af data på passende måde

2. Interface (2 uger)
    a. Opsætning af interface
    b. Implementering af passende kald
    c. Søgning i opbevaret data
    d. Afsending af e-mails

3. GUI (1 uge)
    a. Opsætning af indstillinger
    b. Grafisk søgning i data via GUI

4. Udvikling af eget SMTP/POP3 komponent (optionel) (1 uge)
    a. Udvikling af egen komponent til kommunikation med SMTP/POP3 server

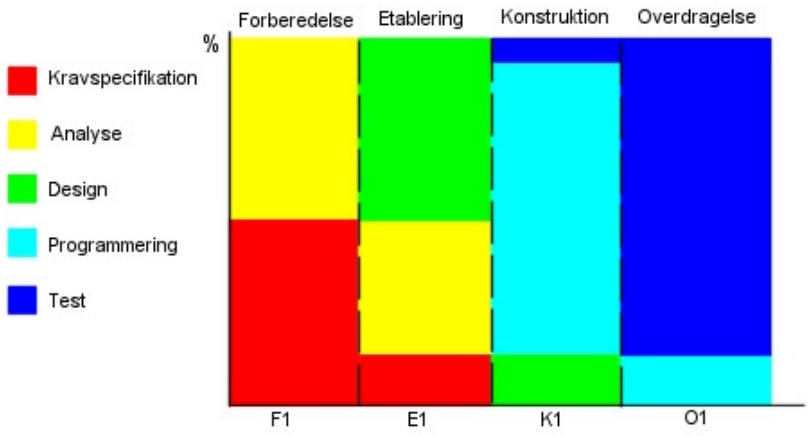
Her under ses en tidsplan, for de ovenstående iterationer, samt andre projekt relaterede emner.

Fra	Til	Aktivitet
01-08-06	07-08-06	Indledende analyse og udformning af overordnet kravspecifikation
08-08-06	21-08-06	Iteration 1
22-08-06	07-09-06	Iteration 2
08-09-06	14-09-06	Iteration 3
15-09-06	21-09-06	Iteration 4
22-09-06	25-09-06	Tests af samlet system
26-09-06	30-09-06	Fejlrettelser efter test
01-09-06	15-10-06	Færdiggørelse af rapport, samt finpudsning af projekt

Iteration 4 udføres kun i tilfælde af at tidsplanen holder, da denne ikke er strengt nødvendig for projektets funktionelle fuldførelse, hvilket vil blive diskuteret nærmere i et senere afsnit.

5.3.3 Arbejdsfordeling

Figur 5.4 beskriver opdelingen af arbejdsbyrden i dette projekt. Det skal dog noteres at der blev foretaget fire iterationer i hver af henholdsvis Etablering og Konstruktions faserne. Således var der fire næsten identiske etablerings og konstruktions iterationer, og derfor er disse ikke angivet specifikt.



Figur 5.3

Det skal ydermere bemærkes, at siden det blev valgt at bruge RUP modellen som udviklingsproces, kan man ikke, som det fremgår af figuren, umiddelbart antage, at den ene fase følger den anden. Som beskrevet tidligere, er RUP modellen ikke så fastlagt som vandfaldsmodellen, og en mere flydende overgang kan antages. Denne figur giver dog et estimat for tidsforbruget.
Bemærk i øvrigt, at 01.10.2006 er sat som skæringsdato for projektet, hvor der her forventes en gennemtestet funktionel prototype. Der er således afsat tid i projektets sidste del, til udelukkende at lave dokumentation.

Kapitel 6

Kravspecifikation

6 Kravspecifikation

6.1 Indledning

I følgende afsnit vil der blive beskrevet en række krav til projektet og dets udførelse. For at få en mest hensigtsmæssig opdeling af disse krav, vil de i dette projekt deles op i tre kategorier: funktionelle krav, ikke-funktionelle krav og krav til tests.
Funktionelle krav er præcise og utvetydige krav til EMS Systemets funktionalitet, mens ikke funktionelle krav der beskriver ideer bag projektet, som performance og brugevenlighed.
Test krav vil beskrive de krav der stilles til en succesfuld test af den prototype af systemet, der ønskes udviklet.

Det skal bemærkes, at det ikke er muligt at få implementeret alle krav i dette projekt, da tidsplanen ikke vil kunne bære dette. Der imod vil der blive lavet en afvejning af kravenes prioritet, hvor en mulig løsning for de ikke implementerede krav, vil blive diskuteret i afsnittet *udvidelser* senere i denne rapport.

6.2 Krav til læseren

Denne rapport beskriver processen, hvori dette projekt bliver til. Fra opsætning af krav, over design beslutninger og til implementering og tests. Det er antaget at læseren har basis kundskaber omkring udvikling af software – dog er ideerne omkring udviklingsstrategien kort beskrevet i afsnit 5.

Dette projekt vil benytte sig af Microsoft's .Net platform, samt programmeringssproget C#. Læseren vil få en introduktion til konceptet bag dette i afsnit 7, men det antages at læseren forstår hoved ideerne i moderne 3-generations programmeringssprog, samt kender til hovedtrækkene bag dataopbevaring i databaser.

Afsnittet omkring analyse, design og implementering vil være tekniske af natur, derfor vil det være påkrævet at læseren har et større indblik i UML og software design. Ønskes det ikke at forstå den tekniske side af dette projekt, kan disse afsnit springes over.

6.3 Funktionelle krav

Funktionelle krav beskriver funktionaliteter, som EMS Systemet skal kunne udføre. Dette kan være en bestemt opgave, en måde opgaven skal udføres på eller hvordan bestandig data gemmes i og af systemet.

For at kunne beslutte hvilke krav der skal have størst prioritet, og dermed gives størst opmærksomhed, gives hvert krav en *rating*.
Et hvert krav vil ydermere blive tildelt et unikt nummer, som der senere kan blive refereret til i denne rapport, samt i en eventuel versionsstyring. Se kapitel 6.3.8

Givet projektet tidsramme, har det været nødvendigt at koncentrere sig om essentielle funktionaliteter, hvormed systemet ikke ville kunne fungere uden. Det er ikke målet at bygge et fuldt færdigt system, både af før omtalte tidsmæssige hensyn, men også på grund af at det forventes, at der vil komme ønsker om ændringer og tilføjelser til systemet, når dette er sat i produktion.

6.3.1 Runtime Platform, §1.1

BOVC bruger som før omtalt Microsoft Windows Server på deres Servere, samt .Net platformen som foretrukne udviklingsplatform. Derfor vil det være et krav til dette projekt, at det bliver udviklet på denne platform. Dette giver dog også god mening, i det .Net er en moderne og lettilgængelig udviklingsplatform, der giver en nemmere og hurtigere udvikling, end tidligere platforme.

6.3.2 Datalager, §1.2

6.3.2.1 Persistent Datalager, §1.2.1

EMS Systemet skal kunne gemme data på en sikker måde, således at det vil være tilgængeligt til enhver tid, indtil denne data ønskes slettet. Dette betyder også at data skal være sikret ved systemnedbrud af enhver art, såsom kritiske systemfejl, server nedbrud osv.
Data opbevaringen skal understøtte basis funktionaliteterne *CRUD* (Create, read, update and delete), og derved give adgang til data gemt i det persistente data lager.

6.3.2.2 Relationel datalager, §1.2.2

Datalageret skal give mulighed for relationel data lagring, således at data gemt i forskellige dele i lageret kan relateres til hinanden, på en sådan måde at der ikke opstår redundant data.

6.3.2.3 Indekseret data, §1.2.3

Datalageret skal give mulighed for indekseret data lagring, således at data kan udtrækkes hurtigt og effektivt ved store datamængder.

6.3.2.4 Forespørgsler, §1.2.4

Datalageret skal give mulighed for at tilgå data via forespørgsler, kaldet *queries*, således at det er muligt at udtrække delmængder af opbevaret data, på tværs af opdelinger i datalageret.

6.3.2.5 Transaktionssupport, §1.2.5

For ikke at opnå inkonsistent data, i et system der vil blive tilgået af flere systemer samtidig, kræves det af datalageret, at det understøtter transaktioner, hvormed der kan behandles flere forespørgsler samtidig, uden inkonsistens opstår.

6.3.2.6 Support for forskellige datalager platforme, §1.2.6

EMS Systemet skal kunne fungere uafhængigt af datalager platformen. Dvs. at der skal supporteres flere forskellige datalager platforme i systemet. Dette krav stilles for at give et fleksibelt system, der ikke låses fast til et enkelt system setup.

6.3.2.7 Unik identifikation af poster §1.2.7

Enhver datamængde i datalageret skal kunne identificeres unikt, således at det til en hver tid er muligt at skelne mellem disse datamængder, selvom deres indhold er præcist ens. Dette krav er vigtigt for at kunne skelne data korrekt fra hinanden.

Ydermere stilles der følgende specificerede krav.

6.3.2.8 Datalager understøttelse, §1.2.8

Den primære databaseplatform hos BOVC er Microsoft SQL Server, og det vil derfor være et krav, at EMS Systemet kan operere med denne platform. Denne platform understøtter de foregående krav stillet under §1.2.

6.3.3 Interface, §1.3

6.3.3.1 Datastruktur, §1.3.1

EMS Systemet giver i dets natur, mulighed for at andre systemer kan koble op mod det. For at give det mest fleksible interface, er det et krav at interfacet bygger på gængse standarder, der er let tilgængelige og mulige at koble op i mod for ethvert andet system, uanset platform.
Ligeledes skal data kunne overføres via dette interface, på en sådan måde, at ethvert system vil kunne læse denne data, uden at implementere dele fra det udvikledes systems platform.

6.3.3.2 Funktionalitet, §1.3.2

Interfacet skal stille følgende funktionalitet til rådighed gennem dets interface.

1. Mulighed for udtræk af e-mail data, på baggrund af på forhånd specificerede parametre. Her under understøttelse for overførsel af vedhæftede filer.
2. Mulighed for at afsende e-mails via EMS Systemet, og en af de hertil knyttede e-mail servere.
3. Mulighed for at slette lagrede e-mails fra EMS Systemets data lager.

6.3.4 E-mail Servere, §1.4

Systemet skal bygges således at det skal lette håndteringen af e-mails i en organisation, og kunne fungere som bindeled mellem et eksisterende system og en e-mail server. Således vil der naturligvis være krav til hvilke funktionaliteter der ønskes, med hensyn til kommunikation osv. med e-mail servere. Disse er beskrevet her under.

6.3.4.1 Forbindelse til E-mail server, §1.4.1

Forbindelse til e-mail server, skal foregå over en standard protokol, således at enhver server der følger denne standard kan understøttes af EMS Systemet. Denne forbindelse skal understøtte password beskyttede e-mail servere.
Det skal være muligt at forbinde til en server over såvel intranet som Internet.

6.3.4.2 Kommunikation med E-mail server, §1.4.2

Følgende kommunikation skal kunne foretages af systemet, mod en specificeret e-mail server.

1. Skabe en password sikret forbindelse til serveren.
2. Hente e-mails.
3. Sende e-mails.
4. Slette e-mails på serveren.
5. Vedhæftede filer skal understøttes, såvel i modtagelse som afsending.

6.3.4.3 Understøttelse for opsætning af flere e-mail servere på en gang, §1.4.3

Systemet skal understøtte at forbinde til flere forskellige e-mail servere på en gang. Således at systemet kan fortage den i §1.4.2 definerede kommunikation med en server, alt efter systemet eget eller en brugers ønske.

6.3.4.4 Enkel opsætning af E-mail server, §1.4.4

Opsætning af e-mail server, skal foregå i en der til beregnet grafisk brugerflade. Disse data skal gemmes i en konfigurationsfil, således at data automatisk hentes ved system genstart. Brugerfladen skal understøtte opsætning af flere e-mail servere.

6.3.5 Håndtering af e-mails, §1.5

Der stilles følgende krav til hvordan systemet skal håndtere e-mails, samt hvorledes disse skal kunne hentes, sorteres og afsendes.

6.3.5.1 Automatisk download, §1.5.1

Systemet skal kunne opsættes således at det automatisk henter e-mails fra en eller flere servere i et givent interval. Således vil systemet holdes opdateret med nyeste e-mails fra serverne.
Intervallet skal kunne specificeres af brugeren på en hensigtsmæssig måde.

6.3.5.2 Opbevaring, §1.5.2

E-mails skal opbevares i et persistent data lager, således at disse kan fremhentes til enhver tid. Se også kravet omkring data lagring §1.2

Det skal være muligt at fremsøge en e-mail ud fra nogle bestemte parametre, således at en e-mail til enhver tid vil kunne skelnes og fremsøges, under forudsætning af at denne er identificerbar ud fra de givne parametre.

6.3.5.3 Afsendelse, §1.5.3

Det skal være muligt at afsende e-mails via systemet, således at brugeren kan benytte en af de opsatte e-mail servere til denne afsendelse. Det skal her under være muligt at afsende e-mails med vedhæftede filer.

6.3.6 Kategorisering/Søge Parametre, §1.6

For at opnå et system, hvormed det er muligt at kategorisere og søge efter e-mails, skal der opsættes en række parametre hvorpå der kategoriseres og hvorpå en bruger kan fremsøge en eller flere e-mails. Her under opstilles en række parametre, som det kræves at systemet kan søge på.

1. E-mail afsender.
2. E-mail modtager (for at kunne skelne mellem flere opsatte e-mail konti).
3. E-mail overskrift.
4. Afsender dato (denne skal kunne angives som et interval).

6.3.7 Grafisk brugerflade, §1.7

For at give den bedst mulige oplevelse for en bruger, der skal opsætte eller bruge systemet direkte, ønskes der udviklet en grafisk brugerflade, til at varetage konfigurationen af systemet, samt diverse andre opgaver.

6.3.7.1 E-mail klient, §1.7.1

Der skal opbygges en grafisk brugerflade, hvormed man kan søge i de gemte e-mails, samt læse indholdet af disse. Således vil systemet kunne fungere som en primitiv e-mail klient.

6.3.7.2 Konfiguration af systemet, §1.7.2

En grafisk brugerflade til ændring af diverse konfigurations data, skal indgå som en del af den samlede GUI. Denne skal indeholde funktionalitet til at understøtte krav §1.4.4 og §1.5.1

6.3.7.3 Automatisk download, §1.7.3

Den grafiske brugerflade skal give mulighed for at starte og stoppe automatisk download af e-mails fra de konfigurerede e-mail servere. Således at systemet stadig er tilgængeligt, selvom der ikke automatisk hentes fra serverne.

6.3.8 Prioritering af funktionelle krav

Her følger en liste over prioritering af de funktionelle krav, samt bemærkninger for denne prioritering. Kravene er vægtet på følgende måde.

1. Ultimativt krav
2. Kritisk krav
3. Vigtigt krav
4. Krav der kan undlades, men vil give en forringelse af projektet
5. Krav der kan undlades
6. *Nice to have*

Risiko analyse vil også fremgå, således at det bliver muligt at bedømme, hvor store risici der er ved et enkelt krav. Det kan ud fra risiko faktoren besluttes, om et krav er for risici fyldt til at implementerer, hvilket så vil skulle omdefinerer projektet. Samtidig giver risiko analysen en idé om hvor mange ressourcer der skal lægges i implementeringen af et krav, og det er derfor en vigtig del af planlægningen.

Vægt	Krav #	Krav	Risici	Bemærkning
1	§1.1	Runtime Platform	**Ingen** Da denne platform er en gennemtestet og kendt platform, antages det ikke som en risiko at bruge denne, frem for en anden.	Krav fra samarbejdsvirksomhed
1	§1.2.8	Understøttelse af datalager (MSSql)	**Høj** Kan denne platform ikke understøttes, vil systemet ikke kunne afvikles hos samarbejdsvirksomheden.	Krav fra samarbejdsvirksomhed
2	§1.3	Interface	**Høj** Interfacet er grundlæg-gende funktionalitet i systemet, der vil være ubrugeligt uden en mulighed for kommuni-kation med andre systemer.	
2	§1.4.1	Forbindelse til E-mailserver	**Medium** Kan der ikke opnås at oprette forbindelse til e-mail server, vil kritisk funktionalitet mangle. Risiko dog nedgraderet til medium, da der kan anvendes eksisterende komponent til forbindelse.	
2	§1.4.2	Kommunikation med E-mailserver	**Medium** Kan der ikke opnås korrekt kommunikation med e-mail server, vil kritisk funktionalitet mangle. Risiko dog nedgraderet til medium, da der kan anvendes eksisterende komponent til kommunikation.	
2	§1.6	Kategorisering/Søge Parametre	**Medium** Søge parametre gør at e-mails kan fremsøges på en måde, så projektet kan fungere som ønsket. Dog ikke en stor udviklingsmæssig opgave.	
3	§1.5.2	Opbevaring af E-mails	**Medium** Større udviklingsmæssig opgave, der rummer mange facetter, og derfor større chance for at tidlige designbeslutninger giver problemer senere i forløbet.	

3	§1.2.1	Persistant Datalager	**Lav** Kendt teknologi understøtter dette krav.	Som følge af prioritering af §1.5.2
3	§1.2.4	Forespørgsler	**Medium** Kendt teknologi understøtter dette krav, men forskellige datalager platforme, understøtter forskellige versioner af samme sprog til forespørgsler.	Som følge af prioritering af §1.2.1
4	§1.2.2	Relationel Datalager	**Lav** Kendt teknologi understøtter dette krav.	
4	§1.2.3	Indekseret Datalager	**Lav** Kendt teknologi understøtter dette krav.	
4	§1.2.5	Transaktionssupport	**Lav** Kendt teknologi understøtter dette krav. Minimum af implementering, der ikke er streng nødvendig.	
4	§1.2.7	Unik identifikation af data	**Lav** Kendt teknologi understøtter dette krav.	
4	§1.4.3	Understøttelse for flere E-mail servere	**Medium** Større implementering, der kræver flere udvidelser af systemet. Tidlige beslutninger kan give problemer senere i forløbet.	
4	§1.4.4	Enkel opsætning af E-mail servere	**Medium** Der skal udvikles en standard for konfiguration	
4	§1.5.1	Automatisk download af e-mails	**Medium** Større implementering, der kræver flere udvidelser af systemet. Tidlige beslutninger kan give problemer senere i forløbet.	
5	§1.5.3	Afsendelse af e-mail	**Lav** Kræver ingen ændringer i eksisterende funktionalitet.	
5	§1.7	GUI	**Medium** Ved implementering af GUI, bliver al funktionalitet *event-driven*[3], hvilket kan give indviklede program-strukturer, og kræver god planlægning af program eksekvering.	
6	§1.2.6	Support for flere datalager platforme	**Medium** Kræver at systemet udvides, så det er muligt at konfigurere det til at bruge en specifik platform, dette gør at systemet skal differentiere på baggrund af konfigurationen, og bruge forskellige datalag, der også skal implementeres.	

[3] Alle hændelser i program afviklingen, vil ske efter brugerens input.

6.4 Ikke funktionelle krav

Følgende ikke-funktionelle krav stilles til projektet. Bemærk at disse krav kan være sværere at påvise implementeret, og derfor vil disse ikke kunne evalueres og prioriteres på samme måde som funktionelle krav. Det er dog stadig vigtigt at opsætte disse krav, så ikke vigtige beslutninger går tabt.

6.4.1 Pålidelighed

For at opnå et praktisk anvendeligt system, er det nødvendigt at skabe et system der er pålideligt, i den forstand at det udføre de opgaver, det bliver stillet.
Derfor er det et krav til EMS Systemet, at det til enhver tid udfører de opgaver det bliver stillet, med den antagelse at eksterne afhængigheder, så som e-mail server osv., er funktionelle.
For at opnå denne pålidelighed, må systemet udvikles således at det testes for enhver tilføjet funktionalitet. Det vil sige at det er påkrævet at systemet testes mellem hver implementerings iteration, for at vise at nyligt implementerede funktionaliteter fungerer efter hensigten, samtidig med at eksisterende funktionalitet bibeholdes.

6.4.2 Kapacitet

Det er vigtigt at systemet udvikles så det har kapacitet til at varetage tusinder af e-mails. Den egentlige kapacitets byrde, ligger dog hovedsageligt hos runtime server og data lager server. Microsofts .Net platform og Microsofts SQL Server, er begge gennemtestede systemer, der skulle kunne klare store datamængder, og derfor vil denne opgave ikke fokusere på kapaciteten for disse.

6.4.3 Brugervenlighed

Da systemet skal kunne opsættes af en udenforstående, er det vigtigt at der opnås en hvis form for brugervenlighed. Systemet vil sagtens kunne fungere uden, men da vil det hurtigt blive fravalgt, alene fordi der ikke er nogen brugere, der forstår at sætte systemet op, eller bruge det.

Brugervenlighed er et abstrakt begreb, som kan være svært at måle. Derfor kan det også være svært at bestemme, om det er lykkedes at skabe et brugervenligt program. Især fordi bruger og udvikler, ser tingene meget forskelligt. Derfor er det et krav at der udformes nogle use-cases, hvori der beskrives ønsket interaktion mellem bruger og system.

6.5 Dokumentation

Det er påkrævet at udviklingen af dette projekt bliver dokumenteret. Dokumentationen begrænser sig ikke til udviklings- og designfasen, men også til kravspecifikation og analysefasen. Forskellige former for dokumentation ønskes udført.

- En API dokumentation af systemet. Indeholdende beskrivelse af klasser, metoder og funktionaliteter.
- Dokumentation for analyse samt design, bl.a. i form af UML diagrammer.
- Use-cases og test af disse, til dokumentation for at systemet opfører sig som ønsket.
- Udførte tests-scripts af systemets funktionaliteter.
- En rapport beskrivende projektets udførelse, samt de 4 ovenstående.

En API dokumentation er vigtig for en mere teknisk bruger, samt til udviklere, der skal videreudvikle systemet. Denne vil være beskrivende af karakter, således at dele af systemet belyses, på en måde så en udvikler, der ikke har arbejdet på projektet, kan sætte sig ind i kildekoden.
Det vil være et krav at analyse og design dokumenteres, således at det gøres forståeligt hvorledes de vigtigere beslutninger i projektet er træffet.

6.6 Succes kriterier

Kriterierne for succes, er at få implementeret krav angivet med prioritering fra 1 til 4, da disse anses, som mere eller mindre vigtige, for et funktionelt system. Ligeledes vil det være et succeskriterium, at systemet er designet på en effektiv måde, der giver lav kobling mellem de forskellige moduler i systemet.

Endnu et kriterium er, at de i analysen opstillede use-cases, bliver implementeret og godkendt ved test. Således vil der skabes et system, der ud over at have den ønskede funktionalitet, også fungere efter overensstemmelse med brugernes ønsker.

Slutteligt er det selvfølgeligt et succeskriterium at test-scripts udføres, og at disse godkendes, således at det påvises at systemets funktionalitet og opførsel er som ønsket.

Det kan ikke undgås at visse kriterier ikke vil blive fuldt opfyldt, da udviklingen af systemet foregår i en meget begrænset periode, derfor vil der også blive foretaget en afvejning, der eventuelt kan godtage ikke-kritiske fejl, og alligevel anse projektet som en succes.

Kapitel 7

Teknologier

7 Teknologier

7.1 Indledning

I dette afsnit, vil der blive præsenteret nogle af de teknologier som dette projekt bygger på. Visse teknologier er i kravspecifikationen allerede fastlåst, således at der ikke kan foretages et valg mellem konkurrerende teknologier, og der vil derfor ensidigt blive beskrevet den i kravspecifikationen specificerede teknologi.

7.2 .Net platformen

I lang tid var .Net platformen blot et koncept introduceret af Microsoft. Der var mange meninger om .Net og lige så mange svar på hvad .Net egentlig var for en størrelse, hvor kun få af disse var helt korrekte.
I dag er holdningen blandt modstandere, at .Net blot er Java teknologi pakket pænt ind, og der intet nyskabende er at finde. Denne påstand er selvfølgeligt ikke helt uden grundlag, men sandheden er langt fra denne.

7.2.1 Hvad er .Net

.Net er en software platform. Det er et sprog-neutral miljø for system-udvikling, der tillader enkel afvikling og kommunikation mellem programmer. Modsat kernel-specifikke oversættere, kan .Net ligesom Java, afvikle et kompileret program på alle typer maskiner, så længe .Net platformen er til stede.

.Net er dog også en række programmer og tjenester, hovedsageligt udviklet af Microsoft, bygget på .Net platformen. Sammen giver de en række produkter, som Visual Studio .Net og tjenester som .Net Passport. Samlet kaldes disse .Net Framework'et.

7.2.2 .Net Framwork

.Net Framework'et har to dele, Common Language Runtime (CLR) hvorpå alle .Net applikationer afvikles og en rækker klassebiblioteker bygget oven på CRR'en.
CLR har bl.a. følgende vigtige features:

- Low-level assembler kode, der kaldes Intermidiate Language (IL), hvortil kompilerede applikationer kompileres.
- Memory Management, her under Garbage Collection.
- Sikkerhedsmoduler, der sørger for at overvåge eksekverende kode, således at chancen for harmfuld kode minimeres.
- Versionskontrol af applikationer og biblioteker.

Her følger et par begreber, nødvendige for forståelsen af .Net:

Managed- and Unmanaged code: Kode der henvender sig til .Net platformen, og indeholde meta-data til at beskrive sig selv. Begge kode typer kan afvikles på .Net platformen, men kun managed-code indeholder information, der sørger for at :net kan garanterer sikker eksekvering samt komplet integration mellem applikationer og komponenter.

Managed Data: CLR giver .Net Framework'et sikkerhed indenfor bl.a. hukommelses allokering, ved at introducerer en række nye faciliteter. Her under, er Garbage Collectoren, den mest kendte.
Managed Data bruges som default ved sprog som C#.Net, vb.Net og j#.Net, mens C++ ikke har dette som default. Der ud over, sætter CLR en række restrektioner på tidligere sprog som fx C++, der mister muligheden for at nedarve fra flere klasser samtidig.

For at kunne give integration mellem komponenter udviklet i forskellige sprog, implementerer frameworket det såkaldte Common Type System (CTS), der beskriver datatyper m.m. på en fælles måde. CTS kan ligeledes sørge for at en applikation ikke tilgår hukommelse der ikke er tildelt denne.

Al .Net kildekode er kopileret til IL-kode, i stedet for kerne-specifik bytekode. Ved eksekvering omdanner Just-in-time (JIT) kompileren, denne kode til kernel-specifik kode, således at den specifikke CPU forstår de enkelte kommandoer. Først når en metode kaldes, bliver denne kompileret, og der er derfor ikke et stort overhead ved opstart af applikationer. Kompileret kode gemmes i en cache, således at den samme IL kode, ikke skal kompileres flere gange.

7.2.3 .Net Komponenter

.Net bygger som tidligere programmeringssprog på komponenter. Udviklere der har arbejdet med komponenter i fx Windows, har ofte set, hvorledes en komponent (oftest Dynamic Link Libraries) ved installationen af et program, har overskrevet en anden komponent med samme navn, der ikke var fuldstændig kompatibel med den nye, og derved fået eksisterende applikationer til at fejle.

.Net adresserer denne problematik, ved helt at gå væk fra den gamle COM model, og bygge en ny model, hvori versionering og unik identifikation af delte komponenter er implementeret. Der vil ikke blive gået i dybden med denne model i denne rapport, men for forståelse af .Net platformen er det her nævnt.

