HiDocuments

Rapporten er udarbejdet af:
Christian Stokbro

Vejleder(e):
Hans Bruun – DTU
Line K. Olsen – HiQ Wise

IMM•DTU
Danmarks Tekniske Universitet
Richard Petersens Plads
Bygning 321
2800 Kgs. Lyngby
Denmark

www.imm.dtu.dk

Tel: (+45) 45 25 33 51

Udgivelsesdato:	4. august 2006
Klasse:	Offentligt
Udgave:	1. udgave - IMM-B.ENG.-2006-56
Bemærkninger:	Denne rapport er indleveret som led i opfyldelse af kravene for opnåelse af Bachelelor på Danmarks Tekniske Universitet. Rapporten repræsenterer 15 ECTS point.

Indholdsfortegnelse

Forord

Som Bachelor projekt har jeg udviklet et dokumentstyringssystem. Projektet er blevet udarbejdet over 10 uger, og består af denne rapport, samt det tilhørende produkt.
Det skal ved hjælp af applikationen være muligt at oprette diverse informationer om et dokument. Produktet er udviklet for firmaet HiQ Wise A/S, der er et konsulent firma inden for IT branchen.
Da der ikke var nogen krav fra firmaets side om, hvilket sprog applikationen skulle udvikles i, har jeg valgt at bruge C#, da jeg gerne selv vil arbejde med dette sprog frem over. Der var heller ikke nogen krav til hvilken database der skulle bruges, derfor har jeg valgt at bruge Microsoft Sql Server, da denne passer godt sammen med C#.

Projektplan

Indledning

Document Management (dokument styring) er en form for Content Management. Content Management er en vigtig teknologi til at håndtere udviklingen af digital informationer.
Document Mangement bruges til at samle informationer om dokumenter i en virksomhed. For hvert dokument bliver det dokumenteret, hvem der har oprettet dokumentet, hvem der har arbejdet på det og hvad der er den nyeste version. Der er også være funktioner til at checke dokumenter ud, så andre ikke kan arbejde på dem, og checke dem ind igen med de ændringer der er foretaget. Dette sikrer at der ikke opstår to forskellige instanser af et dokument, men at der kun er et samlet, med alle ændringer.

Problemformulering

Problemstilling

HiQ Wise har allerede et versions styrings system, ved navn Subversion. Dette system har til opgave at gemme at firmaets dokumenter, samt give mulighed for at checke disse ud og ind. Systemet gemmer alle versioner af dokumenterne.
Der skal udvikles et system der virker sideløbende med Subversion. Dette system skal gemme informationer om dokumenterne, som Subversion ikke indeholder. Disse informationer skal gemmes i en fælles database, og skal så kunne hentes ved hjælp af en applikation der kører på brugernes computer. Systemet skal have en mindre integration til Subversion, da det skal være muligt at linke fra disse registreringer til de relevante dokumenter i Subversion.

Problemafgrænsning

Systemet skal være delt op i en database og en applikation, der kan tilgå oplysningerne i denne. Applikationen laves i C# og databasen i MS SQL. Det skal være muligt at overføre firmaets nuværende database med informationer om dokumenter, til den nye database.
Applikationen skal kunne hente alle informationer ud fra databasen, samt give brugeren mulighed for at søge i disse.

Det skal være muligt at logge ind i systemet, og derved have flere rettigheder. Der skal være muligt at definere om brugere er administratorer eller almindelige brugere.
Brugere, der er logget ind i systemet, skal have mulighed for at oprette nye registreringer, samt opdatere gamle registreringer.
Administratorer i systemet skal have mulighed for at oprette nye brugere, samt slette allerede oprettede brugere.
Det skal også være muligt for administratorer at oprettes samt slette forskellige standard typer af registreringer.
Det skal også være muligt at eksportere specifikke registreringer til en simpel tekst fil så de f.eks. kan importeres til MS Excel.

Kravspecifikation

1. Introduktion
   - *1.1.* Formål: *At fastlægge hvilke krav der stilles til dokumentstyringssystemet der udvikles til HiQ Wise A/S. Disse krav er til prototypen af systemet.*
   - 1.2. Produktet: *Produktet bliver udviklet til HiQ Wise A/S og er lavet så det passer hertil, men da det er muligt at systemet senere skal udbredes til andre HiQ sites, og produktet skal designes ud fra dette.*
   - 1.3. Definitioner
     - 1.3.1. Bruger: *En bruger er den person der benytter systemet på et givent tidspunkt.*
     - 1.3.2. Administrator: *Er en bruger med adgang til alle funktioner i systemet.*
     - 1.3.3. Registrering: *Er navnet på den samling af informationer der knytter sig til hver enkelt dokument.*
2. Generel beskrivelse
   - 2.1. Produktets formål: *Systemet skal gøre brugere i stand til at oprette informationer om firmaets dokumenter, samt ændre og søge på disse. Brugergrænsefladen vil være en applikation, der køres på en Windows maskine.*
   - 2.2. Produktets funktioner:
     - 2.2.1. Vis registreringer: *Det skal være muligt at vise allerede oprettede registreringer i systemet. Det skal også være muligt at søge på disse ud fra forskellige kriterier.*

2.2.2. Opret registrering: *Denne funktion skal oprette en ny registrering ud fra brugerens input.*

2.2.3. Opdater registrering: *Denne funktion opdater en registrering og markerer den gamle version som ugyldig.*

2.2.4. Slet registrering: *Denne funktion skal markere en registrering som slettet.*

2.2.5. Makuler registrering: *Denne funktion fjerner alle informationer om det valgte dokument fra databasen.*

2.2.6. Login: *Bruges til at logge brugere ind i systemet.*

2.2.7. Typer: *Denne funktion viser alle de oprettede typer, grupper og kategorie,r der oprettet.*

2.2.8. Opret typer: *Dette er 3 funktioner, der opretter de forskellige typer dokumenter der findes i systemet. Det vil kun være administratorer, der har adgang til denne.*

2.2.9. Slet typer: *Dette er 3 forskellige funktioner der sletter den pågældende type. Denne funktion kan kun bruges af administratorer*

2.2.10. Opret bruger: *Administratorer vil med denne funktion kunne oprette nye brugere i systemet.*

2.2.11. Slet bruger: *Denne funktion bruges af administratorer til at fjerne brugere fra systemet*

2.2.12. Eksporter: *Denne funktion bruges til at kopiere alle informationer fra de valgte registreringer til en tekst fil.*

2.3. Brugere: *Det vil være ansatte hos HiQ Wise A/S der er systemets brugere.*

2.4. Forudsætninger

2.4.1. Windows XP: *Maskinen som brugeren bruger, forudsættes at have Windows XP eller nyere installeret.*

2.4.2. Microsoft .Net 2.0: *Det forudsætte at .Net 2.0 eller nyere er installeret.*

2.4.3. TortoiseSVN: *Der skal være installeret TortoiseSVN på maskinen, for at åbne Subversion links. Dette kan findes på http://tortoisesvn.tigris.org/.*

3. Specifikke krav

3.1. Grænseflader:

3.1.1. Brugergrænseflade: *Denne designes så den er simpel og overskuelig for brugeren.*

3.2. To delt: *Dokumentstyringssystemet skal være todelt og bestå af en database og en applikation. Det skal være muligt at opdatere applikationen uden at ændre i databasen, så vidt dette er muligt.*

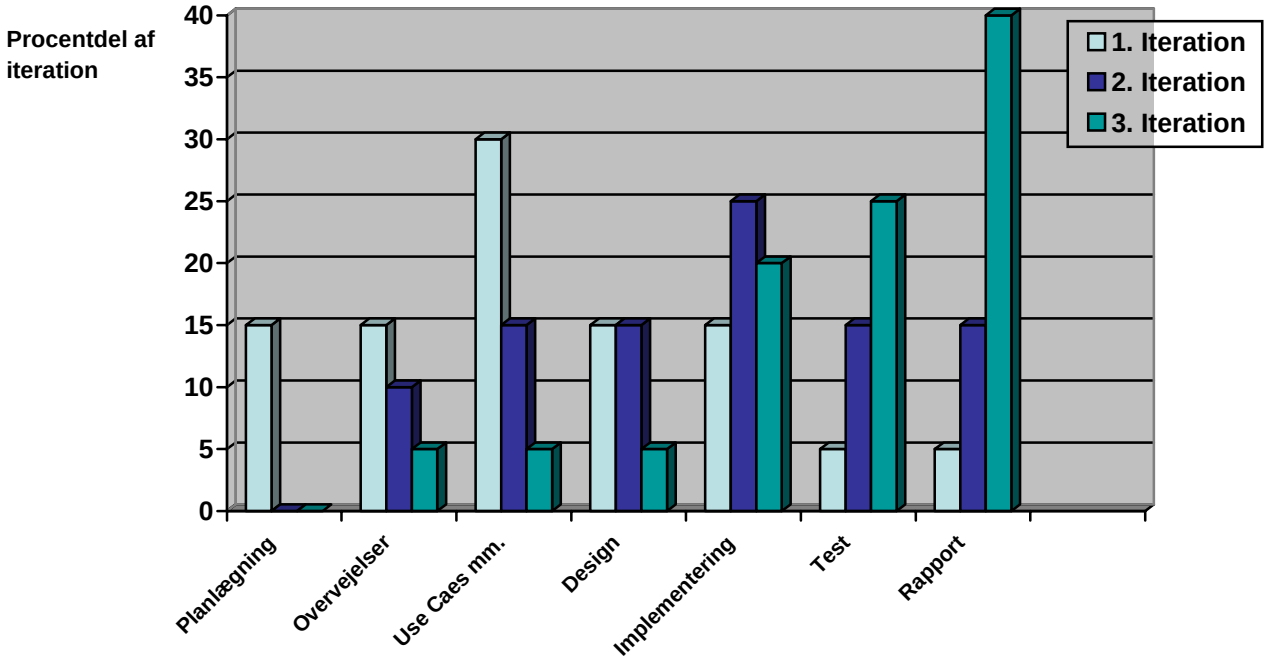
3.3. Sikkerhed: *Databasen skal være beskyttet af et kodeord. Ud over dette skal brugere at systemet være logget ind, for at få adgang til de fleste funktioner.*

3.4. Dokument ID: *Der skal være et specielt dokument ID tilknyttet til hver registrering. Dette ID skal bestå af en kategori, en gruppe samt en type. Ud over dette skal der indikeres hvorvidt dette er et indgående eller udgående dokument. ID'et skal også bestå af en indikation om hvilken konfidentialitet dokumentet har. Der skal også være et fortløbende nummer tilknyttet ID'et, idet dette giver et unikt ID.*

3.5. Import af gammel database: *Det skal være muligt at importere den database HiQ Wise allerede bruger med dokument information.*

Iterationsplan

Se bilag 2 for tidsplan udarbejdet i Microsoft Project.



Det ovenstående diagram viser den tid jeg regner med at fordele min tid under projektet. Jeg har valgt at benytte mig af tre iterationer, da jeg mener at dette vil passe bedst til projektet.

Den første iteration fokuserer på planlægningen, overvejelser samt analyse. I analysen vil jeg bruge Use Case og andre UML værktøjer til at beskrive og klargøre de funktioner der skal udarbejdes. I denne iteration vil jeg også starte på design og implementering af systemet. I første iteration vil jeg også udvikle basis designet til databasen. Dette vil betyde det skal være mulig at gemme registreringer i databasen, samt hvad der ellers vil være brug for fra applikations side. Første iteration vil efter planen være færdig når det er muligt at oprette og se registreringer i systemet, samt slette disse.
Anden iteration vil indeholde videre design og implementering af databasen og applikationen. Fokus i denne iteration er de administrative funktioner, som bruger administration samt en mulighed for at logge ind i systemet.
I tredje og sidste iteration vil de sidste dele af systemet blive implementeret. Dette er f.eks. integrationen med Subversion, samt andre funktioner der kan udarbejdes i løbet af projektet. Det vigtigste i tredje iteration vil dog være testningen af systemet, samt udarbejdningen af den færdig rapport. Dette vil have større betydning en implementeringen af funktioner, hvis der skulle opstå komplikationer undervejs og dette være tilfældet

Risikostyring

Dette afsnit handler om hvilke risici der kan være under projektet.

Person risici

I tilfælde af sygdom vil det stadig være muligt at arbejde på projektet, da jeg er alene om det. I tilfælde af en alvorlig sygdom kan det dog være at projektet skal udskydes.

Data risici

Da der kun er en person på dette projekt er der ikke så mange data risici at tage hensyn til. Da det meste af arbejdet foregår på en bærbar vil den nyeste version altid være at finde her. Ud over dette vil der jævnligt blive taget backup både på denne, samt på en USB key og på en anden computer.

Analyse

Use Cases

Følgende afsnit omhandler de Use Cases der er relevante for systemet, og dettes interaktion med brugerne. Følgende Use Cases findes herunder:

- Registreringsliste
- Opret registrering
- Slet registrering
- Login
- Listning af typer
- Undertyper
- Administration af typer
- Bruger administration
- Eksporter

USE CASE 1	Registreringsliste.	
Formål	At vise systemets registreringer.	
Scope	Det skal virke i HiDocuments	
Forudsætning	SQL Serveren virker.	
Success End Condition	Brugeren vil få vist en list over registreringer ud fra de kriterier denne har specificeret.	
Failed End Condition	Hvis der ikke kommer nogen liste over registreringer.	
Primær Aktør **Sekundær Aktør**	Bruger Ingen	
Trigger	Applikation bliver åbnet / bruger ændrer på listens kriterier.	
DESCRIPTION	**Step**	**Action**

	1	Programmet starter
	2	Liste over alle registreringer hentes i databasen og disse vises i programmet
EXTENSIONS	**Step**	**Branching Action**
		ingen
SUB-VARIATIONS		**Branching Action**
	1a	Bruger vælger at liste registreringer ud fra nye specifikationer.

USE CASE 2	Opret registrering	
Formål	At oprette en ny registrering.	
Scope	Det skal virke i HiDocuments	
Forudsætning	SQL Serveren virker.	
Success End Condition	Det vil være muligt for brugeren at oprette en ny registrering.	
Failed End Condition	Det vil ikke være muligt for brugeren oprette en registrering.	
Primær Aktør **Sekundær Aktør**	Bruger Ingen	
Trigger	Nyt dokument	
DESCRIPTION	**Step**	**Action**
	1	Nyt dokument, skal registreres.
	2	Registreringen oprettes
	3	Registreringen vises i registreringslisten hvis dette er aktuelt.
EXTENSIONS	**Step**	**Branching Action**
	2a	Hvis brugeren ikke er logget ind gøres dette.
SUB-VARIATIONS		**Branching Action**

	2a	Brugeren har valgt at opdatere en allerede eksisterende registrering
	2b	Brugeren bliver bedt om at bekræfte dette valg.
	2c	Hvis brugeren har bekræftet vil den nye registrering blive oprettet i databasen, og den gamle vil blive sat til at være ugyldig.

USE CASE 3	Slet registrering.	
Formål	At markere en registrering som slettet.	
Scope	Det skal virke i HiDocuments	
Forudsætning	SQL Serveren virker.	
Success End Condition	Det vil være muligt for brugeren at slette en allerede oprettet rergistrering.	
Failed End Condition	Det vil ikke være muligt for brugeren at slette registreringen.	
Primær Aktør **Sekundær Aktør**	Bruger Ingen	
Trigger	Dokument skal slettes	
DESCRIPTION	**Step**	**Action**
	1	Brugeren vælger at slette det valgte dokument
	2	Brugeren bekræfter at dokumentet skal slettes
	3	Registreringen bliver markeret som slettet i databasen
EXTENSIONS	**Step**	**Branching Action**
	1a	Hvis brugeren ikke er logget ind vil det ikke være muligt at udføre denne handling.
SUB-VARIATIONS		**Branching Action**
	1a	Hvis brugeren vil fjerne registreringen fra

		databasen, bruges makuler registrering i stedet.

USE CASE 4	Login	
Formål	At logge ind i systemet	
Scope	Det skal virke i HiDocuments	
Forudsætning	SQL Serveren virker.	
Success End Condition	Når brugeren bliver logget ind i systemet og får adgang til de funktioner dette kræver.	
Failed End Condition	Hvis brugeren ikke bliver logget ind.	
Primær Aktør **Sekundær Aktør**	Bruger Ingen	
Trigger	Login vindue bliver åbnet	
DESCRIPTION	**Step**	**Action**
	1	Brugeren indtaster brugernavn og password i den relevante form.
	2	Hvis brugernavn og password bliver godkendt af systemet, får brugeren adgang til flere funktioner ud fra dennes brugerstatus.
EXTENSIONS	**Step**	**Branching Action**
	1a	Hvis brugeren taster forkert information ind, vil denne blive bedt om at indtaste information igen.

USE CASE 5	Listning af typer
Formål	At liste de dokument typer der er oprettet i systemet
Scope	Det skal virke i HiDocuments
Forudsætning	SQL Serveren virker.
Success End	Alle dokument typer i systemet bliver listet.

Condition		
Failed End Condition	Hvis dokument typer der er oprettet i systemet ikke bliver vist.	
Primær Aktør **Sekundær Aktør**	Administrator Ingen	
Trigger	Dokument type vindue bliver åbnet.	
DESCRIPTION	**Step**	**Action**
	1	Dokument type vindue bliver vist.
	2	Alle dokument typer bliver hentet fra databasen og vist i en liste.
EXTENSIONS	**Step**	**Branching Action**
	1a	Hvis brugeren ikke er logget ind som administrator vil denne blive bedt om dette.
	2a	Det er muligt at slette en type ved at markere denne i listen og vælge slet funktionen.

USE CASE 6	Undertyper	
Formål	At administrere under typerne i systemet	
Scope	Det skal virke i HiDocuments	
Forudsætning	SQL Serveren virker.	
Success End Condition	Administratorer kan oprette og slette undertyper.	
Failed End Condition	Hvis det ikke er muligt at oprette eller slette en undertype.	
Primær Aktør **Sekundær Aktør**	Administrator Ingen	
Trigger	Undertype oprettes eller slettes.	
DESCRIPTION	**Step**	**Action**
	1	En af undertype formene åbnes.

	2	En liste over alle oprettede undertyper af den valgte undertype vises.
EXTENSIONS	**Step**	**Branching Action**
	2a	Brugeren kan markere en undertype og slette denne.
SUB-VARIATIONS		**Branching Action**
	1a	Der findes tre undertyper i systemet: Type, Kategori og Gruppe.

USE CASE 7	Administration af typer	
Formål	At administrere systemets typer	
Scope	Det skal virke i HiDocuments	
Forudsætning	SQL Serveren virker.	
Success End Condition	Administratoren opretter eller sletter den valgte type.	
Failed End Condition	Hvis typen ikke bliver oprettet eller slettet.	
Primær Aktør **Sekundær Aktør**	Administrator Ingen	
Trigger	Opret type åbnes fra typer vinduet.	
DESCRIPTION	**Step**	**Action**
	1	Administratoren indtaster informationer om typen.
	2	Typen gemmes i databasen.
EXTENSIONS	**Step**	**Branching Action**
	1a	Hvis administratoren ikke har udfyldt alle informationer, vil der blive informeret om dette.

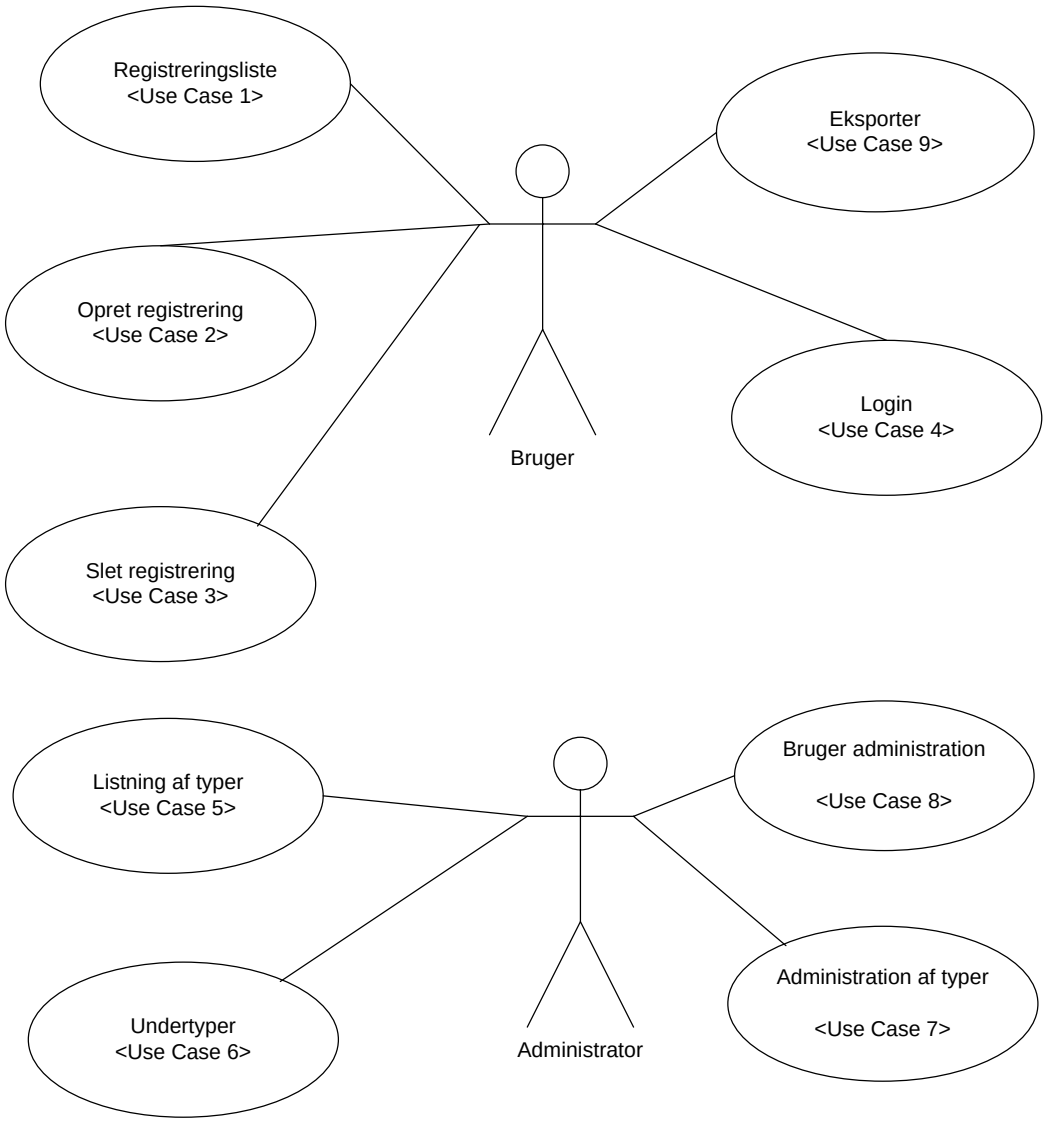
USE CASE 8	Bruger administration

Formål	At administrere brugere i systemet	
Scope	Det skal virke i HiDocuments	
Forudsætning	SQL Serveren virker.	
Success End Condition	Administrator kan oprette og slette brugere.	
Failed End Condition	Hvis det ikke er muligt at oprette eller slette en bruger.	
Primær Aktør **Sekundær Aktør**	Administrator Ingen	
Trigger	Bruger skal oprettes eller slettes	
DESCRIPTION	**Step**	**Action**
	1	Bruger administration formen åbnes.
	2	En liste over alle brugere vises.
	3	Brugeren indtaster alle informationer for en ny bruger for oprette denne.
	4	Bruger bliver oprettet i systemet.
EXTENSIONS	**Step**	**Branching Action**
	2a	Brugeren kan markere en undertype og slette denne.

