Automatisk generering af system til databehandling

Rune Juhl-Petersen

Kgs. Lyngby 2004

IMM 2004-44

1 Forord

En dag sad jeg på min arbejdsplads og udviklede endnu en brugergrænseflade oven på en database. Dette er noget jeg og mange andre har gjort utallige gange før, og derfor ikke noget der forbindes med en spændende opgave. Disse systemer er ikke der jeg brænder for at udvikle, og derfor begyndte jeg at tænke over om der mon ikke var en nemmere måde.

Resultatet af mine overvejelser blev at lave dette projekt. Det oprindelige ide lød noget i retning af et system, der skulle kunne implementere alt ud fra en konfiguration. Det skulle automatisk kunne generere et system ud fra en struktureret kravspecifikation.

Det endelige forslag til projektet er noget mere beskedent. At kunne implementere **alt** er for omfattende til at kunne klares i et eksamens projekt.

2 Abstract

Projektet omhandler generisk databehandling. Der ønskes en måde hvormed generiske data kan håndteres i et system, der stiller en brugergrænseflade til rådighed. Produktet tænkes at reducere fremstillings / opstillingstiden for nye databehandlingssystemer. Derudover tænkes det at kunne formidle data fra andre databaserede systemer, således at brugeren i sidste ende får et ”single point of entry” til al tilgængelig data.

Systemets funktionalitet kan på mange måder sammenlignes med de funktioner databasesystemet Access stiller til rådighed. I forhold til Access vil der være større fokus på distribution af data imellem flere systemer af samme type. Systemet vil kunne generere en slutbrugergrænseflade (i modsætning til Access' brugergrænseflade, der er rettet mod udvikling), og en programmatisk grænseflade. Ligesom Access vil det være muligt at forbinde til eksterne systemer af samme type, og bruge disse som datagrundlag.

Produktet består altså af 4 dele:

- Definition af data.
- Opbevaring af data.
- Præsentation af data til brugere (herunder søgning etc.)
- Integration til eksterne systemer.

Alle grænseflader baseres på XML.

3 Indledning

Der bruges meget tid på at udvikle software til databehandling. Formålet med projektet er at mindske udviklingstiden på nye softwaresystemer til databehandling, samt at gøre fejlmængden på systemerne mindre.

Projektet går ud på at simplificere udviklingens processen af software. Tiden det tager at udvikle softwaren vil blive reduceret, og softwaren vil indeholde færre fejl.

Jeg har erfaret, at kravspecifikationer ofte ikke er specifikke nok til at udvikle et produkt ud fra. Er kravene tvetydige vil slutbrugeren og udviklerne ofte opfatte kravene forskelligt, og produktet vil ikke blive opfattet som en succes. Ved at simplificere udviklingen kan udvikleren koncentrere sig mere om slutbrugerens krav. Ideelt vil det være så simpelt, at slutbrugeren selv kan implementere sit system og derved få nøjagtigt det han forventer.

Projektet går ud på at udvikle en generisk platform der kan bruges til at være basis for nye databehandlingssystemer. Jeg har ofte erfaret, at der er behov for at forskellige databehandlingssystemer skal udveksle data. Det er derfor nærliggende at inkludere standarder for kommunikation i projektet.

Kort sagt er dette projekt lavet for at undgå i fremtiden at udvikle ”trivielt software”, men have tid til at udvikle mere udfordrende software, der skal løse nye problemer.

4 Systemet

Dette system vil udfylde følgende krav:

- Reducere udviklingstiden for databehandlingssystemer med begrænset forretningslogik.
- Simplificere grænseflader, og derved muliggøre programmatisk kommunikation til og fra slutbrugersystemer.
- Standardisere brugergrænsefladerne således, at brugeren får let ved at bruge nye slutbrugersystemer.
- Muliggøre integration imellem flere systemer.

Der er to forskellige interessenter i målgruppen for dette projekt.