7.2.4 Namespaces

Namespaces, der også er kendt fra andre programmeringssprog, er en måde hvorpå et systems komponenter opdeles. Det er en simpel struktureret gruppering af sammenhængende klasser, interfaces, strukturer eller andre namespaces.

7.2.5 Fejlhåndtering

Fejlhåndtering er blevet standardiseret og gjort mere effektivt, via CLR. Således er der i forhold til den gamle COM model standarder for hvorledes fejlhåndtering skal foregå, samt det er muligt at lave fejlhåndtering over processer, i det de er en del af CLR.

7.2.6 ADO.Net

ADO.Net er en klassestruktur integreret i .Net frameworket til at oprette forbindelse til datakilder. Fordi webservices er en vigtig del af Microsofts strategi, er datastrukturene i ADO.Net bygget op omkring XML, hvilket gør at objekter kan sendes gennem standard protokoller som SOAP.
Strukturebe i ADO.Net er opbygget således, at de giver en ensartet facade, på trods af forskellig underliggende strukturer. Således giver det udvikleren en ensartet metode til at tilgå forskellige datakilder.

7.3 Kommunikation

En af det ønskede systems hovedfunktionaliteter er at stille de opsamlede data til rådighed for andre systemer, således at disse kan integrere data i deres eksisterende datastrukturer, eller blot vise data i sammenhæng med de eksisterende data.

7.3.1 Socket kommunikation

Socket kommunikation er en udbredt måde for programmer at kommunikere over et netværk. Systemerne åbner en socket-forbindelse på en specifik port hvormed de kommunikerer – e-mail systemet kunne fungere som server, som andre systemer kunne koble sig op imod.

Denne type forbindelse er rimeligt simpel og pålidelig, men kræver at begge systemer er sat op til at kunne kommunikerer via denne forbindelse, hvilket vil sige at der skal implementeres et modul i bruger systemet til

denne kommunikation. Samtidigt skal der opsættes en standard for kommunikationen, således at systemerne taler samme sprog. Desuden skal der implementeres speciel logik i server-systemet, hvis mange systemer skal kunne kobles op mod dette på en gang.

Socket kommunikation er dog en udbredt kommunikationsform, som alle typer systemer kan implementerer, ligegyldigt hvilket sprog og styresystem de ligger på.

7.3.2 Webservices

Microsoft .NET tilbyder en anden kommunikations form, webservices. Her kan et system tilbyde en række funktionaliteter via Internettet (oftest HTTP protokollen), hvor kommunikationen foregår i XML/SOAP, som er en universelt udbredt standard.

Fordelene her er at kommunikationen netop er forenklet, og det er relativt simpelt at implementerer i eksisterende systemer, da man kan tilgå de tilgængelige funktionaliteter direkte som metoder i systemerne.

En ulempe kan være at i og med kommunikationen foregår via Internettet, kræver webservices en web-server til at eksekvere. Dette giver dog også en fordel, nemlig den at web-serveren varetager kommunikationen, og man derfor ikke behøver at tage hånd om dette i sit system.

7.4 Datakilder

Området for opbevaring af data, har været rimeligt uændret i de seneste år. Der er flere store gennemtestede kommercielle systemer på gaden, de i bund og grund tilbyder de samme features. Samtidig er der mange opensource dataopbevarings systemer, der gratis kan benyttes af enhver. Til dette projekt, der, i forhold til andre projekter, skal bruge et minimum af funktionalitet i dataopbevarings systemet, spiller det ikke den store rolle, hvilet system der vælges. Derfor vil der ikke blive givet en grundig analyse af de forskellige muligheder, men i Analyse afsnittet vil der fortages en kort diskussion om emnet.

Muligheder for eksisterende dataopbevaringssytemer der understøtter kravene i kravspecifikationen er fx:

- Microsoft SQL Server
- Oracle
- MySQL
- Sybase
- m.fl.

7.5 E-mail kommunikation

7.5.1 POP3

Kommunikation med en e-mail server foregår i dag hovedsageligt gennem den aldrende, men pålidelige POP3 protokol. Post Office Protocol 3, er som navnet antyder nummer 3 i rækken af protokoller, der tilbyder kommunikation med e-mail servere, sat op til POP3 kommunikation. Langt de fleste abonnenter på individuelle e-mail konti, bruger POP3 protokollen til at forbinde og kommunikere med deres e-mail server. POP3 protokollen tilbyder ikke megen funktionalitet, men har indtil videre kunne tilfredsstille store dele af markedet. En udvidet version med kryptering af login information, APOP, blev udviklet for over 10 år siden, men bruges ikke ofte.
Når e-mails hentes fra serveren, er der generelt to muligheder – enten at hente e-mails og slette dem fra serveren, eller at hente dem uden at slette dem. Modsat nyere protokoller, som IMAP, kan man ikke specifikt identificere den enkelte e-mail fra forbindelse til forbindelse, hvilket kan give nogle problemer, hvis man ønsker efterlade e-mails på serveren, og der efter synkronisere med serveren.

Her under gives et eksempel på en dialog mellem klient og server, over POP3 protokollen.

```
S: <wait for connection on TCP port 110>
C: <open connection>
S:    +OK POP3 server ready <1896.697170952@dbc.mtview.ca.us>
C:    USER mrose
S:    +OK User acceptet
C:    PASS 123456
S:    +OK Pass acceptet
C:    STAT
S:    +OK 2 320
C:    LIST
S:    +OK 2 messages (320 octets)
S:    1 120
S:    2 200
S:    .
C:    RETR 1
S:    +OK 120 octets
S:    <the POP3 server sends message 1>
S:    .
C:    DELE 1
S:    +OK message 1 deleted
C:    RETR 2
S:    +OK 200 octets
S:    <the POP3 server sends message 2>
S:    .
C:    DELE 2
S:    +OK message 2 deleted
C:    QUIT
S:    +OK dewey POP3 server signing off (maildrop empty)
C:  <close connection>
```

7.5.2 SMTP

Til afsendelse af e-mails, bruges den anonyme SMTP protokol oftest. Denne protokol er om end mere simpel end POP3, og giver ikke brugeren mulighed for at identificere sig selv. Grunden til at den aldrende SMTP protokol stadig benyttes, er simpelthen den, at for mange servere allerede benytter sig af den, samtidigt med at den er så simpel at benytte.

7.5.3 IMAP

Som før omtalt, er der udviklet en mere avanceret protokol, til at læse e-mails fra en e-mail server, nemlig IMAP. Denne protokol har den store forskel fra POP3 protokollen, at man her specifikt kan identificere e-mails fra forbindelse til forbindelse. E-mails efterlades på serveren, indtil brugeren specifikt beordrer dem slettet. IMAP protokollen antager ved synkronisering at det er serveren der indeholder den korrekte liste over e-mails, og synkroniserer så klienter efter den.

7.6 Design Patterns

Design Pattern eller designmønster er en generel løsning til en type problem, der ofte opstår i softwareudvikling. Et design pattern er ikke et endeligt design, der kan programmeres direkte; det er en beskrivelse eller skabelon for, hvordan man løser et problem i mange forskellige situationer.

Design Patterns er et af de store emner i software udvikling i dag, og er derfor ved at være godt beskrevet i litteraturen. Denne rapport vil ikke gå i dybden med disse, men vil i stedet referere til bøger som fx 'Elements of Reusable Object-Oriented Software', også kendt som 'The Gang of Four'.

Kapitel 8

Analyse

8 Analyse

8.1 Indledning

Målet for dette afsnit, er gennem en grundig analyse, at finde den bedst mulige måde at løse den ønskede opgave på. Samtidig vil analysen give et bedre grundlag for at kunne designe et robust og fleksibelt system, der får præcis den funktionalitet der ønskes.

Analysen vil hovedsageligt kigge på større konceptuelle idéer, hvor der kigges på systemet som en række sorte bokse – i et senere afsnit vil indholdet af disse sorte bokse diskuteres – samspillet mellem enhederne i systemet, og samspillet mellem bruger og system.

8.2 Interface

En af det ønskede systems hovedfunktionaliteter er at stille de opsamlede data til rådighed for andre systemer, således at disse kan integrere data i deres eksisterende datastrukturer, eller blot vise data i sammenhæng med de eksisterende data - i det følgende afsnit, vil det blive diskuteret hvorledes denne opgave håndteres bedst.

For at skabe denne kommunikation mellem systemerne, blev der overvejet to teknologier, der er beskrevet her under. Som tidligere omtalt, skal man huske på, at det ønskede system og de eksisterende systemer ofte vil eksekveres på forskellige servere, og derfor skal kommunikationsformen understøtte kommunikation via netværk.

8.2.1 Socket kommunikation

Socket kommunikation er en udbredt måde for programmer at kommunikere over et netværk. Systemerne åbner en socket-forbindelse på en specifik port hvormed de kommunikerer – e-mail systemet kunne fungere som server, som andre systemer kunne koble sig op imod.

Denne type forbindelse er rimeligt simpel og pålidelig, men kræver at begge systemer er sat op til at kunne kommunikerer via denne forbindelse, hvilket vil sige at der skal implementeres et modul i bruger systemet til denne kommunikation. Samtidigt skal der opsættes en standard for kommunikationen, således at systemerne taler samme sprog. Desuden skal der implementeres speciel logik i server-systemet, hvis mange systemer, skal kunne kobles op mod dette på en gang.

Socket kommunikation er dog en udbredt kommunikationsform, som alle typer systemer kan implementerer, ligegyldigt hvilket sprog og styresystem, der benyttes.

8.2.2 Webservices

Microsoft .NET tilbyder en anden kommunikations form, WebServices. Her kan e-mail systemet tilbyde en række funktionaliteter via Internettet (oftest HTTP protokollen), hvor kommunikationen foregår i XML/SOAP, som er en universal udbredt standard.

Fordelene her er at kommunikationen netop er forenklet, og det er relativt simpelt at implementerer i eksisterende systemer, da man kan tilgå de tilgængelige funktionaliteter direkte som metoder i systemerne, ved at bruge Microsofts platform.

En ulempe kan være at i og med kommunikationen foregår via Internettet, kræver WebServices en web-server til at eksekvere. Dette giver dog også en fordel, nemlig den at web-serveren varetager kommunikationen, og man derfor ikke behøver at tage hånd om dette i sit system.

Ved brug af WebServices er der defineret en standard protokol til kommunikationen, hvilket igen betyder forenklet udvikling.

8.2.3 Konklusion

For dette projekt, anses WebServices som den enkleste løsning, der dog ikke mangler funktionalitet, og derfor opfylder §1.3 omkring Interfacet.

Først og fremmest tilbyder WebServices en god metode til at stille sit systems funktionalitet til rådighed via en standard protokol, da en WebService på enkel vis forbinder system og Internettet. Samtidig er det lige så enkelt at tilgå webservicen fra eksisterende systemer, da man kommunikerer via XML og SOAP over HTTP protokollen.
En web-server er så en nødvendighed, men man kan argumentere for at dette er en lille pris at betale, for den enkelthed og store fleksibilitet som WebServices tilbyder.

Da de eksisterende systemer i samarbejdsfirmaet, allerede er bygget op omkring WebServices og web-servere er det også et naturligt valg.

Man kunne dog sagtens, hvis e-mail systemet skulle udbredes yderligere, lave en komponent der stillede et interface via socket kommunikation til rådighed også. Således ville systemet blive endnu mere fleksibelt, og brugere ville kunne vælge deres præfererede kommunikations metode.

8.3 E-mail kommunikation

Som det omtales tidligere, findes der flere forskellige protokoller til kommunikation med e-mail servere. POP3 protokollen er stadig den mest udbredte, som langt de fleste servere understøtter, men efterhånden er IMAP protokollen ved at komme med, pga. dens udvidede funktionalitet.

For at opnå et fuldt fleksibelt system, er det nødvendigt at understøtte dem begge, men der er dog ingen fordele at opnå i EMS Systemet, ved at bruge IMAP. Grunden til dette er, at systemet henter e-mails ned i sit eget lager, og derefter skal slette e-mails fra serveren, da det vil give redundant data at have dem liggende to steder.

8.4 E-mail parsing

At parse en e-mail fra ren tekstdata, til en mere brugelig klasse struktur er ikke et nemt job. E-mail består af to dele. *Headeren*, eller brev hovedet, bestående af nogle specifikke og altid eksisterende felter, hvor nødvendigt data rimeligt nemt kan fremsøges, samt nogle felter der kan variere i indhold og udformning, men som ofte ikke er vigtige for brugeren. Nogle af standard felterne er f.eks. afsender adresse, modtager og emne.

Den anden del af e-mailen er egentlig flere dele. Dette er indholdsdelen.
Indholdet af en e-mail kan være mange ting, netop derfor er denne del delt op i forskellige sektioner, med et felt, der fortæller om hvorledes den enkelte sektion skal behandles – dette kaldes en MIME-type.

Oftest har en e-mail en sektion af typen *text/plain*, der indeholder ren tekst. Dette er selve e-mail teksten, altså selve brevet. Dog er der efterhånden mange, der sender e-mails indeholdende HTML[4] kode, for at opnå et mere farverigt indhold. Det betyder at e-mailens hovedsektion nu ofte er af typen *text/html*, og selve indholdet af e-mailen kan ikke læses direkte, men skal igen parses af en browser eller et system der selv kan renderer HTML kode.

Ud over teksten, kan en e-mail indeholde vedhæftede filer, billeder mm. Denne data ligger som separate sektioner, med felter der fortæller om MIME-type, filnavn og andre oplysninger vedrørende denne data.

[4] HTML (HyperText Markup Language) er et markup sprog, der bruges til at præsenterer tekst og billeder i en browser på en pæn og overskuelig måde.

En e-mail kunne f.eks. se således ud.

```
Return-path:  <martin@fubunet.dk>
Envelope-to:  garfield_ml@stofanet.dk
Delivery-date: Thu, 10 Aug 2006 13:14:54 +0200
Received:     from mail07.talkactive.net ([195.128.174.77])
              by mx01.stofanet.dk (envelope-from
              <martin@fubunet.dk>)
              with smtp id 1GB8V4-0004Ng-1D
              for garfield_ml@stofanet.dk; Thu, 10 Aug 2006 13:14:54 +0200
Received:     (qmail 99204 invoked by uid 80); 10 Aug 2006 11:13:30 -0000
Received:     from 87.48.119.55
              (SquirrelMail authenticated user martin@fubunet.dk)
              by mail07.talkactive.net with HTTP;
              Thu, 10 Aug 2006 13:13:30 +0200 (CEST)
Message-ID:   <42196.87.48.119.55.1155208410.squirrel@mail07.talkactive.net>
Date:         Thu, 10 Aug 2006 13:13:30 +0200 (CEST)
Subject:      Test fra Fubunet
From:         "Martin L" <martin@fubunet.dk>
To:           garfield_ml@stofanet.dk
Reply-To:     martin@fubunet.dk
User-Agent:   SquirrelMail/1.5.1
X-Antivirus:  AVG for E-mail 7.1.405 [268.12.6/453]
Mime-Version: 1.0
Content-Type: multipart/mixed; boundary="----=_20060810131330_15874"

------=_20060810131330_15874
Content-Type: text/plain; charset=iso-8859-1
Content-Transfer-Encoding: 8bit

Test: Martin siger hej.

------=_20060810131330_15874
Content-Type: image/pjpeg; name=image.jpg
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="image.jpg"

/9j/4AAQSkZJRgABAAEA1gCWAAD//gAfTEVBRCBUZWNobm9sb2dpZXMgSW5jLiBWMS4wMQD/2wCE
AAQDAwQDAwQEAwQFBQQFBwwHBwYGBw4KCwgMEQ8SEhEPEBATFRsXExQaFBAQGCAYGhwdHh8eEhch
JCEeJBseHh0BBQUFBwYHDgcHDh0TEBMdHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0d
HR0dHR0dHR0dHR0dHf/EAaIAAAEFAQEBAQEBAAAAAAAAAAABAgMEBQYHCAkKCwEAAwEBAQEB
AQEBAQAAAAAAAAECAwQFBgcICQoLEAACAQMDAgQDBQUEBAAAAX0BAgMABBEFEiExQQYTUWEHInEU
AAN2AKYluKrAldgC5GDx1oHYCPLCqr9unNAnpsMdQwbK7eecdvagVl1P/9k=

------=_20060810131330_15874
Content-Type: text/plain; x-avg=cert; charset=us-ascii
Content-Transfer-Encoding: quoted-printable
Content-Disposition: inline
Content-Description: "AVG certification"

No virus found in this incoming message.
Checked by AVG Free Edition.
Version: 7.1.405 / Virus Database: 268.12.6/453 - Release Date: 20-09-2006

------=_20060810131330_15874--
```

Den ovenstående e-mail består af header, markeret med rød, og 3 sektioner, hver med sin egen farve.

Den grønne sektion er en gammeldags tekst sektion, der er e-mailens tekst indhold. Her blot ”Test: Martin siger hej.”

Den blå sektion længere nede, er en vedhæftet fil, her i form af et *jpeg* formateret billede, med filnavnet ”image.jpg”.

Den lilla sektion er en sektion indsat af en virus scanner, der blot fortæller at e-mailen er scannet for virus, og der ikke blev fundet noget mistænkeligt.

Til sidst er der en sort linie der fortæller, at e-mailen nu ikke indeholder mere data.

For at kunne opnå et fuldt fungerende system, er det nødvendigt at kunne parse e-mails. Det er målet med at få bygget et modul til denne funktionalitet, der skal indgå som en del af modulet til kommunikation med en e-mail server.
Dog er der også den mulighed at benytte et af de mange Open Source[5] moduler, der kan findes på Internettet. Det kan forsvares med, at der ikke er tid til at bygge det ønskede modul således, at det vil kunne parse alle tænkelige kombinationer af e-mails. Samtidig er det heller ikke hovedformålet med denne opgave at forstå opbygningen af en e-mail, samt udvikle en metode til at parse denne – hvilket også er grunden til den blot lette berøring af netop dette emne.

8.5 Datalager

Systemet skal kunne gemme e-mails på en måde, således at de opfylder kravene i §1.2. Der vil følgende blive diskuteret hvilke muligheder der kunne bruges, samt hvilke der anses for passende til dette projekt.

Et datalager kan være mange ting, lige fra tekstfiler til store datawarehouse løsninger. I dette projekt vil der blive koncentreret om 2 mulige løsninger. Tekstbaseret og database baseret datalager.

8.5.1 Tekstbaseret lagring i separate filer

E-mails er i sagens natur ideelle at opbevare i enkelte tekst filer, separeret fra hinanden. En e-mail er som et standard dokument opbygget efter nogle specielle regler, og vil derfor nemt kunne skelnes fra andre typer af dokumenter, hentes ind og dechifreres i en klient og slettes og flyttes.

Tekstbaseret lagring i separate filer har dog sine begrænsninger når det kommer til sammenligning og kategorisering af e-mails, der er kritisk funktionalitet for projektet.
For at sammenligne to eller flere e-mails, må disse hentes ind i en klient, dechifreres og analyseres enkeltvis, for derefter at kategoriseres. Denne megen behandling af enkelte filer, gør tekstbaseret lagring i separate filer til en smule besværligt.

Der ville være fordele ved denne type lagring, hvis e-mails skulle gemmes hos en enkelt bruger, der ønskede at kunne tilgå e-mails med flere forskellige klienter, eller direkte i en teksteditor. Det er dog ikke hensigten med EMS Systemet, at brugere har direkte adgang til e-mails.

8.5.2 Database baseret lagring

Databaser er praktiske til at opbevare store mængder data, der skal kunne kategoriseres og fremsøges på baggrund af specifikke parametre.
Databaser er et persistent medie lige som enkelte filer, og tilbyder en del 'gratis' funktionalitet igennem forespørgsler.

Fordelen ved en database løsning, er den enkle måde hvorpå data kan sammenlignes og udtrækkes fra datalageret. Ved korrekt opbygning af databasen, vil det være muligt at kategoriserer e-mails og opnå en rimeligt optimal løsning, der opfylder kravene i §1.2. Da e-mails og dets vedhæftede filer er ren binær data, er der ingen hindringer for at disse gemmes i en database, sammen med de parametre hvorved de kan fremsøges.

8.5.3 Kombineret løsning

Der kan argumenteres for, at man kunne lave en kombineret løsning, hvor der blev vedligeholdt en database til kategorisering af e-mails, men hvor selve e-mails blev gemt i tekstfiler uden for databasen.
Der er dog ingen umiddelbar fordel i at have e-mails gemt i tekstfiler i dette projekt, og det anses derfor som urelevant at gøre dette. Oven i købet ville dette betyde, at der skulle udvikles et ekstra interface til

[5] OpenSource er en betegnelse der bruges om kode der er mere eller mindre til fri afbenyttelse.

læsning og skrivning af filer på serveren, hvilket ville give længere udvikling og større mulighed for fejl i systemet.

8.6 Aktører og Use Cases

Aktører og use cases bruges til at bestemme hvilke interaktioner et system vil blive udsat for. Ved at specificere et antal aktører og en række scenarier for interaktion mellem disse og systemet, use cases, opstilles der en ramme for hvorledes systemet skal opføre sig, på et overfladisk niveau.
Use cases kan senere bruges som test og verifikation på at systemets overfladiske funktionalitet er som ønsket.

8.6.1 Identifikation af aktører

En måde at finde aktører på er simpelthen at forestille sig hvem, der kan have interesse i at bruge systemet. Desuden er systemet opdelt i mindre del-systemer, som alle kan være aktører i hinanden. Derfor kan disse også tages med som sekundære aktører.

8.6.1.1 Primære aktører

- IT-Medarbejder
  IT-Medarbejderen vil være ansvarlig for opsætningen af systemet, samt konfiguration heraf, og vil derfor komme i kontakt med systemets konfigurations interface.

- Eksternt system
  Det eksterne system vil bruge det udvikledes systems primære funktionaliteter, og vil komme i kontakt, med det funktionelle interface.

- E-mail server
  EMS Systemet vil kommunikere med en række e-mail servere for at udveksle e-mails. Kommunikationen her vil foregå over en kendt protokol.

8.6.1.2 Sekundære aktører

- Hoved system
  Det egentlige system, inklusive konfigurations- og primært interface.

- Datalager
  Datalageret hvor data omkring e-mails gemmes

8.6.2 Use Cases

I de følgende afsnit vil vi præsenterer et udvalg af de udviklede use cases.

Use cases viser hvordan det ønskes at programmet skal fungere. De lægger vægt på den visuelle funktionalitet set fra brugere, og ikke hvad der sker under overfladen i programmet. Use cases beskriver derfor de funktioner som systemet skal kunne udføre, og hvordan det for aktøren ser ud under udførelsen.

Use cases kan deles op i to kategorier, hhv. use cases hvor menneskelige aktører indgår, og use cases mellem systemer.

- Uses cases hvori der indgår menneskelige aktører, findes bedst ved at konsulterer de specifikke aktører, hvorved der kan dannes et indblik i hvorledes denne finder at systemet skal opføre sig. Denne type use cases er især vigtige ved brugerflader, som grafiske interfaces, da brugere og

designere oftest har vidt forskellige opfattelser af hvad der er en god brugerflade. Samtidig vil brugere oftest have en speciel arbejdsgang, som designeren af systemet ikke har kendskab til på forhånd.

- Use cases mellem systemer, er mere tekniske af natur. Oftest vil der være et sæt regler, en protokol, for hvorledes et system interagerer med andre systemer, og derfor vil use cases være bundet af disse.

8.6.2.1 Use case #1

Use Case: 1 – Opsætte en e-mail server

Goal:	At opsætte systemet til at hente e-mails fra en server
Preconditions:	Systemet, database og andre afhængigheder fungerer. Data omkring e-mail server er korrekt.
Scope:	Her ses hele systemet som en sort box, brugeren har kun kontakt med systemet brugergrænseflade.
Success End Condition:	Konfigurationen afsluttes og data gemmes korrekt
Failed End Condition:	Konfigurationen fejler pga. systemfejl
Primary Actor:	Bruger
Trigger:	En bruger ønsker at hente e-mails fra en ny server

Main success scenario:

1. Brugeren åbner systemet
2. Brugeren vælger menupunkter ”Server konfiguration”.
3. Brugeren vælger ”Tilføj Server”.
4. Brugeren indtaster data omkring e-mail server i indtastningsfelterne, og trykker ”Gem”.
5. Data gemmes
6. Bruger genstarter eventuelt automatisk download, hvis denne kører
7. Done

Extensions:

1a. Systemet kører allerede, derfor skal systemet ikke startes op.
4a. Bruger indtaster ugyldigt data
    4a1. Systemet advarer bruger og gemmer ikke
    4a2. Step 4 gentages
5a. Data kan ikke gemmes, pga. systemfejl.
    5a1. Bruger gøres opmærksom på dette

Other information:

Priority: Høj prioritet
Performance Target: 1-2 minutter
Frequency: Sjældent

Open issues:

- Skal systemet eventuelt selv genstarte automatisk download?

8.6.2.2 Use case #2

Use Case: 2 – Ændre systemkonfigurationen

Goal:	At ændre i systemets opsætning
Preconditions:	Systemet, database og andre afhængigheder fungerer.
Scope:	Her ses hele systemet som en sort boks, brugeren har kun kontakt med systemet brugergrænseflade.
Success End Condition:	Konfigurationen afsluttes og data gemmes korrekt
Failed End Condition:	Konfigurationen fejler pga. systemfejl
Primary Actor:	Bruger
Trigger:	En bruger ønsker ændre systemets konfiguration

Main success scenario:

1. Brugeren åbner systemet
2. Brugeren vælger menupunkter ”Konfiguration”.
3. Brugeren ændrer den ønskede indstilling.
4. Data gemmes
5. Bruger genstarter eventuelt automatisk download, hvis denne kører
6. Done

Extensions:

1a. Systemet kører allerede, derfor skal systemet ikke startes op.
3a. Bruger indtaster ugyldigt data
    3a1. Systemet advarer bruger og gemmer ikke
    3a2. Step 4 gentages
4a. Data kan ikke gemmes, pga. systemfejl.
    4a1. Bruger gøres opmærksom på dette

Other information:

Priority: Høj prioritet
Performance Target: Under 1 minut
Frequency: Sjældent

Open issues:

- Skal systemet eventuelt selv genstarte automatisk download?

8.6.2.3 Use case #3

Use Case: 3 – Starte automatisk download

Goal:	At starte automatisk download af e-mails
Preconditions:	Systemet, database og andre afhængigheder fungerer.
Scope:	Her ses hele systemet som en sort boks, brugeren har kun kontakt med systemet brugergrænseflade.
Success End Condition:	Automatisk download starter korrekt
Failed End Condition:	Automatisk download fejler pga. systemfejl
Primary Actor:	Bruger
Trigger:	En bruger ønsker at starte automatisk download

Main success scenario:

1. Brugeren åbner systemet
2. Brugeren trykker på knappen for start af automatisk download.
3. Automatisk download starter.
4. Systemet opdateres, således at det kan ses at automatisk download er i gang.
5. Automatisk download arbejder
6. Done

Extensions:

1a. Systemet kører allerede, derfor skal systemet ikke startes op.
2a. Automatisk download er i gang
    2a1. Bruger kan ikke vælge at trykke på knappen for automatisk download, gå til step 6.
2b. Ingen e-mail servere konfigureret
    2b1. Bruger gøres opmærksom på, at der ikke er konfigureret nogle servere. Gå til step 6.
5a. Automatisk download fejler pga. systemfejl
    5a1. Bruger gøres opmærksom på dette, automatisk download afsluttes.
5b. Automatisk download fejler for en enkelt server
    5b1. Bruger gøres opmærksom på dette, automatisk download kører videre.