USE CASE 9	Eksporter
Formål	At eksportere de valgte registreringer til en simpel tekst fil
Scope	Det skal virke i HiDocuments
Forudsætning	SQL Serveren virker.
Success End Condition	Registreringerne bliver eksporteret
Failed End Condition	Registreringerne bliver ikke eksporteret
Primær Aktør	Bruger

Sekundær Aktør	Ingen	
Trigger	Eksport funktion aktiveres	
DESCRIPTION	**Step**	**Action**
	1	Brugeren bliver bedt om at vælge placering og navn på den fil der eksporteres
	2	Alle viste registreringer bliver eksporteret til fil.

Diagram over de udarbejdede Use Cases:

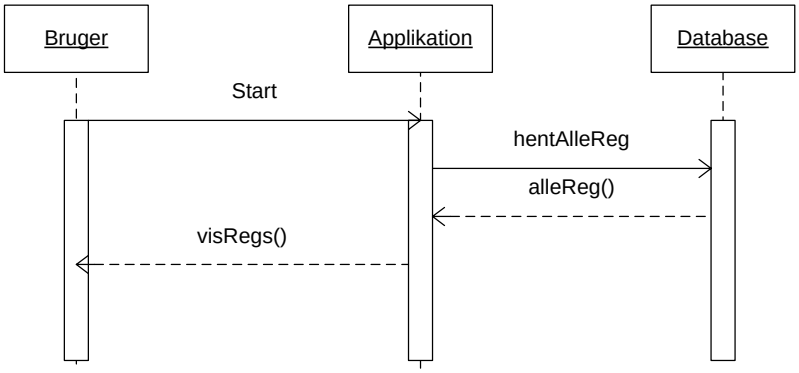


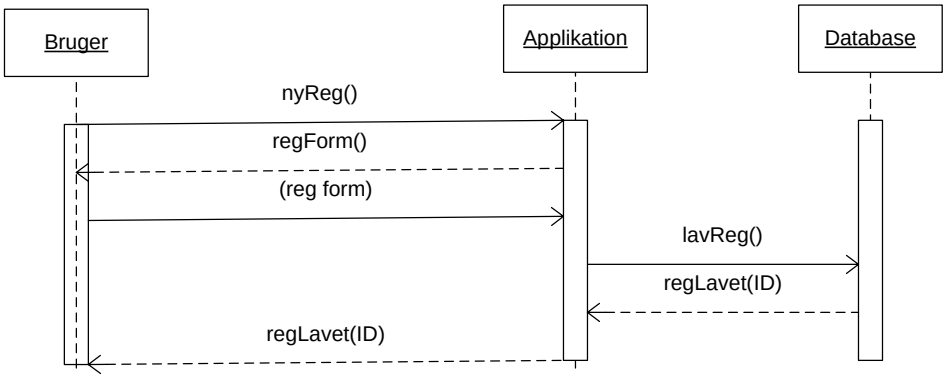
Design

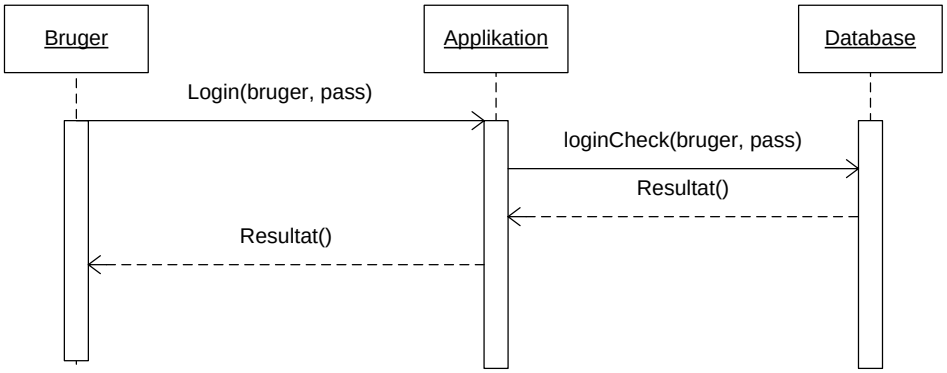
Designet danner grundlag for at hele systemet bliver sikkert, brugervenligt, funktionelt og overskueligt. Designet hjælper med at gøre implementeringen mere overskuelig. I design delen vil de udviklede Use Cases danne grundlag for udviklingen af systemet sekvens diagrammer samt klassediagram. Dette skal vise samspillet mellem de forskellige funktioner og klasser i systemet.

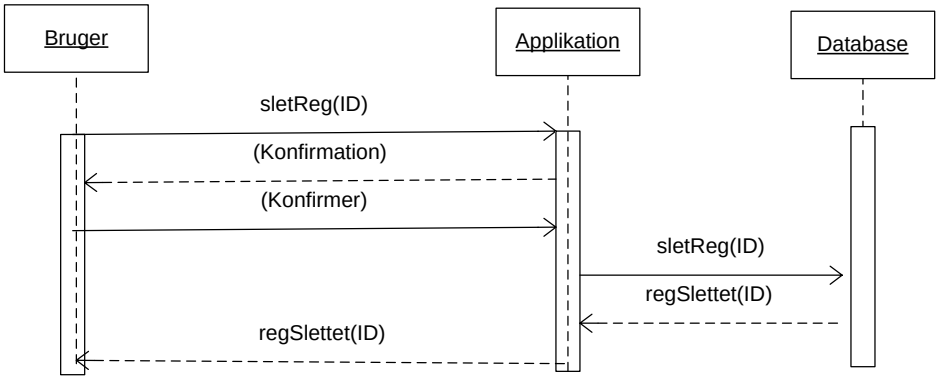
System sekvens diagrammer

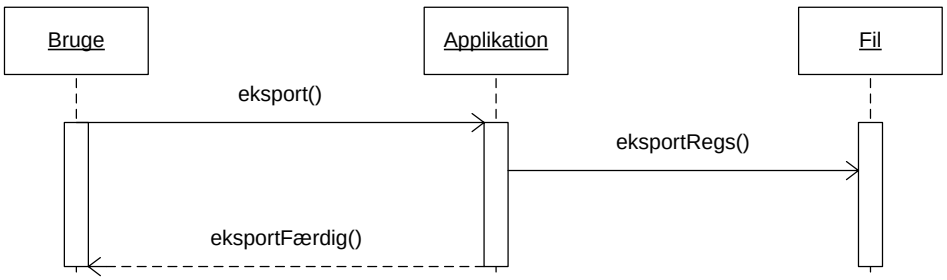
SSD for bruger



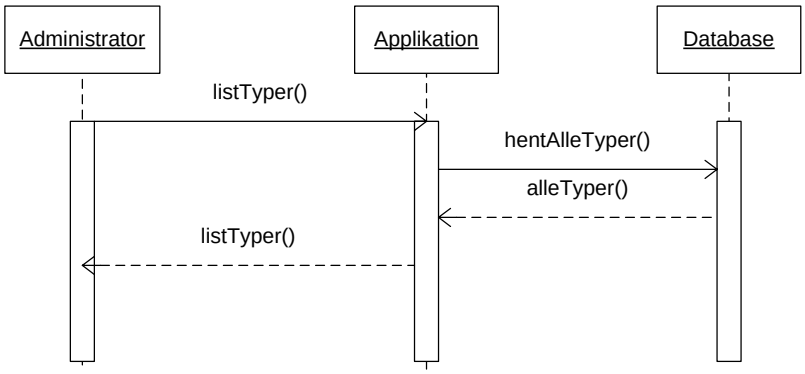


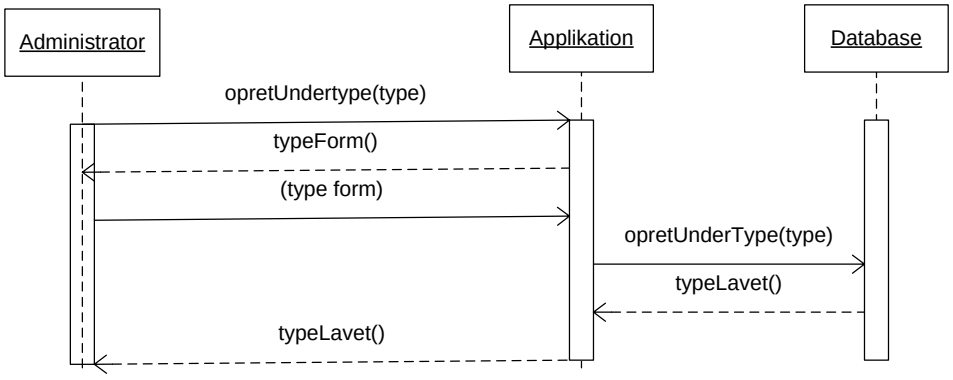

Bruger
Applikation
Database
Login(bruger, pass)
loginCheck(bruger, pass)
Resultat()
Resultat()

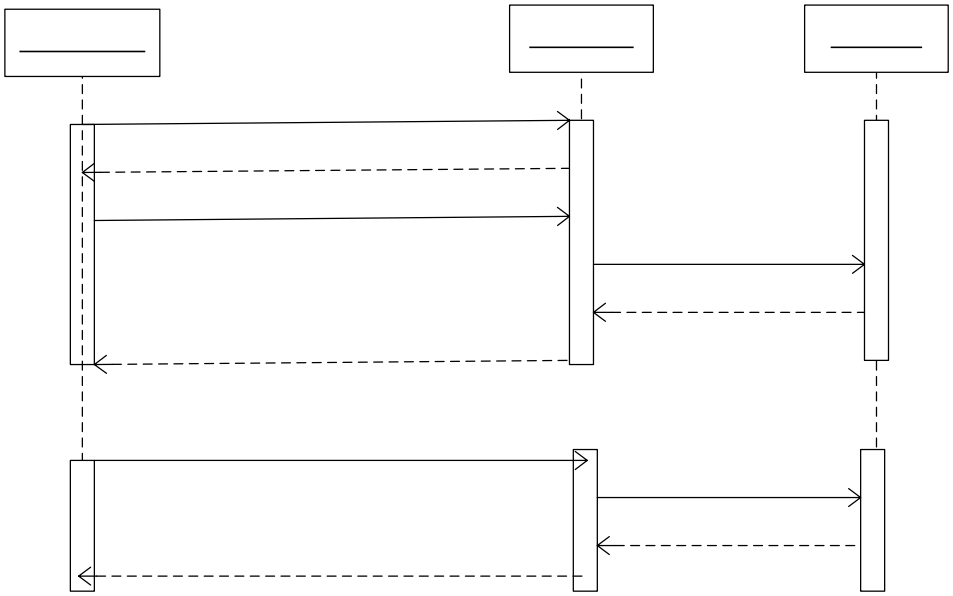

Bruger
Applikation
Database
sletReg(ID)
(Konfirmation)
(Konfirmer)
sletReg(ID)
regSlettet(ID)
regSlettet(ID)

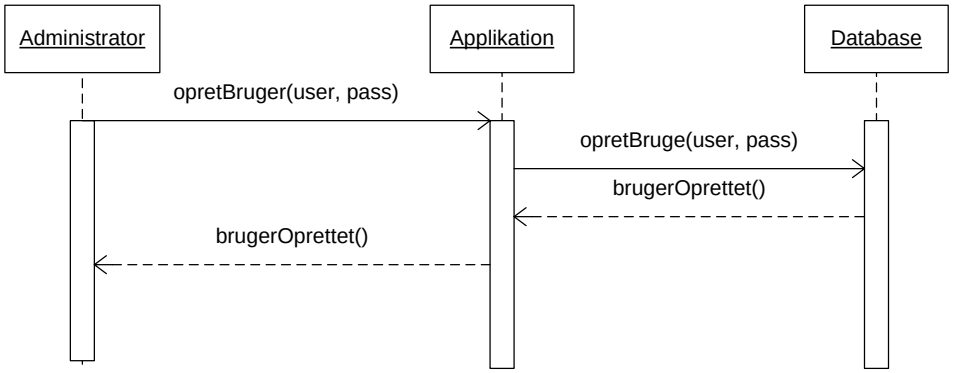

Bruge
Applikation
Fil
eksport()
eksportRegs()
eksportFærdig()

SSD for Administrator









Klassediagram

Klasse diagrammet kan findes i bilag 3.

Design af applikationen

Som det kan ses har jeg brugt UML notation til beskrive designet af applikationen, da jeg mener dette er det bedste redskab man kan bruge når man har med et objekt orienteres sprog at gøre.

Design af databasen

Da det skal være muligt at importere informationer fra firmaets gamle database. Derfor har jeg valgt at opbygge den nye database på lignende måde, da dette vil gøre migrationen fra den gamle database til den nye simplere. Da der ikke er tale om nogen indviklet database, mener jeg ikke at dette vil have den store indflydelse på systemets effektivitet.

Implementering

Herunder beskrives alle applikationens funktioner, ud fra hvilken klasse denne tilhører

Registrations	
Registrations	Dette er klassens konstruktør. Denne initialiserer alle klassens komponenter. Ud over dette sender den også et kald til databasen, for at checke om denne kan findes.
showAllRegs	Kalder en funktion i SQL klassen ud fra hvilke kriterier der er valgt, samt viser resultatet i en listview.
`reg_id_btn_Click`	Hvis brugeren er logget ind vises en NewRegistration form. Hvis brugeren ikke er logget ind bliver denne bedt om dette.
`close_btn_Click`	Åbner en ja/nej dialog hvor brugeren bliver spurgt om programmet skal lukkes. Hvis ja vælges, lukkes programmet.
`Login_btn_Click`	Åbner en Logon form.
`Refresh_btn_Click`	Henter de valgte registreringer fra databasen igen.
`regtypes_btn_Click`	Hvis brugeren er logget ind som Administrator vil en Registrationtypes form blive åbnet. Hvis brugeren ikke har Administrator rettigheder vil denne blive informeret om dette.
`Registrations_list_ItemActivate`	Åbner en UpdateRegistration form til den registrering der er blevet clicked.
`Registrations_list_Click`	Aktiverer slet, makuler og ny version knapperne for den markerede registrering.

	Dette er dog kun tilfældet hvis brugeren er logget ind i systemet.
`del_reg_btn_Click`	Sletter den markerede registrering hvis brugeren konfirmerer dette vha. et OK/Cancel dialog.
`shred_reg_btn_Click`	Fjerner den markerede registrering fra databasen hvis brugeren konfirmerer dette vha. et OK/Cancel dialog.
`Category_box_SelectedIndexChanged`	Kalder showAllRegs funktionen når der bliver ændret på kategori boksen.
`Category_box_TextUpdate`	Kalder showAllRegs når teksten i kategori boksen ændres.
`Selection_box_TextUpdate`	Samme som overstående funktion, men denne funktion kalder også ParameterCheck funktionen.
`Selection_box_SelectedIndexChanged`	Samme som overstående funktion.
`Parameter_text_TextChanged`	Hvis teksten i parameter boksen ændre bliver showAllRegs kaldt.
`ParameterCheck`	Checker om parameter boksen skal være synlig, og sætte synligheden til den korrekte værdi.
`new_version_btn_Click`	Åbner en UpdateRegistration form ud fra den markerede registrering.
`Export`	Udskriver de viste registreringer til en tekst fil, hvis placering og navn bliver bestemt af brugeren vha. et saveFileDialog.
`Export_btn_Click`	Kalder Export funktionen.
HiDocuments	
Main	Åbner Registrations formen.
NewRegistration	
`NewRegistration`	Initialiserer formens komponenter. Ud over

	dette bliver sender/author sat til at være brugerens navn. Kategori og konfidentialitets boksene bliver også fyldt med de valg der findes i databasen.
`Register_btn_Click`	Hvis alle obligatoriske felter er udfyldt i formen kaldes RegArray, RegistrationSql samt setID. Til sidst bliver formen lukket.
`Regtype_text_TextUpdate`	Kalder ShowPanel funktionen når der ændres i Registrerings type boksen.
`RegArray`	Opretter et array med de informationer der er udfyldt i formen.
`Cancel_btn_Click`	Lukker formen.
`ShowPanel`	Checker hvad der er indtastet i registrerings type boksen og viser det relevante panel.
`Regtype_text_SelectedIndexChanged`	Hvis der ændres på indekset i registrerings type boksen bliver ShowPanel kaldt.
`makeID`	Genererer et dokument id ud fra den indtastede information.
`Group_text_Leave`	Kalder makeID, når brugeren forlader gruppe feltet.
`Confidentiality_text_Leave`	Kalder makeID, når brugeren forlader konfidentialitets feltet.
`Regtype_text_Leave`	Kalder makeID, når brugeren forlader registrerings type feltet.
`TypeRadio1_CheckedChanged`	Ændre indholdet af kategori boksen ud fra informationerne i databasen.
`TypeRadio2_CheckedChanged`	Samme som ovenstående funktion.
`TypeRadio3_CheckedChanged`	Samme som ovenstående funktion.
`Category_text_SelectionChangeCommitted`	Aktiverer beskrivelses feltet, samt henter valgmuligheder ud fra databasen. Dette sker når kategori feltet bliver udfyldt.
`desc_text_SelectionChangeCommitted`	Aktiverer gruppe feltet, samt henter

	valgmuligheder ud fra databasen. Dette sker når beskrivelses feltet bliver udfyldt.
UpdateRegistration	
`UpdateRegistration`	Initialiserer klassens komponenter. Ud over dette bliver den valgte registrering hentet fra databasen. Hvis brugeren er logget ind vil selve opdater knappen også blive aktiveret.
Da mange af funktionerne er de samme som i NewRegistration henvises der til denne for informationer om disse. Der vil kun blive beskrevet unikke funktioner nedenfor.	
showReg	Denne funktioner indsætter informationerne om registreringen i de relevante felter i formen.
`Link_text_MouseDoubleClick_1`	Denne funktion åbner et MS Internet Explorer vindue ud fra tekst som findes i link feltet.
Registrationtypes	
`Registrationtypes`	Initialiserer formens komponenter, samt kalder showLinks.
showLinks	Kalder getAllLinks funktionen fra SQL klassen, hvorefter de returnerede links bliver vist i en listview.
`Close_btn_Click`	Lukker formen.
`Cat_btn_Click`	Åbner en Category form.
`Groups_btn_Click`	Åbner en Group form.
`Types_btn_Click`	Åbner en Type form.
`Newlink_btn_Click`	Åbner en Links form.
`Remlink_btn_Click`	Sletter det markerede link fra databasen, hvis brugeren konfirmerer dette.
`Registrationtypes_Enter`	Kalder showLinks.
`Users_btn_Click`	Åbner en Users form.

SQL	
checkAdmin	Checker om brugeren er Administrator.
checkLogin	Checker om username og password passer.
checkName	Henter brugeren fulde navn fra databasen.
checkTypes	Checker om en type eksister.
DelRegistration	Sætter den relevante registrering som slettet i databasen.
getAllLinks	Henter alle links fra databasen.
getAllReg	Henter alle registreringer af den valgte gruppe.
getAllReg(5 variabler)	Henter alle registreringer der passer til de søgte kriterier.
getCatLinks	Henter kategorier ud fra deres id.
getCats	Henter alle kategorier i databasen.
getCatType(int)	Henter forkortelsen af kategori typen ud fra dennes id.
getCatType(string)	Henter forkortelsen af kategori typen, ud fra det fulde navn.
getCatTypeId	Henter id'et for den valgte kategori type ud fra navnet på denne.
getColNames	Henter kolonne navne på registreringerne i databasen.
getConf	Henter forkortelsen af konfidentialiteten.
getConfs	Henter alle konfidentialiteter fra databasen.
getDescLinks	Henter beskrivelser ud fra deres id.
getDescs	Henter alle beskrivelser fra databasen.
getFullCats	Henter alle informationer om kategorier fra databasen.
getFullGroups	Hente alle informationer om alle grupper i databasen.
getGroup(int)	Henter gruppens forkortelse ud fra dennes id.

`getGroup(string)`	Returnerer gruppe forkortelsen af den valgte gruppe.
`getGroupId`	Henter id'et for den valgte gruppe, ud fra dennes navn.
`getGroupL`	Henter gruppe navnet ud fra dennes forkortelse.
`getGroupLinks`	Henter grupper ud fra deres id.
`getGroups`	Henter alle grupper fra databasen.
`getID`	Henter det nyeste ID nummer fra databasen.
`getIndex`	Henter det nyeste index.
`getInstance`	Singleton der returnerer SQL klassen.
`getLinkRows`	Henter antallet af links i databasen.
`getReg`	Returnerer den valgte registrering fra databasen.
`getRows`	Henter antallet af registreringer i databasen.
`getType(int)`	Henter typens navn ud fra dennes id.
`getType(string)`	Henter typens id ud fra beskrivelsen af denne.
`getTypeId`	Henter type id'et vha. typens navn.
`getTypes`	Henter alle typer fra databasen.
`getUsers`	Henter alle brugere fra databasen.
`NewCat`	Opretter en ny kategori i database ud fra de sendte oplysninger.
`NewGroup`	Opretter en ny gruppe i databasen ud fra de givne oplysninger.
`newLink`	Opretter en nyt link i databasen.
`NewType`	Opretter en ny type ud fra de givne oplysninger.
`newUser`	Opretter en ny bruger ud fra de sendte oplysninger.
`RegistrationSql`	Indsætter den aktuelle registrering i

	databasen.
remLink	Fjerner et link fra databasen ud fra dettes beskrivelse.
removeCat	Fjerner den valgte kategori fra databasen.
removeGroup	Fjerner den valgte gruppe fra databasen.
removeType	Fjerner den valgte type fra databasen.
removeUser	Fjerner den valgte bruger fra databasen.
setID	Henter det nyeste ID nummer og adderer dette med 1.
setIndex	Henter det nyeste index og adderer dette med 1.
setInvalid	Sætter den valgte registrering som invalid.
ShredRegistration	Fjerner den relevante registrering fra databasen.
Type	
Type	Initialiserer formens komponenter samt kalder showTypes.
Close_btn_Click	Lukker formen.
showTypes	Viser alle typer i en listview.
Remove_btn_Click	Fjerner den markerede type hvis brugeren konfirmerer dette.
Add_btn_Click	Hvis alle felter i formen er udfyldt vil den indtastede type bliver oprettet i databasen.
Group	
Group	Initialiserer formens komponenter samt kalder showGroups.
Close_btn_Click	Lukker formen.
showGroups	Viser alle grupper fra databasen i et listview.
Remove_btn_Click	Fjerner den markerede gruppe hvis bruger konfirmerer dette.
Add_btn_Click	Opretter den relevante gruppe hvis alle felter

	udfyldt i formen.
Category	
`Category`	Initialiserer formens komponenter samt kalder showCats.
`Close_btn_Click`	Lukker formen.
`showCats`	Viser alle kategorier fra databasen i et listview.
`Remove_btn_Click`	Fjerner den markerede kategori hvis bruger konfirmerer dette.
`Add_btn_Click`	Opretter den relevante kategori hvis alle felter udfyldt i formen.
Links	
`Links`	Initialiserer formens komponenter, samt udfylder de 3 combobokse med mulighederne, fra databasen.
`Close_btn_Click`	Lukker formen.
`Create_btn_Click`	Hvis alle felter er udfyldt korrekt vil linket blive oprettet.
`createLink`	Opretter et array med informationer om linket, fra formen. Samt sender dette til newLink.
Logon	
`Logon`	Initialiserer formens komponenter.
`Close_btn_Click`	Lukker formen.
`Login_btn_Click`	Checker om de indtastede informationer passer, og logger brugeren hvis dette er tilfældet.
Users	
`Users`	Initialiserer formens komponenter, samt kalder showUsers.
`Close_btn_Click`	Lukker formen.