Interessent	**Beskrivelse**	**Problemstilling**
Udvikleren	Udvikleren er den person eller det firma der lever af at udvikle systemer til andre. Dette bliver gjort ud fra kravspecifikationer.	Skal oversætte imellem det datatekniske og slutbrugerens faglige sprog.
Slutbrugeren	Det er slutbrugeren der har brug for nogle databasemæssige egenskaber, som systemet konfigureres til at opfylde	Generel utilfredshed med projekter, da de ikke lever op til forventningerne.

Den del af systemet som udvikleren vil være i kontakt med vil fremover blive benævnt systemet eller produktet.
Den del slutbrugeren kommer i kontakt med benævnes slutbrugersystemet.

4.1 Simplificering

Det er et krav til dette projekt, at udviklingstiden på nye systemer skal reduceres. Dette gøres ved at simplificere udviklingsprocessen.
Udviklingsprocessen kan simplificeres ved at reducere mængden af muligheder i udviklingsprocessen, derved give udviklerne færre ting at tage stilling til. Udvikleren fratages muligheden for at implementere andet end et databasebehandlingssystem.

Da der nu er færre beslutninger at træffe, er der også færre beslutninger, der kan være forkerte. Der er kun de beslutninger tilbage, der vedrører slutbrugerens krav, og der kan derfor fra udviklernes side fokuseres på disse.

Ved at simplificere udviklingen, er det dog klart at det også vil sætte nogle begrænsninger på funktionen af det der udvikles. Da udvikleren er blevet begrænset i sit arbejde allerede før det er gået i gang, er det svært at afvige fra det planlagte spor. Udvikleren kommer til at lave systemer på samlebånd. Han kan godt selv bestemme, hvilke dele der skal indgå, men han kan kun tilføje ting til systemet.

Det kan sammenlignes med skridtet fra maskinkode til mere avancerede programmeringssprog. Jo mere avanceret et programmeringssprog bliver, jo mindre har programmøren kontrol over, hvilke instruktioner der bliver udført på maskinen. Dette system er skridtet efter de avancerede programmeringssprog. Her skal programmøren ikke tænke på bits og bytes og løkker, men kun tænke på hvad slutbrugersystemet til slut skal kunne. Programmøren skal blot definere funktionaliteten. Programmørens opgave bliver at definere de krav som systemet skal implementere. Så når en slutbruger skal have et system implementeret stopper processen når kravspecifikationen er overstået. Hvis slutbrugeren vil lave om i sit system er det bare at ændre i kravspecifikationen. Slutbrugersystemets funktionalitet bliver genereret ud fra kravspecifikationen, og udviklerens traditionelle rolle er blevet automatiseret.

Ulemperne ved at lave udviklingsprocessen så simpel som muligt er at slutbrugeren bliver begrænset i funktionaliteten af slutbrugersystemet. Før man beslutter sig for at bruge produktet, bør man overveje om det nu også er den optimale løsning.

Prisen for udviklingen, standardiseringen og integrationen med andre systemer bør være hovedgrundene for at vælge systemet til at implementere sin funktionalitet.

Performance bør ikke være hvorfor man vælger systemet.