Other information:

Priority: Høj prioritet
Performance Target: Afhængig af e-mail server konfiguration
Frequency: Ofte

Open issues:

- Hvorledes skal bruger gøres opmærksom på at en, eller flere, servere fejler, da dette ikke er en kritisk fejl for alle servere, og derfor ikke stopper automatisk download?

8.6.2.4 Use case #4

Use Case: 4 – Stoppe automatisk download

Goal:	At stoppe automatisk download af e-mails
Preconditions:	Systemet, database og andre afhængigheder fungerer.
Scope:	Her ses hele systemet som en sort boks, brugeren har kun kontakt med systemet brugergrænseflade.
Success End Condition:	Automatisk download afsluttes korrekt
Failed End Condition:	Automatisk download afsluttes med systemfejl
Primary Actor:	Bruger
Trigger:	En bruger ønsker at stoppe automatisk download

Main success scenario:

1. Brugeren åbner systemet
2. Brugeren trykker på knappen for stop af automatisk download.
3. Systemet venter på at færdiggøre en iteration af automatisk doenload
4. Automatisk download stopper.
5. Systemet opdateres, således at det kan ses at automatisk download er stoppet.
6. Done

Extensions:

1a. Systemet kører allerede, derfor skal systemet ikke startes op.
2a. Automatisk download stoppet
    2a1. Bruger kan ikke vælge at trykke på knappen for at stoppe, gå til step 6.
4a. Automatisk download fejler pga. systemfejl
    4a1. Bruger gøres opmærksom på dette, automatisk download afsluttes med fejl.

Other information:

Priority: Høj prioritet
Performance Target: Afhængig af e-mail server konfiguration
Frequency: Ofte

Open issues:

- Er data konsistent, hvis automatisk download af e-mails ikke stoppes korrekt?

8.6.2.5 Use case #5

Use Case: 5 – E-mail søgning via GUI

Goal:	At fremsøge e-mails via GUI'et
Preconditions:	Systemet, database og andre afhængigheder fungerer.
Scope:	Her ses hele systemet som en sort boks, brugeren har kun kontakt med systemet brugergrænseflade.
Success End Condition:	Søgning afsluttes korrekt
Failed End Condition:	Søgning afsluttes med systemfejl
Primary Actor:	Bruger
Trigger:	En bruger ønsker at stoppe foretage en søgning i e-mail lageret via det grafiske brugerinterface

Main success scenario:

1. Brugeren åbner systemet
2. Bruger indtaster søge parametre i de der til hørende felter.
3. Passende e-mails fremsøges
4. E-mails vises i resultatvinduet
5. Done

Extensions:

1a. Systemet kører allerede, derfor skal systemet ikke startes op.
2a. Bruger indtaster ugyldigt data
    2a1. Brugeren gøres opmærksom på dette. Gentag step 2.
3a. Søgningen kan ikke foretages, pga. systemfejl.
    3a1. Bruger gøres opmærksom på dette.
4a. Ingen e-mails passer på søgningen
    4a1. Der vises intet i resultatvinduet

Other information:

Priority: Høj prioritet
Performance Target: Afhængig af søgeparametre samt antal af e-mails i lageret. Dog under 1 minut.
Frequency: Ofte

Open issues:

- Skal der eventuelt vises en "ingen resultater fundet" besked?

8.6.2.6 Use case #6

Use Case: 6 – E-mail søgning via System Interface

Goal:	At fremsøge e-mails via system interfacet
Preconditions:	Systemet, database og andre afhængigheder fungerer. Systemet kører.
Scope:	Her ses hele systemet som en sort boks, brugeren har kun kontakt med systemet system grænsefladen.
Success End Condition:	Søgning afsluttes korrekt
Failed End Condition:	Søgning afsluttes med systemfejl
Primary Actor:	Bruger
Trigger:	En bruger ønsker at stoppe foretage en søgning i e-mail lageret via systeminterfacet.

Main success scenario:

1. Metode med søgeparametre kaldes.
2. Passende e-mails fremsøges
3. Resultat returneres som XML
4. Done

Extensions:

1a. Metode kaldes med ugyldigt data.
    1a1. Fejlbesked returneres, gå til step 4.
2a. Søgningen kan ikke foretages, pga. systemfejl.
    2a1. Fejlbesked returneres, gå til step 4.
3a. Ingen e-mails passer på søgningen
    3a1. Tomt resultat returneres

Other information:

Priority: Høj prioritet
Performance Target: Afhængig af søgeparametre samt antal af e-mails i lageret. Dog under 1 minut.
Frequency: Ofte

Open issues:

8.6.3 Domæne model

Domænemodellen er en oversigt over de koncepter der findes i opgavens domæne. Samtidig giver det et billede af hvordan disse koncepter hænger sammen. Denne kan give designeren et overblik over hvorledes systemet skal designes, ud fra et hvilke koncepter der er identificeret. Et koncept forstås som en egentlig del af systemet, eller et mere abstrakt begreb, som f.eks. en e-mail.

8.6.3.1 Identifikation af koncepter

Følgende koncepter er identificeret for systemet.

- E-mail
  En række data, sammensat på en specifik måde.

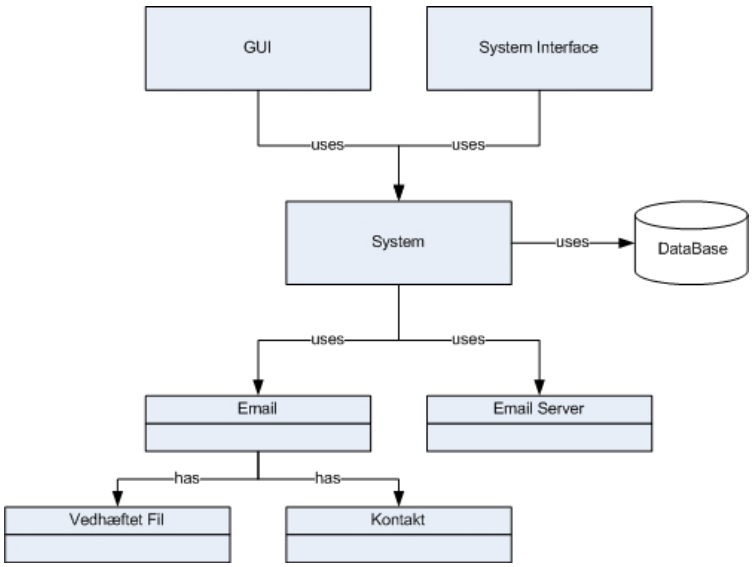
- Vedhæftet fil
  Egentlig en del af en e-mail, men disse kræver speciel varsomhed, og bliver derfor specificeret.

- Kontakt
  Også en underdel af e-mail. Består af flere underdele, og er derfor trukket ud som et enkeltstående koncept.

- E-mail server
  Aktør, men også et begreb i systemet, da systemet vil skulle kunne håndtere disse på en passende måde ifølge §1.4.

8.6.3.2 Domæne diagram

I nedenstående model vises systemet, systemets interne aktører samt de identificerede koncepter, således som de hænger sammen i domænet. Medtaget er datalager samt E-mail server (aktør), hvilke egentlig ligger uden for domænet. Dette er dog gjort for at fremme forståelsen for systemet.

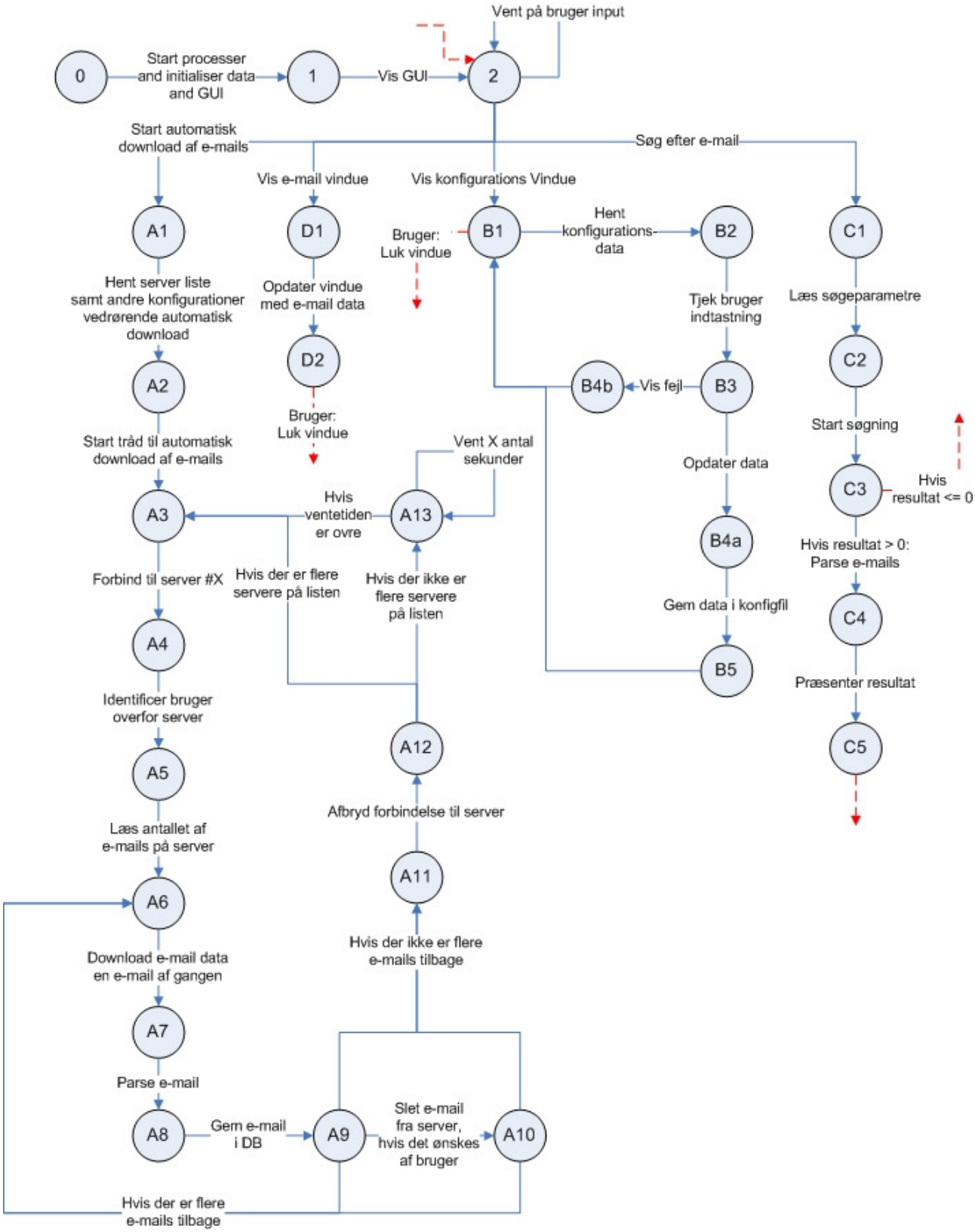


Figur 8.1

8.6.4 Flowchart

Flowcharts beskriver udførelsen af funktionalitet i systemet. Således viser disse hvorledes det forventes, at de enkelte del-funktionaliteters eksekvering vil foregå. Der vil ikke blive gået i detaljer med tekniske detaljer, men således kun *flow'et* i systemet.

8.6.4.1 Flowchart for systemet



Figur 8.2

Røde pile indikerer tilbagevenden til stadie 2. Grenene A, B og D, kan foretages samtidig, da de alle foregår i forskellige processor eller vinduer. Gren C kan kun foretages samtidig med gren A, i så fald at A er startet først.

Kapitel 9

Design

9 Design

9.1 Database struktur og design

9.1.1 Indledning

Som analysen tidligere gjorde rede for, er datalageret blevet udviklet, således at der bliver brugt en database løsning som data-opbevaring. Valget af database er dels afhængig af kravspecifikationen, dels det at visse løsninger er open-sorce, og derved gratis at benytte.
Det følgende design, er blevet udviklet til MySQL database løsningen, men kan i lige så høj grad bruges til MSSQL løsninger, da de nævnte understøtter de samme funktionaliteter.

ER-diagrammet for databasen, giver et billede af databasens struktur – det giver database designeren et overblik over sammenhænge og udformningen af databasen, sådan at det gøres nemmere og mere overskueligt at skabe den fysiske database.

Entiteter og attributter er blevet fastlagt og sat op således, at de kan overholde de krav, der stilles til systemet i dette projekt.

For at få en sammenhængende database, er det stillet som krav at databasen er relationel. Det giver også klart den mest overskuelige design – dog vil det vise sig at dette krav, kan udelades, da database designet ikke giver behov for relationelle sammenhænge.
Visse begreber har et indbygget tilhørsforhold fra begrebsverdenen, og det er disse som bliver implementeret i vores database og oftest kræver relationer.
Man kunne her forestille sig en e-mail og dennes vedhæftede filer – hver vedhæftede fil måtte således have en relationel forbindelse til dennes e-mail, for at kunne skabe sammenhæng i databasen.

For at videreføre kravet om unik identifikation af poster (§1.2.7), er det muligt at benytte sig af en databases "auto increment" funktionalitet. Denne sørger for at enhver ny posts ID bliver genereret af database-systemet, således at ingen poster får samme værdi.

For at efterkomme krav om indeksering (§1.2.3), er det valgt at benytte indbyggede indekserings muligheder i de eksisterende database typer. I MySQL databasen kan man f.eks. ved oprettelsen af en tabel, vælge at en bestemt kolonne skal være indekseret. Dette sørger for at der skabes et indeks på den givne kolonne. Således er e-mails indekseret på f.eks. afsender og afsender dato.
Ved at bruge eksisterende teknologis muligheder for automatiske indeksering, kan man spare en del arbejde med manuelt at lave indeksering, ved f.eks. at splitte tabeller op.

9.1.2 ER-diagram

EMSEmail		
PK	ID	Int
Index	EmailFrom	VarChar(50)
Index	EmailTo	VarChar(50)
Index	EmailSubject	VarChar(100)
Index	EmailDate	DateTime
Index	HasAtt	Binary
	EmailData	LongText

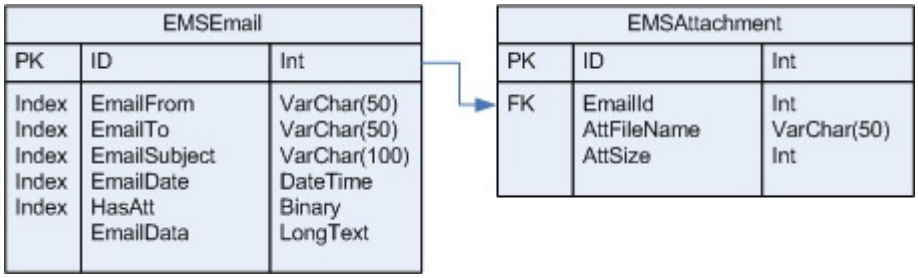
Figur 9.1

9.1.3 Database struktur

Som det ses ud fra ER-diagrammet, indeholder databasen kun en enkelt tabel, hvori data omkring e-mails bliver opbevaret. Således er e-mails gemt på en måde, hvor det er muligt at søge på afsender, modtager, emne, afsender dato samt om e-mailen har vedhæftede filer eller ej. E-mail data er gemt "råt" i den sidste kolonne, hvilket gør at der på ingen måde bliver tabt data.

I og med at databasen kun indeholder en enkelt tabel, gør det at evt. relationer mellem data, ikke er nødvendige eller, for den sags skyld, mulige.

Man kunne argumentere for at opsplitte e-mail tabellen, således at vedhæftede filer, fik sin egen søge tabel, og der blev lavet en relationel sammenhæng mellem de to. Tabellen skulle ikke indeholde data for de vedhæftede filer, men blot fungere som et indeks. Dette ville give den yderligere mulighed, at det også blev muligt at søge e-mails igennem for vedhæftede filer.
Dette er dog ikke blevet foretaget, af rent tidsmæssige grunde, men et muligt udseende for denne database, kan ses af følgende ER-diagram.



Figur 9.2

Ligeledes kunne databasen udvides, således at modtager udtrækkes i en særlig tabel. Således ville det være muligt at søge på en specifik modtager, hvis der skulle være flere modtagere på samme e-mail.

9.1.4 Normalisering

Under konstruktion af databasen, blev der brugt metoder til normalisering, for at tjekke at strukturen fungerede optimalt. Der blev anvendt to metoder: "Boyce Codd Normalform" og "1. til 3. normalform".
Disse metoder bruges bl.a. til at sørge for at databasen ikke indeholder redundant data, hvilket nemt opstår ved databaser med mange tabeller.
Som nævnt tidligere indeholder den udviklede database dog kun en enkelt tabel, og det er derfor ret enkelt at udføre de ovenfor nævnte metoder og bringe databasen på normalform.

9.2 Software

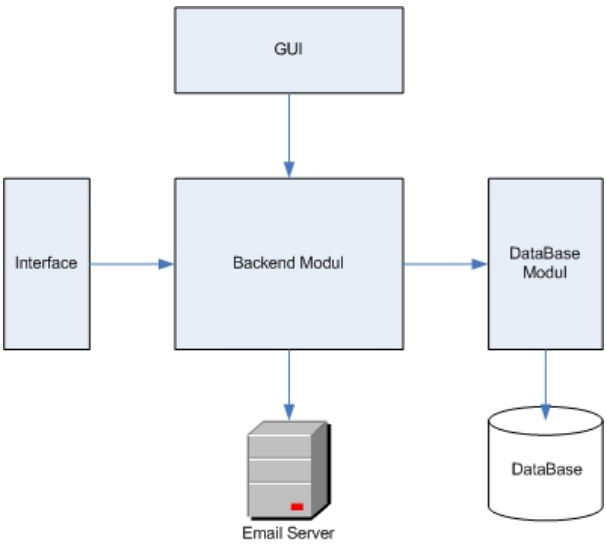
9.2.1 Indledning

Hoved funktionaliteten i projektet ligger i en række moduler, der hver tager vare på del funktionaliteter. En modul opdelt struktur er vigtig, for at give høj fleksibilitet samt lav kobling i projektet, og sikre samtidig et overskueligt design, der kan videreudvikles på en fornuftig måde.
Korrekt anvendt modul-design giver ydermere en hurtigere udviklingsperiode, og sørger for at der laves færre fejl og især færre gentagelser i koden.

I de følgende afsnit vil der blive præsenteret en oversigt over det udvikledes programmels moduler, samt blive givet en forklaring til deres funktionalitet, samt hvorledes opdelingen er blevet valgt som den er.

9.2.2 Modul-design

EMS Systemet er delt op i fire hovedmoduler, samt en række undermoduler. De fire hovedmoduler, hvori datalageret ikke er indregnet, er Interface, GUI, database modul og Backend systemet. Dette er visualiseret på følgende diagram



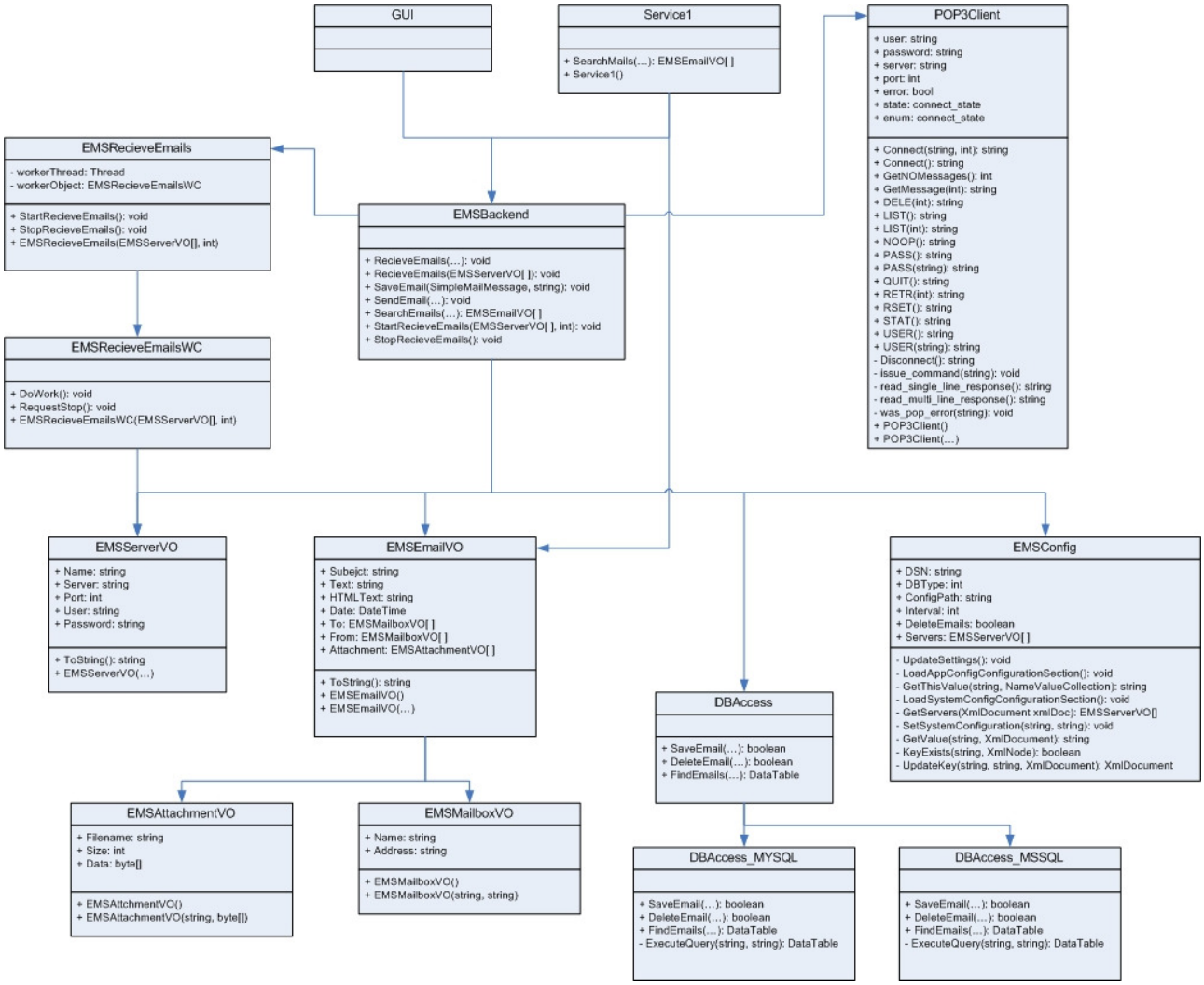
Figur 9.3

Som det kan ses ud fra diagrammet, og som det også antydes af navnet, er Backend modulet det bærende modul, der kobler de andre moduler sammen. Det er i Backend modulet at hovedfunktionaliteten i systemet ligger. Backend modulet er ydermere opdelt i del moduler, hvilket vil blive vist på et følgende klasse-diagram.

Ud over de rent funktionelle moduler, kan man, ved at have fundet domænets koncepter i analysen, give designeren en idé om hvilke klasser der er nødvendige for systemet. Derfor vil læseren også kunne hæfte sig ved, at der er specificerede klasser for e-mails, vedhæftede filer, kontakter og e-mail servere.
Disse fungerer som såvel funktionelle klasser, der stiller funktionalitet, knyttet til det enkelte koncept, til rådighed for systemet, samt anvendes de tre først nævnte også som value-klasser[6], der bruges ved overførsel af data via det udviklede interface.
Dette stiller nogle krav til udformningen af disse klasser, hvilket vil blive diskuteret nærmere i Implementeringsafsnittet.

[6] En value-klasse er en klasse der indeholder en samling af ren data, således at alt data om et koncept er samlet under et.

9.2.3 Klassediagram



Figur 9.4

9.2.4 Kommentarer til klassediagram

For at give læseren en bedre forståelse af de enkelte moduler og klasser, er her en kort beskrivelse af disse.

9.2.4.1 Database interface modul

Dette modul består af en interfaceklasse, samt en række klasser der specifikt er rettet mod en bestemt database løsning.

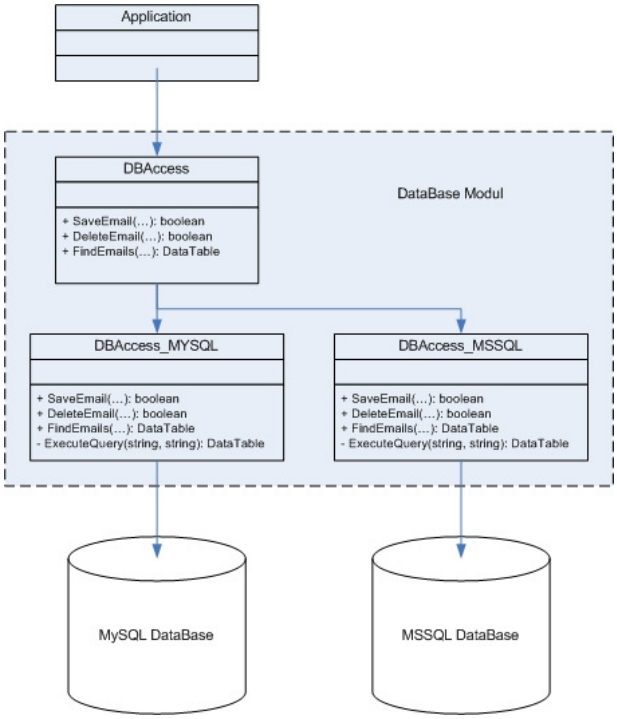
- DBAccess
  Interface klasse, der stiller funktionalitet fra de underliggende klasser til rådighed for de andre moduler. Denne klasse indeholder logik, der ud fra konfigurationsdata bestemmer hvilken underliggende klasse der skal bruges til at kommunikere med den af brugeren valgte database løsning.

- DBAccess_MYSQL
  Dataase interface klasse, der indeholder funktionalitet til at oprette forbindelse og kommunikere med en MySQL database. Her under forespørgsler, specifikt rettet mod MySQL databasen.
  Denne klasse bruger funktionalitet hentet fra MySQL.Data.dll der er nødvendig for projektet, da .Net

frameworket ikke har funktionalitet til kommunikation med en MySQL server.

- DBAccess_MSSQL
  Lige som DBAccess_MySQL klassen, indeholder denne klasse funktionalitet til at forbinde og kommunikere med en specifik database løsning, nemlig MSSQL, samt de nødvendige forespørgsler rettet mod præcist denne database løsning.

Her under ses et diagram over database modulet.



Figur 9.5

9.2.4.2 Backend modulet

Backend modulet er som det fremgår af klasse-diagrammet, det modul der binder de andre moduler sammen. Hoved funktionaliteterne ligger i denne klasse.

- EMSBackend
  Hovedklasse, hvori funktionalitet til behandling af e-mails ligger. Herunder at modtage, gemme og søge efter e-mails.

- EMSConfig
  Konfigurationsklasse der fungerer som bindeled til de forskellige konfigurationsfiler.

- EMSRecieveEmails
  Implementerer funktionalitet, således at e-mails hentes i en separat program-tråd. Dette er nærmere beskrevet i implementeringsafsnittet.

- POP3Client
  Indeholder funktionalitet til forbindelse og kommunikation med en POP3 e-mail server. Implementerer funktioner til diverse POP3 kommandoer, samt visse metoder til at forenkle funktionaliteten for andre klasser.