`showUsers`	Henter alle brugere fra databasen og viser disse i et listview.
`Remove_btn_Click`	Fjerner den markerede bruger fra databasen. Dette kræver dog at brugeren konfirmerer dette.
`Add_btn_Click`	Hvis alle felter i formen er udfyldt vil brugeren blive oprettet i systemet.

Implementering af databasen

Et diagram over det fysiske database diagram kan ses i bilag 4.

Tests

I dette afsnit vil jeg foretage test af produktet, samt beskrive resultatet af testen. Dette bliver gjort i form af testrapporter. Følgende testrapporter vil være at finde nedenfor:

- Test af Registreringslister
- Test af Login
- Test af Opret Registrering
- Test af Opdater Registrering
- Test af Slet Registrering
- Test af Makuler Registrering
- Test af Eksporter
- Test af Opret Kategori
- Test af Slet Kategori
- Test af Opret Gruppe
- Test af Slet Gruppe
- Test af Opret Type
- Test af Slet Type
- Test af Opret Bruger
- Test af Slet Bruger
- Test af Opret Link
- Test af Slet link

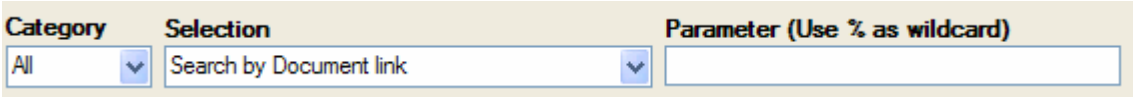
Test af Registreringslister

Test-formål:

Formålet med denne test er at teste om registreringslisten virker.

Testbeskrivelse:

Alle de forskellige kombinationer det er muligt at vælge i listerne, som er vist nedenfor, samt med forskellige parametre.



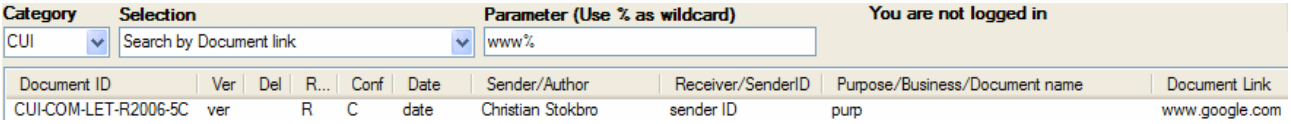
Dette skulle gerne vise de registreringer der passer til de valgte krav.

Test-forudsætninger:

Det forudsættes at .Net 2.0 og applikationen har adgang til MS Sql serveren er installeret. Det forudsættes også at brugeren er logget ind og har de rigtige rettigheder.

Test-resultat:

Da de forskellige tests viste de rigtige resultater , som f.eks.



Da jeg ikke fandt nogen fejl med alle de forskellige kombinater mener jeg at denne funktion virker efter hensigt.

Tilladte afvigelser / fejl:

Ingen.

Test af Login

Test-formål:

Formålet med denne test er at teste om det er muligt for en bruger at logge ind i systemet.

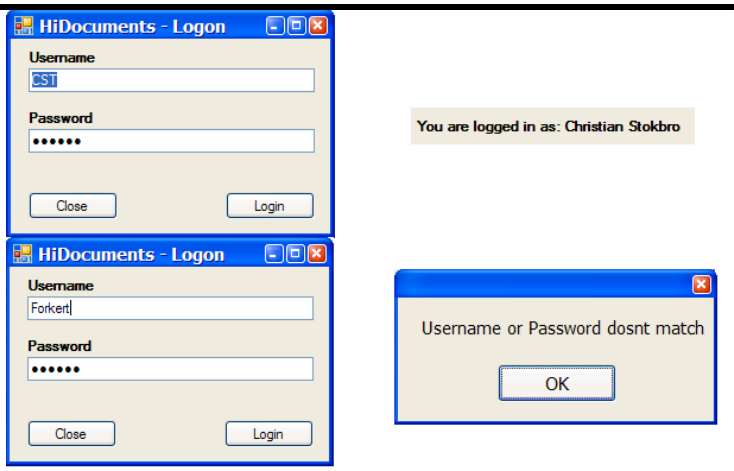
Testbeskrivelse:

Login formen åbnes og der indtastes både en forkert og en rigtigt bruger. Hvis den forkerte bliver afvist og den rigtige godkendt, vil testen være en success.

Test-forudsætninger:

Det forudsættes at .Net 2.0 og applikationen har adgang til MS Sql serveren er installeret. Det forudsættes også at brugeren er logget ind og har de rigtige rettigheder.

Test-resultat:



Da testen forløb som antaget, må det antages af denne del virker.

Tilladte afvigelser / fejl:

Ingen.

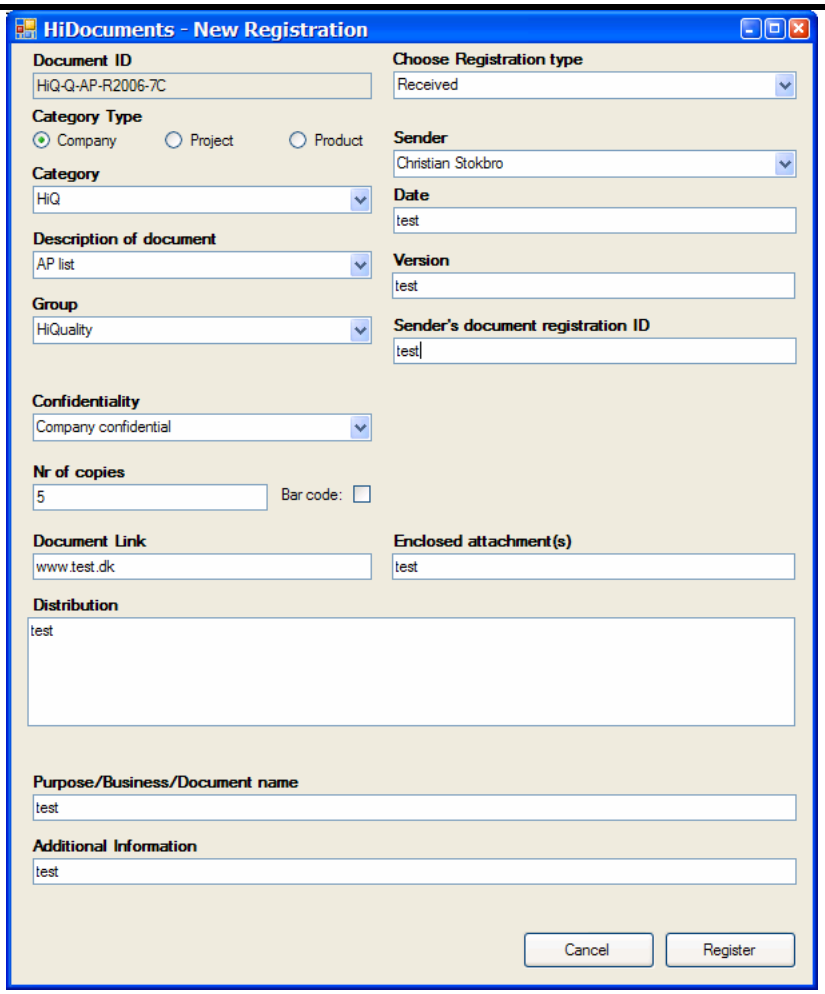
Test af Opret Registrering

Test-formål:

Formålet med denne test er at teste om det er muligt at oprette en registrering

Testbeskrivelse:

Alle felter i opret registrering udfyldes:

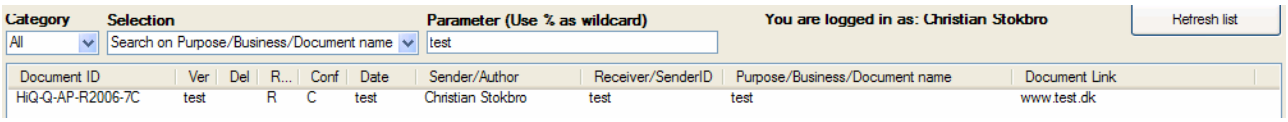


Test-forudsætninger:

Det forudsættes at .Net 2.0 og applikationen har adgang til MS Sql serveren er installeret. Det forudsættes også at brugeren er logget ind og har de rigtige rettigheder.

Test-resultat:

Da den oprettede registrering blev vist i listen, mener jeg også at denne funktion virker.



Tilladte afvigelser / fejl:

Indtastning af f.eks. meget lange ord vil genere fejl i databasen.

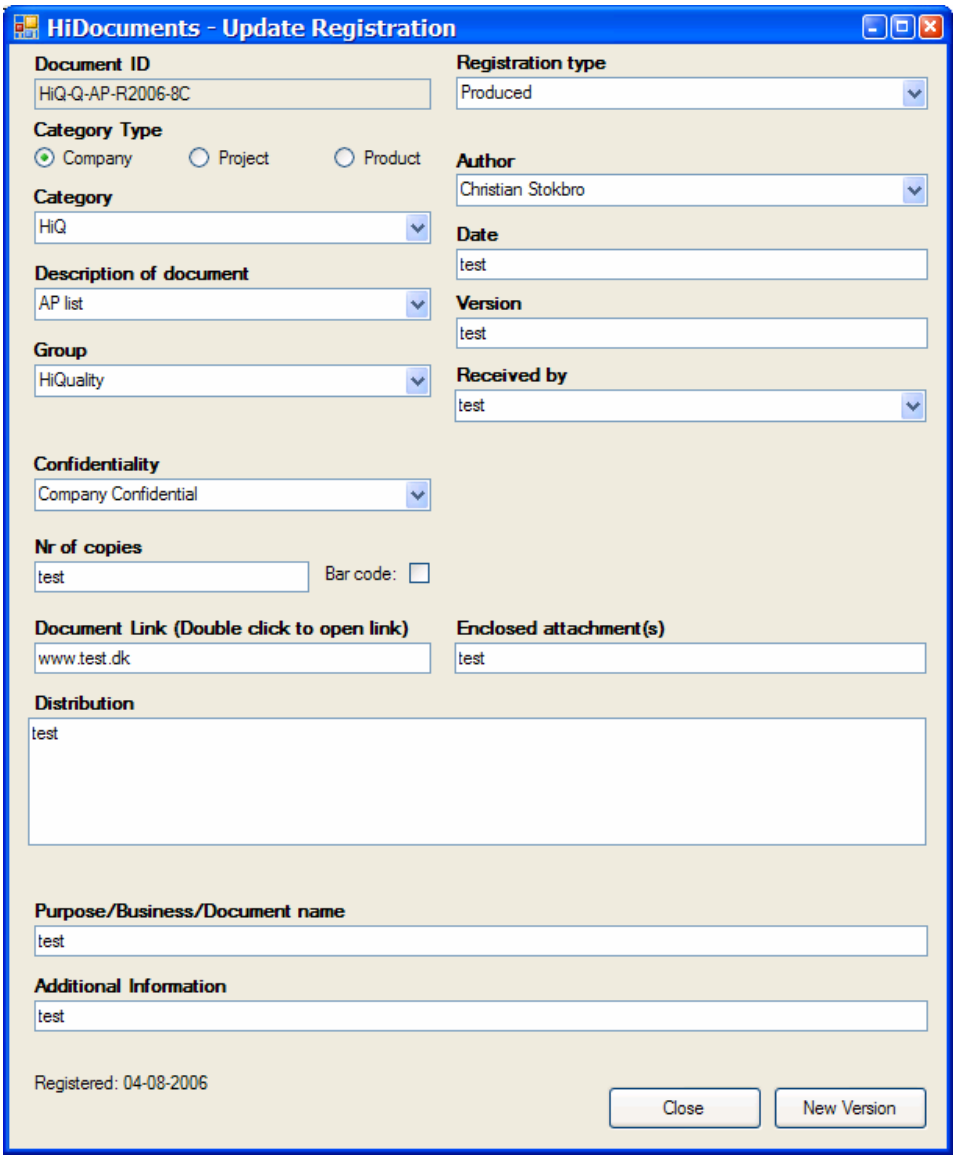
Test af Opdater Registrering

Test-formål:

Formålet med denne test er at teste om det er muligt at opdatere en registrering

Testbeskrivelse:

Et af felterne ændres i forhold til den gamle registrering, som f.eks. registrerings typen:



Hvis registreringen bliver opdateret i systemet vil testen være en succes.

Test-forudsætninger:

Det forudsættes at .Net 2.0 og applikationen har adgang til MS Sql serveren er installeret. Det forudsættes også at brugeren er logget ind og har de rigtige rettigheder.

Test-resultat:

Document ID	Ver	Del	R...
HiQ-Q-AP-R2006-8C	test		R
HiQ-Q-AP-R2006-8C	test		P

Da den oprettede registrering blev vist i listen, og den gamle blev sat til invalid, var testen en succes.

Tilladte afvigelser / fejl:

Indtastning af f.eks. meget lange ord vil genere fejl i databasen.

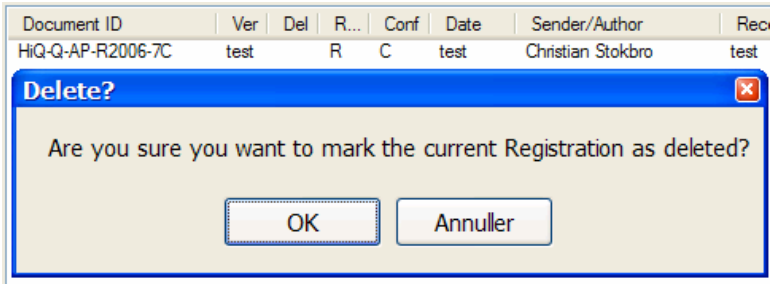
Test af Slet Registrering

Test-formål:

Formålet med denne test er at teste om det er muligt at markere en registrering som slettet.

Testbeskrivelse:

Der vælges en registrering og listen og trykkes slet, som nedenfor:



Testen vil være en succes hvis registreringen bliver markeret som slettet.

Test-forudsætninger:

Det forudsættes at .Net 2.0 og applikationen har adgang til MS Sql serveren er installeret. Det forudsættes også at brugeren er logget ind og har de rigtige rettigheder.

Test-resultat:

Document ID	Ver	Del
HiQ-Q-AP-R2006-7C	test	x

Da registreringen blev markeret som slettet er denne test en succes.

Tilladte afvigelser / fejl:

Ingen.

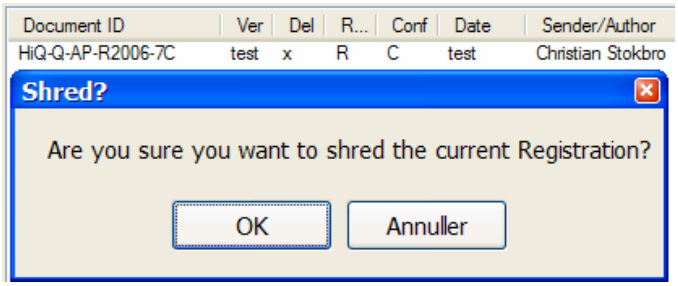
Test af Makuler Registrering

Test-formål:

Formålet med denne test er at teste om det er muligt fjerne er registrering fra systemet

Testbeskrivelse:

Der vælges en registrering og listen og trykkes makuler, som nedenfor:

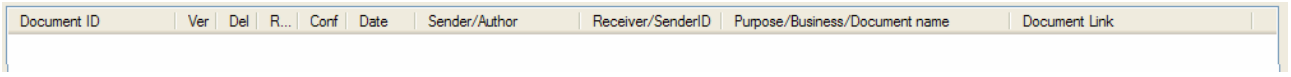


Testen vil være en succes hvis registreringen bliver fjernet.

Test-forudsætninger:

Det forudsættes at .Net 2.0 og applikationen har adgang til MS Sql serveren er installeret. Det forudsættes også at brugeren er logget ind og har de rigtige rettigheder.

Test-resultat:



Da registreringen blev fjernet fra systemet virker denne del efter hensigt.

Tilladte afvigelser / fejl:

Ingen.

Test af Eksporter

Test-formål:

Formålet med denne test er at teste om eksport funktionen virker.

Testbeskrivelse:

Der søges på en registrering i listen og eksport vælges. Herefter importeres tekst filen i Microsoft Excel, da dette giver et bedre overblik.

Testen vil være en succes hvis der vises et fornuftigt resultat i Excel.

Test-forudsætninger:

Det forudsættes at .Net 2.0 og applikationen har adgang til MS Sql serveren er installeret. Det forudsættes også at brugeren er logget ind og har de rigtige rettigheder.

Test-resultat:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	DocID	Cat_type	Category	Desc_doc	Group_	Conf	Copies	Bar	Link	Purpose	Ainfo	Regtype	Author_Sender	Date	Version
2	HiQ-Q-AP-R2006-8C	Company	HiQ	AP list	HiQuality	Company Confidential	test	FALSE	www.test.dk	test	test	Produced	Christian Stokbro	test	test

Det ovenstående resultat er en del af Excel importen, og ud fra dette ser det ud til at denne del virker korrekt.

Tilladte afvigelser / fejl:

Ingen.

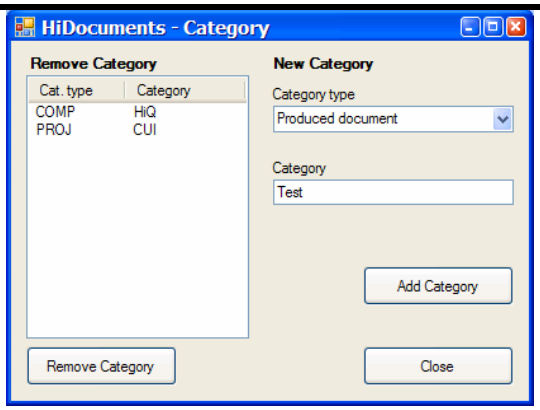
Test af Opret Kategori

Test-formål:

Formålet med denne test er at teste om det er muligt at oprette en kategori

Testbeskrivelse:

Alle felterne i opret kategori formen udfyldes, som nedenfor, og tilføj kategori vælges.

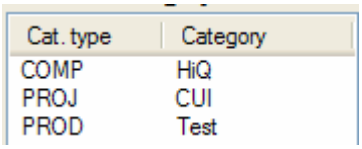


Hvis kategorien bliver oprettet vil testen være en succes.

Test-forudsætninger:

Det forudsættes at .Net 2.0 og applikationen har adgang til MS Sql serveren er installeret. Det forudsættes også at brugeren er logget ind og har de rigtige rettigheder.

Test-resultat:



Cat. type	Category
COMP	HiQ
PROJ	CUI
PROD	Test

Kategorien er blevet oprettet, så testen var en succes.

Tilladte afvigelser / fejl:

Indtastning af f.eks. meget lange ord vil genere fejl i databasen.

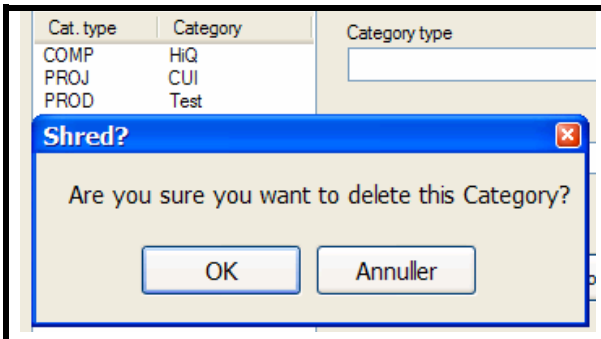
Test af Slet Kategori

Test-formål:

Formålet med denne test er at teste om det er muligt at slette en kategori

Testbeskrivelse:

Kategorien markeres og der vælges fjern:

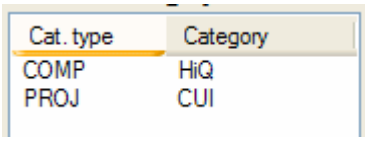


Hvis kategorien bliver fjernet er testen en succes.

Test-forudsætninger:

Det forudsættes at .Net 2.0 og applikationen har adgang til MS Sql serveren er installeret. Det forudsættes også at brugeren er logget ind og har de rigtige rettigheder.

Test-resultat:



Da kategorien blev fjernet var testen en succes.

Tilladte afvigelser / fejl:

Ingen.

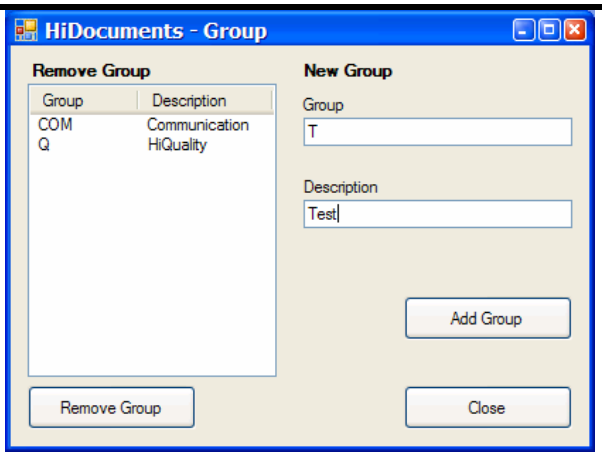
Test af Opret Gruppe

Test-formål:

Formålet med denne test er at teste om det er muligt at oprette en gruppe

Testbeskrivelse:

Alle felterne i opret gruppe formen udfyldes, som nedenfor, og tilføj gruppe vælges.