- EMSEmailVO
  Modsvarer konceptet *e-mail*, og indeholder funktionalitet der knytter sig til e-mails. Indeholder også lister over vedhæftede filer samt kontakter, der er knyttet til en enkelt e-mail. Fungerer også som

value-klasse.

- EMSMailboxVO
  Modsvarer konceptet *kontakt*, og indeholder funktionalitet der knytter sig til kontakter. Fungerer også som value-klasse.

- EMSAttachmentVO
  Modsvarer konceptet *vedhæftet fil*, og indeholder funktionalitet der knytter sig til vedhæftede filer. Fungerer også som value-klasse.

- EMSServerVO
  Modsvarer konceptet *e-mail server*, og indeholder funktionalitet der knytter sig til disse. Fungerer også som value-klasse.

9.2.4.3 EMSGUI

EMSGUI er modulet som giver en grafisk brugerflade, hvormed en menneskelig bruger kan benytte systemet, konfigurere dette, samt påbegynde automatisk download af e-mails fra de valgte e-mail servere. Modulet består af en række Windows-forms klasser, der interagere med hinanden, samt benytter Backend modulet, til at fortage de funktionaliteter brugeren ønsker.

9.2.4.4 EMSWebService

EMSWebService er interfacet, hvorigennem en eksternt bruger, oftest et andet system, kan tilgå e-mail data, samt benytte de søgefunktionaliteter der er i det EMS Systemet.
Dette modul består af en enkelt web-service klasse, der benytter sig af Backend modulet. Dette modul er uafhængigt af resten af systemet, og eksekveres via en web-server, som f.eks. Microsoft IIS.

9.2.5 Grafisk brugerinterface

Det grafiske brugerinterface er designet, således at det skal opfylde de krav der er stillet i kravspecifikationen, samt overholde de i analysen præsenterede use cases. Således vil det grafiske brugerinterface leve op til de funktionelle krav, og samtidig opføre sig på den ønskede måde.

Ud fra analysen og use cases er det valgt at brugerinterfacet skal bestå af et hovedvindue, samt nogle undervinduer. Således skal hovedfunktionaliteten ligge let tilgængelig i hovedvinduet, mens konfigurationsværktøjer skal ligge i undervinduer. En fremsøgt e-mail vil også skulle læses i et undervindue, således at flere e-mail kan åbnes på en gang, samt denne struktur giver et renest og mest anvendelige design.

9.2.6 System Interface

Ud fra analysen blev det klart at den enkleste, men samtidig også mest anvendelige, løsning er at implementerer systemets interface som en WebService. Dette gør at der ikke behøves at tages nogle design beslutninger, da .Net frameworket og Web-serveren giver de strukturer og den funktionalitet der skal bruges.

Af kravspecifikationen gives det, at der skal gives mulighed for fremsøgning og overførsel af e-mails, hvilket nemt opnås ved at implementerer en enkelt metode. Implementeringen af denne diskuteres i implementeringsafsnittet.

Kapitel 10

Implementering

10 Implementering

10.1 Database

Implementering af database strukturen her ikke har stillet de store udfordringer, i det strukturen, som vist på ER-diagrammet, ikke er avanceret. Det betyder samtidig at forespørgsler også er rimeligt simple. I de følgende afsnit, vil beslutninger omkring databasen og forespørgsler blive diskuteret.
Der kan findes nærmere beskrivelser af enkelte klasser, samt deres variable og metoder i appendikset.

10.1.1 Oprettelse af tabeller og data

Oprettelse af tabeller, efter strukturen fundet i tidligere kapitler, er blevet foretaget via *create* scripts i SQL. Udformningen af disse, dikteres direkte af ER-diagrammet.
Visse forholdsregler skal dog tages, i det forskellige database systemer bruger en smule forskellige versioner af SQL, samtidig med at hvert system har deres egne felt typer, som ikke altid er identiske.
F.eks. bruges der i MySQL et felt af typen *longtext*, til opbevaring af den fulde e-mail data – denne type findes ikke i MSSQL, og derfor må det være af typen *text* eller *ntext*.
De forskellige forskelle i SQL sproget skal kendes, og der er ingen vej uden om at skrive forskellige versioner til de hvert database system.
Create scripts kan findes som bilag.

En meget vigtig pointe er, at for at få referentiel integritet implementeret, som kravspecifikationen kræver, har det været nødvendigt, at oprette tabeller af typen *InnoDB* og ikke standard tabeltypen for MySQL.
Det samme gælder for transaktionsstyring, der kun er korrekt implementeret i *InnoDB* tabel-typen.

10.1.2 SQL forespørgsler

Der er, som tidligere omtalt, kun brug for forholdsvis simple SQL forespørgsler i systemet. Men disse, skal lige som *create* scripts, være tilpasset de enkelte databaser. Tre typer forespørgsler er brugt – såkaldte *INSERT-, SELECT-* samt *DELETE* statements.

INSERT og *DELETE* statements bruges til hhv. at indsætte ny data i databasen og slette eksisterende data. *SELECT* statements bruges til at udtrække data, på baggrund af parametre og giver mulighed for sorteringer af data i udtrækket.

For at få et udtræk, på netop de ønskede parametre, har det været nødvendigt at lave en forespørgsel der dynamisk bygges op, med netop de ønskede parametre. Hvis der f.eks. skal udtrækkes e-mails med en speciel afsender, bruges følgende udtræk

```
SELECT * FROM EMSEmail WHERE EmailFrom LIKE 'navn@server.com';
```

Ønsker man i stedet at udtrække e-mail med en afsender dato indenfor en bestemt periode, bruges følgende

```
SELECT * FROM EMSEmail WHERE EmailDate BETWEEN '2006-10-14' AND '2006-10-
15';
```

Det skal så være muligt at kombinere disse søgeparametre, således at der udtrækkes med begge parametre samtidig

```
SELECT * FROM EMSEmail WHERE EmailDate BETWEEN '2006-10-14' AND '2006-10-15'
AND EmailFrom LIKE 'navn@server.com';
```

Jo flere parametre der tilføjes, jo mere specifik en søgning vil der blive foretaget.

Det antages at læseren er bekendt med SQL, så derfor vil der ikke blive gået yderligere i dybden med en forklaring af syntaksen.

Oftest ses det i større projekter, at databaselaget opdeles i flere lag – med en manager klasse der varetager et specifikt område. I dette projekt er dette dog fravalgt, idet kommunikation med databasen ikke er så kompliceret, og derfor ville det give mere besvær end gavn, at dele modulet yderligere op.

10.2 Backend Modul

Backend modulet indeholder, som tidligere nævnt, funktionaliteter der er fælles for flere dele af systemet. Det er i dette modul, at funktionaliteter fra de andre moduler knyttes sammen, og her at funktionalitet til at hente e-mails fra en e-mail server ligger.

For at opnå højst mulig genbrugelighed samt mest overskuelige kode, var det vigtigt at få delt koden op i nogle mindre moduler, eller klasser.
Der er to typer af klasser i backend modulet; dataklasser og funktionalitets klasser.

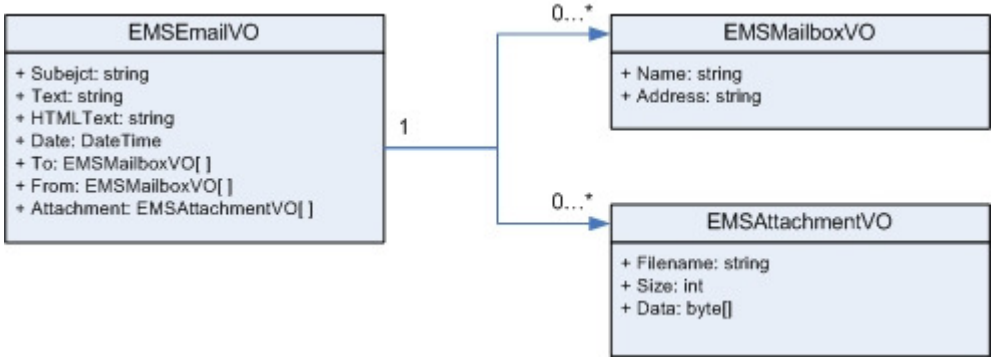
10.2.1 Dataklasser

Som det kan ses af klassediagrammet i forrige kapitel, er der lavet klasser, der indeholder data omkring et enkelt koncept, som f.eks. en e-mail.
Disse klasser indeholder dog også enkelte metoder og private variable, der bruges internt i systemet.

Dataklasserne er konstrueret således, at de kan virke som value-objekter, der kan sendes via en web-service. Når klasser sendes via en web-service, vil alle metoder, samt private variable blive 'glemt', og kun offentlige variable vil blive medtaget.
Dette er præcis det der bliver udnyttet, således at klassen, efter at være blevet overført via webservicen, kun indeholder data omkring en e-mail. Det giver brugeren mulighed for at læse indholdet af klassens vigtige variable, uden at denne forvirres af ubetydeligt data og unødvendige metoder.

For at holde styr på en e-mails vedhæftede filer og modtagere, er disse lagt i arrays af dataklasser, under e-mail dataklassen. Således kan denne struktur nemlig også overføres via web-servicen, og brugeren får en struktur der er nem at overskue.

Strukturen for e-mails er som følger.



Figur 10.1

10.2.2 Funktionsklasser

De nødvendige funktionaliteter kan deles op i 3 dele.

1. At hente og parse e-mails
2. At gemme, slette og søge efter e-mails i lageret
3. At opbevare og opdaterer brugerindstillinger

10.2.2.1 Download og parsing af E-mails

Funktionalitet omkring at hente e-mails blev implementeret i klassen *POP3Client*. Implementeringen af denne er rimeligt simpel. Der er først og fremmest brug for at kunne oprette en forbindelse til e-mail serveren. Denne forbindelse oprettes ved hjælp af den i .Net indbyggede *TCPClient* klasse og et *Stream* objekt.
Her ud over er der implementeret metoder, der afsender POP3 kommandoer via den før omtalte forbindelse, og eventuelt, vha. hjælpemetoder, læser og returnere svaret fra serveren.
Enkelte metoder bygger oven på de rene POP3 kommando metoder, således at der laves et lidt mere brugervenligt interface.

Parsing af e-mails blev ikke implementeret i dette system, da der ikke var nok tid. I stedet blev der brugt en komponent *Mail.dll* der kunne gøre dette. Komponenten er udviklet af Pavel Lesnikowski. Denne komponent er inkluderet i projektet som en midlertidig løsning, men fungerer udmærket.
Komponenten indeholder også metoder til forbindelse til e-mail servere, men denne del bliver ikke benyttet.

For at systemet skulle kunne hente e-mails med automatisk, som der ønskes i kravspecifikationen, blev det besluttet, at denne funktionalitet skulle implementeres i en separat tråd.
Således kan brugeren af systemet benytte sig af det grafiske brugerinterface samtidig med at systemet henter e-mails fra serveren, og det forenkler også automatiseringen. Tråden kan sættes på standby i et tidsrum, hvilket medvirker, at systemet kun henter e-mails fra serveren med et vist mellemrum. Dette mellemrum kan så defineres af brugeren, og brugeren kan også vælge helt at suspendere automatisk download, hvorved tråden simpelthen nedlægges.
Her er det dog valgt at implementere det således, at brugeren ikke kan afbryde download af e-mails midt i processen, da dette vil kunne give uhensigtsmæssige fejl og ukonsistent data. I stedet venter systemet med at afbryde download til alle e-mails er hentet, for derefter at stoppe automatisk download.

En del af POP3 protokollens sekvens for korrekt dialog med e-mail serveren, er ikke implementeret i *POP3Client* klassen – hvilket ville have været mest hensigtsmæssigt. Der har dog ikke været nok tid, til at få denne detalje på plads.

10.2.2.2 Søgning og lagring af e-mails

Søgning og lagring af e-mails sker gennem kald til metoder *DBAccess* klassen. Herfra returneres e-mail data fra søgningen, der bliver konverteret til klasse strukturen, som kan ses på figur 10.1.

10.2.2.3 System konfiguration

For at få et system der er anvendeligt, er det nødvendigt for en bruger at kunne konfigurere systemet, dette er også nævnt i kravspecifikationen. Derfor er der blevet udviklet en klasse til at håndtere brugerkonfigurations data.
Opbevaring af brugerdata gøres ved at gemme denne data i XML filer, således at data er gemt på en overskuelig måde og på en måde hvorpå de kan hentes igen efter system genstart.

XML filen er bygget op af to sektioner: En sektion med konfigurationer omkring forskellige ting i systemet, som f.eks. hvor længe der skal gå mellem automatisk download af e-mails fra e-mail serveren. Denne sektion indeholder et fast antal noder eller konfigurationsfelter, hvori data kan ændres efter brugerens behov.

Den anden del, er sektionen for e-mail servere. Her Kan der tilføjes og fjernes data omkring servere, der ønskes at hente e-mails fra. Sektionen varierer altså i antal af noder.

EMSConfig klassen indeholder metoder til at finde og opdatere noder i konfigurationsfilen, samt stiller konfigurationsdata til rådighed for resten af systemet, gennem offentlige variable.

10.3 Database klasser

Database klasserne er systemets interface til databasen. Disse indeholder en række metoder til at udtrække, slette, opdatere og indsætte data i databasen.
Som det nævnes i design-kapitlet, er database interfacet delt op i 3 dele.

1. En interface klasse, der ud fra brugerens konfiguration af systemet, finder ud af hvilken underklasse der skal benyttes, for at kommunikere med den valgte database type.
2. En klasse til kommunikation med en MySQL database
3. En klasse til kommunikation med en MSSSQL database

10.3.1 Interface klassen

Ved hjælp af konfigurations modulet, ses det hvilken database type, som brugeren har valgt at koble op i mod. Som før omtalt, er der forskel på opkobling samt kommunikation med forskellige database løsninger, og derfor er der blevet udviklet forskellige klasser til kommunikation med disse.
Interface klassen sender simpelthen det specifikke metodekald videre til den passende databaseklasse, hvilket gør, at systemet kan kobles op i mod flere typer af database løsning.
Hvis det var nødvendigt, kunne man her udvide systemet, til at indeholde flere database-specifikke klasser, således, at systemet understøttede endnu flere databaseløsninger.

10.3.2 De database-specifikke klasser

Microsoft .Net har moduler til understøttelse af kommunikation med MSSQL databaser indbygget – disse kan nemt inkluderes i projektet, og bruges uden større problemer. Ved udtræk af data bruges en *SqlAdapter* klasse, der letter synkronisering af data mellem database og system. Denne står selv for at oprette forbindelse til databasen, hente data ud fra en angiven SQL forespørgsel, og returnerer en tabel med data, som senere kan læses.
Ved andre operationer, der ikke returnere data, er det enklere at varetage forbindelse og udførelsen af SQL forespørgslen selv. Det gøres via et *SqlConnection* og *SqlCommand* objekt.

Transaktionsstyring har ikke været nødvendig i dette projekt, da der kun arbejdes på en tabel af gangen, og der kun foretages en enkelt SQL forespørgsel per session. Dog kunne man forestille sig den tænkte udvidelse af databasen (se 9.1.3), hvor der skulle oprettes rækker i flere tabeller, hver gang en e-mail skulle gemmes. For at undgå ukonsistent data, ville der her være brug for at systemet implementerede transaktionsstyring.

10.4 WebService klassen

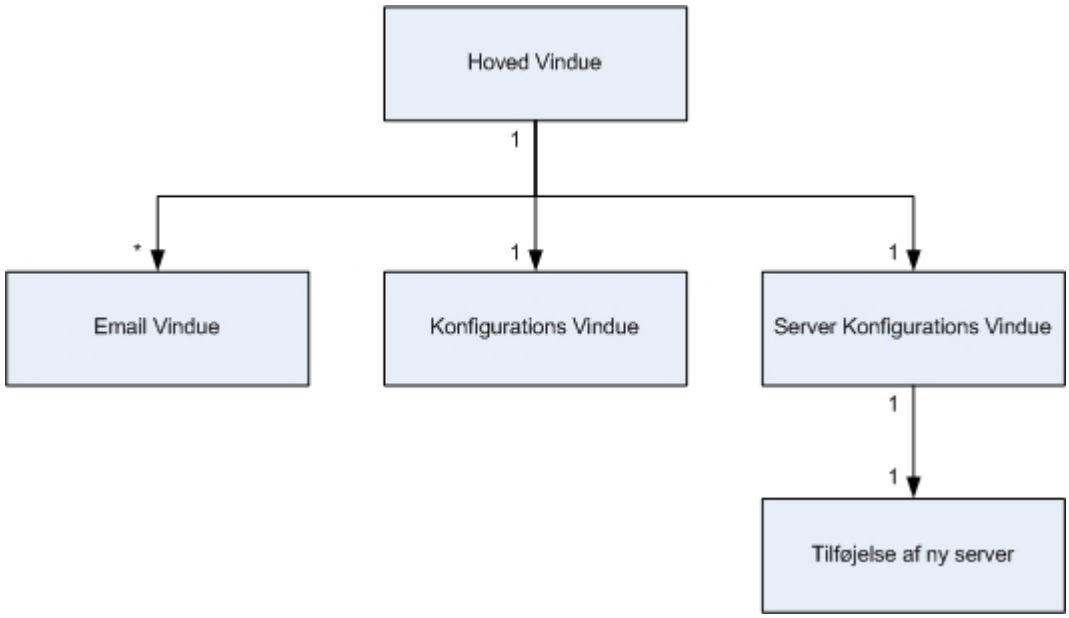
WebServic'en er meget enkelt bygget op. Den indeholder nogle enkelte metoder, der igen kalder passende metoder i Backend modulet, og returnere data der returneres her fra.
En enkelt udvidelse til disse metoder er, at der sørges for at en tom søgestreng bliver konverteret til en streng indeholdende et *wildcard*, hvilket betyder at søgningen vil fremsøge alt i stedet for intet, ved tomme søgestrenge. Dette er dog kun gældende for felter af typen *string,* da det ikke giver nogen mening at sende en tom Boolean eller DataTime.

10.5 Grafisk Bruger Interface

Det grafiske brugerinterface er implementeret, så det giver en let og overskuelig tilgang til systemet.
Der er lagt vægt på, at de mest brugte funktioner ligger nemt tilgængelige, og konfigurationsdata gemmes væk – da det forventes at disse ikke bruges ofte.

For at få en brugervanlig struktur, er brugerinterfacet bygget op af vinduer. Disse grupperer funktionalitet i samme kategori, og giver mulighed for et mere brugervenligt og overskueligt design.

Det grafiske brugerinterface er bygget således op.



Figur 10.2

10.5.1 Hoved vinduet

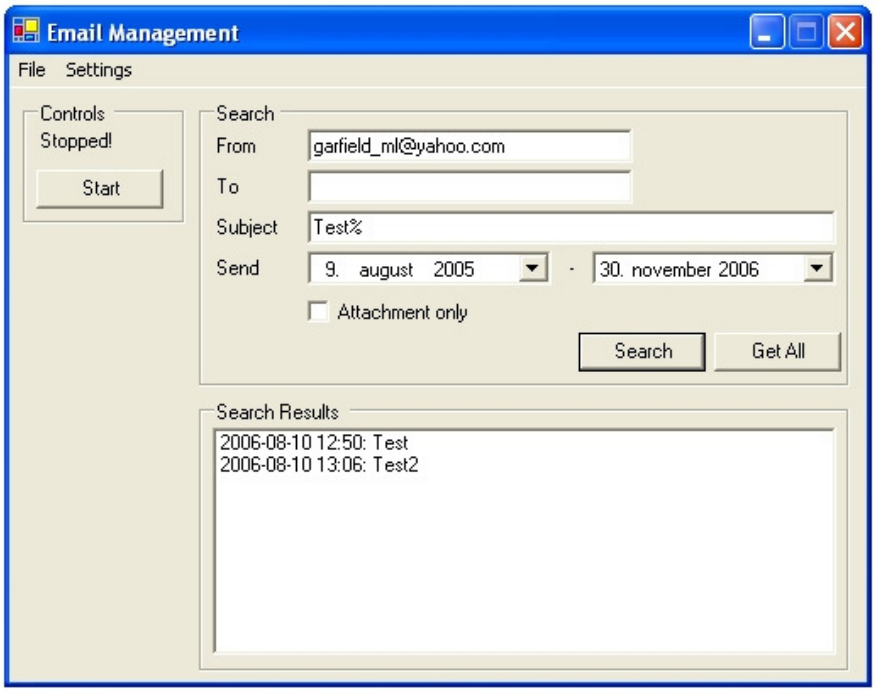
Hoved vinduet indeholder to del funktionaliteter. Den første og vigtigste er muligheden for at starte og stoppe automatisk download af e-mails. Dette er forenklet til en enkelt knap og en label, der fortæller om status for denne funktionalitet. Der er ikke brug for yderligere muligheder for brugerinteraktion her, og da der også er tale om en prototype af systemet, er denne løsning valgt, for at forenkle processen.

Her ud over er der funktionalitet til at søge efter e-mails i systemet. Denne funktionalitet er inkluderet, således at en bruger kan få et hurtigt overblik over hvad der ligger i systemet, om der er nogen e-mails der ikke er blevet hentet videre ind i forretningens systemer, eller simpelthen bruges som en simpel mail klient. Igen er dette brugerinterface ret enkelt, og kunne bruge nogle ekstra muligheder, men det stiller basis funktionaliteter til rådighed, på en overskuelig måde.

Når en søgning er blevet foretaget, vil e-mails der passer på søgekriterierne blive listet i feltet "*search results*". Herefter kan brugeren klikke på en ønsket e-mail i denne boks, og han vil blive præsenteret for et nyt vindue med mere detaljerede oplysninger og indhold af e-mailen.

Det er muligt a bruge *wildcards* i disse søgefelter også. Således kan der søges på delstrenge i *from, to* og *subject* felterne.
Dato felterne er lavet vha. .nets indbyggede kalender systemer, der på en nem og overskuelig måde, giver brugeren mulighed for at vælge dato. Læg i øvrigt mærke til, at datoformatet i disse bokse automatisk tilpasses Windows datoformatet.

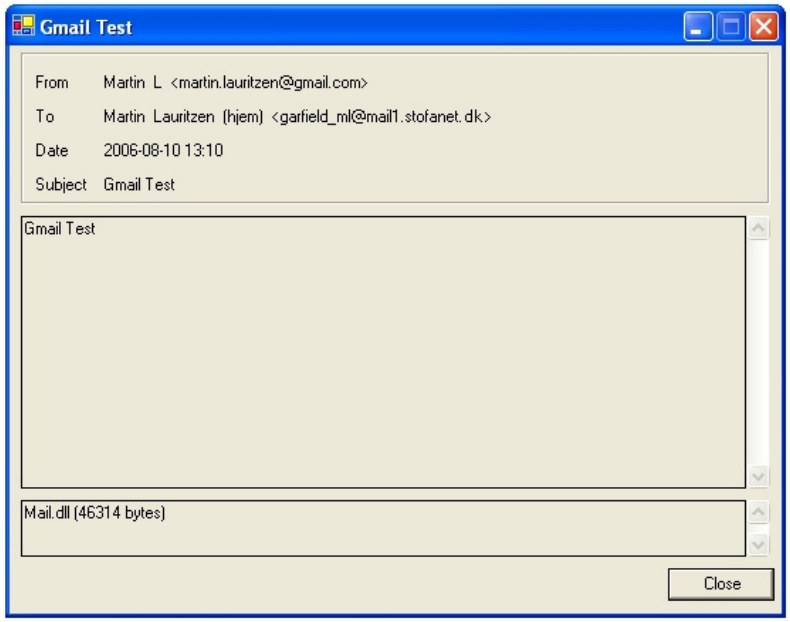
Her under ses et screenshot af hoved vinduet.



Figur 10.3

10.5.2 E-mail vindue

E-mail vinduet viser en e-mails indhold, som en e-mail klient normalt ville. Det er forsøgt byget op nogenlunde som de fleste andre e-mail klienter, da de fleste brugere er vant til at benytte eksisterende systemer. Dette er dog som før omtalt en prototype, og det er f.eks. ikke muligt at se indholdet af en HTML formateret e-mail korrekt og ej heller gemme vedhæftede filer på sin maskine.

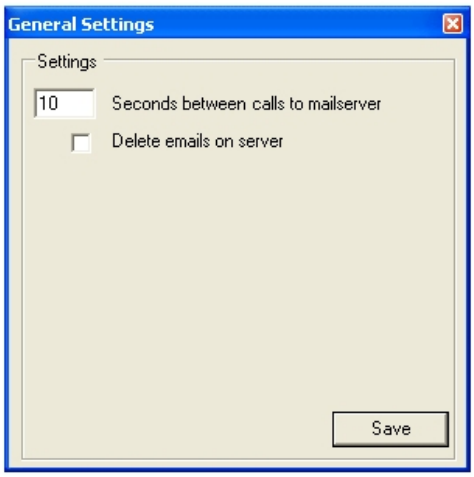


Figur 10.4

10.5.3 Konfigurationsvindue

Dette vindue giver brugeren mulighed for at ændre på systemets konfiguration. Indtil videre er disse rimeligt få, men man kunne tænke sig at dette vindue kunne udvides, hvis flere funktionaliteter tilføjes.
Der er lavet sikkerhed, således at brugeren ikke kan komme til, at indtaste en ugyldig værdi for antallet af sekunder mellem hver forbindelse til e-mail serverne, under automatisk download af e-mails.

Dette vindue er beskyttet af et *singleton pattern*, således at det kun er muligt at åbne en enkelt instans af vinduet. Således giver det en vis beskyttelse mod, at konfigurationsdata bliver opdateret forkert.

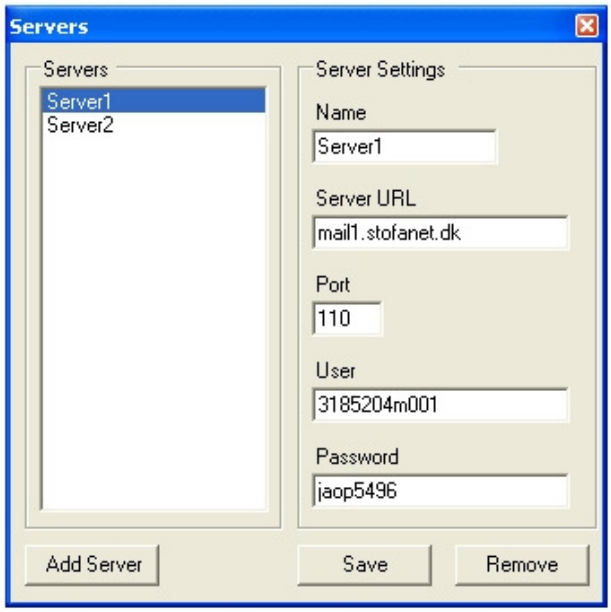


Figur 10.5

10.5.4 Server konfigurationsvindue

Server konfigurationsvinduet giver mulighed for at se opsatte e-mail servere, samt ændre i opsætningerne. Vinduet er enkelt bygget op, med en liste til højre, samt felter til mere specifik information om konfigurationen til en valgt e-mail server.
Ydermere er det muligt at tilføje servere til listen, hvilket dog ikke er blevet færdig implementeret, og det er derfor nødvendigt at tilføje servere manuelt i konfigurationsfilen.

Ligesom Konfigurationsvinduet, er dette vindue beskyttet af et *singleton pattern*.