Hvis gruppen bliver oprettet vil testen være en succes.

Test-forudsætninger:

Det forudsættes at .Net 2.0 og applikationen har adgang til MS Sql serveren er installeret. Det forudsættes også at brugeren er logget ind og har de rigtige rettigheder.

Test-resultat:

Group	Description
COM	Communication
Q	HiQuality
T	Test

Gruppen er blevet oprettet, så testen var en succes.

Tilladte afvigelser / fejl:

Indtastning af f.eks. meget lange ord vil genere fejl i databasen.

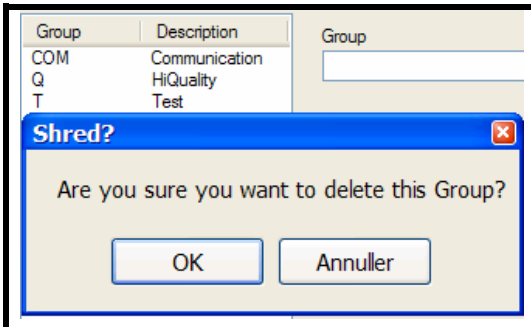
Test af Slet Gruppe

Test-formål:

Formålet med denne test er at teste om det er muligt at slette en gruppe.

Testbeskrivelse:

Gruppen markeres og der vælges fjern:

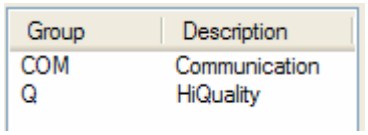


Hvis gruppen bliver fjernet er testen en succes.

Test-forudsætninger:

Det forudsættes at .Net 2.0 og applikationen har adgang til MS Sql serveren er installeret. Det forudsættes også at brugeren er logget ind og har de rigtige rettigheder.

Test-resultat:



Da gruppen blev fjernet var testen en succes.

Tilladte afvigelser / fejl:

Ingen.

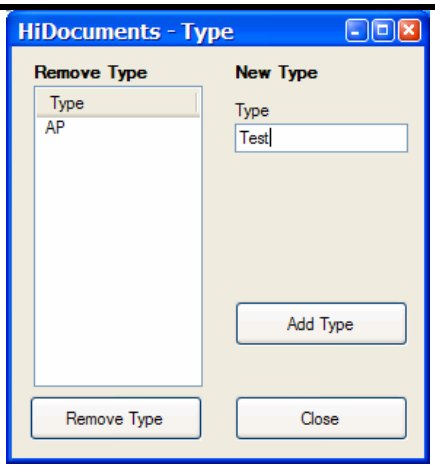
Test af Opret Type

Test-formål:

Formålet med denne test er at teste om det er muligt at oprette en type.

Testbeskrivelse:

Alle felterne i opret type formen udfyldes, som nedenfor, og tilføj type vælges.

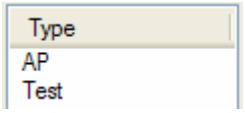


Hvis typen bliver oprettet vil testen være en succes.

Test-forudsætninger:

Det forudsættes at .Net 2.0 og applikationen har adgang til MS Sql serveren er installeret. Det forudsættes også at brugeren er logget ind og har de rigtige rettigheder.

Test-resultat:



Typen er blevet oprettet, så testen var en succes.

Tilladte afvigelser / fejl:

Indtastning af f.eks. meget lange ord vil genere fejl i databasen.

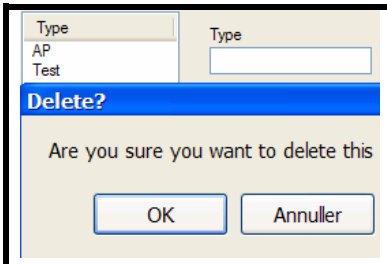
Test af Slet Type

Test-formål:

Formålet med denne test er at teste om det er muligt at slette en type.

Testbeskrivelse:

Typen markeres og der vælges fjern:

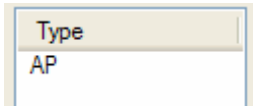


Hvis typen bliver fjernet er testen en succes.

Test-forudsætninger:

Det forudsættes at .Net 2.0 og applikationen har adgang til MS Sql serveren er installeret. Det forudsættes også at brugeren er logget ind og har de rigtige rettigheder.

Test-resultat:



Da typen blev fjernet var testen en succes.

Tilladte afvigelser / fejl:

Ingen.

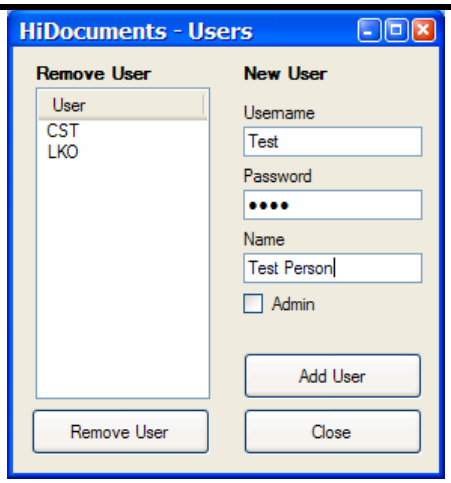
Test af Opret Bruger

Test-formål:

Formålet med denne test er at teste om det er muligt at oprette en ny bruger

Testbeskrivelse:

Alle felterne i opret bruger formen udfyldes, som nedenfor, og tilføj bruger vælges.

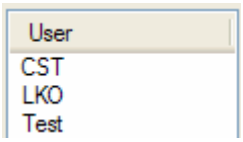


Hvis brugeren bliver oprettet vil testen være en succes.

Test-forudsætninger:

Det forudsættes at .Net 2.0 og applikationen har adgang til MS Sql serveren er installeret. Det forudsættes også at brugeren er logget ind og har de rigtige rettigheder.

Test-resultat:



Brugeren er blevet oprettet, så testen var en succes.

Tilladte afvigelser / fejl:

Indtastning af f.eks. meget lange ord vil genere fejl i databasen.

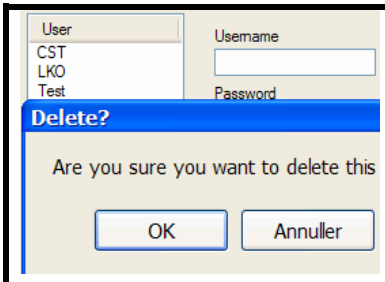
Test af Slet Bruger

Test-formål:

Formålet med denne test er at teste om det er muligt at slette en bruger.

Testbeskrivelse:

Brugeren markeres og der vælges fjern:

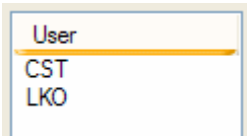


Hvis brugeren bliver fjernet er testen en succes.

Test-forudsætninger:

Det forudsættes at .Net 2.0 og applikationen har adgang til MS Sql serveren er installeret. Det forudsættes også at brugeren er logget ind og har de rigtige rettigheder.

Test-resultat:



Da Brugeren blev fjernet var testen en succes.

Tilladte afvigelser / fejl:

Ingen.

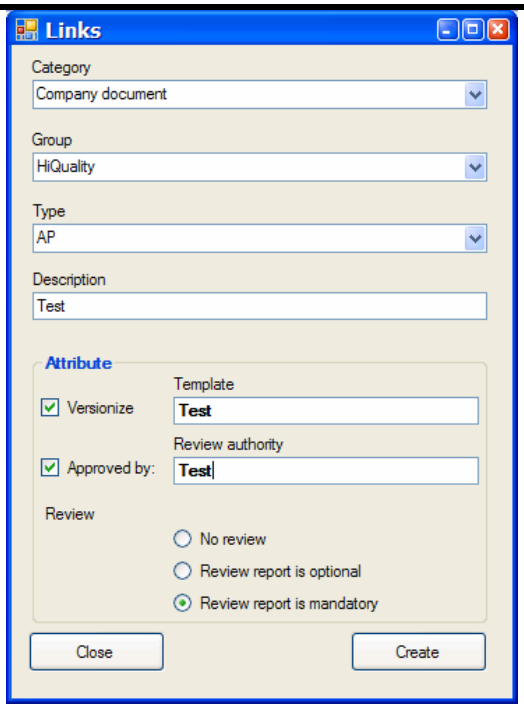
Test af Opret Link

Test-formål:

Formålet med denne test er at teste om det er muligt at oprette et nyt link.

Testbeskrivelse:

Alle felterne i opret link formen udfyldes, som nedenfor, og tilføj og opret vælges.



Hvis linket bliver oprettet vil testen være en succes.

Test-forudsætninger:

Det forudsættes at .Net 2.0 og applikationen har adgang til MS Sql serveren er installeret. Det forudsættes også at brugeren er logget ind og har de rigtige rettigheder.

Test-resultat:

Cat	Group	Description of group	Type	Description	Ver	Rev	App	Revirew authority	Template
COMP	Q	HiQuality	AP	Test	X	M	X	Test	Test

Linket er blevet oprettet, så testen var en succes.

Tilladte afvigelser / fejl:

Indtastning af f.eks. meget lange ord vil genere fejl i databasen.

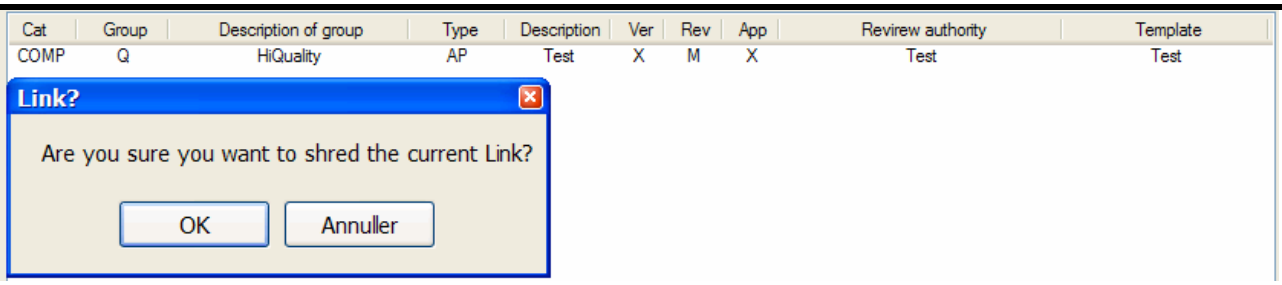
Test af Slet Link

Test-formål:

Formålet med denne test er at teste om det er muligt at slette et link.

Testbeskrivelse:

Linket markeres og der vælges fjern:

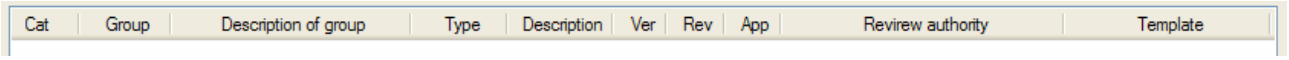


Hvis linket bliver fjernet er testen en succes.

Test-forudsætninger:

Det forudsættes at .Net 2.0 og applikationen har adgang til MS Sql serveren er installeret. Det forudsættes også at brugeren er logget ind og har de rigtige rettigheder.

Test-resultat:



Da linket blev fjernet var testen en succes.

Tilladte afvigelser / fejl:

Ingen.

Fejl og mangler

Der findes stadig en del fejl og mangler i systemet. Bl.a. kan man indtaste ”forkerte” værdier i databasen og herved opnå fejl. Jeg har valgt ikke at bruge tid på dette, da programmet kun skal bruges af ansatte hos HiQ Wise, der ikke vil have et motiv for dette. Der ville også stadig være forbedringer til systemet. Det er f.eks. nogen gange nødvendigt at opdatere registreringslisten manuelt.

Konklusion

Jeg har i dette projekt fået udviklet et dokumentstyringssystem til HiQ Wise A/S, der overholder de krav der blev stillet til dette. Der er dog stadig mange mulige udvidelser samt forbedringer til dette, men dette ligger i fremtiden.
Det har været spændende og lærerigt at udarbejde et produkt som rent faktisk skulle bruges af nogen, da de projekter jeg har arbejdet med før kun var som en del af kurser. Dette gjorde at slutproduktet blev ”glemt” så snart kurset var overstået.
Jeg har prøvet at holde min tids- og iterations plan, men dette blev desværre svært da jeg startede projektet med at lægge mig syg i halvanden uge. Jeg mener dog selv at jeg har indhentet den tabte tid og udviklet et produkt der vil blive brugt i firmaet.
Det har været sjovt at arbejde alene om et projekt, da jeg ikke har gjort dette før under nogen kurser på DTU.
Jeg har i projektet prøvet at bruge alle de værktøjer jeg har lært under mit studie. Jeg har dog mest prøvet at bruge de metoder jeg lærte i faget OOAD og Databaser, nemlig Objekt Orienteres Analyse and Design, samt Unified Process.

Jeg har under projektet snakket meget med min vejleder på HiQ Wise, samt andre ansatte. Dette har været både motiverende for arbejdet, samt en god hjælp til hvordan programmet skulle designes.
Jeg har forsøgt at udarbejde et simpelt funktionelt design, da brugen af versions og dokument styring vil være nyt for nogen af brugerne.
Afleveringen af projektet til DTU vil heller ikke være den sidste del af selve projektet, der mangler stadig opsætningen af systemet i firmaet, samt introduktionen af systemet til medarbejderne. Dette er med til at jeg måske ser en mulig fremtid i HiQ Wise, især med hensyn til videreudvikling og support af systemet.

Litteraturliste

www.google.com

www.msdn.om

www.w3schools.com

subversion.tigris.org

tortoisesvn.tigris.org

Hans Van Vliet – Software Engineering (ISBN 0-471-97508-7)

Craig Larman – Applying UML and Patterns (ISBN 0-13-092569-1)

Charles Petzold – Programming Microsoft Windows with C# (ISBN 0-7356-1370-2)

Ramakrishnan/Gehrke – Database Management Systems (ISBN 0-07-115110-9)

Bilag

Bilag 1 – Kildekode

HiDocuments.cs

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace HiDocuments
{
    static class HiDocuments
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Registrations());
        }
    }
}
```

Registrations.cs

```
using System;
using System.Collections.Generic;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;

namespace HiDocuments
{
    public partial class Registrations : Form
    {
        static bool loggedon = false;
        static string username = "";
        static bool admin = false;
        static bool dbnotfound = true;
        SQL sqlc = SQL.getInstance();
        string[][] regs = null;

       /* Dette er klassens konstruktør. Denne initialiserer alle klassens komponenter.
        * Ud over dette sender den også et kald til databasen, for at checke om denne kan
findes. */
        public Registrations()
        {
            InitializeComponent();
```

```
        try
        {
            int test = sqlc.getID();
            dbnotfound = false;

        }
        catch (Exception e)
        {
            Console.Write((e.ToString()));
        }
        if (dbnotfound)
        {

            MessageBox.Show("No Database connected");
        }
        else
        {

            this.showAllRegs();
        }
    }
    /* Kalder en funktion i SQL klassen ud fra hvilke kriterier der er valgt,
     * samt viser resultatet i en listview. */
    private void showAllRegs()
    {
        if (loggedon)
        {
            Loginname.Text = "You are logged in as: " + username;
        }
        Registrations_list.Items.Clear();
        int n = 0;


        switch (Selection_box.Text)
        {
            default:
                if (Category_box.Text == "HiQ")
                {
                    regs = sqlc.getAllReg("HiQ");
                }
                else if (Category_box.Text == "CUI")
                {
                    regs = sqlc.getAllReg("CUI");
                }
                else
                {
                    regs = sqlc.getAllReg("%");
                }
                break;
            case "Valid registrations":
                if (Category_box.Text == "HiQ")
                {
                    regs = sqlc.getAllReg("HiQ", "Invalid", "Deleted", "0", "0", "AND");
                }
                else if (Category_box.Text == "CUI")
                {
                    regs = sqlc.getAllReg("CUI", "Invalid", "Deleted", "0", "0", "AND");
                }
                else
```

```
            {
                regs = sqlc.getAllReg("%", "Invalid", "Deleted", "0", "0", "AND");
            }
            break;
        case "Invalid registrations":
            if (Category_box.Text == "HiQ")
            {
                regs = sqlc.getAllReg("HiQ", "Invalid", "Deleted", "1", "0", "AND");
            }
            else if (Category_box.Text == "CUI")
            {
                regs = sqlc.getAllReg("CUI", "Invalid", "Deleted", "1", "0", "AND");
            }
            else
            {
                regs = sqlc.getAllReg("%", "Invalid", "Deleted", "1", "0", "AND");
            }
            break;
        case "Search on document ID":
            if (Category_box.Text == "HiQ")
            {
                if (!(Parameter_text.Text == ""))
                {
                    regs = sqlc.getAllReg("HiQ", "DocID", Parameter_text.Text);
                }
            }
            else if (Category_box.Text == "CUI")
            {
                if (!(Parameter_text.Text == ""))
                {
                    regs = sqlc.getAllReg("CUI", "DocID", Parameter_text.Text);
                }
            }
            else
            {
                if (!(Parameter_text.Text == ""))
                {
                    regs = sqlc.getAllReg("%", "DocID", Parameter_text.Text);
                }
            }
            break;
        case "Search on Purpose/Business/Document name":
            if (Category_box.Text == "HiQ")
            {
                if (!(Parameter_text.Text == ""))
                {
                    regs = sqlc.getAllReg("HiQ", "Purpose", Parameter_text.Text);
                }
            }
            else if (Category_box.Text == "CUI")
            {
                if (!(Parameter_text.Text == ""))
                {
                    regs = sqlc.getAllReg("CUI", "Purpose", Parameter_text.Text);
                }
            }
            else
            {
                if (!(Parameter_text.Text == ""))
```

```
                    {
                        regs = sqlc.getAllReg("%", "Purpose", Parameter_text.Text);
                    }
                }
                break;
            case "Search on sender, author or receiver":
                if (Category_box.Text == "HiQ")
                {
                    if (!(Parameter_text.Text == ""))
                    {
                        regs = sqlc.getAllReg("HiQ", "Author_Sender", "Recby_senderid",
Parameter_text.Text, Parameter_text.Text, "OR");
                    }
                }
                else if (Category_box.Text == "CUI")
                {
                    if (!(Parameter_text.Text == ""))
                    {
                        regs = sqlc.getAllReg("CUI", "Author_Sender", "Recby_senderid",
Parameter_text.Text, Parameter_text.Text, "OR");
                    }
                }
                else
                {
                    if (!(Parameter_text.Text == ""))
                    {
                        regs = sqlc.getAllReg("%", "Author_Sender", "Recby_senderid",
Parameter_text.Text, Parameter_text.Text, "OR");
                    }
                }
                break;
            case "Search by Document link":
                if (Category_box.Text == "HiQ")
                {
                    if (!(Parameter_text.Text == ""))
                    {
                        regs = sqlc.getAllReg("HiQ", "Link", Parameter_text.Text);
                    }
                }
                else if (Category_box.Text == "CUI")
                {
                    regs = sqlc.getAllReg("CUI", "Link", Parameter_text.Text);
                }
                else
                {
                    regs = sqlc.getAllReg("%", "Link", Parameter_text.Text);
                }
                break;
            case "Produced":
                if (Category_box.Text == "HiQ")
                {
                    regs = sqlc.getAllReg("HiQ", "Regtype", "Produced");
                }
                else if (Category_box.Text == "CUI")
                {
                    regs = sqlc.getAllReg("CUI", "Regtype", "Produced");
                }
                else
                {
```

```
                    regs = sqlc.getAllReg("%", "Regtype", "Produced");
                }
                break;
            case "Received":
                if (Category_box.Text == "HiQ")
                {
                    regs = sqlc.getAllReg("HiQ", "Regtype", "Received");
                }
                else if (Category_box.Text == "CUI")
                {
                    regs = sqlc.getAllReg("CUI", "Regtype", "Received");
                }
                else
                {
                    regs = sqlc.getAllReg("%", "Regtype", "Received");
                }
                break;
            case "Confidentiality (C/CR)":
                if (Category_box.Text == "HiQ")
                {
                    regs = sqlc.getAllReg("HiQ", "Conf", "Conf", "Company confidential",
"Company restricted", "OR");
                }
                else if (Category_box.Text == "CUI")
                {
                    regs = sqlc.getAllReg("CUI", "Conf", "Conf", "Company confidential",
"Company restricted", "OR");
                }
                else
                {
                    regs = sqlc.getAllReg("%", "Conf", "Conf", "Company confidential",
"Company restricted", "OR");
                }
                break;
            case "Deleted":
                if (Category_box.Text == "HiQ")
                {
                    regs = sqlc.getAllReg("HiQ", "Deleted", "1");
                }
                else if (Category_box.Text == "CUI")
                {
                    regs = sqlc.getAllReg("CUI", "Deleted", "1");
                }
                else
                {
                    regs = sqlc.getAllReg("%", "Deleted", "1");
                }
                break;

        }
        if (regs != null)
        {
            string[] regList = new string[10];
            while (n < regs.Length)
            {
                if (regs[n][0] != null)
                {
                    regList[0] = regs[n][0];
                    regList[1] = regs[n][14];
```

```
                    if (regs[n][21].Equals("True"))
                    {
                        regList[2] = "x";
                    }
                    else
                    {
                        regList[2] = "";
                    }

                    if (regs[n][11].Equals("Received"))
                    {
                        regList[3] = "R";
                    }
                    else if (regs[n][11].Equals("Produced"))
                    {
                        regList[3] = "P";
                    }
                    regList[4] = sqlc.getConf(regs[n][5]);
                    regList[5] = regs[n][13];
                    regList[6] = regs[n][12];
                    regList[7] = regs[n][15];
                    regList[8] = regs[n][9];
                    regList[9] = regs[n][8];

                    ListViewItem regItem = new ListViewItem(regList);
                    Registrations_list.Items.Add(regItem);
                }
                n++;

            }
        }

    }
    /* Hvis brugeren er logget ind vises en NewRegistration form.
     * Hvis brugeren ikke er logget ind bliver denne bedt om dette.*/

    private void reg_id_btn_Click(object sender, EventArgs e)
    {
        if (loggedon)
        {
            Form NewRegistration = new NewRegistration();
            NewRegistration.Show();
        }
        else
        {
            MessageBox.Show("Please log in to use this feature");
        }
    }
    /* Åbner en ja/nej dialog hvor brugeren bliver spurgt om programmet skal lukkes.
     * Hvis ja vælges, lukkes programmet. */
    private void close_btn_Click(object sender, EventArgs e)
    {
        DialogResult result = MessageBox.Show("Are you sure you want to exit HiDocuments?",
"Exit?", MessageBoxButtons.YesNo);
        if (result == DialogResult.Yes)
        {
            this.Close();
        }
    }
```

```
        /* Åbner en Login form.*/
        private void Login_btn_Click(object sender, EventArgs e)
        {
            Form Logon = new Logon();
            Logon.Show();

        }


        /* Henter de valgte registreringer fra databasen igen. */
        private void Refresh_btn_Click(object sender, EventArgs e)
        {
            this.showAllRegs();
        }
        /* Hvis brugeren er logget ind som Administrator vil en Registrationtypes form blive
åbnet.
         * Hvis brugeren ikke har Administrator rettigheder vil denne blive informeret om
dette. */
        private void regtypes_btn_Click(object sender, EventArgs e)
        {
            if (admin)
            {
                Form Registrationtypes = new Registrationtypes();
                Registrationtypes.Show();
            }
            else
            {
                MessageBox.Show("You need to be an Administrator to use this feature");
            }
        }
        /* Åbner en UpdateRegistration form til den registrering der er blevet clicked. */
        private void Registrations_list_ItemActivate(object sender, EventArgs e)
        {
            Form UpdateRegistration = new
UpdateRegistration(Registrations_list.SelectedItems[0].Text);
            UpdateRegistration.Show();
        }

        public static bool Loggedon
        {
            get
            {
                return loggedon;
            }
            set
            {
                loggedon = value;
            }
        }

        public static bool Admin
        {
            get
            {
                return admin;
            }
            set
            {
                admin = value;
```

```
            }
        }

        public static string Username
        {
            get
            {
                return username;
            }
            set
            {
                username = value;
            }
        }

        /* Aktiverer slet, makuler og ny version knapperne for den markerede registrering.
         * Dette er dog kun tilfældet hvis brugeren er logget ind i systemet. */
        private void Registrations_list_Click(object sender, EventArgs e)
        {
            if (loggedon)
            {
                del_reg_btn.Enabled = true;
                shred_reg_btn.Enabled = true;
                new_version_btn.Enabled = true;
            }
        }
        /* Sletter den markerede registrering hvis brugeren konfirmerer dette vha. et OK/Cancel
dialog. */
        private void del_reg_btn_Click(object sender, EventArgs e)
        {
            DialogResult result = MessageBox.Show("Are you sure you want to mark the current
Registration as deleted?", "Delete?", MessageBoxButtons.OKCancel);
            if (result == DialogResult.OK)
            {
                sqlc.DelRegistration(Registrations_list.SelectedItems[0].Text);
            }
        }
        /* Fjerner den markerede registrering fra databasen hvis brugeren konfirmerer dette
vha. et OK/Cancel dialog. */
        private void shred_reg_btn_Click(object sender, EventArgs e)
        {
            DialogResult result = MessageBox.Show("Are you sure you want to shred the current
Registration?", "Shred?", MessageBoxButtons.OKCancel);
            if (result == DialogResult.OK)
            {

                sqlc.ShredRegistration(Registrations_list.SelectedItems[0].Text);
            }
        }
        /* Kalder showAllRegs funktionen når der bliver ændret på kategori boksen.  */
        private void Category_box_SelectedIndexChanged(object sender, EventArgs e)
        {

            this.showAllRegs();

        }
        /* Kalder showAllRegs når teksten i kategori boksen ændres.  */
        private void Category_box_TextUpdate(object sender, EventArgs e)
        {
```

```
            this.showAllRegs();
        }
        /* Samme som overstående funktion, men denne funktion kalder også ParameterCheck
funktionen. */
        private void Selection_box_TextUpdate(object sender, EventArgs e)
        {
            this.ParameterCheck();
            this.showAllRegs();
        }
        /* Samme som overstående funktion. */
        private void Selection_box_SelectedIndexChanged(object sender, EventArgs e)
        {
            this.ParameterCheck();
            this.showAllRegs();
        }
        /* Hvis teksten i parameter boksen ændre bliver showAllRegs kaldt. */
        private void Parameter_text_TextChanged(object sender, EventArgs e)
        {
            this.showAllRegs();
        }
        /* Checker om parameter boksen skal være synlig, og sætte synligheden til den korrekte
værdi. */
        private void ParameterCheck()
        {
            if (Selection_box.Text == "Search on document ID" || Selection_box.Text == "Search
on Purpose/Business/Document name" || Selection_box.Text == "Search on sender, author or
receiver" || Selection_box.Text == "Search by Document link")
            {
                Parameter_label.Visible = true;
                Parameter_text.Visible = true;
                Parameter_text.Enabled = true;
            }
            else
            {
                Parameter_label.Visible = false;
                Parameter_text.Visible = false;
                Parameter_text.Enabled = false;
            }
        }
        /* Åbner en UpdateRegistration form ud fra den markerede registrering.  */
        private void new_version_btn_Click(object sender, EventArgs e)
        {
            Form UpdateRegistration = new
UpdateRegistration(Registrations_list.SelectedItems[0].Text);
            UpdateRegistration.Show();
        }
        /* Udskriver de viste registreringer til en tekst fil, h
         * vis placering og navn bliver bestemt af brugeren vha. et saveFileDialog. */
        private void Export()
        {
            SaveFileDialog saveFileDialog = new SaveFileDialog();
            saveFileDialog.Filter = "Text file|*.txt";
            saveFileDialog.Title = "Export view";
            saveFileDialog.ShowDialog();


            if (saveFileDialog.FileName != "")
            {
```

```
                FileInfo t = new FileInfo(saveFileDialog.FileName);
                StreamWriter Tex = t.CreateText();
                string[] names = sqlc.getColNames();
                for (int j = 0; j < names.Length; j++)
                {
                    Tex.Write(names[j] + ",");
                }
                Tex.Write(Tex.NewLine);

                for (int n = 0; n < regs.Length; n++)
                {
                    for (int i = 0; i < regs[n].Length; i++)
                    {
                        Tex.Write(regs[n][i] + ",");
                    }
                    Tex.Write(Tex.NewLine);
                }
                Tex.Close();

            }
        }
        /* Kalder Export funktionen */
        private void Export_btn_Click(object sender, EventArgs e)
        {
            this.Export();
        }
    }



}
```

NewRegistration.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace HiDocuments
{
    public partial class NewRegistration : Form
    {
        SQL sqlc = SQL.getInstance();
        static string[] sqlA = new string[20];
        int cattype = 2;
        /* Initialiserer formens komponenter. Ud over dette bliver sender/author sat til at
være brugerens navn.
         * Kategori og konfidentialitets boksene bliver også fyldt med de valg der findes i
databasen. */
        public NewRegistration()
        {
            InitializeComponent();
            Author_text_out.Text = Registrations.Username;
```

```
            Sender_text_inc.Text = Registrations.Username;
            Category_text.Items.AddRange(sqlc.getCatLinks(cattype));
            Confidentiality_text.Items.AddRange(sqlc.getConfs());

        }
        /* Hvis alle obligatoriske felter er udfyldt i formen kaldes RegArray,
         * RegistrationSql samt setID. Til sidst bliver formen lukket. */
        private void Register_btn_Click(object sender, EventArgs e)
        {
            if (Category_text.Text == "" || desc_text.Text == "" || Group_text.Text == "" ||
Confidentiality_text.Text == ""
                || Regtype_text.Text == "")
            {
                MessageBox.Show("Please fill in the required fields");
            }
            if (Regtype_text.Text == "Received")
            {
                if(Sender_text_inc.Text == "" || Date_text_inc.Text == "" )
                {
                     MessageBox.Show("Please fill in the required fields");
                }
                else
                {
                    RegArray();
                sqlc.RegistrationSql(sqlA);
                sqlc.setID();
                this.Close();

                }

            }
            if (Regtype_text.Text == "Produced")
            {
                if(Author_text_out.Text == "" || Date_text_out.Text == "")
                {
                     MessageBox.Show("Please fill in the required fields");

                }
                else
                {
                    RegArray();
                sqlc.RegistrationSql(sqlA);
                sqlc.setID();
                this.Close();

                }
            }

        }
        /* Kalder ShowPanel funktionen når der ændres i Registrerings type boksen */
        private void Regtype_text_TextUpdate(object sender, EventArgs e)
        {
            ShowPanel();
        }
        /* Opretter et array med de informationer der er udfyldt i formen. */
        private void RegArray()
        {
            sqlA[0] = DocIDText.Text;
            if(TypeRadio1.Checked)
```

```
            {
                sqlA[1] = TypeRadio1.Text;
            }
            else if(TypeRadio2.Checked)
            {
                sqlA[1] = TypeRadio2.Text;
            }
            else
            {
                sqlA[1] = TypeRadio3.Text;
            }
            sqlA[2] = Category_text.Text;
            sqlA[3] = desc_text.Text;
            sqlA[4] = Group_text.Text;
            sqlA[5] = Confidentiality_text.Text;
            sqlA[6] = Examples_text.Text;
            if (Barbox.Checked)
            {
                sqlA[7] = "Checked";
            }
            else
            {
                sqlA[7] = "Unchecked";
            }
            sqlA[8] = Link_text.Text;
            sqlA[9] = Purpose_text.Text;
            sqlA[10] = AddInfo_text.Text;
            sqlA[11] = Regtype_text.Text;
            if (Regtype_text.Text.Equals("Received"))
            {
                sqlA[12] = Sender_text_inc.Text;
                sqlA[13] = Date_text_inc.Text;
                sqlA[14] = Version_text_inc.Text;
                sqlA[15] = SenderID_text_inc.Text;
                sqlA[16] = Ess_text_inc.Text;
                sqlA[17] = BelongMis_text_inc.Text;
                sqlA[18] = Attach_text.Text;
                sqlA[19] = Distribution_text.Text;
            }
            else if (Regtype_text.Text.Equals("Produced"))
            {
                sqlA[12] = Author_text_out.Text;
                sqlA[13] = Date_text_out.Text;
                sqlA[14] = Version_text_out.Text;
                sqlA[15] = Recby_text_out.Text;
                sqlA[16] = Ess_text_out.Text;
                sqlA[17] = BelongArc_text_out.Text;
                sqlA[18] = Attach_text.Text;
                sqlA[19] = Distribution_text.Text;
            }
        }
        /* Lukker formen. */
        private void Cancel_btn_Click(object sender, EventArgs e)
        {
            this.Close();
        }
        /* Checker hvad der er indtastet i registrerings type boksen og viser det relevante
panel. */
        private void ShowPanel()
```

```
{
    if (Regtype_text.Text.Equals("Received"))
    {
        Out_panel.Hide();
        Inc_panel.Show();
    }
    else if (Regtype_text.Text.Equals("Produced"))
    {
        Inc_panel.Hide();
        Out_panel.Show();
    }
}
/* Hvis der ændres på indekset i registrerings type boksen bliver ShowPanel kaldt. */
private void Regtype_text_SelectedIndexChanged(object sender, EventArgs e)
{
    ShowPanel();
}
/* Genererer et dokument id ud fra den indtastede information. */
private void makeID()
{
        string docID;
        docID = Category_text.Text;
        if (Group_text.Text.Length > 0)
        {
            docID = docID + "-" + sqlc.getGroup(Group_text.Text);
        }
        if (desc_text.Text.Length > 0)
        {
            docID = docID + "-" + sqlc.getType(desc_text.Text);
        }
        if (Regtype_text.Text.Length > 0)
        {
            docID = docID + "-" + Regtype_text.Text.Remove(1);
        }
        if (Regtype_text.Text.Length > 0)
        {
            docID = docID + DateTime.Today.Year.ToString();
        }
        docID = docID + "-" + sqlc.getID().ToString();
        docID = docID + sqlc.getConf(Confidentiality_text.Text);

        DocIDText.Text = docID;
}


/* Kalder makeID, når brugeren forlader gruppe feltet.  */
private void Group_text_Leave(object sender, EventArgs e)
{
    makeID();
}

private void Confidentiality_text_Leave(object sender, EventArgs e)
{
    makeID();
}

private void Regtype_text_Leave(object sender, EventArgs e)
{
    makeID();
```

```
        }
        /* Ændre indholdet af kategori boksen ud fra informationerne i databasen. */
        private void TypeRadio1_CheckedChanged(object sender, EventArgs e)
        {
            if (TypeRadio1.Checked)
            {
                cattype = 1;
                Category_text.Items.Clear();
                Category_text.Items.AddRange(sqlc.getCatLinks(cattype));
            }
        }
        /* Ændre indholdet af kategori boksen ud fra informationerne i databasen. */
        private void TypeRadio2_CheckedChanged(object sender, EventArgs e)
        {
            if (TypeRadio2.Checked)
            {
                cattype = 2;
                Category_text.Items.Clear();
                Category_text.Items.AddRange(sqlc.getCatLinks(cattype));
            }
        }
        /* Ændre indholdet af kategori boksen ud fra informationerne i databasen. */
        private void TypeRadio3_CheckedChanged(object sender, EventArgs e)
        {
            if (TypeRadio3.Checked)
            {
                cattype = 3;
                Category_text.Items.Clear();

                Category_text.Items.AddRange(sqlc.getCatLinks(cattype));
            }
        }
        /* Aktiverer beskrivelses feltet, samt henter valgmuligheder ud fra databasen.
         * Dette sker når kategori feltet bliver udfyldt. */
        private void Category_text_SelectionChangeCommitted(object sender, EventArgs e)
        {

                desc_text.Enabled = true;
                desc_text.Items.Clear();
                desc_text.Items.AddRange(sqlc.getDescLinks(cattype));


            makeID();

        }
        /* Aktiverer gruppe feltet, samt henter valgmuligheder ud fra databasen.
         * Dette sker når beskrivelses feltet bliver udfyldt. */
        private void desc_text_SelectionChangeCommitted(object sender, EventArgs e)
        {

                Group_text.Enabled = true;
                Group_text.Items.Clear();
                Group_text.Items.AddRange(sqlc.getGroupLinks(desc_text.Text));

            makeID();
        }
    }
```

```
}
```

UpdateRegistration.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using SHDocVw;
using System.Runtime.InteropServices;

/* Da mange af funktionerne er de samme som i NewRegistration
 * henvises der til denne for informationer om disse.
 * Der vil kun blive beskrevet unikke funktioner nedenfor */


namespace HiDocuments
{
    public partial class UpdateRegistration : Form
    {
        int cattype;
        SQL sqlc = SQL.getInstance();
        static string[] sqlA = new string[23];
        /* Initialiserer klassens komponenter.
         * Ud over dette bliver den valgte registrering hentet fra databasen.
         * Hvis brugeren er logget ind vil selve opdater knappen også blive aktiveret.  */
        public UpdateRegistration(string ID)
        {

            InitializeComponent();
            this.showReg(ID);
            Confidentiality_text.Items.AddRange(sqlc.getConfs());

            if (Registrations.Loggedon)
            {
                Register_btn.Enabled = true;
            }

        }


        private void Register_btn_Click(object sender, EventArgs e)
        {
            if (Category_text.Text == "" || desc_text.Text == "" || Group_text.Text == "" ||
Confidentiality_text.Text == ""
                || Regtype_text.Text == "")
            {
                MessageBox.Show("Please fill in the required fields");
            }
            if (Regtype_text.Text == "Received")
            {
                if (Sender_text_inc.Text == "" || Date_text_inc.Text == "")
                {
```

```
                    MessageBox.Show("Please fill in the required fields");
                }
                else
                {
                    DialogResult result = MessageBox.Show("Are you sure you want to update
registration?", "Update?", MessageBoxButtons.OKCancel);
                    if (result == DialogResult.OK)
                    {
                        RegArray();
                        sqlc.setInvalid(sqlA[0]);
                        sqlc.RegistrationSql(sqlA);

                        this.Close();
                    }

                }

            }
            if (Regtype_text.Text == "Produced")
            {
                if (Author_text_out.Text == "" || Date_text_out.Text == "")
                {
                    MessageBox.Show("Please fill in the required fields");

                }
                else
                {
                    DialogResult result = MessageBox.Show("Are you sure you want to update
registration?", "Update?", MessageBoxButtons.OKCancel);
                    if (result == DialogResult.OK)
                    {
                        RegArray();
                        sqlc.setInvalid(sqlA[0]);
                        sqlc.RegistrationSql(sqlA);

                        this.Close();
                    }

                }
            }

        }

        private void Regtype_text_TextUpdate(object sender, EventArgs e)
        {
            ShowPanel();
        }
        private void RegArray()
        {
            sqlA[0] = DocIDText.Text;
            if(TypeRadio1.Checked)
            {
                 sqlA[1] = TypeRadio1.Text;
            }
            else if(TypeRadio2.Checked)
            {
                 sqlA[1] = TypeRadio2.Text;
            }
            else
```

```
        {
             sqlA[1] = TypeRadio3.Text;
        }
        sqlA[2] = Category_text.Text;
        sqlA[3] = desc_text.Text;
        sqlA[4] = Group_text.Text;
        sqlA[5] = Confidentiality_text.Text;
        sqlA[6] = Examples_text.Text;
        if (Barbox.Checked)
        {
            sqlA[7] = "Checked";
        }
        else
        {
            sqlA[7] = "Unchecked";
        }
        sqlA[8] = Link_text.Text;
        sqlA[9] = Purpose_text.Text;
        sqlA[10] = AddInfo_text.Text;
        sqlA[11] = Regtype_text.Text;
        if (Regtype_text.Text.Equals("Received"))
        {
            sqlA[12] = Sender_text_inc.Text;
            sqlA[13] = Date_text_inc.Text;
            sqlA[14] = Version_text_inc.Text;
            sqlA[15] = SenderID_text_inc.Text;
            sqlA[16] = Ess_text_inc.Text;
            sqlA[17] = BelongMis_text_inc.Text;
            sqlA[18] = Attach_text.Text;
            sqlA[19] = Distribution_text.Text;
        }
        else if (Regtype_text.Text.Equals("Produced"))
        {
            sqlA[12] = Author_text_out.Text;
            sqlA[13] = Date_text_out.Text;
            sqlA[14] = Version_text_out.Text;
            sqlA[15] = Recby_text_out.Text;
            sqlA[16] = Ess_text_out.Text;
            sqlA[17] = BelongArc_text_out.Text;
            sqlA[18] = Attach_text.Text;
            sqlA[19] = Distribution_text.Text;
        }
    }

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

    private void ShowPanel()
    {
        if (Regtype_text.Text.Equals("Received"))
        {
            Out_panel.Hide();
            Inc_panel.Show();
        }
        else if (Regtype_text.Text.Equals("Produced"))
        {
            Inc_panel.Hide();
```

```
            Out_panel.Show();
        }
    }

    private void Regtype_text_SelectedIndexChanged(object sender, EventArgs e)
    {
        ShowPanel();
    }
    /* Denne funktioner indsætter informationerne om registreringen i de relevante felter i
formen.  */
    private void showReg(string ID)
    {
        string[] regs = new string[24];
        regs = sqlc.getReg(ID);

        DocIDText.Text = regs[0];
        if (regs[1].Equals("Company"))
        {
            TypeRadio1.Checked = true;

        }
        else if (regs[1].Equals("Project"))
        {
            TypeRadio2.Checked = true;

        }
        else
        {
            TypeRadio3.Checked = true;

        }
        Category_text.Text = regs[2];
        desc_text.Text = regs[3];
        Group_text.Text = regs[4];
        Confidentiality_text.Text = regs[5];
        Examples_text.Text = regs[6];
        if (regs[7].Equals("True"))
        {
            Barbox.Checked = true;
        }
        else
        {
            Barbox.Checked = false;
        }
        Link_text.Text = regs[8];
        Purpose_text.Text = regs[9];
        AddInfo_text.Text = regs[10];
        Regtype_text.Text = regs[11];
        if (Regtype_text.Text.Equals("Received"))
        {
            Sender_text_inc.Text = regs[12];
            Date_text_inc.Text = regs[13];
            Version_text_inc.Text = regs[14];
            SenderID_text_inc.Text = regs[15];
            Ess_text_inc.Text = regs[16];
            BelongMis_text_inc.Text = regs[17];
            Attach_text.Text = regs[18];
            Distribution_text.Text = regs[19];
        }
```

```
        else if (Regtype_text.Text.Equals("Produced"))
        {
            Author_text_out.Text = regs[12];
            Date_text_out.Text = regs[13];
            Version_text_out.Text = regs[14];
            Recby_text_out.Text = regs[15];
            Ess_text_out.Text = regs[16];
            BelongArc_text_out.Text = regs[17];
            Attach_text.Text = regs[18];
            Distribution_text.Text = regs[19];
        }
        Dated.Text = Dated.Text + regs[20].Remove(10);
    }

    private void Link_text_MouseDoubleClick(object sender, MouseEventArgs e)
    {

    }

    private void makeID()
    {
        string docID;
        docID = Category_text.Text;
        if (Group_text.Text.Length > 0)
        {
            docID = docID + "-" + sqlc.getGroup(Group_text.Text);
        }
        if (desc_text.Text.Length > 0)
        {
            docID = docID + "-" + sqlc.getType(desc_text.Text);
        }
        if (Regtype_text.Text.Length > 0)
        {
            docID = docID + "-" + Regtype_text.Text.Remove(1);
        }
        if (Regtype_text.Text.Length > 0)
        {
            docID = docID + DateTime.Today.Year.ToString();
        }
        docID = docID + "-" + sqlc.getID().ToString();
        docID = docID + sqlc.getConf(Confidentiality_text.Text);

        DocIDText.Text = docID;
    }



    private void Group_text_Leave(object sender, EventArgs e)
    {
        makeID();
    }

    private void Confidentiality_text_Leave(object sender, EventArgs e)
    {
        makeID();
    }

    private void Regtype_text_Leave(object sender, EventArgs e)
    {
```

```
        makeID();
    }

    private void TypeRadio1_CheckedChanged(object sender, EventArgs e)
    {
        if (TypeRadio1.Checked)
        {
            cattype = 1;
            Category_text.Items.Clear();
            Category_text.Items.AddRange(sqlc.getCatLinks(cattype));
        }
    }

    private void TypeRadio2_CheckedChanged(object sender, EventArgs e)
    {
        if (TypeRadio2.Checked)
        {
            cattype = 2;
            Category_text.Items.Clear();
            Category_text.Items.AddRange(sqlc.getCatLinks(cattype));
        }
    }

    private void TypeRadio3_CheckedChanged(object sender, EventArgs e)
    {
        if (TypeRadio3.Checked)
        {
            cattype = 3;
            Category_text.Items.Clear();

            Category_text.Items.AddRange(sqlc.getCatLinks(cattype));
        }
    }

    private void Category_text_SelectionChangeCommitted(object sender, EventArgs e)
    {

        desc_text.Enabled = true;
        desc_text.Items.Clear();
        desc_text.Items.AddRange(sqlc.getDescLinks(cattype));


        makeID();

    }

    private void desc_text_SelectionChangeCommitted(object sender, EventArgs e)
    {