Figur 10.6

Kapitel 11

Tests

11 Tests

11.1 Indledning

Test er blevet foretaget af undertegnede samt følgende frivillige testere.

- Lærke Thorndal-Debes
- Thomas Lauritzen
- Kasper Storm

Alle har gennemført samtlige tests, med undtagelse af Database Tests, der kun er foretaget af undertegnede.

11.2 Email konti

Følgende e-mail konti blev benyttet under de følgende tests. Nævnes der ikke en specifik e-mail adresse i test scriptet, er testen løbet igennem for alle e-mail konti.

Afsender #1:
En standard Yahoo WebMail konto
Garfield_ml@yahoo.com
Webinterface blev benyttet til at oprette og afsende e-mails.

Afsender #2:
En standard Hotmail WebMail konto
Garfield_ml@hotmail.com
Webinterface blev benyttet til at oprette og afsende e-mails.

Afsender #3:
Stofanet POP3 MailServer
Navn: Garfield_ml@stofanet.dk
Server: mail1.stofanet.dk
Port: 110
Brugernavn: 3185204m001
Password: ******
MS Outlook Express blev benyttet til at oprette og afsende e-mails.

Modtager #1:
Stofanet POP3 MailServer
Navn: Garfield_ml@stofanet.dk
Server: mail1.stofanet.dk
Port: 110
Brugernavn: 3185204m001
Password: ******

11.3 Database

Følgende database systemer blev benyttet under tests. Tests hvor der ikke specifikt er noteret hvilken database der er brugt, er gennemløbet med alle typer. Disse databaser kører lokalt på samme maskine som EMSSystemet, med mindre andet er noteret.

- MySQL Server Database version 4.0.20
- MSSQL Server version 2002

11.4 Test System

For at teste System Interfacet, er der udviklet et simpelt test system. Dette benytter EMS Systemets WebService til at trække e-mails ud fra databasen. Herefter vil e-mails blive udskrevet til skærmen og eventuelle vedhæftede filer gemt på harddisken.
Dette test system afvikles på samme maskine som EMS Systemet med mindre andet er angivet i de specifikke tests.

11.5 Tests

For at sikre, at data bliver gemt korrekt er det nødvendigt at teste nogle forskellige scenarier. Her under vil der blive fortaget nogle tests, der viser om databasen kan håndtere data korrekt.

De følgende tests bliver foretaget på en Campaq Presarion 2100 Laptop, med Windows XP SP2.
Følgende software er installeret.

- .Net framework 1.1 og 2.0
- MySQL Database
- AVG Free Edition Virus Scanner
- MS Visual Studio .Net

11.5.1 E-mails med store data mængder

11.5.1.1 Formål

En e-mail kan indeholde flere vedhæftede filer, og ofte op til ca. 10 MB. Det er vigtigt at databasen kan håndtere data, i denne størrelses orden.
Ifølge MySQL dokumentationen, kan et felt af typen *LongText* indeholde 2^32 + 4 bytes, hvilket svarer til ca. 4 gigabytes. Teoretisk skulle der derfor ikke være nogen problemer med at gemme data på op til 10 MB.

For at teste det, blev der sendt en stor mail til en e-mail adresse, hvor systemet skulle hente den ned fra, og gemme den korrekt. Der efter skulle mailen kunne læses korrekt via såvel GUI'et samt et eksternt test program, der via System Interfacet udtrak e-mailen og gemte dens vedhæftede filer på harddisken.

11.5.1.2 Bemærkninger:

Ingen

11.5.1.3 Testscript:

1. Opret e-mail, vedhæft minimum to filer med en samlet størrelse på over 5 MB. Afsend denne e-mail til modtageradresse #1.
2. Tjek at mail er kommet frem til modtageradresse, via eksternt software. Tjek at størrelse af e-mailen passer.
3. Download mails med EMS Systemet.
4. Tjek at mail er gemt i test databasen.
5. Start test system, og hent e-mails via system interface
6. Tjek at mail udskrives korrekt til skærmen
7. Tjek at vedhæftede filer er gemt korrekt på harddisken.
8. Åben vedhæftede filer i passende program
9. Tjek at vedhæftede filer er identiske med filer der i første omgang blev vedhæftet filen.

Testen betegnes som succes, hvis alle skridt gennemføres korrekt.

11.5.1.4 Første gennemløb af testen.

1. En mail blev sendt til en af de i systemet opsatte e-mail adresser. Mailen indeholder tekst + 2 jpeg billeder af hver ca. 3 MB.
2. Mailen kom frem til e-mail adressen, hvor det blev tjekket at den havde den korrekte størrelse (ca. 6 MB).
3. Systemet blev sat til at hente mails.
4. Mailen blev downloadet af systemet og blev forsøgt gemt i databasen.
5. MySQL Database fejler.

MySQL test databasen fejler, da den kun er konfigureret til at kunne håndtere pakker på op til 1 MB per transaktion.

11.5.1.5 Andet gennemløb.

Indstillingen for maksimal pakke størrelse for MySQL ændres til 20 MB.

1. En mail blev sendt til en af de i systemet opsatte e-mail adresser. Mailen indeholder tekst + 2 jpeg billeder af hver ca. 2 MB.
2. Mailen kom frem til e-mail adressen, hvor det blev tjekket at den havde den korrekte størrelse (ca. 4 MB).
3. Systemet blev sat til at hente mails.
4. Mailen downloades af systemet og gemmes i databasen.
5. Test System forbinder til Systemet og henter e-mail via System Interface.
6. E-mail udskrives korrekt til skærmen.
7. Vedhæftede filer gemmes på harddisken.
8. Disse fremfindes manuelt, og åbnes i Windows Picture and Fax Viewer.
9. Billederne er identiske med dem, der blev vedhæftet e-mailen i første omgang.
10. Testen slut.

11.5.1.6 Resultat

Andet gennemløb af testen endte med succes.
Testen endte derfor i succes.

11.5.2 HTML e-mails

11.5.2.1 Formål

At teste at e-mails indeholdende HTML i brødteksten bliver downloadet, gemt og sendt korrekt via System Interfacet.

11.5.2.2 Bemærkninger

Følgende brødtekst skal bruges til test e-mailen

```
<html>
<head>
<style type="text/css">
<!-- DIV {margin:0px;} -->
</style>
</head>
<body>
<div style="font-family:times new roman, new york, times, serif;font-size:12pt">
<DIV>Dette er en test e-mail med html kode:</DIV>
<DIV><STRONG>Denne tekst er fed</STRONG></DIV>
<DIV><EM>Denne tekst er kursiv</EM></DIV>
<DIV>Her er nogle punkter:</DIV>
<UL>
<LI>1</LI>
<LI>2</LI>
<LI>3</LI>
<LI>4</LI></UL>
<P><U>Slut</U></P>
</div>
<br>
</body>
</html>
```

11.5.2.3 Test Script

1. Opret e-mail, indeholdende HTML i brødteksten. Brødtekst ses oven for.
   Afsend e-mail til modtager adresse #1.
2. Tjek at e-mail er kommet frem.
3. Download mails med EMS Systemet.
4. Tjek at e-mail er gemt i test databasen.
5. Tjek at e-mail vises korrekt i det EMS Systemets GUI.
6. Start test system, og hent e-mails via system interface
7. Tjek at mail udskrives korrekt til skærmen

Testen betegnes som succes, hvis alle skridt gennemføres korrekt.

11.5.2.4 Første gennemløb

1. E-mail skrives og afsendes
2. Mailen kommer frem til modtager adresse.
3. System startes og sættes til at hente e-mails.
4. E-mail er herefter gemt i test databasen
5. Der søges efter e-mail i systemet, denne findes, og læses korrekt via systemets GUI
6. Test system startes, og henter e-mails.
7. E-mail udskrives korrekt til skærmen.
8. Test slut.

11.5.2.5 Resultat

Testen forløb som forventet, uden problemer. E-mail blev downloadet og vist korrekt i EMS Systemet egen GUI, samt hentet og vist korrekt af test systemet.

11.5.3 Fremsøgning af e-mails via GUI

11.5.3.1 Formål

At teste, at søgefunktionaliteten via EMS Systemets GUI fungerer som forventet.
Denne test indeholder flere testscripts.

11.5.3.2 Bemærkninger

For at teste denne funktionalitet, er det nødvendigt at have nogle e-mails liggende i databasen. Disse skal have en passende variation i afsender, modtager, dato og emne, for at kunne teste korrekt. E-mails med følgende indhold skal ligge i systemet.

```
Email #1
Afsender:     hans@hotmail.com
Modtager:     info@minserver.com
Dato:         2006-01-01
Emne:         Test mail nummer 1

Email #2
Afsender:     peter@gmail.com
Modtager:     kundeservice@server.firma.dk
Dato:         2005-12-31
Emne:         Test e-mail nummer 2

Email #3
Afsender:     julie_11@hotmail.com
Modtager:     info@minserver.com
Dato:         2006-04-12
Emne:         E-mail test #3

Email #4
Afsender:     peter@gmail.com
Modtager:     info@minserver.com
Dato:         2006-01-02
Emne:         Dette er den 4. e-mail

Email #5
Afsender:     mark@hotmail.com
Modtager:     kundeservice@minserver.com
Dato:         2004-11-02
Emne:         Wqæøå
```

11.5.3.3 Test Script #1

1. Indtast følgende søgeparametre i EMS Systemets søgefelter:
   Afsender: ”peter@gmail.com”
   Start søgning
2. Tjek at søgning returnerer det forventede

Forventet resultat:
Alle e-mails med den specifikke streng ”hans@hotmail.com” i afsender feltet findes.
E-mail: #2 og #4

11.5.3.4 Test Script #2

1. Indtast følgende søgeparametre i EMS Systemets søgefelter:
   Modtager: ”%@minserver.com”
   Start søgning
2. Tjek at søgning returnerer det forventede

Forventet resultat:
Alle e-mails med del-strengen ”@minserver.com” til sidst i modtager feltet findes.
E-mail: #1, #3, #4 og #5

11.5.3.5 Test Script #3

1. Indtast følgende søgeparametre i EMS Systemets søgefelter:
   Dato start: ”2006-01-01”

Dato slut: ”2006-12-31”
Start søgning
2. Tjek at søgning returnerer det forventede

Forventet resultat:
Alle e-mails med en dato indenfor år 2006 findes.
E-mail: #1, #3 og #4

11.5.3.6 Test Script #4

1. Indtast følgende søgeparametre i EMS Systemets søgefelter:
   Subject: ”%e-mail%”
   Start søgning
2. Tjek at søgning returnerer det forventede

Forventet resultat:
Alle e-mails med del-strengen ”e-mail” i emne feltet findes.
E-mail: #2, #3 og #4

11.5.3.7 Test Script #5

1. Indtast følgende søgeparametre i EMS Systemets søgefelter:
   Dato start: ”2006-01-01”
   Dato slut: ”2006-12-31”
2. Subject: ”%e-mail%”
   Start søgning
3. Tjek at søgning returnerer det forventede

Forventet resultat:
Alle e-mails med del-strengen ”e-mail” i emne feltet, samt en dato indenfor år 2006 findes.
E-mail: #3 og #4

11.5.3.8 Første gennemløb

Her gennemløbes alle testscripts på samme måde.

1. Data indtastes i indtastningsfelt, søgning startes
2. Resultat afstemmes med det ønskede
3. Test slut

11.5.3.9 Resultat

Resultatet var ens for alle testscripts, de fremsøgte e-mails passede overens med det ønskede resultat. Testen er derfor en succes.

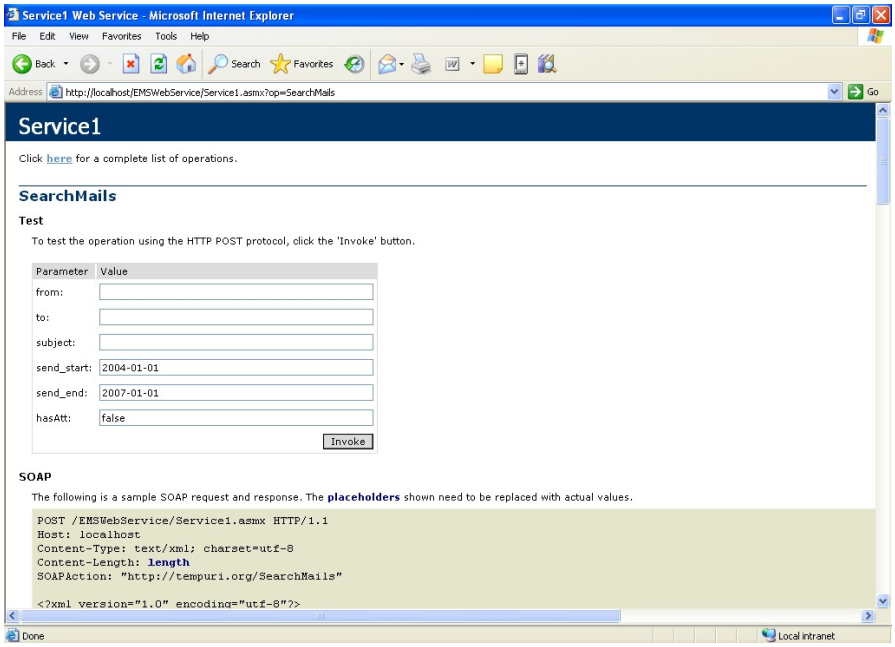
11.5.4 Fremsøgning af e-mails via System Interface

Denne test er næsten lig testen *Fremsøgning af e-mails via GUI*, i det at de to interfaces benytter den samme underliggende funktionalitet. Derfor vil der ikke blive gået i detaljer med denne test.
Der blev foretaget de samme indtastninger i som i den før omtalte test, blot blev System Interfacet direkte benyttet.
Dette kan gøres vha. en standard *browser*[7], da enhver WebService i .net har indbygget et standard Web Interface. Se screenshot neden for.

[7] En *browser* er et program der bruges til at kunne benytte internetsider. Eksempler på browsere er MS Internet Explorer, Firefox og Opera.

Her skal det dog bemærkes, at der er mulighed for at lave fejlindtastninger i datofelterne, da indtastningen skal være formateret som en korrekt dato. Der er ikke taget højde for dette i teksten, da det standard web-interfacet kun forventes benyttet til tests.

Resultatet på testen blev som forventet, da alle søgninger resulterede i de forventede e-mails.
Testen er derfor succesfuld.



Figur 11.1

11.5.5 Distribueret system

11.5.5.1 Formål

At teste at EMS Systemet fungerer korrekt, når det er distribueret over flere servere. Samt at et eksternt system, fra en ekstern server, kan tilgå EMS Systemets system interface.

11.5.5.2 Bemærkninger

Test databasen er ændret til følgende:

MySQL version 4.1.11
Server: db.fubunet.dk
Database: web90792
Bruger: web90792
Password: ******

EMS Systemet eksekveres stadig på tidligere omtalte maskine (IP: 10.1.0.3).

Test system eksekveres på en maskine på samme lokalnetværk som EMS Systemets maskine (IP: 10.1.0.4). Lokalnetværket er et standard Microsoft netværk, der kører via TCP/IP protokollen. Netværket er trådløst, og er forbundet via en trådløs D-Link router – der er ikke opsat intern *firewall*[8] i netværket.

11.5.5.3 Test Script

[8] På dansk også kaldet "brandmur".

1. Eksekver test system på ekstern maskine.
2. Fremsøg e-mails via test system og EMS Systemets system interface
3. Tjek om e-mails udskrives korrekt til skærmen
4. Tjek at vedhæftede filer gemmes korrekt på den lokale harddisk

11.5.5.4 Første gennemløb

1. Test system eksekveres på test maskine
2. E-mails hentes
3. E-mails udskrives til skærmen
4. Vedhæftede filer er identiske med det forventede
5. Test slut

11.5.5.5 Resultat

Testen udføres som forventet. Der opstår ingen netværksfejl, eller korrupt data.
Testen er en succes.

11.5.6 Ingen forbindelse til database

11.5.6.1 Formål

At teste hvordan systemet opfører sig, hvis forbindelsen til databasen forsvinder.

11.5.6.2 Bemærkninger

Det ønskes at der skal opstå en fejlmeddelelse i en popup boks, der fortæller brugeren at der er opstået en kritisk fejl, og en beskrivelse af fejlen.

Der er to test script til denne test. Script #1 omhandler e-mail søgning og script #2 omhandler automatisk download af e-mails.

11.5.6.3 Test Script #1

1. EMS Systemet startes som normalt
2. Forbindelse til database afbrydes
3. Der fortages en søgning af e-mails
4. Fejlbesked ventes at fremkomme

11.5.6.4 Test Script #2

1. EMS Systemet startes som normalt
2. Automatisk download startes
3. Forbindelse til database afbrydes
4. Fejlbesked ventes at fremkomme kort efter

11.5.6.5 Første gennemløb af Script #1

1. EMS Systemet startes som normalt
2. Test Databasen lukkes ned, så den ikke længere er tilgængelig
3. Søgning foretages
4. Fejlbesked opstår

11.5.6.6 Resultat af script #1

Systemet opførte sig som forventet, der opstod en fejl, som blev vist for brugeren vha. en fejlbesked i en popup boks.

11.5.6.7 Første gennemløb af Script #2

1. EMS Systemet startes som normalt
2. Automatisk download starter
3. Test Databasen lukkes ned, så den ikke længere er tilgængelig
4. Fejlbesked opstår ikke. Programmet fortsætter ufortrødent.

11.5.6.8 Resultat af script #2

Systemet fejler, i det det ikke fortæller brugeren, at der er opstået en fejl i forbindelsen til databasen. For brugeren ser alting normalt ud.
Dette er en kritisk fejl der skal rettes, test ikke succesfuld.

11.5.7 Indtastning af ugyldigt data i GUI

11.5.7.1 Formål

At teste at det grafiske bruger interface opføre sig som ventet ved indtastning af ugyldig data.

11.5.7.2 Bemærkninger

Der testes både i e-mail søgning, samt indtastningsfelter i konfigurations indstillings vinduerne.

11.5.7.3 Test Script #1

Dette script udføres på e-mail søgning, i felterne *from*, *to* og *subject*.

Indtastet test data	Resultat	Bestået?
info@server.com	Foretager søgning	OK
Mit emne	Foretager søgning	OK
1000	Foretager søgning	OK
-1	Foretager søgning	OK
%Test%	Foretager søgning	OK
756.04	Foretager søgning	OK
W¤uu(){	Foretager søgning	OK
*	Foretager søgning	OK
’ 1=1	Foretager søgning	OK
\” 1=1	Foretager søgning	OK
200 hundrede a’er	Kan ikke indtastes, højst 50 karakterer (100 for subject)	OK
Tomt felt	Foretager søgning	OK

11.5.7.4 Test Script #2

Dette script udføres på e-mail søgning, i feltet *seconds between calls to mailsservers*.

Indtastet test data	Resultat	Bestået?
5	Gemmer data	OK
9999	Gemmer data	OK
1	Gemmer data	OK
0	Udskriver fejlbesked	OK
-1	Udskriver fejlbesked	OK
1000001	Kan ikke indtastes, højst 4	OK

	karakterer	
7.04	Udskriver fejlbesked	OK
aagg	Udskriver fejlbesked	OK
*	Udskriver fejlbesked	OK
1+1	Udskriver fejlbesked	OK
Tomt felt	Udskriver fejlbesked	OK

11.5.8 Pålidelighed

11.5.8.1 Formål

At teste at systemet opføre sig korrekt over en længere tidsperiode.

11.5.8.2 Bemærkninger

Denne test blev udført over en periode på godt 12 timer, og med forskellige e-mail konti som modtager og afsender.

11.5.8.3 Test Script

1. Start automatisk download af e-mails
2. Tjek at systemet henter og gemmer e-mails i databasen
3. Vent ½ time
4. Send e-mail til en af de konfigurerede e-mail konti
5. Tjek at systemet henter e-mailen
6. Gentag 4 og 5 et passende antal gange.

11.5.8.4 Første gennemløb

1. Automatisk download startet
2. Systemet henter e-mails som forventet
3. –
4. E-mail afsendes til en af de i systemet konfigurerede e-mail konti
5. E-mailen hentes korrekt af systemet, efter at denne er kommet frem
6. Step 4 og 5 gentaget med pauser på ca. ½ time.
7. Test slut

11.5.8.5 Resultat

Testen resulterede i succes, da systemet opførte sig som det skulle.

11.5.9 Kapacitet

11.5.9.1 Formål

At systemet kan håndtere en e-mail konti indeholdende mange e-mails.

11.5.9.2 Bemærkninger

Systemet blev testet for mængder af normale e-mails. Der er i en tidligere test blevet testet for e-mails indeholdende store mængder data.

11.5.9.3 Test script

1. Send et passende antal mail, til en i systemet konfigurerede e-mail konto.
2. Start download.
3. Tjek at systemet downloader og gemmer e-mails korrekt.

11.5.9.4 Første gennemløb

1. Der sendes 200 e-mails til en i system konfigureret e-mail konto.
2. Automatisk download startes
3. Systemet påbegynder download
4. Efter lidt tid er alle e-mails hentet korrekt.
5. Test slut

11.5.9.5 Resultat

Systemet opførte sig som forventet, de mange e-mails tager lidt tid at hente, men det er kun normalt. Testen endte med succes.

11.6 Tests af Use Cases

Følgende tests er blevet foretaget, for at sikre at use cases bliver fulgt af systemet.

Test 1	
Test type:	Use-case test
Test af:	1 – Opsætte en e-mail server
Dato for oprettelse:	15.09.2006
Dato for udførsel:	10.10.2006
Fortaget af:	Martin Lauritzen
Test forløb: Det har ikke været muligt at foretage denne test, da den nødvendige funktionalitet ikke er blevet implementeret.	
Test konklusion:	

Test 2	
Test type:	Use-case test
Test af:	2 – Ændre systemkonfigurationen
Dato for oprettelse:	15.09.2006
Dato for udførsel:	10.10.2006
Fortaget af:	Martin Lauritzen, Lærke Thorndal-Debes og Thomas Lauritzen
Test forløb: Vi har gennemgået de forskellige trin i use-casen, og repliceret de, i use-casen, beskrevne fejl i programmet. Alt lod til at virke, og trinene blev fulgt tilfredsstillende.	
Test konklusion: Testen endte i succes, da konfigurationen blev opdateret korrekt.	

Test 3

Test type:	Use-case test
Test af:	3 – Starte automatisk download
Dato for oprettelse:	10.09.2006
Dato for udførsel:	11.10.2006
Fortaget af:	Martin Lauritzen og Kasper Storm

Test forløb:
Vi har gennemgået de forskellige trin i use-casen, og repliceret de, i use-casen, beskrevne fejl i programmet. Step 5a og 5b blev ikke udført korrekt. Brugeren gøres ikke opmærksom på disse fejl.

Test konklusion:
Testen fejler, i det step 5a og 5b ikke udføres korrekt. Dette er ikke en kritisk funktionel fejl, men bør rettes i en fremtidig opdatering.

Test 4

Test type:	Use-case test
Test af:	4 – Stoppe automatisk download
Dato for oprettelse:	10.09.2006
Dato for udførsel:	11.10.2006
Fortaget af:	Martin Lauritzen og Kasper Storm

Test forløb:
Vi har gennemgået de forskellige trin i use-casen, og repliceret de, i use-casen, beskrevne fejl i programmet. Step 4a blev ikke udført korrekt. Brugeren gøres ikke opmærksom på denne fejl.

Test konklusion:
Testen fejler, i det step 5a ikke udføres korrekt. Dette er ikke en kritisk funktionel fejl, men bør rettes i en fremtidig opdatering.

Test 5

Test type:	Use-case test
Test af:	5 – E-mail søgning via GUI
Dato for oprettelse:	12.09.2006
Dato for udførsel:	09.10.2006
Fortaget af:	Martin Lauritzen og Lærke Thorndal-Debes

Test forløb:
Vi har gennemgået de forskellige trin i use-casen, og repliceret de, i use-casen, beskrevne fejl i programmet. Alt lod til at virke, og trinene blev fulgt tilfredsstillende.

Test konklusion:
Testen endte i succes, da søgningen blev foretaget korrekt.

Test 6

Test type:	Use-case test
Test af:	6 – E-mail søgning via System Interface
Dato for oprettelse:	12.09.2006
Dato for udførsel:	09.10.2006
Fortaget af:	Martin Lauritzen og Lærke Thorndal-Debes

Test forløb:
Vi har gennemgået de forskellige trin i use-casen, og repliceret de, i use-casen, beskrevne fejl i programmet. Alt lod til at virke, og trinene blev fulgt tilfredsstillende.
Step 2a fejler, hvis testen foretages via WebServicens standard WebInterface, da dette ikke kan håndtere fejlmeddelelser. Dette anses dog ikke som en fejl, da WebServicens WebInterface ikke skal bruges til andet end tests.

Test konklusion:
Testen endte i succes, da søgningen blev foretaget korrekt.

Kapitel 12

Status, forbedring og udvidelser

12 Status, forbedringer og udvidelser

12.1 Indledning

I dette kapitel vil det blive gennemgået, hvor langt projektet nåede i forhold til de opsatte mål. Det vil blive diskuteret om krav fra kravspecifikationen blev nået, og hvordan de eventuelt kan nås, i en fremtidig udvidelse af systemet. Samtidig vil det blive diskuteret, hvilke kendte svagheder systemet har, og hvorledes disse indvirker på systemets funktionalitet.

12.2 Opfyldelse af kravspecifikation

Kravspecifikationen giver et godt mål for, hvorledes projektet er skredet frem. Der blev opsat en række krav, som projektet skulle opfylde, og ved at gennemgå disse, kan det ses, hvor langt projektet er nået, og om projektet er blevet udviklet færdigt.

12.2.1 Funktionelle krav

Vægt	Krav #	Krav	Er kravet opfyldt (ja/nej)?
1	§1.1	Runtime Platform	Ja
1	§1.2.8	Understøttelse af datalager (MSSql)	Ja
2	§1.3	Interface	Ja
2	§1.4.1	Forbindelse til E-mailserver	Ja
2	§1.4.2	Kommunikation med E-mailserver	Ja
2	§1.6	Kategorisering/Søge Parametre	Ja
3	§1.5.2	Opbevaring af E-mails	Ja
3	§1.2.1	Persistant Datalager	Ja
3	§1.2.4	Forespørgsler	Ja
4	§1.2.2	Relationel Datalager	Ja – ikke aktuelt
4	§1.2.3	Indekseret Datalager	Ja
4	§1.2.5	Transaktionssupport	Ja
4	§1.2.7	Unik identifikation af data	Ja
4	§1.4.3	Understøttelse for flere E-mail servere	Ja
4	§1.4.4	Enkel opsætning af E-mail servere	Nej
4	§1.5.1	Automatisk download af e-mails	Ja
5	§1.5.3	Afsendelse af e-mail	Nej
5	§1.7	GUI	Ja
6	§1.2.6	Support for flere datalager platforme	Ja

Som det kan ses ud fra ovenstående liste, er alle krav med en prioritet på 3 eller højere implementeret. Dette gør at vi kan konstatere, at der er blevet udviklet at funktionelt system, der opfylder vigtigste af de specifikke funktionelle krav.

Kravet §1.4.4 er dog ikke opfyldt – dette har en prioritet på 4, hvilket gør, at projektet ikke er fuldt har nået sit må, om at implementere alle krav med en prioritet på 4 eller højere. Funktionaliteten beskrevet i dette krav, er dog ikke således, at EMS Systemet ikke er funktionelt, men giver det et handikap i forhold til brugervenlighed – som dog også kan være et stort problem.