        Group_text.Enabled = true;
        Group_text.Items.Clear();
        Group_text.Items.AddRange(sqlc.getGroupLinks(desc_text.Text));

        makeID();
    }

    private void Link_text_MouseDoubleClick_1(object sender, MouseEventArgs e)
    {
        InternetExplorer explorer = new InternetExplorer();
```

```
            if (explorer != null)
            {
                explorer.Visible = true;
                object x = null;
                explorer.Navigate(@Link_text.Text, ref x, ref x, ref x, ref x);
            }

        }
    }
}
```

Logon.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace HiDocuments
{
    public partial class Logon : Form
    {
        SQL sqlc = SQL.getInstance();
        /* Initialiserer formens komponenter. */
        public Logon()
        {
            InitializeComponent();
        }
        /* Lukker formen */
        private void Close_btn_Click(object sender, EventArgs e)
        {
            this.Close();
        }
        /* Checker om de indtastede informationer passer,
         * og logger brugeren hvis dette er tilfældet. */
        private void Login_btn_Click(object sender, EventArgs e)
        {

                                if(sqlc.checkLogin(Username_text.Text, Password_text.Text))
                                {
                Registrations.Loggedon = true;
                Registrations.Username = sqlc.checkName(Username_text.Text);
                Registrations.Admin = sqlc.checkAdmin(Username_text.Text);
                this.Close();

                                }
                                else if(Username_text.Text == "" || Password_text.Text == "")
                                {
                                        MessageBox.Show("Username or Password is missing");
                                }
                                else
                                {
```

```
                                        MessageBox.Show("Username or Password dosnt
match");
                                }
        }
    }
}
```

Registrationtypes.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace HiDocuments
{
    public partial class Registrationtypes : Form
    {
        SQL sqlc = SQL.getInstance();

        /* Initialiserer formens komponenter, samt kalder showLinks. */
        public Registrationtypes()
        {
            InitializeComponent();
            showLinks();
        }
        /* Kalder getAllLinks funktionen fra SQL klassen,
         * hvorefter de returnerede links bliver vist i en listview. */
        private void showLinks()
        {

            Typelist.Items.Clear();
            string[][] links = sqlc.getAllLinks();
            int n = 0;
            if (links != null)
            {

                while (n < links.Length)
                {
                    if (links[n][4] != null)
                    {
                        ListViewItem linkItem = new ListViewItem(links[n]);
                        Typelist.Items.Add(linkItem);
                    }
                    n++;

                }
            }

        }
        /* Lukker formen */
        private void Close_btn_Click(object sender, EventArgs e)
        {
            this.Close();
```

```
        }
        /* Åbner en kateogri form */
        private void Cat_btn_Click(object sender, EventArgs e)
        {
            Form Category = new Category();
            Category.Show();
        }
        /* Åbner en gruppe form */
        private void Groups_btn_Click(object sender, EventArgs e)
        {
            Form Group = new Group();
            Group.Show();
        }
        /* Åbner en type form */
        private void Types_btn_Click(object sender, EventArgs e)
        {
            Form Type = new Type();
            Type.Show();
        }
        /* Åbner en links form */
        private void Newlink_btn_Click(object sender, EventArgs e)
        {
            Form Links = new Links();
            Links.Show();
        }
        /* Sletter det markerede link fra databasen, hvis brugeren konfirmerer dette. */
        private void Remlink_btn_Click(object sender, EventArgs e)
        {

            DialogResult result = MessageBox.Show("Are you sure you want to shred the current
Link?", "Link?", MessageBoxButtons.OKCancel);
            if (result == DialogResult.OK)
            {

                sqlc.remLink(Typelist.SelectedItems[0].SubItems[4].Text);
                this.showLinks();
            }

        }
        /* Kalder showLinks */
        private void Registrationtypes_Enter(object sender, EventArgs e)
        {
            this.showLinks();
        }

        private void Users_btn_Click(object sender, EventArgs e)
        {
            Form users = new Users();
            users.Show();
        }

    }
}
```

Users.cs

```
using System;
```

```
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace HiDocuments
{
    public partial class Users : Form
    {
        SQL sqlc = SQL.getInstance();
        /* Initialiserer formens komponenter, samt kalder showUsers. */
        public Users()
        {
            InitializeComponent();
            this.showUsers();
        }
        /* Lukker formen */
        private void Close_btn_Click(object sender, EventArgs e)
        {
            this.Close();
        }
        /* Henter alle brugere fra databasen og viser disse i et listview. */
        private void showUsers()
        {
            Users_list.Items.Clear();
            string[] users = sqlc.getUsers();

            string userList = "";
            int n = 0;
            if (users != null)
            {
                while (n < users.Length)
                {
                    userList = users[n];

                    ListViewItem userItem = new ListViewItem(userList);
                    Users_list.Items.Add(userItem);
                    n++;
                }
            }
        }
        /* Fjerner den markerede bruger fra databasen. Dette kræver dog at brugeren konfirmerer
dette. */
        private void Remove_btn_Click(object sender, EventArgs e)
        {
            DialogResult result = MessageBox.Show("Are you sure you want to delete this User?",
"Delete?", MessageBoxButtons.OKCancel);
            if (result == DialogResult.OK)
            {
                sqlc.removeUser(Users_list.SelectedItems[0].Text);
                this.showUsers();
            }
        }
        /* Hvis alle felter i formen er udfyldt vil brugeren blive oprettet i systemet.  */
        private void Add_btn_Click(object sender, EventArgs e)
        {
            if (Usernamebox.Text == "" || Passbox.Text == "" || Namebox.Text == "")
```

```
            {
                MessageBox.Show("Please fill out all the fields");
            }
            else
            {
                sqlc.newUser(Usernamebox.Text, Passbox.Text, Namebox.Text, adminbox.Checked);
                 this.showUsers();
            }
        }
    }
}
```

Type.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace HiDocuments
{
    public partial class Type : Form
    {
        SQL sqlc = SQL.getInstance();
        /* Initialiserer formens komponenter samt kalder showTypes. */
        public Type()
        {
            InitializeComponent();
            this.showTypes();
        }
        /* Lukker formen */
        private void Close_btn_Click(object sender, EventArgs e)
        {
            this.Close();
        }
        /* Viser alle typer i en listview. */
        private void showTypes()
        {
            Type_list.Items.Clear();
            string[] types = sqlc.getTypes();

            string typeList = "";
            int n = 0;
            if (types != null)
            {
                while (n < types.Length)
                {
                    typeList = types[n];

                    ListViewItem typeItem = new ListViewItem(typeList);
                    Type_list.Items.Add(typeItem);
                    n++;
                }
```

```
            }
        }
        /* Fjerner den markerede type hvis brugeren konfirmerer dette. */
        private void Remove_btn_Click(object sender, EventArgs e)
        {
            DialogResult result = MessageBox.Show("Are you sure you want to delete this Type?",
"Delete?", MessageBoxButtons.OKCancel);
            if (result == DialogResult.OK)
            {
                sqlc.removeType(Type_list.SelectedItems[0].Text);
                this.showTypes();
            }
        }
        /* Hvis alle felter i formen er udfyldt vil den indtastede type bliver oprettet i
databasen. */
        private void Add_btn_Click(object sender, EventArgs e)
        {
            if (Typebox.Text == "")
            {
                MessageBox.Show("Please fill out all the fields");
            }
            else
            {
                sqlc.NewType(Typebox.Text);
                 this.showTypes();
            }
        }
    }
}
```

Group.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace HiDocuments
{
    public partial class Group : Form
    {
        SQL sqlc = SQL.getInstance();
        /* Initialiserer formens komponenter samt kalder showGroups. */
        public Group()
        {
            InitializeComponent();
            this.showGroups();
        }
        /* Lukker formen */
        private void Close_btn_Click(object sender, EventArgs e)
        {
            this.Close();
        }
        /* Viser alle grupper fra databasen i et listview. */
```

```
        private void showGroups()
        {
            Group_list.Items.Clear();
            string[][] groups = sqlc.getFullGroups();

            string[] groupList = new string[2];
            int n = 0;
            if (groups != null)
            {
                while (n < groups.Length)
                {
                    groupList[0] = groups[n][0];
                    groupList[1] = groups[n][1];

                    ListViewItem groupItem = new ListViewItem(groupList);
                    Group_list.Items.Add(groupItem);
                    n++;
                }
            }
        }
        /* Fjerner den markerede gruppe hvis bruger konfirmerer dette. */
        private void Remove_btn_Click(object sender, EventArgs e)
        {
            DialogResult result = MessageBox.Show("Are you sure you want to delete this
Group?", "Shred?", MessageBoxButtons.OKCancel);
            if (result == DialogResult.OK)
            {
                sqlc.removeGroup(Group_list.SelectedItems[0].Text);
                this.showGroups();
            }
        }
        /* Opretter den relevante gruppe hvis alle felter udfyldt i formen. */
        private void Add_btn_Click(object sender, EventArgs e)
        {
            if (Groupbox.Text == "" || Descriptionbox.Text == "")
            {
                MessageBox.Show("Please fill out all the fields");
            }
            else
            {
                sqlc.NewGroup(Groupbox.Text, Descriptionbox.Text);
                 this.showGroups();
            }
        }
    }
}
```

Category.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
```

```
namespace HiDocuments
{
    public partial class Category : Form
    {
        SQL sqlc = SQL.getInstance();
        /* Initialiserer formens komponenter samt kalder showCats. */
        public Category()
        {
            InitializeComponent();
            Categorytypebox.Items.AddRange(sqlc.getCatTypes());
            this.showCats();
        }
        /* Lukker formen. */
        private void Close_btn_Click(object sender, EventArgs e)
        {
            this.Close();
        }
        /* Viser alle kategorier fra databasen i et listview. */
        private void showCats()
        {
            Category_list.Items.Clear();
            string[][] cats = sqlc.getFullCats();

            string[] catList = new string[2];
            int n = 0;
            if (cats != null)
            {
                while (n < cats.Length)
                {
                    catList[1] = cats[n][0];
                    catList[0] = cats[n][1];

                    ListViewItem catItem = new ListViewItem(catList);
                    Category_list.Items.Add(catItem);
                    n++;
                }
            }
        }
        /* Fjerner den markerede kategori hvis bruger konfirmerer dette. */
        private void Remove_btn_Click(object sender, EventArgs e)
        {
            DialogResult result = MessageBox.Show("Are you sure you want to delete this
Category?", "Shred?", MessageBoxButtons.OKCancel);
            if (result == DialogResult.OK)
            {
                sqlc.removeCat(Category_list.SelectedItems[0].SubItems[1].Text);
                this.showCats();
            }
        }
        /* Opretter den relevante kategori hvis alle felter udfyldt i formen. */
        private void Add_btn_Click(object sender, EventArgs e)
        {
            if (Categorybox.Text == "" || Categorytypebox.Text == "")
            {
                MessageBox.Show("Please fill out all the fields");
            }
            else
            {
```

```
                sqlc.NewCat(Categorytypebox.Text, Categorybox.Text);
                this.showCats();
            }
        }
    }
}
```

Links.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace HiDocuments
{
    public partial class Links : Form
    {
        SQL sqlc = SQL.getInstance();
        /* Initialiserer formens komponenter, samt udfylder de 3 combobokse med mulighederne,
fra databasen. */
        public Links()
        {
            InitializeComponent();
            Categorybox.Items.AddRange(sqlc.getCatTypes());
            Groupbox.Items.AddRange(sqlc.getGroups());
            Typebox.Items.AddRange(sqlc.getTypes());
        }
        /* Lukker formen */
        private void Close_btn_Click(object sender, EventArgs e)
        {
            this.Close();
        }
        /* Hvis alle felter er udfyldt korrekt vil linket blive oprettet. */
        private void Create_btn_Click(object sender, EventArgs e)
        {
            if (Categorybox.Text == "" || Groupbox.Text == "" || Typebox.Text == "" ||
Descbox.Text == "")
            {
                MessageBox.Show("Please fill out the appropriate fields");
            }
            else
            {
                this.createLink();
                this.Close();
            }
        }
        /* Opretter et array med informationer om linket, fra formen. Samt sender dette til
newLink.  */
        private void createLink()
        {
            string[] linkA = new string[9];
```

```
            linkA[0] = Categorybox.Text;
            linkA[1] = Groupbox.Text;
            linkA[2] = Typebox.Text;
            linkA[3] = Descbox.Text;
            linkA[4] = Verbox.Checked.ToString();
            linkA[5] = Templatebox.Text;
            linkA[6] = Approvebox.Checked.ToString();
            linkA[7] = Authbox.Text;
            if (Norew_btn.Checked)
            {
                linkA[8] = " ";
            }
            else if (Optionalrev_btn.Checked)
            {
                linkA[8] = "O";
            }
            else
            {
                linkA[8] = "M";
            }
            sqlc.newLink(linkA);
        }
    }
}
```

SQL.cs

```
using System;
using System.Collections.Generic;
using System.Collections;
using System.Text;
using System.Data;
using System.Data.Sql;
using System.Data.SqlTypes;
using System.Data.SqlClient;
using Microsoft.SqlServer.Server;


namespace HiDocuments
{
    class SQL
    {
        static string db = "HiQWise";
        static string userid = "Hiq";
        static string pass = "hiq123";
        static string server = "CST";

        static SQL instance = new SQL();

        /* Singleton der returnerer SQL klassen. */
        public static SQL getInstance()
        {
            return instance;
        }
        [SqlProcedure()]
        /* Indsætter den aktuelle registrering i databasen.  */
```

```
        public void RegistrationSql(string[] sqlA)
        {

            using (SqlConnection conn = new SqlConnection("user id= " +userid+";password = " +
pass + "; server = " +server +"; database =" + db + ";"))
            {
                int index = this.getIndex();
                try
                {
                    int barbox;
                    {
                        if (sqlA[7].Equals("Checked"))
                        {
                            barbox = 1;
                        }
                        else
                        {
                            barbox = 0;
                        }

                        SqlCommand InsertReg = new SqlCommand();
                        InsertReg.CommandText = "INSERT INTO Registration " +
                            "(DocID, Cat_type, Category, Desc_doc, Group_, Conf, Copies, Bar,
Link, " +
                          "Purpose, Ainfo, Regtype, Author_Sender, Date, Version,
Recby_senderid, Ess, " +
                           " Belongnr, Attachment, Distribution, Regdate, Deleted, Invalid,
Indexnr) VALUES " +
                                "('" + sqlA[0] + "','" + sqlA[1] + "','" + sqlA[2] + "', " +
                               " '" + sqlA[3] + "','" + sqlA[4] + "','" + sqlA[5] + "','" +
sqlA[6] + "', "+
                               " " + barbox + ",'" + sqlA[8] + "','" + sqlA[9] + "','" +
sqlA[10] + "', "+
                               " '" + sqlA[11] + "','" + sqlA[12] + "','" + sqlA[13] + "','" +
sqlA[14] + "', "+
                               " '" + sqlA[15] + "','" + sqlA[16] + "','" + sqlA[17] + "','" +
sqlA[18] + "',"+
                               " '" + sqlA[19] + "','" + DateTime.Today + "',0,0, +'" + index
+"')";
                        InsertReg.Connection = conn;
                        conn.Open();
                        InsertReg.ExecuteNonQuery();
                        conn.Close();
                        this.setIndex();
                    }
                }
                catch (SqlException ie)
                {
                    Console.WriteLine(ie.ToString());
                    throw ie;
                }
            }
        }

        public static string Userid
        {
            get
            {
                return userid;
```

```
            }
            set
            {
                userid = value;
            }
        }

        public static string DB
        {
            get
            {
                return db;
            }
            set
            {
                db = value;
            }
        }

        public static string Pass
        {
            get
            {
                return pass;
            }
            set
            {
                pass = value;
            }
        }

        public static string Server
        {
            get
            {
                return server;
            }
            set
            {
                server = value;
            }
        }
        /* Sætter den relevante registrering som slettet i databasen.  */
        public void DelRegistration(string ID)
        {
            using (SqlConnection conn = new SqlConnection("user id= " +userid+";password = " +
pass + "; server = " +server +"; database =" + db + ";"))
            {
                SqlCommand DelReg = new SqlCommand();
                DelReg.CommandText = "UPDATE Registration SET Deleted = '1' WHERE DocID = '"
+ID+ "'";
                DelReg.Connection = conn;
                conn.Open();
                DelReg.ExecuteNonQuery();
                conn.Close();



            }
```

```
        }
        /* Fjerner den relevante registrering fra databasen. */
        public void ShredRegistration(string ID)
        {
            using (SqlConnection conn = new SqlConnection("user id= " +userid+";password = " +
pass + "; server = " +server +"; database =" + db + ";"))
            {
                SqlCommand ShredReg = new SqlCommand();
                ShredReg.CommandText = "DELETE FROM Registration WHERE DocID = '" + ID + "'";
                ShredReg.Connection = conn;
                conn.Open();
                ShredReg.ExecuteNonQuery();
                conn.Close();
            }
        }
        /* Sætter den valgte registrering som invalid. */
        public void setInvalid(string ID)
        {
            using (SqlConnection conn = new SqlConnection("user id= " +userid+";password = " +
pass + "; server = " +server +"; database =" + db + ";"))
            {
                SqlCommand DelReg = new SqlCommand();
                DelReg.CommandText = "UPDATE Registration SET Invalid= '1' WHERE DocID = '" +
ID + "'";
                DelReg.Connection = conn;
                conn.Open();
                DelReg.ExecuteNonQuery();
                conn.Close();
            }
        }
        /* Henter det nyeste ID og adderer dette med 1. */
        public void setID()
        {
            using (SqlConnection conn = new SqlConnection("user id= " +userid+";password = " +
pass + "; server = " +server +"; database =" + db + ";"))
            {
                try
                {
                    {
                        int id = this.getID();
                        int nid = id + 1;
                        SqlCommand updateID = new SqlCommand();
                        updateID.CommandText = "UPDATE ID SET DocID = '" + nid + "' WHERE DocID
= '" + id + "'";
                        updateID.Connection = conn;
                        conn.Open();
                        updateID.ExecuteNonQuery();
                        conn.Close();

                    }
                }
                catch (SqlException ie)
                {
                    Console.WriteLine(ie.ToString());
                    throw ie;
                }
            }
        }
        /* Henter det nyeste index og adderer dette med 1. */
```

```
        public void setIndex()
        {
            using (SqlConnection conn = new SqlConnection("user id= " +userid+";password = " +
pass + "; server = " +server +"; database =" + db + ";"))
            {
                try
                {
                    {
                        int index = this.getIndex();
                        int nindex = index + 1;
                        SqlCommand updateIndex = new SqlCommand();
                        updateIndex.CommandText = "UPDATE Index_nr SET Indexnr = '" + nindex +
"' WHERE Indexnr = '" + index + "'";
                        updateIndex.Connection = conn;
                        conn.Open();
                        updateIndex.ExecuteNonQuery();
                        conn.Close();

                    }
                }
                catch (SqlException ie)
                {
                    Console.WriteLine(ie.ToString());
                    throw ie;
                }
            }
        }


        /* Henter antallet af registreringer i databasen. */
        public int getRows()
        {
            using (SqlConnection conn = new SqlConnection("user id= " +userid+";password = " +
pass + "; server = " +server +"; database =" + db + ";"))
            {
                try
                {
                    {

                        SqlCommand getRows = new SqlCommand("SELECT COUNT(*) FROM
Registration", conn);
                        conn.Open();
                        int rows = (int)getRows.ExecuteScalar();
                        conn.Close();

                        return rows;

                    }
                }
                catch (SqlException ie)
                {
                    Console.WriteLine(ie.ToString());
                    throw ie;
                }
            }
        }
        /* Henter antallet af links i databasen. */
        public int getLinkRows()
        {
```

```
            using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password = "
+ pass + "; server = " + server + "; database =" + db + ";"))
            {
                try
                {
                    {

                        SqlCommand getRows = new SqlCommand("SELECT COUNT(*) FROM
Registrationtypes", conn);
                        conn.Open();
                        int rows = (int)getRows.ExecuteScalar();
                        conn.Close();

                        return rows;

                    }
                }
                catch (SqlException ie)
                {
                    Console.WriteLine(ie.ToString());
                    throw ie;
                }
            }
        }
        /* Henter det nyeste ID fra databasen.  */
        public int getID()
        {
            using (SqlConnection conn = new SqlConnection("user id= " +userid+";password = " +
pass + "; server = " +server +"; database =" + db + ";"))
            {
                try
                {
                    {
                        SqlCommand getID = new SqlCommand("SELECT DocID FROM ID", conn);
                        conn.Open();
                        int id = (int)getID.ExecuteScalar();
                        conn.Close();
                        return id;
                    }
                }
                catch (SqlException ie)
                {
                    Console.WriteLine(ie.ToString());
                    throw ie;
                }
            }
        }
        /* Henter det nyeste index. */
        public int getIndex()
        {
            using (SqlConnection conn = new SqlConnection("user id= " +userid+";password = " +
pass + "; server = " +server +"; database =" + db + ";"))
            {
                try
                {
                    {
                        SqlCommand getIndex = new SqlCommand("SELECT Indexnr FROM Index_nr",
conn);
                        conn.Open();
```

```
                    int index = (int)getIndex.ExecuteScalar();
                    conn.Close();
                    return index;
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }
        }
    }
    /* Returnerer den valgte registrering fra databasen.  */
    public string[] getReg(string ID)
    {
        try
            {
                using (SqlConnection conn = new SqlConnection("user id= "
+userid+";password = " + pass + "; server = " +server +"; database =" + db + ";"))
                {
            string[] regs = new string[24];
            SqlDataReader regReader = null;
            int i = 0;
            SqlCommand regCommand = new SqlCommand("SELECT * FROM Registration WHERE DocID
= '" + ID + " '", conn);
            conn.Open();
            regReader = regCommand.ExecuteReader();
            if (regReader.HasRows)
            {
                while (regReader.Read())
                {
                    for (i = 0; i < regReader.FieldCount; i++)
                    {
                        if (i == 7 || i == 21 || i == 22)
                        {
                            regs[i] = regReader.GetBoolean(i).ToString();
                        }
                        else if (i == 23)
                        {
                            regs[i] = regReader.GetInt32(i).ToString();
                        }
                        else
                        {
                            regs[i] = regReader.GetString(i);
                        }
                    }


                }
            }
            else
            {
                regs = null;
            }

            regReader.Close();
            conn.Close();
            return regs;
        }
```

```
        }
        catch (SqlException ie)
        {
            Console.WriteLine(ie.ToString());
            throw ie;
        }

        }
        /* Returnerer gruppe forkortelsen af den valgte gruppe. */
        public string getGroup(string group)
        {
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " +userid+";password =
" + pass + "; server = " +server +"; database =" + db + ";"))
                {
                    SqlCommand regCommand = new SqlCommand("SELECT Group_short FROM
Registration_group WHERE Group_long = '" + group + "'", conn);
                    conn.Open();
                    SqlDataReader regReader = null;
                    string gr = null;
                    regReader = regCommand.ExecuteReader();
                    while (regReader.Read())
                    {
                        gr = regReader.GetString(0);

                    }
                    regReader.Close();
                    conn.Close();
                    return gr;
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }

        }
        /* Henter gruppe navnet ud fra dennes forkortelse. */
        public string getGroupL(string group)
        {
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password
= " + pass + "; server = " + server + "; database =" + db + ";"))
                {
                    SqlCommand regCommand = new SqlCommand("SELECT Group_long FROM
Registration_group WHERE Group_short = '" + group + "'", conn);
                    conn.Open();
                    SqlDataReader regReader = null;
                    string gr = null;
                    regReader = regCommand.ExecuteReader();
                    while (regReader.Read())
                    {
                        gr = regReader.GetString(0);

                    }
                    regReader.Close();
```

```
                    conn.Close();
                    return gr;
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }

        }
        /* Henter alle grupper fra databasen.  */
        public string[] getGroups()
        {
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " +userid+";password =
" + pass + "; server = " +server +"; database =" + db + ";"))
                {
                    SqlCommand regCommand = new SqlCommand("SELECT Group_long FROM
Registration_group", conn);
                    SqlCommand rowCommand = new SqlCommand("SELECT COUNT(*)FROM
Registration_group", conn);


                    SqlDataReader regReader = null;
                    conn.Open();
                    int rows = (int)rowCommand.ExecuteScalar();
                    string[] groups = new string[rows];
                    regReader = regCommand.ExecuteReader();
                    int i = 0;
                    while (regReader.Read())
                    {
                        groups[i] = regReader.GetString(0);
                        i++;
                    }

                    regReader.Close();
                    conn.Close();
                    return groups;
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }

        }
        /* Henter alle brugere fra databasen.  */
        public string[] getUsers()
        {
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password
= " + pass + "; server = " + server + "; database =" + db + ";"))
                {
                    SqlCommand regCommand = new SqlCommand("SELECT Username FROM Login", conn);
                    SqlCommand rowCommand = new SqlCommand("SELECT COUNT(*)FROM Login", conn);
```

```
                    SqlDataReader regReader = null;
                    conn.Open();
                    int rows = (int)rowCommand.ExecuteScalar();
                    string[] users = new string[rows];
                    regReader = regCommand.ExecuteReader();
                    int n = 0;
                    while (regReader.Read())
                    {
                        users[n] = regReader.GetString(0);
                        n++;

                    }

                    regReader.Close();
                    conn.Close();
                    return users;
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }

        }
        /* Henter forkortelsen af kategori typen, ud fra det fulde navn. */
        public string getCatType(string cattype)
        {
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password
= " + pass + "; server = " + server + "; database =" + db + ";"))
                {
                    SqlCommand regCommand = new SqlCommand("SELECT Categorytype_short FROM
Registration_categorytype WHERE Categorytype_long = '" + cattype + " '", conn);
                    conn.Open();
                    SqlDataReader regReader = null;
                    string ct = null;
                    regReader = regCommand.ExecuteReader();
                    while (regReader.Read())
                    {
                        ct = regReader.GetString(0);

                    }
                    regReader.Close();
                    conn.Close();
                    return ct;
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }

        }
        /* Henter forkortelsen af kategori typen, ud fra det fulde navn. */
```

```
        public string[] getCatTypes()
        {
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password
= " + pass + "; server = " + server + "; database =" + db + ";"))
                {
                    SqlCommand regCommand = new SqlCommand("SELECT Categorytype_long FROM
Registration_categorytype", conn);
                    SqlCommand rowCommand = new SqlCommand("SELECT COUNT(*)FROM
Registration_categorytype", conn);


                    SqlDataReader regReader = null;
                    conn.Open();
                    int rows = (int)rowCommand.ExecuteScalar();
                    string[] cattypes = new string[rows];
                    regReader = regCommand.ExecuteReader();
                    int i = 0;
                    while (regReader.Read())
                    {
                        cattypes[i] = regReader.GetString(0);
                        i++;
                    }

                    regReader.Close();
                    conn.Close();
                    return cattypes;
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }

        }
        /* Henter alle kategorier i databasen. */
        public string[] getCats()
        {
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password
= " + pass + "; server = " + server + "; database =" + db + ";"))
                {
                    SqlCommand regCommand = new SqlCommand("SELECT Category FROM
Registration_category", conn);
                    SqlCommand rowCommand = new SqlCommand("SELECT COUNT(*)FROM
Registration_category", conn);


                    SqlDataReader regReader = null;
                    conn.Open();
                    int rows = (int)rowCommand.ExecuteScalar();
                    string[] cats = new string[rows];
                    regReader = regCommand.ExecuteReader();
                    int i = 0;
                    while (regReader.Read())
                    {
```

```
                        cats[i] = regReader.GetString(0);
                        i++;
                    }

                    regReader.Close();
                    conn.Close();
                    return cats;
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }

        }

        /* Checker om en type eksister. */
        public bool checkTypes(string table, string col, string var)
        {
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password
= " + pass + "; server = " + server + "; database =" + db + ";"))
                {
                    SqlCommand rowCommand = new SqlCommand("SELECT COUNT(*)FROM " + table + "
WHERE " + col + " = '" + var + "'", conn);

                    conn.Open();
                    int rows = (int)rowCommand.ExecuteScalar();
                    conn.Close();
                    if (rows > 0)
                    {
                        return true;
                    }
                    else
                    {
                        return false;
                    }
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }
        }

        /* Henter alle beskrivelser fra databasen.  */
        public string[] getDescs()
        {
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password
= " + pass + "; server = " + server + "; database =" + db + ";"))
                {
                    SqlCommand regCommand = new SqlCommand("SELECT Description FROM
Registrationtypes", conn);
```

```
                    SqlCommand rowCommand = new SqlCommand("SELECT COUNT(*)FROM
Registrationtypes", conn);


                    SqlDataReader regReader = null;
                    conn.Open();
                    int rows = (int)rowCommand.ExecuteScalar();
                    string[] descs = new string[rows];
                    regReader = regCommand.ExecuteReader();
                    int i = 0;
                    while (regReader.Read())
                    {

                        descs[i] = regReader.GetString(0);
                        i++;
                    }

                    regReader.Close();
                    conn.Close();
                    return descs;
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }

        }
        /* Henter alle typer fra databasen. */
        public string[] getTypes()
        {
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " +userid+";password =
" + pass + "; server = " +server +"; database =" + db + ";"))
                {
                    SqlCommand regCommand = new SqlCommand("SELECT Type FROM
Registration_type", conn);
                    SqlCommand rowCommand = new SqlCommand("SELECT COUNT(*)FROM
Registration_type", conn);


                    SqlDataReader regReader = null;
                    conn.Open();
                    int rows = (int)rowCommand.ExecuteScalar();
                    string[] types = new string[rows];
                    regReader = regCommand.ExecuteReader();
                    int i = 0;
                    while (regReader.Read())
                    {

                        types[i] = regReader.GetString(0);
                        i++;
                    }

                    regReader.Close();
                    conn.Close();
                    return types;
```

```
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }

        }
        /* Henter forkortelsen af konfidentialiteten. */
        public string getConf(string conf)
        {
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " +userid+";password =
" + pass + "; server = " +server +"; database =" + db + ";"))
                {
                    SqlCommand regCommand = new SqlCommand("SELECT Confidentiality_short FROM
Registration_conf WHERE Confidentiality_long = '" + conf + " '", conn);
                    conn.Open();
                    SqlDataReader regReader = null;
                    string con = null;
                    regReader = regCommand.ExecuteReader();
                    while (regReader.Read())
                    {
                        con = regReader.GetString(0);

                    }
                    regReader.Close();
                    conn.Close();
                    return con;
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }

        }
        /* Henter alle konfidentialiteter fra databasen. */
        public string[] getConfs()
        {
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " +userid+";password =
" + pass + "; server = " +server +"; database =" + db + ";"))
                {
                    SqlCommand regCommand = new SqlCommand("SELECT Confidentiality_long FROM
Registration_conf", conn);
                    SqlCommand rowCommand = new SqlCommand("SELECT COUNT(*)FROM
Registration_conf", conn);


                    SqlDataReader regReader = null;
                    conn.Open();
                    int rows = (int)rowCommand.ExecuteScalar();
                    string[] confs = new string[rows];
                    regReader = regCommand.ExecuteReader();
```

```
                    int i = 0;
                    while (regReader.Read())
                    {
                        confs[i] = regReader.GetString(0);
                        i++;
                    }

                    regReader.Close();
                    conn.Close();
                    return confs;
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }

        }
        /* Checker om username og password passer. */
        public bool checkLogin(string User, string Pass)
        {
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " +userid+";password =
" + pass + "; server = " +server +"; database =" + db + ";"))
                {
                    SqlCommand regCommand = new SqlCommand("SELECT Pass FROM Login WHERE
Username = '" + User + " '", conn);
                    conn.Open();
                    SqlDataReader regReader = null;
                    string con = null;
                    regReader = regCommand.ExecuteReader();
                    while (regReader.Read())
                    {
                        con = regReader.GetString(0);
                    }
                    regReader.Close();
                    conn.Close();
                    return Pass.Equals(con);
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }
        }
        /* Checker om brugeren er Administrator. */
        public bool checkAdmin(string User)
        {
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password
= " + pass + "; server = " + server + "; database =" + db + ";"))
                {
                    SqlCommand regCommand = new SqlCommand("SELECT Admin FROM Login WHERE
Username = '" + User + " '", conn);
                    conn.Open();
```

```
                    SqlDataReader regReader = null;
                    bool con = false;
                    regReader = regCommand.ExecuteReader();
                    while (regReader.Read())
                    {
                        con = regReader.GetBoolean(0);
                    }
                    regReader.Close();
                    conn.Close();
                    return con;
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }
        }
        /* Henter brugeren fulde navn fra databasen. */
        public string checkName(string User)
        {
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password
= " + pass + "; server = " + server + "; database =" + db + ";"))
                {
                    SqlCommand regCommand = new SqlCommand("SELECT Name FROM Login WHERE
Username = '" + User + " '", conn);
                    conn.Open();
                    SqlDataReader regReader = null;
                    string con = null;
                    regReader = regCommand.ExecuteReader();
                    while (regReader.Read())
                    {
                        con = regReader.GetString(0);
                    }
                    regReader.Close();
                    conn.Close();
                    return con;
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }
        }

        public string[][] getAllReg(string group)
        {
            try
            {
               using (SqlConnection conn = new SqlConnection("user id= " +userid+";password = "
+ pass + "; server = " +server +"; database =" + db + ";"))
                 {
                    int nrRows = this.getRows();
                    int n = 0;
                    string[][] regs = new string[nrRows][];
                    for (int j = 0; j < nrRows; j++)
```

```
                    {
                        regs[j] = new string[24];
                    }

                    SqlDataReader regReader = null;
                    SqlCommand regCommand = new SqlCommand("SELECT * FROM Registration WHERE
Category LIKE '"+ group +"'", conn);
                    regCommand.Connection = conn;
                    conn.Open();
                    regReader = regCommand.ExecuteReader();
                    if (regReader.HasRows)
                    {
                        while (regReader.Read())
                        {

                            for (int i = 0; i < regReader.FieldCount; i++)
                            {
                                if (i == 7 || i == 21 || i == 22)
                                {
                                    regs[n][i] = regReader.GetBoolean(i).ToString();

                                }
                                else if (i == 23)
                                {
                                    regs[n][i] = regReader.GetInt32(i).ToString();
                                }
                                else
                                {
                                    regs[n][i] = regReader.GetString(i);
                                }

                            }
                            n++;

                        }
                    }
                    else
                    {
                        regs = null;
                    }


                    regReader.Close();
                    conn.Close();
                    return regs;
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                return null;
               // throw ie;
            }

        }
        /* Henter alle registreringer af den valgte gruppe. */
        public string[][] getAllReg(string group, string col, string var)
        {
            try
```

```
            {
                using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password
= " + pass + "; server = " + server + "; database =" + db + ";"))
                {
                    int nrRows = this.getRows();
                    int n = 0;
                    string[][] regs = new string[nrRows][];
                    for (int j = 0; j < nrRows; j++)
                    {
                        regs[j] = new string[24];
                    }

                    SqlDataReader regReader = null;
                    SqlCommand regCommand = new SqlCommand("SELECT * FROM Registration WHERE "
+ col + " LIKE'" + var + "' AND Category LIKE '"+ group +"'", conn);
                    regCommand.Connection = conn;
                    conn.Open();
                    regReader = regCommand.ExecuteReader();
                    if (regReader.HasRows)
                    {
                        while (regReader.Read())
                        {
                            for (int i = 0; i < regReader.FieldCount; i++)
                            {
                                if (i == 7 || i == 21 || i == 22)
                                {
                                    regs[n][i] = regReader.GetBoolean(i).ToString();

                                }
                                else if (i == 23)
                                {
                                    regs[n][i] = regReader.GetInt32(i).ToString();
                                }
                                else
                                {
                                    regs[n][i] = regReader.GetString(i);
                                }

                            }
                            n++;

                        }
                    }
                    else
                    {
                        regs = null;
                    }

                    regReader.Close();
                    conn.Close();
                    return regs;
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }
```

```
        }
        /* Henter alle registreringer der passer til de søgte kriterier. */
        public string[][] getAllReg(string group, string col1, string col2, string var, string
var2, string andor)
        {
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password
= " + pass + "; server = " + server + "; database =" + db + ";"))
                {
                    int nrRows = this.getRows();
                    int n = 0;
                    string[][] regs = new string[nrRows][];
                    for (int j = 0; j < nrRows; j++)
                    {
                        regs[j] = new string[24];
                    }

                    SqlDataReader regReader = null;
                    SqlCommand regCommand = new SqlCommand("SELECT * FROM Registration WHERE
("+col1+" LIKE '"+ var +"' "+ andor +" "+col2+" LIKE '"+var2+"') AND Category LIKE
'"+group+"'", conn);
                    regCommand.Connection = conn;
                    conn.Open();
                    regReader = regCommand.ExecuteReader();
                    if (regReader.HasRows)
                    {
                        while (regReader.Read())
                        {
                            for (int i = 0; i < regReader.FieldCount; i++)
                            {
                                if (i == 7 || i == 21 || i == 22)
                                {
                                    regs[n][i] = regReader.GetBoolean(i).ToString();

                                }
                                else if (i == 23)
                                {
                                    regs[n][i] = regReader.GetInt32(i).ToString();
                                }
                                else
                                {
                                    regs[n][i] = regReader.GetString(i);
                                }

                            }
                            n++;

                        }
                    }
                    else
                    {
                        regs = null;
                    }

                    regReader.Close();
                    conn.Close();
                    return regs;
                }
```

```
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }

        }
        /* Opretter en ny kategori i database, ud fra de sendte oplysninger. */
        public void NewCat(string cattype, string cat)
        {
            using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password = "
+ pass + "; server = " + server + "; database =" + db + ";"))
            {
                try
                {
                    SqlCommand getID = new SqlCommand("SELECT MAX(Category_id) FROM
Registration_category", conn);
                    SqlCommand getCatTypeId = new SqlCommand("SELECT Categorytype_id FROM
Registration_categorytype WHERE Categorytype_long = '" + cattype + "'");
                    getCatTypeId.Connection = conn;
                    SqlDataReader reader = null;
                    SqlCommand InsertCat = new SqlCommand();
                    conn.Open();
                    int typeid = (int)getCatTypeId.ExecuteScalar();
                    int id = 0;
                    reader = getID.ExecuteReader();
                    while (reader.Read())
                    {
                        try{
                            id = reader.GetInt32(0);
                            id++;
                        }
                        catch(Exception e)
                        {
                            Console.WriteLine(e);
                            id = 1;
                        }
                    }

                    InsertCat.CommandText = "INSERT INTO Registration_category " +
                            "(Category_id, Categorytype_id, Category) VALUES " +
                                "(" + id + "," + typeid + ",'" + cat + "')";
                    InsertCat.Connection = conn;
                    reader.Close();
                    InsertCat.ExecuteNonQuery();

                    conn.Close();
                }
                catch (SqlException ie)
                {
                    Console.WriteLine(ie.ToString());
                    throw ie;
                }
            }

        }
        /* Opretter en ny gruppe i databasen, ud fra de givne oplysninger. */
        public void NewGroup(string group_short, string group_long)
```

```
        {
            using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password = "
+ pass + "; server = " + server + "; database =" + db + ";"))
            {
                try
                {
                    SqlCommand getID = new SqlCommand("SELECT MAX(Group_id) FROM
Registration_group", conn);
                    SqlDataReader reader = null;
                    SqlCommand InsertGroup = new SqlCommand();
                    conn.Open();
                    int id = 0;
                    reader = getID.ExecuteReader();
                    while (reader.Read())
                    {
                        try
                        {
                            id = reader.GetInt32(0);
                            id++;
                        }
                        catch (Exception e)
                        {
                            Console.WriteLine(e);
                            id = 1;
                        }
                    }

                    InsertGroup.CommandText = "INSERT INTO Registration_group " +
                            "(Group_id, Group_short, Group_long) VALUES " +
                                "(" + id + ",'" + group_short + "','" + group_long + "')";
                    InsertGroup.Connection = conn;
                    reader.Close();
                    InsertGroup.ExecuteNonQuery();
                    conn.Close();
                }
                catch (SqlException ie)
                {
                    Console.WriteLine(ie.ToString());
                    throw ie;
                }
            }

        }
        /* Opretter en ny bruger ud fra de sendte oplysninger. */
        public void newUser(string usename, string password, string name, bool admin)
        {
            using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password = "
+ pass + "; server = " + server + "; database =" + db + ";"))
            {
                try
                {

                    SqlCommand InsertUser = new SqlCommand();
                    InsertUser.CommandText = "INSERT INTO Login VALUES ('" + usename + "','" +
password + "','"+admin+"','"+name+"')";
                    InsertUser.Connection = conn;
                    conn.Open();
                    ;
                    InsertUser.ExecuteNonQuery();
```

```
                    conn.Close();
                }
                catch (SqlException ie)
                {
                    Console.WriteLine(ie.ToString());
                    throw ie;
                }
            }

        }
        /* Opretter en ny type ud fra de givne oplysninger. */
        public void NewType(string type)
        {
            using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password = "
+ pass + "; server = " + server + "; database =" + db + ";"))
            {
                try
                {
                    SqlCommand getID = new SqlCommand("SELECT MAX(Type_id) FROM
Registration_type", conn);
                    SqlDataReader reader = null;
                    SqlCommand InsertType = new SqlCommand();
                    conn.Open();
                    int id = 0;
                    reader = getID.ExecuteReader();
                    while (reader.Read())
                    {
                        try
                        {
                            id = reader.GetInt32(0);
                            id++;
                        }
                        catch (Exception e)
                        {
                            Console.WriteLine(e);
                            id = 1;
                        }
                    }

                    InsertType.CommandText = "INSERT INTO Registration_Type " +
                            "(Type_id, Type) VALUES (" + id + ",'" + type + "')";
                    InsertType.Connection = conn;
                    reader.Close();
                    InsertType.ExecuteNonQuery();
                    conn.Close();
                }
                catch (SqlException ie)
                {
                    Console.WriteLine(ie.ToString());
                    throw ie;
                }
            }

        }

        /* Fjerner den valgte kategori fra databasen. */
        public void removeCat(string cat)
        {
```

```
            using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password = "
+ pass + "; server = " + server + "; database =" + db + ";"))
            {
                SqlCommand remCat = new SqlCommand();
                remCat.CommandText = "DELETE FROM Registration_category WHERE Category = '" +
cat + "'";
                remCat.Connection = conn;
                conn.Open();
                remCat.ExecuteNonQuery();
                conn.Close();
            }
        }
        /* Fjerner den valgte type fra databasen. */
        public void removeType(string type)
        {
            using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password = "
+ pass + "; server = " + server + "; database =" + db + ";"))
            {
                SqlCommand remType = new SqlCommand();
                remType.CommandText = "DELETE FROM Registration_type WHERE Type = '" + type +
"'";
                remType.Connection = conn;
                conn.Open();
                remType.ExecuteNonQuery();
                conn.Close();
            }
        }
        /* Fjerner den valgte bruger fra databasen. */
        public void removeUser(string user)
        {
            using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password = "
+ pass + "; server = " + server + "; database =" + db + ";"))
            {
                SqlCommand remUser = new SqlCommand();
                remUser.CommandText = "DELETE FROM Login WHERE Username = '" + user + "'";
                remUser.Connection = conn;
                conn.Open();
                remUser.ExecuteNonQuery();
                conn.Close();
            }
        }
        /* Fjerner den valgte gruppe fra databasen. */
        public void removeGroup(string group)
        {
            using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password = "
+ pass + "; server = " + server + "; database =" + db + ";"))
            {
                SqlCommand remGroup = new SqlCommand();
                remGroup.CommandText = "DELETE FROM Registration_group WHERE Group_short = '" +
group + "'";
                remGroup.Connection = conn;
                conn.Open();
                remGroup.ExecuteNonQuery();
                conn.Close();
            }
        }
        /* Henter alle informationer om en kategori fra databasen.  */
        public string[][] getFullCats()
        {
```

```
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password
= " + pass + "; server = " + server + "; database =" + db + ";"))
                {
                    SqlCommand regCommand = new SqlCommand("SELECT
Registration_category.Category, Registration_categorytype.Categorytype_short " +
                        "FROM Registration_category LEFT JOIN Registration_categorytype ON
Registration_category.Categorytype_id=Registration_categorytype.Categorytype_id", conn);
                    SqlCommand rowCommand = new SqlCommand("SELECT COUNT(*)FROM
Registration_category", conn);


                    SqlDataReader regReader = null;
                    conn.Open();
                    int n = 0;
                    int rows = (int)rowCommand.ExecuteScalar();
                    string[][] cats = new string[rows][];
                    for (int j = 0; j < rows; j++)
                    {
                        cats[j] = new string[2];
                    }
                    regReader = regCommand.ExecuteReader();

                    if (regReader.HasRows)
                    {
                        while (regReader.Read())
                        {

                            for (int i = 0; i < regReader.FieldCount; i++)
                            {
                                cats[n][i] = regReader.GetString(i);

                            }
                            n++;

                        }
                    }
                    else
                    {
                        cats = null;
                    }

                    regReader.Close();
                    conn.Close();
                    return cats;
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }

        }
        /* Hente alle informationer om alle grupper i databasen. */
        public string[][] getFullGroups()
        {
            try
```

```
            {
                using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password
= " + pass + "; server = " + server + "; database =" + db + ";"))
                {
                    SqlCommand regCommand = new SqlCommand("SELECT Group_short, Group_long FROM
Registration_group", conn);
                    SqlCommand rowCommand = new SqlCommand("SELECT COUNT(*)FROM
Registration_group", conn);