Kravet §1.5.3 er heller ikke opfyldt. Funktionaliteten beskrevet i dette krav, giver en funktionel mangel i projektet, som dog kan undværes, da det ikke ødelægger hovedformålet med projektet.

12.2.2 Ikke funktionelle krav

De ikke funktionelle krav kan være sværere at bedømme, med hensyn til om de er opnået eller ej. De er mere udefinerede i deres udtryk, og det kan være svært at sætte en grænse for hvornår kravet er nået. Det vil dog blive forsøgt, at give et indblik i hvad der tæller for og imod.

12.2.2.1 Pålidelighed

Pålidelighed er opnået til en vis grad, i det systemet vha. af tests, har vist sig at fungere ganske udmærket. Hovedfunktionaliteterne fungerer som forventet, og målet med projektet nås. Tests har dog vist at der er visse problemer med fejlbeskeder, som kan give et problem for især pålidelighed. Man må som bruger forvente, at der gives notifikation på eventuelle fejltilstande, hvilket der ikke altid gør i EMS Systemet. Dette er et seriøst problem, der bør rettes hurtigst muligt, for at brugeren skal få en bedst mulig oplevelse.

12.2.2.2 Kapacitet

Systemet har gennem tests vist at det kan varetage e-mails med et tungt indhold, kan fremsøge disse igen og sende dem korrekt via System Interfacet. Samtidig har det vist, at der ikke er problemer med et overvældende antal e-mails på en gang. Derfor må man antage at systemet kan klare den kapacitet, som det forventes at det bliver udsat for.

12.2.2.3 Brugervenlighed

Samme problem som beskrevet omkring pålidelighed gælder også ved brugervenlighed. Systemet skal have en bedre fejlbehandling, som vist af de udførte tests.
Der ud over, er det ikke implementerede funktionelle krav §1.4.4, også en belastning for brugervenligheden, da det ikke på en enkel måde, er muligt at opsætte e-mail servere. Man kan selvfølgelig pointere, at det *er* muligt at opsætte e-mail servere, ved at ændre den relativt overskuelige konfigurationsfil – men set fra et brugerveligheds synspunkt, er det ikke godt nok.
Dog har systemet klaret de fleste use cases, således at eksterne testere har fundet systemet godt nok, i forhold til de opsatte use case krav. Desuden er der blevet implementeret en enkel GUI, som giver en bruger en langt bedre oplevelse, end hvis systemet havde været rent tekstbaseret, eller simpelthen ikke havde et brugerinterface, og alle konfigurationer skulle laves direkte i konfigurationsfilen.

12.3 Konklusion på tests

Hoveddelen af de udførte tests, har fået et positivt resultat, og har ikke givet anledning til ønsker om ændringer af systemet. Der har dog, som tidligere omtalt, været enkelte problemer, som bør adresseres i en videre udvikling af systemet.
De opsatte tests, giver dog udtryk for et system der i de fleste tilfælde fungerer korrekt, og bestemt kan benyttes til det givne formål.

Enkelte tests kan dog aldrig give det fulde billede på, hvorledes et system fungerer. Først når systemet er kommet i produktion, eller benyttes som det ville i produktion, vil det med fuldstændig sikkerhed, vise sig om systemet fungerer tilfredsstillende. Denne type tests, har dog ikke været mulig at foretage i dette projekt, da der ikke har været tid, personale eller materiel til det.

12.4 Forbedringer og udvidelser

Det første der bør tages hånd om, ved en videreudvikling af systemet, er de før omtalte problemer, der har vist sig ved tests. Når disse er adresseret, kan man tage hul på, at opfylde de ikke implementerede krav, fra kravspecifikationen. Her efter bør man, efter en periode med systemet i produktionslignende omgivelser, lave en kritisk vurdering af, hvorledes systemet fungerer og samtidig vurderer, om der er dele, der bør ændres eller optimeres.

Systemet har fået den funktionalitet der ønskedes, og da det er et mindre system, der skal ligge som mellemled, er der ikke mange udvidelser der giver mening.

En udvidelse, som der blev omtalt i analysen, kunne dog være mulighed for flere system interfaces, via anden teknologi, som den omtalte via en socket forbindelse. Således skulle der opsættes et sæt regler for hvorledes denne kommunikation kunne udføres, og der skulle laves et modul til behandling af disse forbindelser.
Man kunne forstille sig, at systemet fungerede som server, hvortil klienter kunne tilkoble sig, netop hvad webserveren håndterer, for det implementerede system interface. Når kontakten blev opnået, skulle der være et defineret sæt af kommandoer, som interfacet reagerede på, også kaldet en protokol. Denne kan sammenlignes med fx POP3 protokollen, som er omtalt tidligere.
Der skulle umiddelbart ikke ændres i de andre moduler, i det de samme funktionaliteter skulle eksponeres, gennem dette nye interface, som dem, der eksponeres i den allerede implementerede WebService.

Systemet kunne ligeledes udvides med mulighed for at arbejde med flere forskellige database løsninger, hvilket ville give en større fleksibilitet. Systemet er gjort parat til dette, således at der blot skulle tilføjes et bestemt modul til kommunikation med de nye database typer, samt ændres en smule i databaselagets interface, således at det nye modul bliver tilgængeligt. Disse moduler, ville i høj grad ligne de eksisterende, og en stor del genbrug ville kunne foretages, i det mange database løsninger har en høj grad af identiske træk og de fleste bruger en form af SQL sproget til forespørgsler.

Kapitel 13

Konklusion

13 Konklusion

13.1 Formål

Formålet med dette projekt, har været udvikling og implementring af et system, til at opfylde et bestemt behov i en organisation. Der er blevet givet en problemstilling, og ud fra denne er systemet blevet udviklet. Udviklingen af projektet, er gjort på baggrund af en analyse af hvorledes, der bedst muligt kan opfylde den problemstilling, der er stillet.

Det forventes at det udviklede system, kan dække det specifikke behov, det er designet til. Samarbejdfirmaet har udtrykt interesse, i at få det implementeret i deres arbejdsgange, og dette har netop været målet for projektet – et fuldt funktionelt system, der kunne bruges i forbindelse med firmaets eksisterende systemer.

13.2 Opsummering

Alt i alt har det været et spændende projekt, der har givet udfordringer i forskellige programmerings grene og teknikker. Det er lykkedes at få lavet et system, der kan kommunikerer på en fornuftig måde og udføre den i kravspecifikationen og opgaveformuleringen krævede funktionalitet.
Tiden har som altid i sådanne projekter været en afgørende faktor, hvilket også er grunden til, at der er visse dele af systemet, der ikke fungerer optmalt.

Systemet er designet til en specifik opgave, og gennem de udførte tests, har det vist sig, at systemet dækker de behov, der er blevet identificeret. Som nævnt, er der også blevet identificeret en række problemer, som burde rettes. Derfor kunne der være brug for, at fortage en process, hvori fejlretning og problemløsning af systemet er prioriteret højst.

13.3 Evaluering af resultater

Resultatet af projektet, er et system, der kan opfylde kravet for funktionalitet i det stillede problem. På trods af de identificerede problemer, er systemet blevet udviklet, således at det kan udvides og benyttes som det er. Tests har vist, at systemet virker efter hensigten, og at det kan bruges ved større belastninger.

Ananlysen gav et design, der viste sig at kunne udføre den ønskede opgave, og gøre det på en måde, der gør systemet anvendeligt og brugervenligt.

Det anbefales at der foretages retelser af de i testen fundne problemer, og at systemet udvides med de ikke implementerede funktionaliteter, men systemet kan dog implementeres i firmaets organisation som det er, og vil udfylde den ønskede opgave tilfredsstillende.

Dokumentationen er skrevet, således at den giver et indblik i projektets udvikling, og hvilke oervejelser der ligger til grund for det udviklede system. Den fungerer samtidig som dokumentation for, at systemet kan udføre den ønskede funktionalitet, på en måde, der er effektiv og rimelig fejlfri.

Systemet vil udføre den rolle, det er designet til, på en sådan måde, at behandlingen af e-mails lettes i den organisation det implemeneteres i. E-mails vil blive langt bedre integreret i de nuværende systemer, hvilket vil gøre arbejdsgangen lettere for de ansatte, og give en mere enkel og fejlfri kommunikation med kunder via e-mail. Samtidig giver det de ansatte bedre mulighed for at kunne genskabe og få adgang til historisk e-mail data. Således vil organisationens arbejdsgang blive effektviseret og bedre organiseret, samtidig med at der vil blive skabt en bedre service over for kunden.

Kapitel 14

Appendiks

14 Appendiks

14.1 Litteratur

- Programming Microsoft ASP.NET
  Dino Esposito
  Microsoft Press 2003
  ISBN: 0-7356-1903-4

- Programming Microsoft Windows with C#
  Charles Petzold
  Microsoft Press 2002
  ISBN: 0-7356-1370-2

- Design Patterns
  Elements of Reusable Object-Oriented Software
  Eric Gamma, Richard Helm, Ralph Johnson, John Vlissides
  Addison-Wesley 1995
  ISBN: 0-201-63361-2

- The Rational Unified Process – An Introduction (Second Edition)
  Philippe Kruchten
  Addison-Wesley
  ISBN: 0-201-70710-1

- Use Case Driven Object Modelling With UML
  Doug Rosenberg with Kendall Scott
  Addison-Wesley
  ISBN: 0-201-43289-7

- Applying UML and Patterns
  An introduction to Object-Oriented Analysis and Design and the Unified Process
  Craig Larman
  Prentice Hall, Inc.
  ISBN: 0-13-092569-1

- World Wide Web Consortium
  http://www.w3.org

- Wikipedia
  http://www.wikipedia.com

- MSDN
  Microsoft online documentation
  http://msdn.microsoft.com

- Retrieve Mail From a POP3 Server Using C#
  Agus Kurniawan
  http://www.codeproject.com/csharp/popapp.asp

14.2 Kildekode

Her vedlægges alt kildekode til projektet.

14.2.1 EMSBackend

14.2.1.1 EMSBackend.cs

```
using System;
using System.IO;
using System.Data;

using EMSBackend.ValueObjects;
using EMSBackend.POP3;

using EMSDatabase;
using Lesnikowski.Mail;     // Email parser

namespace EMSBackend
{
  public class EMSBackend
  {
      static EMSRecieveEmails re;

      public static void RecieveEmails(string server, int port, string user, string password)
      {
          //Use own POP3 Connecter to connect
          POP3.POP3client pop3 = new POP3.POP3client();

          try
          {
              //Open connection, set username and password
              pop3.Connect(server,port);
              pop3.USER(user);
              pop3.PASS(password);

              //Get number of messages
              int noe = pop3.GetNOMessages();

              for(int i = 1; i<=noe; i++)
              {
                  try
                  {
                      //Receive email
                      string temp = pop3.GetMessage(i);

                      //Parse email
                      SimpleMailMessage simpleMail= SimpleMailMessage.Parse(temp);

                      //Print email to text file: test only
                      //PrintEmail(simpleMail);

                      //Save email i DB
                      if(!SaveEmail(simpleMail, temp))
                      {
                          throw new DataException("Database not responding.");
                      }

                      // Delete message from POP3 server
                      if(EMSConfig.DeleteEmails)
                      {
                          pop3.DELE(i);
                      }
                  }
                  catch (DataException ex)
                  {
                      throw new DataException(ex.Message);
                  }
                  catch (Exception ex)
                  {
                      //Do nothing
                      //Go to next email
                  }
```

```
            }
        }
        catch(System.Net.Sockets.SocketException sex)
        {
            // Server error
        }
        catch(IOException ioex)
        {
            // IO error
        }
        catch(DataException ex)
        {
            throw;
        }
        catch(Exception ex)
        {
            //Error
        }
        finally
        {
            // Close connection
            pop3.QUIT();
        }
    }


    public static void RecieveEmails(EMSServerVO[] servers)
    {
        foreach(EMSServerVO s in servers)
        {
            EMSBackend.RecieveEmails(s.Server,s.Port,s.User,s.Password);
        }
    }


    private static bool SaveEmail(SimpleMailMessage smm, string mailData)
    {
        string from = ((MailBox)smm.From[0]).Address;
        string to = ((MailBox)smm.To[0]).Address;
        bool hasatt = (smm.Attachments.Count > 0)? true:false;
        return DBAccess.SaveEmail(Convert.ToInt32(EMSConfig.DBType), EMSConfig.DSN, from, to,
smm.Subject, mailData, smm.Date, hasatt);
    }


    private static void PrintEmail(SimpleMailMessage smm)
    {
        StreamWriter sw = null;
        try
        {
            //Open file stream
            sw = new StreamWriter(@"Test.txt",true);

            // Write out email's subject
            sw.WriteLine( smm.Subject );

            // Write out email's from.
            foreach(MailBox mb in smm.From)
            {
                sw.Write("From: ");
                // Write out name, if it was provided
                if (mb.Name != null)
                    sw.Write( mb.Name );
                // Write out address
                sw.WriteLine("<{0}>", mb.Address);
            }

            // Write out email's body.
            if (smm.HtmlData != null)
                sw.WriteLine( smm.HtmlDataString ); // HTML email
            else
                sw.WriteLine( smm.TextDataString );   // Text email


            // Write out Attachments
```

```
            foreach(MimeData attachment in smm.Attachments)
            {
                sw.WriteLine(attachment.FileName);
            }
        }
        catch(Exception ex)
        {
            if(sw != null)
            {
                sw.WriteLine(ex.Message);             // Invalid email format
            }
        }
        finally
        {
            if(sw != null)
            {
                sw.Flush();
                sw.Close();
            }
        }
    }


    public static EMSEmailVO[] SearchEmails(string from, string to, string subject, DateTime
send_start, DateTime send_end, bool hasAtt)
    {
        /*try
        {
            System.Xml.XmlDocument SystemConfig = new System.Xml.XmlDocument();
            SystemConfig.Load("C:\\System.config");
        }
        catch(Exception ex)
        {
            string exp = ex.Message;
        }*/
        DataTable data = DBAccess.FindEmails(Convert.ToInt32(EMSConfig.DBType), EMSConfig.DSN,
from, to, subject, send_start, send_end, hasAtt);

        EMSEmailVO[] result = new EMSEmailVO[data.Rows.Count];
        int i = 0;
        foreach(DataRow dr in data.Rows)
        {
            SimpleMailMessage smm = SimpleMailMessage.Parse(dr["EmailData"].ToString());

            int j;
            EMSMailboxVO[] tomb = new EMSMailboxVO[smm.To.Count];
            j = 0;
            foreach(MailBox m in smm.To)
            {
                tomb[j] = new EMSMailboxVO(m.Name, m.Address);
                j++;
            }

            EMSMailboxVO[] frommb = new EMSMailboxVO[smm.From.Count];
            j = 0;
            foreach(MailBox m in smm.From)
            {
                frommb[j] = new EMSMailboxVO(m.Name, m.Address);
                j++;
            }

            EMSAttachmentVO[] attach = new EMSAttachmentVO[smm.Attachments.Count];
            j = 0;
            foreach(MimeData md in smm.Attachments)
            {
                string filename = md.FileName != null ? md.FileName : "file"+j;
                attach[j] = new EMSAttachmentVO(filename,md.Data);
                j++;
            }

            result[i] = new
EMSEmailVO(tomb,frommb,smm.Subject,smm.TextDataString,smm.HtmlDataString,smm.Date,attach);
            i++;
        }
        return result;
```

```
        }


        public static void StartRecieveEmails(EMSServerVO[] servers, int interval)
        {
            re = new EMSRecieveEmails(servers, interval);
            re.StartRecieveEmails();
        }


        public static void StopRecieveEmails()
        {
            if(re != null)
            {
                re.StopRecieveEmails();
            }
        }
    }
}
```

14.2.1.2 EMSConfig.cs

```
using System;
using System.IO;
using System.Configuration;
using System.Collections.Specialized;
using System.Xml;
using EMSBackend.ValueObjects;

namespace EMSBackend
{
  public enum DBEnum
  {
      MySql = 1,
      MSSql = 2
  }


  public class EMSConfig
  {
      private static string dsn;
      private static string configPath;
      private static DBEnum dbtype;
      private static int interval;
      private static bool deleteEmails;
      private static EMSServerVO[] servers;

      public static string DSN
      {
          get
          {
              return dsn;
          }
      }

      public static DBEnum DBType
      {
          get
          {
              return dbtype;
          }
      }

      public static string ConfigPath
      {
          get
          {
              return configPath;
          }
      }

      public static int Interval
      {
          get
          {
              return interval;
          }
      }

      public static bool DeleteEmails
      {
          get
          {
              return deleteEmails;
          }
          set
          {
              deleteEmails = value;
          }
      }

      public static EMSServerVO[] Servers
      {
```

```
        get
        {
            if(servers != null)
            {
                return servers;
            }
            else
            {
                return new EMSServerVO[0];
            }
        }
        set
        {
            servers = value;
        }
    }


    static EMSConfig()
    {
        UpdateSettings();
    }


    private static void UpdateSettings()
    {
        configPath = "System.config";
        LoadSystemConfigConfigurationSection();
    }

    #region System.Config

    //Loads Configuration settings from System.config
    public static void LoadSystemConfigConfigurationSection()
    {
        XmlDocument SystemConfig = new XmlDocument();
        SystemConfig.Load(ConfigPath);
        interval = Convert.ToInt32(GetValue("Interval",SystemConfig));
        deleteEmails = Convert.ToBoolean(GetValue("DeleteEmails",SystemConfig));
        dsn = GetValue("DSN",SystemConfig);
        dbtype = (DBEnum)Enum.Parse(typeof(DBEnum), GetValue("DBType",SystemConfig),true);
        servers = GetServers(SystemConfig);
    }


    //Finds all servers
    private static EMSServerVO[] GetServers(XmlDocument xmlDoc)
    {
        XmlNode serverSettingsNode = xmlDoc.SelectSingleNode("configuration/serverSettings");
        EMSServerVO[] res = new EMSServerVO[serverSettingsNode.ChildNodes.Count];
        int i = 0;
        foreach(XmlNode childnode in serverSettingsNode)
        {
            res[i] = new EMSServerVO(
                childnode.Attributes["name"].Value,
                childnode.Attributes["server"].Value,
                Convert.ToInt32(childnode.Attributes["port"].Value),
                childnode.Attributes["user"].Value,
                childnode.Attributes["password"].Value);
            i++;
        }
        return res;
    }


    //Updates a given configuration setting
    public static void SetSystemConfiguration(string key, string keyvalue)
    {
        XmlDocument SystemConfig = new XmlDocument();
        SystemConfig.Load(ConfigPath);

        UpdateKey(key, keyvalue, SystemConfig);

        SystemConfig.Save(ConfigPath);
```

```
            //Reload system config settings
            LoadSystemConfigConfigurationSection();
        }


        //Locates a XmlNode by key
        private static string GetValue(string strKey, XmlDocument xmlDoc)
        {
            XmlNode appSettingsNode = xmlDoc.SelectSingleNode("configuration/appSettings");

            if (!KeyExists(strKey,appSettingsNode))
            {
                throw new ArgumentNullException("Key", "<" + strKey + "> does not exist in the
configuration. Update failed.");
            }

            // Attempt to locate the requested node.
            foreach (XmlNode childNode in appSettingsNode)
            {
                if (childNode.Attributes["key"].Value == strKey)
                {
                    return childNode.Attributes["value"].Value;
                }
            }

            return string.Empty;
        }


        // Determines if a key exists in an XmlDocument
        private static bool KeyExists(string strKey, XmlNode xmlNode)
        {
            // Attempt to locate the requested setting.
            foreach (XmlNode childNode in xmlNode)
            {
                if (childNode.Attributes["key"].Value == strKey)
                    return true;
            }
            return false;
        }


        // Updates a key within an XmlDocument
        private static XmlDocument UpdateKey(string strKey, string newValue, XmlDocument xmlDoc)
        {
            XmlNode appSettingsNode = xmlDoc.SelectSingleNode("configuration/appSettings");

            if (!KeyExists(strKey,appSettingsNode))
            {
                throw new ArgumentNullException("Key", "<" + strKey + "> does not exist in the
configuration. Update failed.");
            }

            // Attempt to locate the requested setting.
            foreach (XmlNode childNode in appSettingsNode)
            {
                if (childNode.Attributes["key"].Value == strKey)
                {
                    childNode.Attributes["value"].Value = newValue;
                }
            }

            return xmlDoc;
        }

        #endregion
    }
}
```

14.2.1.3 EMSRecieveEmail

```
using System;
using System.Threading;
using EMSBackend.ValueObjects;

namespace EMSBackend
{
  public class EMSRecieveEmails
  {
      Thread workerThread;
      EMSRecieveEmailsWC workerObject;

      public EMSRecieveEmails(EMSServerVO[] servers, int interval)
      {
          // Create the thread object. This does not start the thread.
          workerObject = new EMSRecieveEmailsWC(servers, interval);
          workerThread = new Thread(new ThreadStart(workerObject.DoWork));
      }

      public void StartRecieveEmails()
      {
          // Start the worker thread.
          workerThread.Start();

          // Loop until worker thread activates.
          while (!workerThread.IsAlive);
      }

      public void StopRecieveEmails()
      {
          // Request that the worker thread stop itself:
          workerObject.RequestStop();

          // Use the Join method to block the current thread
          // until the object's thread terminates.
          workerThread.Join();
      }
  }

  public class EMSRecieveEmailsWC
  {
      private volatile bool shouldStop;
      private int interval;
      private EMSServerVO[] servers;

      public EMSRecieveEmailsWC(EMSServerVO[] servers, int interval)
      {
          this.interval = interval;
          this.servers = servers;
      }

      public void DoWork()
      {
          while (!shouldStop)
          {
              Console.WriteLine("Starting download from server(s)!");
              EMSBackend.RecieveEmails(servers);
              Console.WriteLine("Sleeping");
              Thread.Sleep(interval*1000);
          }
          Console.WriteLine("Stopped");
      }

      public void RequestStop()
      {
          shouldStop = true;
      }
  }
}
```

14.2.1.4 EMSEmailVO

```
using System;
using System.Collections;

namespace EMSBackend.ValueObjects
{
  [Serializable]
  public class EMSEmailVO
  {
      private string subject, text, htmltext;
      private DateTime date;
      private EMSMailboxVO[] to, from;
      private EMSAttachmentVO[] attachment;

      public string Subject
      {
          get
          {
              if(subject != null)
              {
                  return subject;
              }
              else
              {
                  return string.Empty;
              }
          }
          set
          {
              subject = value;
          }
      }

      public string Text
      {
          get
          {
              if(text != null)
              {
                  return text;
              }
              else
              {
                  return string.Empty;
              }
          }
          set
          {
              text = value;
          }
      }

      public string HTMLText
      {
          get
          {
              if(htmltext != null)
              {
                  return htmltext;
              }
              else
              {
                  return string.Empty;
              }
          }
          set
          {
              htmltext = value;
          }
      }


      public DateTime Date
```

```
{
    get
    {
        return date;
    }
    set
    {
        date = value;
    }
}


public EMSMailboxVO[] To
{
    get
    {
        if(to != null)
        {
            return to;
        }
        else
        {
            to = new EMSMailboxVO[1];
            to[0].Name = "None";
            to[0].Address = "None";
            return to;
        }
    }
    set
    {
        to = value;
    }
}

public EMSMailboxVO[] From
{
    get
    {
        if(from != null)
        {
            return from;
        }
        else
        {
            from = new EMSMailboxVO[1];
            from[0].Name = "None";
            from[0].Address = "None";
            return to;
        }
    }
    set
    {
        from = value;
    }
}


public EMSAttachmentVO[] Attachment
{
    get
    {
        if(attachment != null)
        {
            return attachment;
        }
        else
        {
            return new EMSAttachmentVO[0];;
        }
    }
    set
    {
        attachment = value;
    }
}
```

```
      public EMSEmailVO()
      {
      }


      public EMSEmailVO(EMSMailboxVO[] to, EMSMailboxVO[] from, string subject, string text, string
htmltext, DateTime date, EMSAttachmentVO[] attachment)
      {
          this.Subject = subject;
          this.Text = text;
          this.HTMLText = htmltext;
          this.Date = date;
          this.To = to;
          this.From = from;
          this.Attachment = attachment;
      }


      public override string ToString()
      {
          return Date.ToString("yyyy-MM-dd HH:mm") + ": " + Subject;
      }
  }


  [Serializable]
  public class EMSMailboxVO
  {
      private string name, address;

      public string Name
      {
          get
          {
              if(name != null)
              {
                  return name;
              }
              else
              {
                  return string.Empty;
              }
          }
          set
          {
              name = value;
          }
      }

      public string Address
      {
          get
          {
              if(address != null)
              {
                  return address;
              }
              else
              {
                  return string.Empty;
              }
          }
          set
          {
              address = value;
          }
      }


      public EMSMailboxVO()
      {
      }
```

```
        public EMSMailboxVO(string name, string address)
        {
            this.name = name;
            this.address = address;
        }

    }


    [Serializable]
    public class EMSAttachmentVO
    {
        private string filename;
        private int size;
        private byte[] data;

        public string Filename
        {
            get
            {
                return filename;
            }
            set
            {
                filename = value;
            }
        }

        public int Size
        {
            get
            {
                return size;
            }
            set
            {
                size = value;
            }
        }

        public byte[] Data
        {
            get
            {
                if(data != null)
                {
                    return data;
                }
                else
                {
                    return new byte[0];
                }
            }
            set
            {
                data = value;
            }
        }


        public EMSAttachmentVO()
        {
        }


        public EMSAttachmentVO(string filename, byte[] data)
        {
            this.Filename = filename;
            this.data = data;
            this.size = data.Length;
        }

    }
}
```

14.2.1.5 EMSServerVO

```
using System;

namespace EMSBackend.ValueObjects
{
    public class EMSServerVO
    {
        string name, server, user, password;
        int port;

        public string Name
        {
              get {return name;}
              set {name = value;}
        }

        public string Server
        {
              get {return server;}
              set {server = value;}
        }

        public int Port
        {
              get {return port;}
              set {port = value;}
        }

        public string User
        {
              get {return user;}
              set {user = value;}
        }

        public string Password
        {
              get {return password;}
              set {password = value;}
        }


        public EMSServerVO(string name, string server, int port, string user, string password)
        {
              this.Name = name;
              this.Server = server;
              this.Port = port;
              this.User = user;
              this.Password = password;
        }

        public override string ToString()
        {
              return Name;
        }
    }
}
```

14.2.1.6 POP3Client

```
using System;
using System.IO ;
using System.Net;
using System.Net.Sockets ;

namespace EMSBackend.POP3
{

    public class POP3client
    {
        public enum connect_state {
            disc,
            AUTHORIZATION,
            TRANSACTION,
            UPDATE
        };

        public string user;
        public string pwd;
        public string server;
        public int port;
        public bool error;
        public connect_state state = connect_state.disc ;

        private TcpClient Server;
        private NetworkStream NetStrm;
        private StreamReader  RdStrm;
        private string Data;
        private byte[] szData;
        private string CRLF = "\r\n";

        public POP3client()
        {
        }


        public POP3client(string server,int port,string user,string password)
        {
            this.server = server;
            this.user = user;
            this.pwd = password;
            this.port = port;
        }


        #region Utility Methods

        public string Connect (string server, int port)
        {
            this.server = server;
            this.port = port;
            return(Connect());
        }


        public string Connect()
        {
            // Create connection
            Server = new TcpClient(server,port);


            try
            {
                // Initialization
                NetStrm = Server.GetStream();
                RdStrm = new StreamReader(Server.GetStream());

                //The pop session is now in the AUTHORIZATION state
                state = connect_state.AUTHORIZATION ;
                return(RdStrm.ReadLine ());
            }
            catch(InvalidOperationException ex)
```

```
    {
        return("Error: "+ex.ToString());
    }

}


public int GetNOMessages()
{

    if (state==connect_state.TRANSACTION)
    {
        string temp = STAT();
        string[] temparr;
        temparr = temp.Split(' ');
        return Convert.ToInt32(temparr[1]);
    }
    else

    {
        //the pop command STAT is only valid in the TRANSACTION state
        return (-1);
    }
}


public string GetMessage(int msg_number)
{
    //Remove "+OK" and CRLF
    string temp = RETR(msg_number);
    temp = temp.Remove(0,6);
    return temp;
}


private string disconnect()
{
    string temp="Disconnected Successfully.";
    if(state != connect_state.disc)
    {
        //close connection
        NetStrm.Close();
        RdStrm.Close();
        Server.Close();
        NetStrm = null;
        RdStrm = null;
        Server = null;
        state = connect_state.disc ;
    }
    else
    {
        temp = "Not Connected.";
    }
    return(temp);
}


private void issue_command(string command)
{
    Data = command + CRLF;
    szData = System.Text.Encoding.ASCII.GetBytes(Data.ToCharArray());
    NetStrm.Write(szData,0,szData.Length);
}


private string read_single_line_response()
{
    string temp;
    try
    {
        temp = RdStrm.ReadLine();
        was_pop_error(temp);
        return(temp);
    }
    catch(InvalidOperationException ex)
```

```
        {
            return("Error in read_single_line_response(): " + ex.ToString ()) ;
        }

    }


    private string read_multi_line_response()
    {
        string temp = string.Empty;
        string szTemp;

        try
        {
            szTemp = RdStrm.ReadLine();
            was_pop_error(szTemp);
            if(!error)
            {

                while(szTemp != ".")
                {
                    temp += szTemp+CRLF;
                    szTemp = RdStrm.ReadLine();
                }
            }
            else
            {
                temp = szTemp;
            }
            return(temp);
        }
        catch(InvalidOperationException ex)
        {
            return("Error in read_multi_line_response(): " + ex.ToString ());
        }
    }


    private void was_pop_error(string response)
    {
        //Check if server believes an error has occured

        if(response.StartsWith ("-"))
        {
            //If the first character of the response is "-", then the
            //POP3 server has encountered an error executing the last
            //command send by the client
            error = true;
        }
        else
        {
            //No Error
            error = false;
        }
    }


    #endregion

    #region POP Commands

    public string DELE(int msg_number)
    {
        string temp;

        if (state != connect_state.TRANSACTION )
        {
            //DELE is only valid when the pop session is in the TRANSACTION STATE
            temp="Connection state not = TRANSACTION";
        }
        else
        {
            issue_command("DELE " + msg_number.ToString ());
            temp=read_single_line_response();
        }
```

```
            return(temp);
        }

        public string LIST()
        {
            string temp="";
            if (state != connect_state.TRANSACTION )
            {
                //the pop command LIST is only valid in the TRANSACTION state
                temp="Connection state not = TRANSACTION";
            }
            else
            {
                issue_command ("LIST");
                temp=read_multi_line_response();
            }
            return(temp);
        }

        public string LIST(int msg_number)
        {
            string temp="";

            if (state != connect_state.TRANSACTION )
            {
                //the pop command LIST is only valid in the TRANSACTION state
                temp="Connection state not = TRANSACTION";
            }
            else
            {
                issue_command ("LIST " + msg_number.ToString ());
                temp=read_single_line_response();  //when the message number is supplied, expect a
single line response
            }
            return(temp);

        }

        public string NOOP()
        {
            string temp;
            if (state != connect_state.TRANSACTION )
            {
                //the pop command NOOP is only valid in the TRANSACTION state
                temp="Connection state not = TRANSACTION";
            }
            else
            {
                issue_command ("NOOP");
                temp=read_single_line_response();

            }
            return(temp);

        }
        public string PASS()
        {
            string temp;
            if (state != connect_state.AUTHORIZATION)
            {
                //the pop command PASS is only valid in the AUTHORIZATION state
                temp="Connection state not = AUTHORIZATION";
            }
            else
            {
                if (pwd !=null)
                {
                    issue_command ("PASS " + pwd);
                    temp=read_single_line_response();

                    if (!error)
                    {
                        //transition to the Transaction state
                        state=connect_state.TRANSACTION;
                    }
```

```
        }
        else
        {
            temp="No Password set.";
        }
    }
    return(temp);
}
public string PASS(string password)
{
    this.pwd = password;
    return(PASS());
}

public string QUIT()
{
    //QUIT is valid in all pop states

    string temp;
    if (state !=connect_state.disc)
    {
        issue_command ("QUIT");
        temp=read_single_line_response();
        temp += CRLF + disconnect();

    }
    else
    {
        temp="Not Connected.";
    }
    return(temp);

}
public string RETR (int msg)
{
    string temp="";
    if (state != connect_state.TRANSACTION )
    {
        //the pop command RETR is only valid in the TRANSACTION state
        temp="Connection state not = TRANSACTION";
    }
    else
    {
        // retrieve mail with number mail parameter
        issue_command ("RETR "+ msg.ToString ());
        temp=read_multi_line_response();
    }
    return(temp);

}

public string RSET()
{
    string temp;
    if (state != connect_state.TRANSACTION )
    {
        //the pop command STAT is only valid in the TRANSACTION state
        temp="Connection state not = TRANSACTION";
    }
    else
    {
        issue_command("RSET");
        temp=read_single_line_response();
    }
    return(temp);

}

public string STAT()
{
    string temp;
    if (state==connect_state.TRANSACTION)
    {
        issue_command("STAT");
        temp=read_single_line_response();
```

```
            return(temp);
        }
        else

        {
            //the pop command STAT is only valid in the TRANSACTION state
            return ("Connection state not = TRANSACTION");
        }
    }

    public string USER()
    {
        string temp;
        if (state != connect_state.AUTHORIZATION)
        {
            //the pop command USER is only valid in the AUTHORIZATION state
            temp="Connection state not = AUTHORIZATION";
        }
        else
        {
            if (user !=null)
            {
                issue_command("USER "+ user);
                temp=read_single_line_response();
            }
            else
            {   //no user has been specified
                temp="No User specified.";
            }
        }
        return(temp);
    }

    public string USER(string user)
    {
        this.user=user;
        return(USER());
    }


    #endregion
  }

}
```

14.2.2 EMSDatabase

14.2.2.1 DataAccess.cs

```
using System;
using System.Data;

namespace EMSDatabase
{
    public class DBAccess
    {
        #region email

        public static bool SaveEmail(int DBType, string connstr, string from, string to, string
subject, string email, DateTime send, bool hasAtt)
        {
            bool res = false;
            switch(DBType)
            {
                case 1:
                    res = EMSDatabase.DBAccess_MySQL.SaveEmail(connstr, from, to, subject, email,
send, hasAtt);
                    break;
                case 2:
                    res = EMSDatabase.DBAccess_MSSQL.SaveEmail(connstr, from, to, subject, email,
send, hasAtt);
                    break;
            }
            return res;
        }


        public static DataTable FindEmails(int DBType, string connstr, string from, string to,
string subject, DateTime send_start, DateTime send_end, bool hasAtt)
        {
            DataTable res = new DataTable();
            switch(DBType)
            {
                case 1:
                    res = EMSDatabase.DBAccess_MySQL.FindEmails(connstr, from, to, subject,
send_start, send_end, hasAtt);
                    break;
                case 2:
                    res = EMSDatabase.DBAccess_MSSQL.FindEmails(connstr, from, to, subject,
send_start, send_end, hasAtt);
                    break;
            }
            return res;
        }


        #endregion
    }
}
```

14.2.2.2 DataAccess_MSSQL.cs

```
using System;
using System.Data;
using System.Data.SqlClient;

namespace EMSDatabase
{
    public class DBAccess_MSSQL
    {
        private static DataTable ExecuteQuery(string cmd_str, string conn_str)
        {
            SqlDataAdapter adapter = new SqlDataAdapter(cmd_str, conn_str);
            DataTable data = new DataTable();
            adapter.Fill(data);
            return data;
        }


        public static bool SaveEmail(string connstr, string from, string to, string subject, string
email, DateTime send, bool hasAtt)
        {
            bool ok;
            String conn_str = connstr;
            String cmd_str =
                "INSERT INTO EMSEmail " +
                "(EmailFrom, EmailTo, EmailSubject, EmailData, EmailDate, HasAtt) VALUES "+
                "(@from, @to, @subject, @emaildata, @emaildate, @hasatt)";

            SqlConnection conn = new SqlConnection(conn_str);
            SqlCommand cmd = new SqlCommand(cmd_str, conn);
            cmd.Parameters.Add("@from", SqlDbType.VarChar).Value = from;
            cmd.Parameters.Add("@to", SqlDbType.VarChar).Value = to;
            cmd.Parameters.Add("@subject", SqlDbType.VarChar).Value = subject;
            cmd.Parameters.Add("@emaildata", SqlDbType.VarChar).Value = email;
            cmd.Parameters.Add("@emaildate", SqlDbType.DateTime).Value = send.ToString("yyyy-MM-dd
hh:mm:ss");
            cmd.Parameters.Add("@hasatt", SqlDbType.Bit).Value = hasAtt;

            try
            {
                conn.Open();
                cmd.ExecuteNonQuery();
                ok = true;
            }
            catch(Exception ex)
            {
                ok = false;
            }
            finally
            {
                if(conn != null)
                {
                    conn.Close();
                }
            }
            return ok;
        }


        public static DataTable FindEmails(string connstr, string from, string to, string subject,
DateTime send_start, DateTime send_end, bool hasAtt)
        {
            String conn_str = connstr;
            String cmd_str =
                "SELECT * FROM EMSEmail " +
                "WHERE ";

            if(from != null && from != string.Empty)
            {
                cmd_str += "(EmailFrom LIKE '" + from + "') AND ";
            }

            if(to != null && to != string.Empty)
```

```
            {
                cmd_str += "(EmailTo LIKE '" + to + "') AND ";
            }

            if(subject != null && subject != string.Empty)
            {
                cmd_str += "(EmailSubject LIKE '" + subject + "') AND ";
            }

            if(send_start != DateTime.MinValue && send_end != DateTime.MinValue)
            {
                cmd_str += "(EmailDate BETWEEN '" + send_start.ToString("yyyy-MM-dd hh:mm:ss") + "'
AND '" + send_end.ToString("yyyy-MM-dd hh:mm:ss") + "') AND ";
            }

            if(hasAtt)
            {
                cmd_str += "(HasAtt = 1) AND ";
            }

            //To make sure that there is no "AND" in the end
            cmd_str += "1=1";

            try
            {
                return ExecuteQuery(cmd_str, conn_str);
            }
            catch(Exception ex)
            {
                throw new Exception("Database not responding.");
            }

        }

    }
}
```

14.2.2.3 DataAccess_MySQL.cs

```
using System;
using System.Data;
using MySql.Data.MySqlClient;

namespace EMSDatabase
{
    public class DBAccess_MySQL
    {
        private static DataTable ExecuteQuery(string cmd_str, string conn_str)
        {
            MySqlDataAdapter adapter = new MySqlDataAdapter(cmd_str, conn_str);
            DataTable data = new DataTable();
            adapter.Fill(data);
            return data;
        }


        public static bool SaveEmail(string connstr, string from, string to, string subject, string
email, DateTime send, bool hasAtt)
        {
            bool ok;
            String conn_str = connstr;
            String cmd_str =
                "INSERT INTO EMSEmail " +
                "(EmailFrom, EmailTo, EmailSubject, EmailData, EmailDate, HasAtt) VALUES "+
                "(?from, ?to, ?subject, ?emaildata, ?emaildate, ?hasatt)";

            MySqlConnection conn = new MySqlConnection(conn_str);
            MySqlCommand cmd = new MySqlCommand(cmd_str, conn);
            cmd.Parameters.Add("?from", MySqlDbType.String).Value = from;
            cmd.Parameters.Add("?to", MySqlDbType.String).Value = to;
            cmd.Parameters.Add("?subject", MySqlDbType.String).Value = subject;
            cmd.Parameters.Add("?emaildata", MySqlDbType.String).Value = email;
            cmd.Parameters.Add("?emaildate", MySqlDbType.Datetime).Value = send.ToString("yyyy-MM-dd
hh:mm:ss");
            cmd.Parameters.Add("?hasatt", MySqlDbType.Bit).Value = hasAtt;

            try
            {
                conn.Open();
                cmd.ExecuteNonQuery();
                ok = true;
            }
            catch(Exception ex)
            {
                ok = false;
            }
            finally
            {
                if(conn != null)
                {
                    conn.Close();
                }
            }
            return ok;
        }


        public static DataTable FindEmails(string connstr, string from, string to, string subject,
DateTime send_start, DateTime send_end, bool hasAtt)
        {
            String conn_str = connstr;
            String cmd_str =
                "SELECT * FROM EMSEmail " +
                "WHERE ";

            if(from != null && from != string.Empty)
            {
                cmd_str += "(EmailFrom LIKE '" + from + "') AND ";
            }

            if(to != null && to != string.Empty)
```

```
            {
                cmd_str += "(EmailTo LIKE '" + to + "') AND ";
            }

            if(subject != null && subject != string.Empty)
            {
                cmd_str += "(EmailSubject LIKE '" + subject + "') AND ";
            }

            if(send_start != DateTime.MinValue && send_end != DateTime.MinValue)
            {
                cmd_str += "(EmailDate BETWEEN '" + send_start.ToString("yyyy-MM-dd hh:mm:ss") + "'
AND '" + send_end.ToString("yyyy-MM-dd hh:mm:ss") + "') AND ";
            }

            if(hasAtt)
            {
                cmd_str += "(HasAtt = 1) AND ";
            }

            //To make sure that there is no "AND" in the end
            cmd_str += "1=1";

            try
            {
                return ExecuteQuery(cmd_str, conn_str);
            }
            catch(Exception ex)
            {
                throw new Exception("Database not responding.");
            }

        }

    }
}
```

14.2.3 EMSWebService

14.2.3.1 Service1.asmx

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Web;
using System.Web.Services;
using EMSBackend;
using EMSBackend.ValueObjects;

namespace EMSWebService
{
    public class Service1 : System.Web.Services.WebService
    {
        public Service1()
        {
            InitializeComponent();
        }


        #region Component Designer generated code

        private IContainer components = null;

        private void InitializeComponent()
        {
        }

        protected override void Dispose( bool disposing )
        {
            if(disposing && components != null)
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #endregion

        [WebMethod]
        public EMSEmailVO[] SearchMails(string from, string to, string subject, DateTime send_start,
DateTime send_end, bool hasAtt)
        {
            return EMSBackend.EMSBackend.SearchEmails(
                from != string.Empty ? from : "%",
                to != string.Empty ? to : "%",
                subject != string.Empty ? subject : "%",
                send_start,
                send_end,
                hasAtt
            );
        }
    }
}
```

14.2.4 EMSGUI

14.2.4.1 MainForm.cs

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;

using EMSBackend;
using EMSBackend.ValueObjects;

namespace EMSGUI
{
    public class MainForm : System.Windows.Forms.Form
    {
        private System.Windows.Forms.GroupBox GroupBox_Search;
        private System.Windows.Forms.Button Button_Search;
        private System.Windows.Forms.GroupBox GroupBox_Emails;
        private System.Windows.Forms.Button Button_Start;
        private System.Windows.Forms.Label Label_State;
        private System.ComponentModel.Container components = null;
        private System.Windows.Forms.MainMenu mainMenu1;
        private System.Windows.Forms.MenuItem menuItem1;
        private System.Windows.Forms.MenuItem menuItem2;
        private System.Windows.Forms.MenuItem menuItem3;
        private System.Windows.Forms.MenuItem menuItem4;
        private System.Windows.Forms.MenuItem menuItem5;
        private System.Windows.Forms.Label label6;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.TextBox TextBox_From;
        private System.Windows.Forms.TextBox TextBox_To;
        private System.Windows.Forms.TextBox TextBox_Subject;
        private System.Windows.Forms.DateTimePicker DateTimePicker_Start;
        private System.Windows.Forms.DateTimePicker DateTimePicker_End;
        private System.Windows.Forms.CheckBox CheckBox_Attachment;
        private System.Windows.Forms.GroupBox GroupBox_Results;
        private System.Windows.Forms.ListBox ListBox_Results;

        private bool state;
        private System.Windows.Forms.Button Button_All;
        private EMSEmailVO[] results = null;

        [STAThread]
        static void Main()
        {
            Application.Run(new MainForm());
        }


        public MainForm()
        {
            TestConnections();
            InitializeComponent();
            InitializeData();
        }

        private void TestConnections()
        {
            try
            {
                EMSBackend.EMSBackend.SearchEmails("x", "x", "x", DateTime.MinValue,
DateTime.MinValue, false);
            }
            catch(Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
```

```
private void InitializeData()
{
}


protected override void Dispose( bool disposing )
{
    if( disposing )
    {
        if(components != null)
        {
            components.Dispose();
        }
    }
    base.Dispose( disposing );
}


#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    this.GroupBox_Search = new System.Windows.Forms.GroupBox();
    this.Button_All = new System.Windows.Forms.Button();
    this.label4 = new System.Windows.Forms.Label();
    this.label3 = new System.Windows.Forms.Label();
    this.label2 = new System.Windows.Forms.Label();
    this.label1 = new System.Windows.Forms.Label();
    this.CheckBox_Attachment = new System.Windows.Forms.CheckBox();
    this.DateTimePicker_End = new System.Windows.Forms.DateTimePicker();
    this.DateTimePicker_Start = new System.Windows.Forms.DateTimePicker();
    this.TextBox_Subject = new System.Windows.Forms.TextBox();
    this.TextBox_To = new System.Windows.Forms.TextBox();
    this.TextBox_From = new System.Windows.Forms.TextBox();
    this.Button_Search = new System.Windows.Forms.Button();
    this.label6 = new System.Windows.Forms.Label();
    this.GroupBox_Emails = new System.Windows.Forms.GroupBox();
    this.Button_Start = new System.Windows.Forms.Button();
    this.Label_State = new System.Windows.Forms.Label();
    this.mainMenu1 = new System.Windows.Forms.MainMenu();
    this.menuItem1 = new System.Windows.Forms.MenuItem();
    this.menuItem3 = new System.Windows.Forms.MenuItem();
    this.menuItem2 = new System.Windows.Forms.MenuItem();
    this.menuItem4 = new System.Windows.Forms.MenuItem();
    this.menuItem5 = new System.Windows.Forms.MenuItem();
    this.GroupBox_Results = new System.Windows.Forms.GroupBox();
    this.ListBox_Results = new System.Windows.Forms.ListBox();
    this.GroupBox_Search.SuspendLayout();
    this.GroupBox_Emails.SuspendLayout();
    this.GroupBox_Results.SuspendLayout();
    this.SuspendLayout();
    //
    // GroupBox_Search
    //
    this.GroupBox_Search.Controls.Add(this.Button_All);
    this.GroupBox_Search.Controls.Add(this.label4);
    this.GroupBox_Search.Controls.Add(this.label3);
    this.GroupBox_Search.Controls.Add(this.label2);
    this.GroupBox_Search.Controls.Add(this.label1);
    this.GroupBox_Search.Controls.Add(this.CheckBox_Attachment);
    this.GroupBox_Search.Controls.Add(this.DateTimePicker_End);
    this.GroupBox_Search.Controls.Add(this.DateTimePicker_Start);
    this.GroupBox_Search.Controls.Add(this.TextBox_Subject);
    this.GroupBox_Search.Controls.Add(this.TextBox_To);
    this.GroupBox_Search.Controls.Add(this.TextBox_From);
    this.GroupBox_Search.Controls.Add(this.Button_Search);
    this.GroupBox_Search.Controls.Add(this.label6);
    this.GroupBox_Search.Location = new System.Drawing.Point(112, 8);
    this.GroupBox_Search.Name = "GroupBox_Search";
    this.GroupBox_Search.Size = new System.Drawing.Size(384, 168);
    this.GroupBox_Search.TabIndex = 20;
```

```
this.GroupBox_Search.TabStop = false;
this.GroupBox_Search.Text = "Search";
//
// Button_All
//
this.Button_All.Location = new System.Drawing.Point(304, 136);
this.Button_All.Name = "Button_All";
this.Button_All.TabIndex = 8;
this.Button_All.Text = "Get All";
this.Button_All.Click += new System.EventHandler(this.Button_All_Click);
//
// label4
//
this.label4.Location = new System.Drawing.Point(216, 88);
this.label4.Name = "label4";
this.label4.Size = new System.Drawing.Size(16, 20);
this.label4.TabIndex = 19;
this.label4.Text = "-";
this.label4.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
//
// label3
//
this.label3.Location = new System.Drawing.Point(8, 88);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(48, 20);
this.label3.TabIndex = 20;
this.label3.Text = "Send";
this.label3.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
//
// label2
//
this.label2.Location = new System.Drawing.Point(8, 64);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(48, 20);
this.label2.TabIndex = 20;
this.label2.Text = "Subject";
this.label2.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
//
// label1
//
this.label1.Location = new System.Drawing.Point(8, 40);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(48, 20);
this.label1.TabIndex = 20;
this.label1.Text = "To";
this.label1.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
//
// CheckBox_Attachment
//
this.CheckBox_Attachment.Location = new System.Drawing.Point(64, 112);
this.CheckBox_Attachment.Name = "CheckBox_Attachment";
this.CheckBox_Attachment.TabIndex = 6;
this.CheckBox_Attachment.Text = "Attachment only";
//
// DateTimePicker_End
//
this.DateTimePicker_End.Location = new System.Drawing.Point(232, 88);
this.DateTimePicker_End.Name = "DateTimePicker_End";
this.DateTimePicker_End.Size = new System.Drawing.Size(144, 20);
this.DateTimePicker_End.TabIndex = 5;
//
// DateTimePicker_Start
//
this.DateTimePicker_Start.Location = new System.Drawing.Point(64, 88);
this.DateTimePicker_Start.Name = "DateTimePicker_Start";
this.DateTimePicker_Start.Size = new System.Drawing.Size(144, 20);
this.DateTimePicker_Start.TabIndex = 4;
//
// TextBox_Subject
//
this.TextBox_Subject.Location = new System.Drawing.Point(64, 64);
this.TextBox_Subject.MaxLength = 100;
this.TextBox_Subject.Name = "TextBox_Subject";
this.TextBox_Subject.Size = new System.Drawing.Size(312, 20);
this.TextBox_Subject.TabIndex = 3;
```

```
this.TextBox_Subject.Text = "";
//
// TextBox_To
//
this.TextBox_To.Location = new System.Drawing.Point(64, 40);
this.TextBox_To.MaxLength = 50;
this.TextBox_To.Name = "TextBox_To";
this.TextBox_To.Size = new System.Drawing.Size(192, 20);
this.TextBox_To.TabIndex = 2;
this.TextBox_To.Text = "";
this.TextBox_To.TextChanged += new System.EventHandler(this.TextBox_To_TextChanged);
//
// TextBox_From
//
this.TextBox_From.Location = new System.Drawing.Point(64, 16);
this.TextBox_From.MaxLength = 50;
this.TextBox_From.Name = "TextBox_From";
this.TextBox_From.Size = new System.Drawing.Size(192, 20);
this.TextBox_From.TabIndex = 1;
this.TextBox_From.Text = "";
//
// Button_Search
//
this.Button_Search.Location = new System.Drawing.Point(224, 136);
this.Button_Search.Name = "Button_Search";
this.Button_Search.TabIndex = 7;
this.Button_Search.Text = "Search";
this.Button_Search.Click += new System.EventHandler(this.Button_Search_Click);
//
// label6
//
this.label6.Location = new System.Drawing.Point(8, 16);
this.label6.Name = "label6";
this.label6.Size = new System.Drawing.Size(48, 20);
this.label6.TabIndex = 20;
this.label6.Text = "From";
this.label6.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
//
// GroupBox_Emails
//
this.GroupBox_Emails.Controls.Add(this.Button_Start);
this.GroupBox_Emails.Controls.Add(this.Label_State);
this.GroupBox_Emails.Location = new System.Drawing.Point(8, 8);
this.GroupBox_Emails.Name = "GroupBox_Emails";
this.GroupBox_Emails.Size = new System.Drawing.Size(96, 72);
this.GroupBox_Emails.TabIndex = 20;
this.GroupBox_Emails.TabStop = false;
this.GroupBox_Emails.Text = "Controls";
//
// Button_Start
//
this.Button_Start.Location = new System.Drawing.Point(8, 40);
this.Button_Start.Name = "Button_Start";
this.Button_Start.TabIndex = 10;
this.Button_Start.Text = "Start";
this.Button_Start.Click += new System.EventHandler(this.Button_Start_Click);
//
// Label_State
//
this.Label_State.Location = new System.Drawing.Point(8, 16);
this.Label_State.Name = "Label_State";
this.Label_State.Size = new System.Drawing.Size(80, 16);
this.Label_State.TabIndex = 20;
this.Label_State.Text = "Stopped!";
//
// mainMenu1
//
this.mainMenu1.MenuItems.AddRange(new System.Windows.Forms.MenuItem[] {

                                            this.menuItem1,

                                            this.menuItem2});
//
// menuItem1
//
```

```
            this.menuItem1.Index = 0;
            this.menuItem1.MenuItems.AddRange(new System.Windows.Forms.MenuItem[] {

                                                    this.menuItem3});
            this.menuItem1.Text = "File";
            //
            // menuItem3
            //
            this.menuItem3.Index = 0;
            this.menuItem3.Text = "Exit";
            this.menuItem3.Click += new System.EventHandler(this.menuItem3_Click);
            //
            // menuItem2
            //
            this.menuItem2.Index = 1;
            this.menuItem2.MenuItems.AddRange(new System.Windows.Forms.MenuItem[] {

                                                    this.menuItem4,

                                                    this.menuItem5});
            this.menuItem2.Text = "Settings";
            //
            // menuItem4
            //
            this.menuItem4.Index = 0;
            this.menuItem4.Text = "General";
            this.menuItem4.Click += new System.EventHandler(this.menuItem4_Click);
            //
            // menuItem5
            //
            this.menuItem5.Index = 1;
            this.menuItem5.Text = "Servers";
            this.menuItem5.Click += new System.EventHandler(this.menuItem5_Click);
            //
            // GroupBox_Results
            //
            this.GroupBox_Results.Controls.Add(this.ListBox_Results);
            this.GroupBox_Results.Location = new System.Drawing.Point(112, 184);
            this.GroupBox_Results.Name = "GroupBox_Results";
            this.GroupBox_Results.Size = new System.Drawing.Size(384, 160);
            this.GroupBox_Results.TabIndex = 20;
            this.GroupBox_Results.TabStop = false;
            this.GroupBox_Results.Text = "Search Results";
            //
            // ListBox_Results
            //
            this.ListBox_Results.Location = new System.Drawing.Point(8, 16);
            this.ListBox_Results.Name = "ListBox_Results";
            this.ListBox_Results.Size = new System.Drawing.Size(368, 134);
            this.ListBox_Results.TabIndex = 9;
            this.ListBox_Results.SelectedIndexChanged += new
System.EventHandler(this.ListBox_Results_SelectedIndexChanged);
            //
            // MainForm
            //
            this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
            this.ClientSize = new System.Drawing.Size(506, 350);
            this.Controls.Add(this.GroupBox_Results);
            this.Controls.Add(this.GroupBox_Search);
            this.Controls.Add(this.GroupBox_Emails);
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;
            this.MaximizeBox = false;
            this.Menu = this.mainMenu1;
            this.Name = "MainForm";
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "Email Management";
            this.GroupBox_Search.ResumeLayout(false);
            this.GroupBox_Emails.ResumeLayout(false);
            this.GroupBox_Results.ResumeLayout(false);
            this.ResumeLayout(false);

        }
        #endregion

        private void Button_Search_Click(object sender, System.EventArgs e)
```

```
        {
            try
            {
                this.ListBox_Results.Items.Clear();
                results = EMSBackend.EMSBackend.SearchEmails(
                    TextBox_From.Text != string.Empty ? TextBox_From.Text : "%",
                    TextBox_To.Text != string.Empty ? TextBox_To.Text : "%",
                    TextBox_Subject.Text != string.Empty ? TextBox_Subject.Text : "%",
                    this.DateTimePicker_Start.Value,
                    this.DateTimePicker_End.Value,
                    this.CheckBox_Attachment.Checked
                    );
                foreach(EMSEmailVO evo in results)
                {
                    this.ListBox_Results.Items.Add(evo);
                }
            }
            catch(Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }


        private void Button_Start_Click(object sender, System.EventArgs e)
        {
            if(state)
            {
                EMSBackend.EMSBackend.StopRecieveEmails();
                Label_State.Text = "Stopped!";
                Button_Start.Text = "Start";
                state = !state;
            }
            else
            {
                EMSBackend.EMSBackend.StartRecieveEmails(EMSConfig.Servers,EMSConfig.Interval);
                Label_State.Text = "Running!";
                Button_Start.Text = "Stop";
                state = !state;
            }

        }

        private void menuItem3_Click(object sender, System.EventArgs e)
        {
            Application.Exit();
        }

        private void menuItem4_Click(object sender, System.EventArgs e)
        {
            SettingsForm sf = new SettingsForm();
            sf.Show();
        }

        private void menuItem5_Click(object sender, System.EventArgs e)
        {
            ServerForm sf = new ServerForm();
            sf.Show();
        }

        private void ListBox_Results_SelectedIndexChanged(object sender, System.EventArgs e)
        {

            EmailForm ef = new EmailForm(results[((ListBox)sender).SelectedIndex]);
            ef.Show();
        }

        private void Button_All_Click(object sender, System.EventArgs e)
        {
            this.ListBox_Results.Items.Clear();
            try
            {
                results = EMSBackend.EMSBackend.SearchEmails("%", "%", "%", DateTime.MinValue,
DateTime.MaxValue, false);
                foreach(EMSEmailVO evo in results)
```

```
                {
                    this.ListBox_Results.Items.Add(evo);
                }
            }
            catch(Exception ex)
            {
                MessageBox.Show(ex.Message);
            }

        }

        private void TextBox_To_TextChanged(object sender, System.EventArgs e)
        {

        }
    }
}
```