                    SqlDataReader regReader = null;
                    conn.Open();
                    int n = 0;
                    int rows = (int)rowCommand.ExecuteScalar();
                    string[][] groups = new string[rows][];
                    for (int j = 0; j < rows; j++)
                    {
                        groups[j] = new string[2];
                    }
                    regReader = regCommand.ExecuteReader();

                    if (regReader.HasRows)
                    {
                        while (regReader.Read())
                        {

                            for (int i = 0; i < regReader.FieldCount; i++)
                            {
                                groups[n][i] = regReader.GetString(i);

                            }
                            n++;

                        }
                    }
                    else
                    {
                        groups = null;
                    }

                    regReader.Close();
                    conn.Close();
                    return groups;
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }

        }
        /* Opretter en nyt link i databasen. */
        public void newLink(string[] linkA)
        {
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password
= " + pass + "; server = " + server + "; database =" + db + ";"))
```

```
                {
                    SqlCommand nLink = new SqlCommand();
                    int catType = this.getCatTypeId(linkA[0]);
                    int group = this.getGroupId(linkA[1]);
                    int type = this.getTypeId(linkA[2]);


                    nLink.CommandText = "INSERT INTO Registrationtypes VALUES " +
                        "(" + catType + "," + group + ","+ type + ",'" + linkA[3] +
                        "','" +linkA[4] + "','" + linkA[8] + "','" + linkA[6] + "','"+ linkA[7]
+"','"+ linkA[5] + "')";
                    nLink.Connection = conn;
                    conn.Open();
                    nLink.ExecuteNonQuery();
                    conn.Close();
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }


        }
        /* Henter id'et for den valgte kategori type ud fra navnet på denne. */
        public int getCatTypeId(string cattype)
        {
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password
= " + pass + "; server = " + server + "; database =" + db + ";"))
                {
                    SqlCommand regCommand = new SqlCommand("SELECT Categorytype_id FROM
Registration_categorytype WHERE Categorytype_long = '" + cattype + " '", conn);
                    conn.Open();
                    int ct = (int)regCommand.ExecuteScalar();
                    conn.Close();
                    return ct;
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }

        }
        /* Henter forkortelsen af kategori typen ud fra dennes id. */
        public string getCatType(int cattype)
        {
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password
= " + pass + "; server = " + server + "; database =" + db + ";"))
                {
                    SqlCommand regCommand = new SqlCommand("SELECT Categorytype_short FROM
Registration_categorytype WHERE Categorytype_id = '" + cattype + " '", conn);
                    conn.Open();
```

```
                    string ct = regCommand.ExecuteScalar().ToString();
                    conn.Close();
                    return ct;
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }

        }
        /* Henter id'et for den valgte gruppe, ud fra dennes navn. */
        public int getGroupId(string group)
        {
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password
= " + pass + "; server = " + server + "; database =" + db + ";"))
                {
                    SqlCommand regCommand = new SqlCommand("SELECT Group_id FROM
Registration_group WHERE Group_long = '" + group + " '", conn);
                    conn.Open();
                    int gr = (int)regCommand.ExecuteScalar();
                    conn.Close();
                    return gr;
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }

        }
        /* Henter gruppens forkortelse ud fra dennes id. */
        public string getGroup(int group)
        {
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password
= " + pass + "; server = " + server + "; database =" + db + ";"))
                {
                    SqlCommand regCommand = new SqlCommand("SELECT Group_short FROM
Registration_group WHERE Group_id = '" + group + " '", conn);
                    conn.Open();
                    string gr = regCommand.ExecuteScalar().ToString();
                    conn.Close();
                    return gr;
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }

        }
        /* Henter type id'et vha. typens navn. */
```

```
        public int getTypeId(string type)
        {
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password
= " + pass + "; server = " + server + "; database =" + db + ";"))
                {
                    SqlCommand regCommand = new SqlCommand("SELECT Type_id FROM
Registration_type WHERE Type = '" + type + " '", conn);
                    conn.Open();
                    int t = (int)regCommand.ExecuteScalar();
                    conn.Close();
                    return t;
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }

        }
        /* Henter typens navn ud fra dennes id. */
        public string getType(int type)
        {
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password
= " + pass + "; server = " + server + "; database =" + db + ";"))
                {
                    SqlCommand regCommand = new SqlCommand("SELECT Type FROM Registration_type
WHERE Type_id = '" + type + " '", conn);
                    conn.Open();
                    string t = regCommand.ExecuteScalar().ToString();
                    conn.Close();
                    return t;
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }

        }
        /* Henter typens id ud fra beskrivelsen af denne. */
        public string getType(string desc)
        {
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password
= " + pass + "; server = " + server + "; database =" + db + ";"))
                {
                    SqlCommand getid = new SqlCommand("SELECT Type_id FROM Registrationtypes
WHERE Description ='" +desc+"'",conn);

                    conn.Open();
                    int id = (int)getid.ExecuteScalar();
```

```
                    SqlCommand regCommand = new SqlCommand("SELECT Type FROM Registration_type
WHERE Type_id = '" + id + " '", conn);
                    string t = regCommand.ExecuteScalar().ToString();
                    conn.Close();
                    return t;
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }

        }
        /* Henter alle links fra databasen. */
        public string[][] getAllLinks()
        {
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password
= " + pass + "; server = " + server + "; database =" + db + ";"))
                {
                    int nrRows = this.getLinkRows() ;
                    int n = 0;
                    string[][] links = new string[nrRows][];
                    for (int j = 0; j < nrRows; j++)
                    {
                        links[j] = new string[10];
                    }

                    SqlDataReader regReader = null;
                    SqlCommand regCommand = new SqlCommand("SELECT * FROM Registrationtypes",
conn);
                    regCommand.Connection = conn;
                    conn.Open();
                    regReader = regCommand.ExecuteReader();
                    if (regReader.HasRows)
                    {
                        while (regReader.Read())
                        {

                            for (int i = 0; i < regReader.FieldCount; i++)
                            {
                                switch (i)
                                {
                                    case 0:
                                        links[n][i] = this.getCatType(regReader.GetInt32(i));
                                        break;

                                    case 1:
                                        links[n][i] = this.getGroup(regReader.GetInt32(i));
                                        break;

                                    case 2:
                                        links[n][i] = this.getType(regReader.GetInt32(i));
                                        break;

                                    case 4:
                                        if (regReader.GetBoolean(i))
```

```
                                    {
                                        links[n][i+1] = "X";
                                    }
                                    else
                                    {
                                        links[n][i+1] = "";
                                    }
                                    break;
                                case 6:
                                    if (regReader.GetBoolean(i))
                                    {
                                        links[n][i+1] = "X";
                                    }
                                    else
                                    {
                                        links[n][i+1] = "";
                                    }
                                    break;

                                default:
                                    if (i == 3)
                                    {
                                        links[n][3] = links[n][2];
                                        links[n][2] = this.getGroupL(links[n][1]);
                                    }
                                    links[n][i+1] = regReader.GetString(i);
                                    break;
                            }

                        }

                        n++;

                    }
                }
                else
                {
                    links = null;
                }


                regReader.Close();
                conn.Close();
                return links;
            }
        }
        catch (SqlException ie)
        {
            Console.WriteLine(ie.ToString());
            throw ie;
        }

    }
    /* Fjerner et link fra databasen ud fra dettes beskrivelse. */
    public void remLink(string desc)
    {
        using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password = "
+ pass + "; server = " + server + "; database =" + db + ";"))
        {
```

```
                SqlCommand ShredReg = new SqlCommand();
                ShredReg.CommandText = "DELETE FROM Registrationtypes WHERE Description = '" +
desc + "'";
                ShredReg.Connection = conn;
                conn.Open();
                ShredReg.ExecuteNonQuery();
                conn.Close();
            }
        }
        /* Henter kategorier ud fra deres id. */
        public string[] getCatLinks(int cattype)
        {
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password
= " + pass + "; server = " + server + "; database =" + db + ";"))
                {
                    SqlCommand regCommand = new SqlCommand("SELECT Category FROM
Registration_category WHERE Categorytype_id ='"+cattype+"'" , conn);
                    SqlCommand rowCommand = new SqlCommand("SELECT COUNT(*)FROM
Registration_category WHERE Categorytype_id ='"+cattype+"'", conn);


                    SqlDataReader regReader = null;
                    conn.Open();
                    int n = 0;
                    int rows = (int)rowCommand.ExecuteScalar();
                    string[] cats = new string[rows];

                    regReader = regCommand.ExecuteReader();

                    if (regReader.HasRows)
                    {
                        while (regReader.Read())
                        {


                                cats[n] = regReader.GetString(0);

                                n++;

                        }
                    }


                    regReader.Close();
                    conn.Close();
                    return cats;
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }

        }
        /* Henter beskrivelser ud fra deres id.*/
        public string[] getDescLinks(int cattype)
```

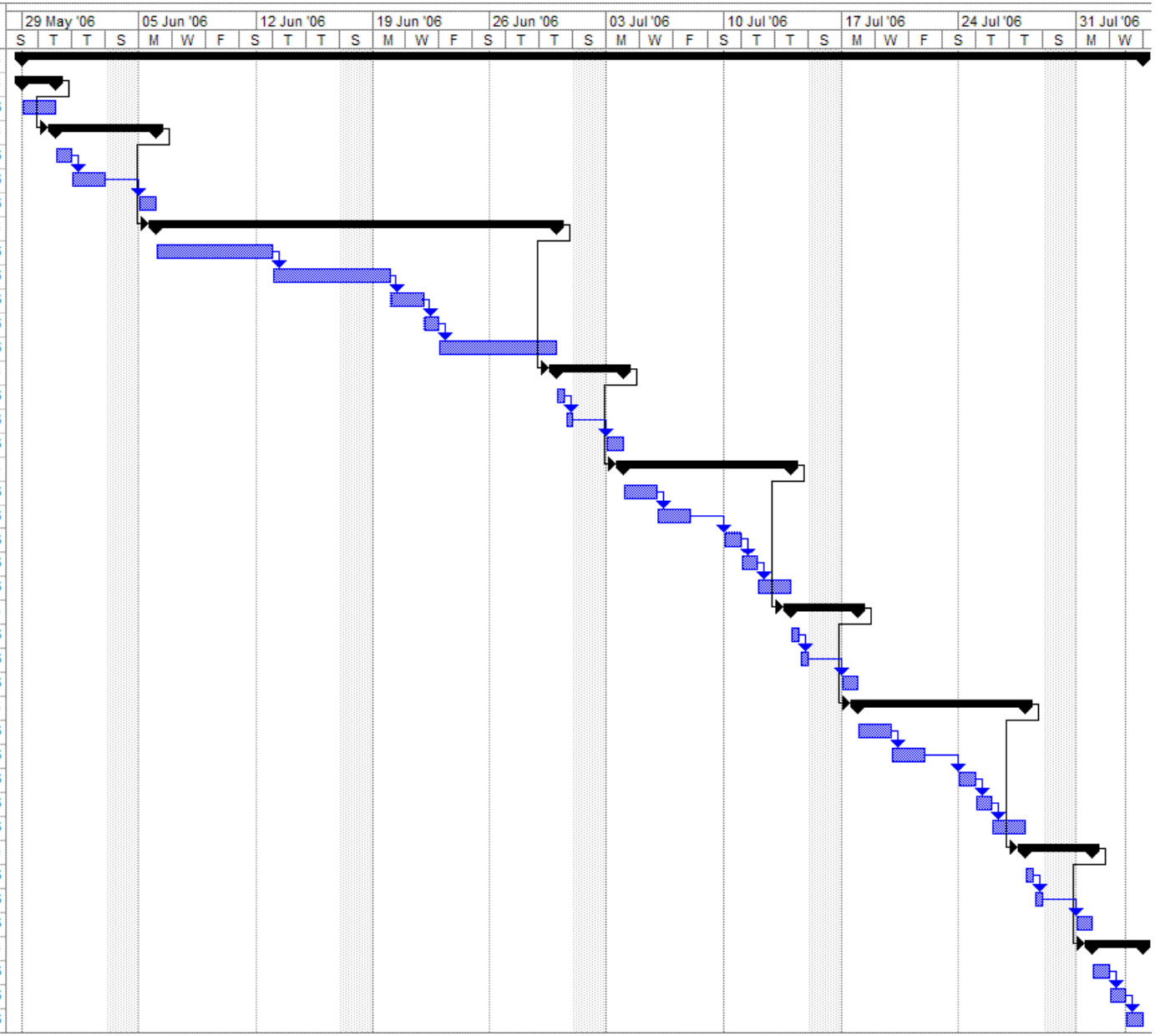
```
        {
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password
= " + pass + "; server = " + server + "; database =" + db + ";"))
                {
                    SqlCommand regCommand = new SqlCommand("SELECT Description FROM
Registrationtypes WHERE Categorytype_id =" + cattype, conn);
                    SqlCommand rowCommand = new SqlCommand("SELECT COUNT(*)FROM
Registrationtypes WHERE Categorytype_id =" + cattype, conn);


                    SqlDataReader regReader = null;
                    conn.Open();
                    int n = 0;
                    int rows = (int)rowCommand.ExecuteScalar();
                    string[] descs = new string[rows];

                    regReader = regCommand.ExecuteReader();

                    if (regReader.HasRows)
                    {
                        while (regReader.Read())
                        {


                            descs[n] = regReader.GetString(0);

                            n++;

                        }
                    }


                    regReader.Close();
                    conn.Close();
                    return descs;
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }

        }
        /* Henter grupper ud fra deres id. */
        public string[] getGroupLinks(string desc)
        {
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password
= " + pass + "; server = " + server + "; database =" + db + ";"))
                {
                    SqlCommand regCommand = new SqlCommand("SELECT
Registration_group.Group_long FROM Registrationtypes " +
                        "INNER JOIN Registration_group ON Registrationtypes.Group_id =
Registration_group.Group_id "+
                        "WHERE(Registrationtypes.Description = N'"+desc+"')", conn);
```

```
                    SqlCommand rowCommand = new SqlCommand("SELECT COUNT(*) FROM
Registrationtypes " +
                        "INNER JOIN Registration_group ON Registrationtypes.Group_id =
Registration_group.Group_id " +
                        "WHERE(Registrationtypes.Description = N'" + desc + "')", conn);

                    SqlDataReader regReader = null;
                    conn.Open();
                    int n = 0;
                    int rows = (int)rowCommand.ExecuteScalar();
                    string[] groups = new string[rows];

                    regReader = regCommand.ExecuteReader();

                    if (regReader.HasRows)
                    {
                        while (regReader.Read())
                        {


                            groups[n] = regReader.GetString(0);

                            n++;

                        }
                    }


                    regReader.Close();
                    conn.Close();
                    return groups;
                }
            }
            catch (SqlException ie)
            {
                Console.WriteLine(ie.ToString());
                throw ie;
            }

        }
        /* Henter kolonne navne på registreringerne i databasen. */
        public string[] getColNames()
        {
            try
            {
                using (SqlConnection conn = new SqlConnection("user id= " + userid + ";password
= " + pass + "; server = " + server + "; database =" + db + ";"))
                {
                    SqlCommand regCommand = new SqlCommand("SELECT * FROM Registration", conn);

                    SqlDataReader regReader = null;
                    conn.Open();
                    int n = 0;
                    string[] names = null;
                    regReader = regCommand.ExecuteReader();

                    if (regReader.HasRows)
                    {
                        while (regReader.Read())
```

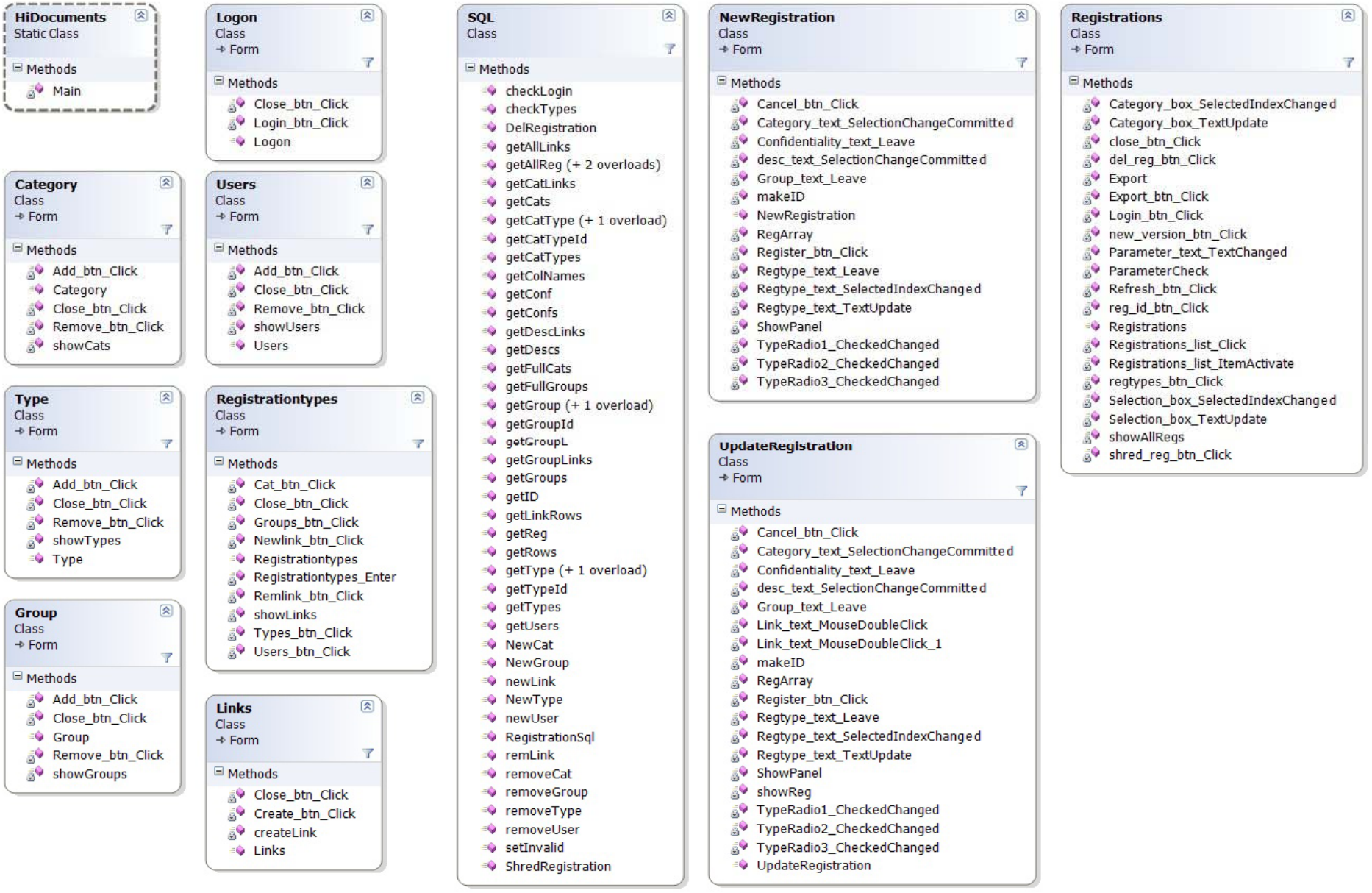
```
                    {
                        if (names == null)
                        {
                            names = new string[regReader.FieldCount];
                        }

                        while (n < names.Length)
                        {
                            names[n] = regReader.GetName(n);
                            n++;
                        }



                    }
                }


                regReader.Close();
                conn.Close();
                return names;
            }
        }
        catch (SqlException ie)
        {
            Console.WriteLine(ie.ToString());
            throw ie;
        }

    }
}


}
```

Bilag 2 - Tidsplan

Task Name	Duration	Start	Finish
Dokumentstyringssystem	**49 days**	**Mon 29-05-06**	**Thu 03-08-06**
Inception	**2 days**	**Mon 29-05-06**	**Tue 30-05-06**
Dokumentation	2 days	Mon 29-05-06	Tue 30-05-06
Basis Database	**4 days**	**Wed 31-05-06**	**Mon 05-06-06**
Design	1 day	Wed 31-05-06	Wed 31-05-06
Kodning	2 days	Thu 01-06-06	Fri 02-06-06
Dokumentation	1 day	Mon 05-06-06	Mon 05-06-06
Basis Program	**18 days**	**Tue 06-06-06**	**Thu 29-06-06**
Design	5 days	Tue 06-06-06	Mon 12-06-06
Kodning	5 days	Tue 13-06-06	Mon 19-06-06
Test	2 days	Tue 20-06-06	Wed 21-06-06
Rettelser	1 day	Thu 22-06-06	Thu 22-06-06
Dokumentation	5 days	Fri 23-06-06	Thu 29-06-06
Database Opdatering	**2 days**	**Fri 30-06-06**	**Mon 03-07-06**
Design	0,5 days	Fri 30-06-06	Fri 30-06-06
Kodning	0,5 days	Fri 30-06-06	Fri 30-06-06
Dokumentation	1 day	Mon 03-07-06	Mon 03-07-06
Administrator funktioner	**8 days**	**Tue 04-07-06**	**Thu 13-07-06**
Design	2 days	Tue 04-07-06	Wed 05-07-06
Kodning	2 days	Thu 06-07-06	Fri 07-07-06
Test	1 day	Mon 10-07-06	Mon 10-07-06
Rettelser	1 day	Tue 11-07-06	Tue 11-07-06
Dokumentation	2 days	Wed 12-07-06	Thu 13-07-06
Database Opdatering	**2 days**	**Fri 14-07-06**	**Mon 17-07-06**
Design	0,5 days	Fri 14-07-06	Fri 14-07-06
Kodning	0,5 days	Fri 14-07-06	Fri 14-07-06
Dokumentation	1 day	Mon 17-07-06	Mon 17-07-06
Integration til Subversion	**8 days**	**Tue 18-07-06**	**Thu 27-07-06**
Design	2 days	Tue 18-07-06	Wed 19-07-06
Kodning	2 days	Thu 20-07-06	Fri 21-07-06
Test	1 day	Mon 24-07-06	Mon 24-07-06
Rettelser	1 day	Tue 25-07-06	Tue 25-07-06
Dokumentation	2 days	Wed 26-07-06	Thu 27-07-06
Database Rettelser	**2 days**	**Fri 28-07-06**	**Mon 31-07-06**
Test	0,5 days	Fri 28-07-06	Fri 28-07-06
Rettelser	0,5 days	Fri 28-07-06	Fri 28-07-06
Dokumentation	1 day	Mon 31-07-06	Mon 31-07-06
Rapport	**3 days**	**Tue 01-08-06**	**Thu 03-08-06**
Dokumentation	1 day	Tue 01-08-06	Tue 01-08-06
Rettelser	1 day	Wed 02-08-06	Wed 02-08-06
Sidste dokumentation	1 day	Thu 03-08-06	Thu 03-08-06



Bilag 3 - Klassediagram



Bilag 4 – Database diagram