14.2.4.2 EmailForm.cs

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;

using EMSBackend.ValueObjects;

namespace EMSGUI
{
    public class EmailForm : System.Windows.Forms.Form
    {
        private System.Windows.Forms.GroupBox GroupBox_Info;
        private System.Windows.Forms.Label label6;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Label Label_From;
        private System.Windows.Forms.Label Label_To;
        private System.Windows.Forms.Label Label_Date;
        private System.Windows.Forms.TextBox TextBox_Body;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Label Label_Subject;
        private System.Windows.Forms.Button Button_Close;
        private System.Windows.Forms.TextBox TextBox_Att;
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.Container components = null;

        public EmailForm(EMSEmailVO email)
        {
            InitializeComponent();
            foreach(EMSMailboxVO mvo in email.From)
            {
                this.Label_From.Text += mvo.Name + " <" + mvo.Address + "> ";
            }

            foreach(EMSMailboxVO mvo in email.To)
            {
                this.Label_To.Text += mvo.Name + " <" + mvo.Address + "> ";
            }

            this.Label_Date.Text = email.Date.ToString("yyyy-MM-dd HH:mm");
            this.Label_Subject.Text = email.Subject;
            this.TextBox_Body.Text = email.HTMLText != null && !email.HTMLText.Equals(string.Empty) ?
email.HTMLText : email.Text;

            foreach(EMSAttachmentVO avo in email.Attachment)
            {
                this.TextBox_Att.Text += avo.Filename + " (" + avo.Size + " bytes)" +
Environment.NewLine;
            }

            this.Text = email.Subject;

        }

        protected override void Dispose( bool disposing )
        {
            if( disposing )
            {
                if(components != null)
                {
                    components.Dispose();
                }
            }
            base.Dispose( disposing );
        }


        #region Windows Form Designer generated code
        /// <summary>
```

```
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    this.GroupBox_Info = new System.Windows.Forms.GroupBox();
    this.Label_Subject = new System.Windows.Forms.Label();
    this.label3 = new System.Windows.Forms.Label();
    this.label2 = new System.Windows.Forms.Label();
    this.label1 = new System.Windows.Forms.Label();
    this.label6 = new System.Windows.Forms.Label();
    this.Label_From = new System.Windows.Forms.Label();
    this.Label_To = new System.Windows.Forms.Label();
    this.Label_Date = new System.Windows.Forms.Label();
    this.TextBox_Body = new System.Windows.Forms.TextBox();
    this.Button_Close = new System.Windows.Forms.Button();
    this.TextBox_Att = new System.Windows.Forms.TextBox();
    this.GroupBox_Info.SuspendLayout();
    this.SuspendLayout();
    //
    // GroupBox_Info
    //
    this.GroupBox_Info.Controls.Add(this.Label_Subject);
    this.GroupBox_Info.Controls.Add(this.label3);
    this.GroupBox_Info.Controls.Add(this.label2);
    this.GroupBox_Info.Controls.Add(this.label1);
    this.GroupBox_Info.Controls.Add(this.label6);
    this.GroupBox_Info.Controls.Add(this.Label_From);
    this.GroupBox_Info.Controls.Add(this.Label_To);
    this.GroupBox_Info.Controls.Add(this.Label_Date);
    this.GroupBox_Info.Location = new System.Drawing.Point(8, 0);
    this.GroupBox_Info.Name = "GroupBox_Info";
    this.GroupBox_Info.Size = new System.Drawing.Size(528, 112);
    this.GroupBox_Info.TabIndex = 0;
    this.GroupBox_Info.TabStop = false;
    //
    // Label_Subject
    //
    this.Label_Subject.Location = new System.Drawing.Point(56, 88);
    this.Label_Subject.Name = "Label_Subject";
    this.Label_Subject.Size = new System.Drawing.Size(464, 20);
    this.Label_Subject.TabIndex = 10;
    this.Label_Subject.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
    //
    // label3
    //
    this.label3.Location = new System.Drawing.Point(8, 88);
    this.label3.Name = "label3";
    this.label3.Size = new System.Drawing.Size(48, 20);
    this.label3.TabIndex = 9;
    this.label3.Text = "Subject";
    this.label3.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
    //
    // label2
    //
    this.label2.Location = new System.Drawing.Point(8, 64);
    this.label2.Name = "label2";
    this.label2.Size = new System.Drawing.Size(48, 20);
    this.label2.TabIndex = 8;
    this.label2.Text = "Date";
    this.label2.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
    //
    // label1
    //
    this.label1.Location = new System.Drawing.Point(8, 40);
    this.label1.Name = "label1";
    this.label1.Size = new System.Drawing.Size(48, 20);
    this.label1.TabIndex = 7;
    this.label1.Text = "To";
    this.label1.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
    //
    // label6
    //
    this.label6.Location = new System.Drawing.Point(8, 16);
    this.label6.Name = "label6";
```

```
this.label6.Size = new System.Drawing.Size(48, 20);
this.label6.TabIndex = 6;
this.label6.Text = "From";
this.label6.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
//
// Label_From
//
this.Label_From.Location = new System.Drawing.Point(56, 16);
this.Label_From.Name = "Label_From";
this.Label_From.Size = new System.Drawing.Size(464, 20);
this.Label_From.TabIndex = 6;
this.Label_From.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
//
// Label_To
//
this.Label_To.Location = new System.Drawing.Point(56, 40);
this.Label_To.Name = "Label_To";
this.Label_To.Size = new System.Drawing.Size(464, 20);
this.Label_To.TabIndex = 6;
this.Label_To.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
//
// Label_Date
//
this.Label_Date.Location = new System.Drawing.Point(56, 64);
this.Label_Date.Name = "Label_Date";
this.Label_Date.Size = new System.Drawing.Size(464, 20);
this.Label_Date.TabIndex = 6;
this.Label_Date.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
//
// TextBox_Body
//
this.TextBox_Body.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;
this.TextBox_Body.Location = new System.Drawing.Point(8, 120);
this.TextBox_Body.Multiline = true;
this.TextBox_Body.Name = "TextBox_Body";
this.TextBox_Body.ReadOnly = true;
this.TextBox_Body.ScrollBars = System.Windows.Forms.ScrollBars.Both;
this.TextBox_Body.Size = new System.Drawing.Size(528, 192);
this.TextBox_Body.TabIndex = 11;
this.TextBox_Body.TabStop = false;
this.TextBox_Body.Text = "";
//
// Button_Close
//
this.Button_Close.Location = new System.Drawing.Point(464, 368);
this.Button_Close.Name = "Button_Close";
this.Button_Close.TabIndex = 1;
this.Button_Close.Text = "Close";
this.Button_Close.Click += new System.EventHandler(this.Button_Close_Click);
//
// TextBox_Att
//
this.TextBox_Att.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;
this.TextBox_Att.Location = new System.Drawing.Point(8, 320);
this.TextBox_Att.Multiline = true;
this.TextBox_Att.Name = "TextBox_Att";
this.TextBox_Att.ReadOnly = true;
this.TextBox_Att.ScrollBars = System.Windows.Forms.ScrollBars.Both;
this.TextBox_Att.Size = new System.Drawing.Size(528, 40);
this.TextBox_Att.TabIndex = 12;
this.TextBox_Att.TabStop = false;
this.TextBox_Att.Text = "";
//
// EmailForm
//
this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
this.ClientSize = new System.Drawing.Size(544, 400);
this.Controls.Add(this.TextBox_Att);
this.Controls.Add(this.Button_Close);
this.Controls.Add(this.TextBox_Body);
this.Controls.Add(this.GroupBox_Info);
this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;
this.MaximizeBox = false;
this.Name = "EmailForm";
this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
```

```
            this.Text = "Email";
            this.GroupBox_Info.ResumeLayout(false);
            this.ResumeLayout(false);

        }
        #endregion

        private void Button_Close_Click(object sender, System.EventArgs e)
        {
            this.Close();
        }

    }
}
```

14.2.4.3 ServerForm.cs

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;

using EMSBackend;
using EMSBackend.ValueObjects;

namespace EMSGUI
{
    public class ServerForm : System.Windows.Forms.Form
    {
        private System.Windows.Forms.GroupBox GroupBox_Servers;
        private System.Windows.Forms.GroupBox GroupBox_Settings;
        private System.Windows.Forms.ListBox ListBox_Servers;
        private System.Windows.Forms.TextBox TextBox_Name;
        private System.Windows.Forms.TextBox TextBox_Server;
        private System.Windows.Forms.TextBox TextBox_User;
        private System.Windows.Forms.TextBox TextBox_Password;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.TextBox TextBox_Port;
        private System.Windows.Forms.Button Button_Add;
        private System.Windows.Forms.Button Button_Remove;
        private System.Windows.Forms.Button Button_Save;

        private System.ComponentModel.Container components = null;

        public ServerForm()
        {
            InitializeComponent();
            InitializeData();
        }


        protected override void Dispose( bool disposing )
        {
            if( disposing )
            {
                if(components != null)
                {
                    components.Dispose();
                }
            }
            base.Dispose( disposing );
        }


        #region Windows Form Designer generated code
        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.GroupBox_Servers = new System.Windows.Forms.GroupBox();
            this.ListBox_Servers = new System.Windows.Forms.ListBox();
            this.GroupBox_Settings = new System.Windows.Forms.GroupBox();
            this.TextBox_Port = new System.Windows.Forms.TextBox();
            this.label5 = new System.Windows.Forms.Label();
            this.label4 = new System.Windows.Forms.Label();
            this.label3 = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.label1 = new System.Windows.Forms.Label();
            this.TextBox_Password = new System.Windows.Forms.TextBox();
            this.TextBox_User = new System.Windows.Forms.TextBox();
            this.TextBox_Server = new System.Windows.Forms.TextBox();
            this.TextBox_Name = new System.Windows.Forms.TextBox();
```

```
            this.Button_Add = new System.Windows.Forms.Button();
            this.Button_Remove = new System.Windows.Forms.Button();
            this.Button_Save = new System.Windows.Forms.Button();
            this.GroupBox_Servers.SuspendLayout();
            this.GroupBox_Settings.SuspendLayout();
            this.SuspendLayout();
            //
            // GroupBox_Servers
            //
            this.GroupBox_Servers.Controls.Add(this.ListBox_Servers);
            this.GroupBox_Servers.Location = new System.Drawing.Point(8, 8);
            this.GroupBox_Servers.Name = "GroupBox_Servers";
            this.GroupBox_Servers.Size = new System.Drawing.Size(144, 264);
            this.GroupBox_Servers.TabIndex = 0;
            this.GroupBox_Servers.TabStop = false;
            this.GroupBox_Servers.Text = "Servers";
            //
            // ListBox_Servers
            //
            this.ListBox_Servers.Location = new System.Drawing.Point(8, 16);
            this.ListBox_Servers.Name = "ListBox_Servers";
            this.ListBox_Servers.Size = new System.Drawing.Size(128, 238);
            this.ListBox_Servers.TabIndex = 1;
            this.ListBox_Servers.SelectedIndexChanged += new
System.EventHandler(this.ListBox_Servers_SelectedIndexChanged);
            //
            // GroupBox_Settings
            //
            this.GroupBox_Settings.Controls.Add(this.TextBox_Port);
            this.GroupBox_Settings.Controls.Add(this.label5);
            this.GroupBox_Settings.Controls.Add(this.label4);
            this.GroupBox_Settings.Controls.Add(this.label3);
            this.GroupBox_Settings.Controls.Add(this.label2);
            this.GroupBox_Settings.Controls.Add(this.label1);
            this.GroupBox_Settings.Controls.Add(this.TextBox_Password);
            this.GroupBox_Settings.Controls.Add(this.TextBox_User);
            this.GroupBox_Settings.Controls.Add(this.TextBox_Server);
            this.GroupBox_Settings.Controls.Add(this.TextBox_Name);
            this.GroupBox_Settings.Location = new System.Drawing.Point(160, 8);
            this.GroupBox_Settings.Name = "GroupBox_Settings";
            this.GroupBox_Settings.Size = new System.Drawing.Size(168, 264);
            this.GroupBox_Settings.TabIndex = 1;
            this.GroupBox_Settings.TabStop = false;
            this.GroupBox_Settings.Text = "Server Settings";
            //
            // TextBox_Port
            //
            this.TextBox_Port.Location = new System.Drawing.Point(8, 136);
            this.TextBox_Port.MaxLength = 4;
            this.TextBox_Port.Name = "TextBox_Port";
            this.TextBox_Port.Size = new System.Drawing.Size(40, 20);
            this.TextBox_Port.TabIndex = 4;
            this.TextBox_Port.Text = "";
            //
            // label5
            //
            this.label5.Location = new System.Drawing.Point(8, 120);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(100, 16);
            this.label5.TabIndex = 0;
            this.label5.Text = "Port";
            //
            // label4
            //
            this.label4.Location = new System.Drawing.Point(8, 216);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(100, 16);
            this.label4.TabIndex = 0;
            this.label4.Text = "Password";
            //
            // label3
            //
            this.label3.Location = new System.Drawing.Point(8, 168);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(100, 16);
```

```
this.label3.TabIndex = 0;
this.label3.Text = "User";
//
// label2
//
this.label2.Location = new System.Drawing.Point(8, 72);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(100, 16);
this.label2.TabIndex = 0;
this.label2.Text = "Server URL";
//
// label1
//
this.label1.Location = new System.Drawing.Point(8, 24);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(100, 16);
this.label1.TabIndex = 0;
this.label1.Text = "Name";
//
// TextBox_Password
//
this.TextBox_Password.Location = new System.Drawing.Point(8, 232);
this.TextBox_Password.Name = "TextBox_Password";
this.TextBox_Password.Size = new System.Drawing.Size(144, 20);
this.TextBox_Password.TabIndex = 6;
this.TextBox_Password.Text = "";
//
// TextBox_User
//
this.TextBox_User.Location = new System.Drawing.Point(8, 184);
this.TextBox_User.Name = "TextBox_User";
this.TextBox_User.Size = new System.Drawing.Size(144, 20);
this.TextBox_User.TabIndex = 5;
this.TextBox_User.Text = "";
//
// TextBox_Server
//
this.TextBox_Server.Location = new System.Drawing.Point(8, 88);
this.TextBox_Server.Name = "TextBox_Server";
this.TextBox_Server.Size = new System.Drawing.Size(144, 20);
this.TextBox_Server.TabIndex = 3;
this.TextBox_Server.Text = "";
//
// TextBox_Name
//
this.TextBox_Name.Location = new System.Drawing.Point(8, 40);
this.TextBox_Name.MaxLength = 20;
this.TextBox_Name.Name = "TextBox_Name";
this.TextBox_Name.Size = new System.Drawing.Size(104, 20);
this.TextBox_Name.TabIndex = 2;
this.TextBox_Name.Text = "";
//
// Button_Add
//
this.Button_Add.Location = new System.Drawing.Point(8, 280);
this.Button_Add.Name = "Button_Add";
this.Button_Add.TabIndex = 2;
this.Button_Add.Text = "Add Server";
this.Button_Add.Click += new System.EventHandler(this.button1_Click);
//
// Button_Remove
//
this.Button_Remove.Location = new System.Drawing.Point(248, 280);
this.Button_Remove.Name = "Button_Remove";
this.Button_Remove.TabIndex = 3;
this.Button_Remove.Text = "Remove";
this.Button_Remove.Click += new System.EventHandler(this.button2_Click);
//
// Button_Save
//
this.Button_Save.Location = new System.Drawing.Point(160, 280);
this.Button_Save.Name = "Button_Save";
this.Button_Save.TabIndex = 4;
this.Button_Save.Text = "Save";
this.Button_Save.Click += new System.EventHandler(this.Button_Save_Click);
```

```
            //
            // ServerForm
            //
            this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
            this.ClientSize = new System.Drawing.Size(330, 312);
            this.Controls.Add(this.Button_Save);
            this.Controls.Add(this.Button_Remove);
            this.Controls.Add(this.Button_Add);
            this.Controls.Add(this.GroupBox_Settings);
            this.Controls.Add(this.GroupBox_Servers);
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedToolWindow;
            this.Name = "ServerForm";
            this.ShowInTaskbar = false;
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "Servers";
            this.TopMost = true;
            this.GroupBox_Servers.ResumeLayout(false);
            this.GroupBox_Settings.ResumeLayout(false);
            this.ResumeLayout(false);

        }
        #endregion

        private void InitializeData()
        {
            EMSServerVO[] servers = EMSConfig.Servers;
            foreach(EMSServerVO s in servers)
            {
                this.ListBox_Servers.Items.Add(s);
            }
        }

        private void ListBox_Servers_SelectedIndexChanged(object sender, System.EventArgs e)
        {
            this.TextBox_Name.Text = ((EMSServerVO)((ListBox)sender).SelectedItem).Name;
            this.TextBox_Server.Text = ((EMSServerVO)((ListBox)sender).SelectedItem).Server;
            this.TextBox_Port.Text = ((EMSServerVO)((ListBox)sender).SelectedItem).Port.ToString();
            this.TextBox_User.Text = ((EMSServerVO)((ListBox)sender).SelectedItem).User;
            this.TextBox_Password.Text = ((EMSServerVO)((ListBox)sender).SelectedItem).Password;
        }

        private void button2_Click(object sender, System.EventArgs e)
        {

        }

        private void button1_Click(object sender, System.EventArgs e)
        {

        }

        private void Button_Save_Click(object sender, System.EventArgs e)
        {

        }
    }
}
```

14.2.4.4 SettingsForm.cs

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;

using EMSBackend;

namespace EMSGUI
{
    public class SettingsForm : System.Windows.Forms.Form
    {
        private System.Windows.Forms.GroupBox GroupBox_Settings;
        private System.Windows.Forms.Button Button_Save;
        private System.Windows.Forms.TextBox TextBox_SettingInterval;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.CheckBox CheckBox_DeleteEmails;
        private System.ComponentModel.Container components = null;

        public SettingsForm()
        {
            InitializeComponent();
            InitializeData();
        }


        protected override void Dispose( bool disposing )
        {
            if( disposing )
            {
                if(components != null)
                {
                    components.Dispose();
                }
            }
            base.Dispose( disposing );
        }


        #region Windows Form Designer generated code
        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.GroupBox_Settings = new System.Windows.Forms.GroupBox();
            this.label2 = new System.Windows.Forms.Label();
            this.CheckBox_DeleteEmails = new System.Windows.Forms.CheckBox();
            this.label1 = new System.Windows.Forms.Label();
            this.TextBox_SettingInterval = new System.Windows.Forms.TextBox();
            this.Button_Save = new System.Windows.Forms.Button();
            this.GroupBox_Settings.SuspendLayout();
            this.SuspendLayout();
            //
            // GroupBox_Settings
            //
            this.GroupBox_Settings.Controls.Add(this.label2);
            this.GroupBox_Settings.Controls.Add(this.CheckBox_DeleteEmails);
            this.GroupBox_Settings.Controls.Add(this.label1);
            this.GroupBox_Settings.Controls.Add(this.TextBox_SettingInterval);
            this.GroupBox_Settings.Controls.Add(this.Button_Save);
            this.GroupBox_Settings.Location = new System.Drawing.Point(8, 8);
            this.GroupBox_Settings.Name = "GroupBox_Settings";
            this.GroupBox_Settings.Size = new System.Drawing.Size(280, 264);
            this.GroupBox_Settings.TabIndex = 3;
            this.GroupBox_Settings.TabStop = false;
            this.GroupBox_Settings.Text = "Settings";
            //
            // label2
            //
```

```
    this.label2.Location = new System.Drawing.Point(56, 48);
    this.label2.Name = "label2";
    this.label2.Size = new System.Drawing.Size(200, 20);
    this.label2.TabIndex = 4;
    this.label2.Text = "Delete emails on server";
    this.label2.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
    //
    // CheckBox_DeleteEmails
    //
    this.CheckBox_DeleteEmails.Location = new System.Drawing.Point(32, 48);
    this.CheckBox_DeleteEmails.Name = "CheckBox_DeleteEmails";
    this.CheckBox_DeleteEmails.Size = new System.Drawing.Size(16, 24);
    this.CheckBox_DeleteEmails.TabIndex = 3;
    //
    // label1
    //
    this.label1.Location = new System.Drawing.Point(56, 24);
    this.label1.Name = "label1";
    this.label1.Size = new System.Drawing.Size(200, 20);
    this.label1.TabIndex = 2;
    this.label1.Text = "Seconds between calls to mailserver";
    this.label1.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
    //
    // TextBox_SettingInterval
    //
    this.TextBox_SettingInterval.Location = new System.Drawing.Point(8, 24);
    this.TextBox_SettingInterval.MaxLength = 4;
    this.TextBox_SettingInterval.Name = "TextBox_SettingInterval";
    this.TextBox_SettingInterval.Size = new System.Drawing.Size(40, 20);
    this.TextBox_SettingInterval.TabIndex = 1;
    this.TextBox_SettingInterval.Text = "";
    //
    // Button_Save
    //
    this.Button_Save.Location = new System.Drawing.Point(200, 232);
    this.Button_Save.Name = "Button_Save";
    this.Button_Save.TabIndex = 0;
    this.Button_Save.Text = "Save";
    this.Button_Save.Click += new System.EventHandler(this.Button_Save_Click);
    //
    // SettingsForm
    //
    this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
    this.ClientSize = new System.Drawing.Size(294, 276);
    this.Controls.Add(this.GroupBox_Settings);
    this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedToolWindow;
    this.MinimizeBox = false;
    this.Name = "SettingsForm";
    this.ShowInTaskbar = false;
    this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
    this.Text = "General Settings";
    this.TopMost = true;
    this.GroupBox_Settings.ResumeLayout(false);
    this.ResumeLayout(false);

}
#endregion

private void InitializeData()
{
    this.TextBox_SettingInterval.Text = EMSConfig.Interval.ToString();
    this.CheckBox_DeleteEmails.Checked = EMSConfig.DeleteEmails;
}

private void Button_Save_Click(object sender, System.EventArgs e)
{
    try
    {
        int interval = Convert.ToInt32(this.TextBox_SettingInterval.Text);
        if(interval < 0 || interval > 9999)
        {
            throw new Exception();
        }

        EMSConfig.SetSystemConfiguration("Interval", interval.ToString());
```

```
                EMSConfig.SetSystemConfiguration("DeleteEmails",
this.CheckBox_DeleteEmails.Checked.ToString());
                MessageBox.Show("Settings saved!\n");
            }
            catch(Exception ex)
            {
                MessageBox.Show("One or more settings contain errors.\nSettings not saved!");
            }
        }
    }
}
```

14.2.5 Andet kildekode

14.2.5.1 System.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <appSettings>
        <add key="Interval" value="3"/>
        <add key="DeleteEmails" value="true"/>
        <add key="DBType" value="MySQL"/>
        <add key="DSN" value="server=db.fubunet.dk; user id=web90792; password=4r2n9f9j85;
database=web90792; pooling=false;" />
    </appSettings>
    <serverSettings>
        <add name="Server1" server="mail1.stofanet.dk" port="110" user="3185204m001"
password="jaop5496"/>
        <add name="Server2" server="localhost" port="110" user="TestUser" password="123456"/>
    </serverSettings>
</configuration>
```

14.2.5.2 CreateScript for DataBase Tabel

Dette createscript er optimeret for MySQL version 4.

```
CREATE TABLE `EMSEmail` (
  `ID` int(11) NOT NULL auto_increment,
  `EmailFrom` varchar(50) NOT NULL default '',
  `EmailTo` varchar(50) NOT NULL default '',
  `EmailSubject` varchar(100) NOT NULL default '',
  `EmailData` longtext NOT NULL,
  `EmailDate` datetime NOT NULL default '0000-00-00 00:00:00',
  `HasAtt` binary(1) NOT NULL default '',
  PRIMARY KEY  (`ID`),
  KEY `EmailDate` (`EmailDate`),
  KEY `HasAtt` (`HasAtt`),
  FULLTEXT KEY `EmailFrom` (`EmailFrom`),
  FULLTEXT KEY `EmailTo` (`EmailTo`),
  FULLTEXT KEY `EmailSubject` (`EmailSubject`)
) TYPE=MyISAM AUTO_INCREMENT=1 ;
```