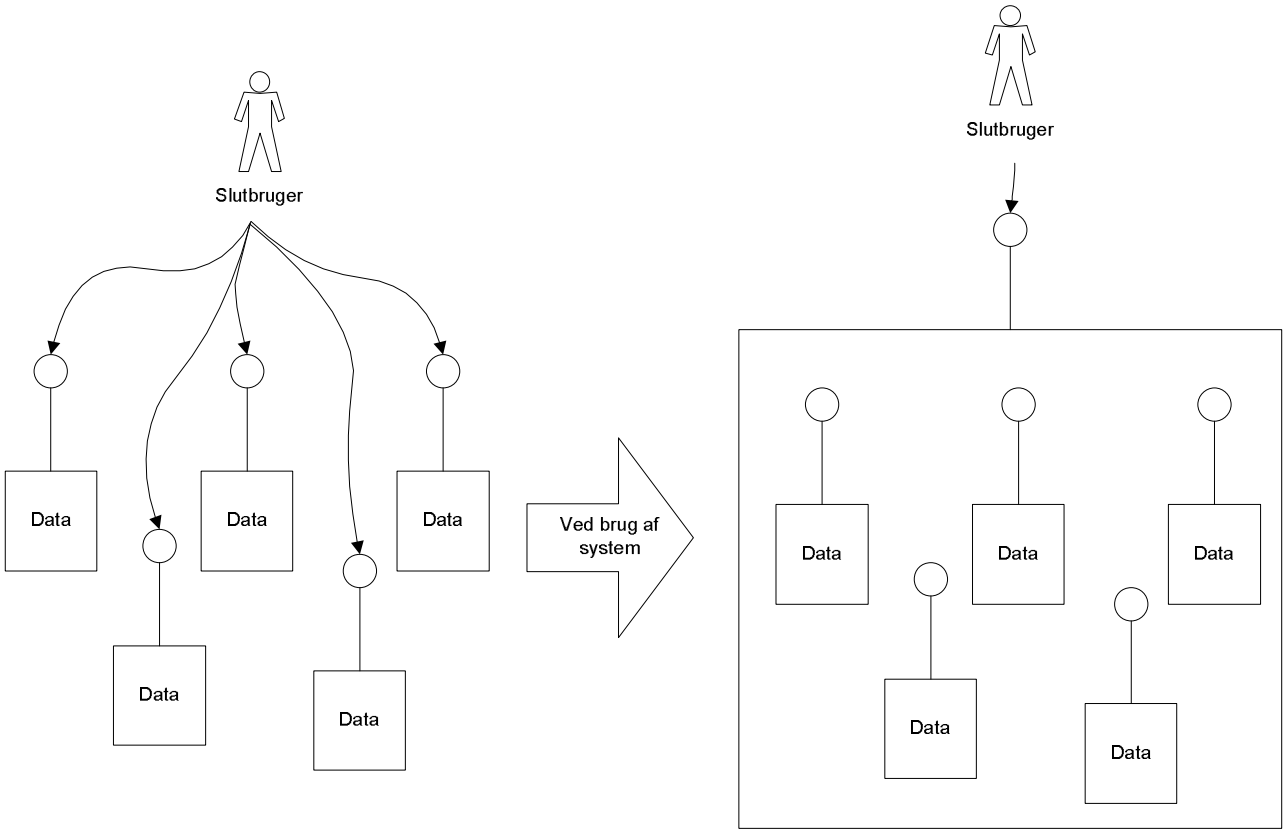
4.2 Integration

Slutbrugersystemerne får simple og standardiserede grænseflader til at tilgå data igennem. Dette gælder for brugergrænsefladen, men også for den programmatiske grænseflade. Da grænsefladerne er standardiserede betyder det, at de **alle** overholder standarder. Det er derfor muligt at forudsige, hvorledes der kan kommunikeres til et slutbrugersystem. Det vil sige, at man programmatisk kan bruge grænsefladerne, selvom man ikke kender ret meget til systemet.

Da vi nu ved hvordan man kan kommunikere til alle slutbrugersystemer, kan slutbrugersystemerne selv bruge denne egenskab. Et slutbrugersystem kan bruge et andet slutbrugersystem til at søge efter data o.lign. På denne måde kan data distribueres, og ejeren af data kan stadig have det lokalt, men lade andre søge i det.

Ved at understøtte en serviceorienteret struktur, kan slutbrugersystemerne sættes til at bruge hinandens ressourcer, uden at de bliver meget tæt bundet på de andres systemers implementation. Det vil i praksis foregå således, at der bliver lavet nogle kontrakter på, hvad data der skal udveksles. Kontrakterne er filer der overfor systemet beskriver hvad der skal udveksles.

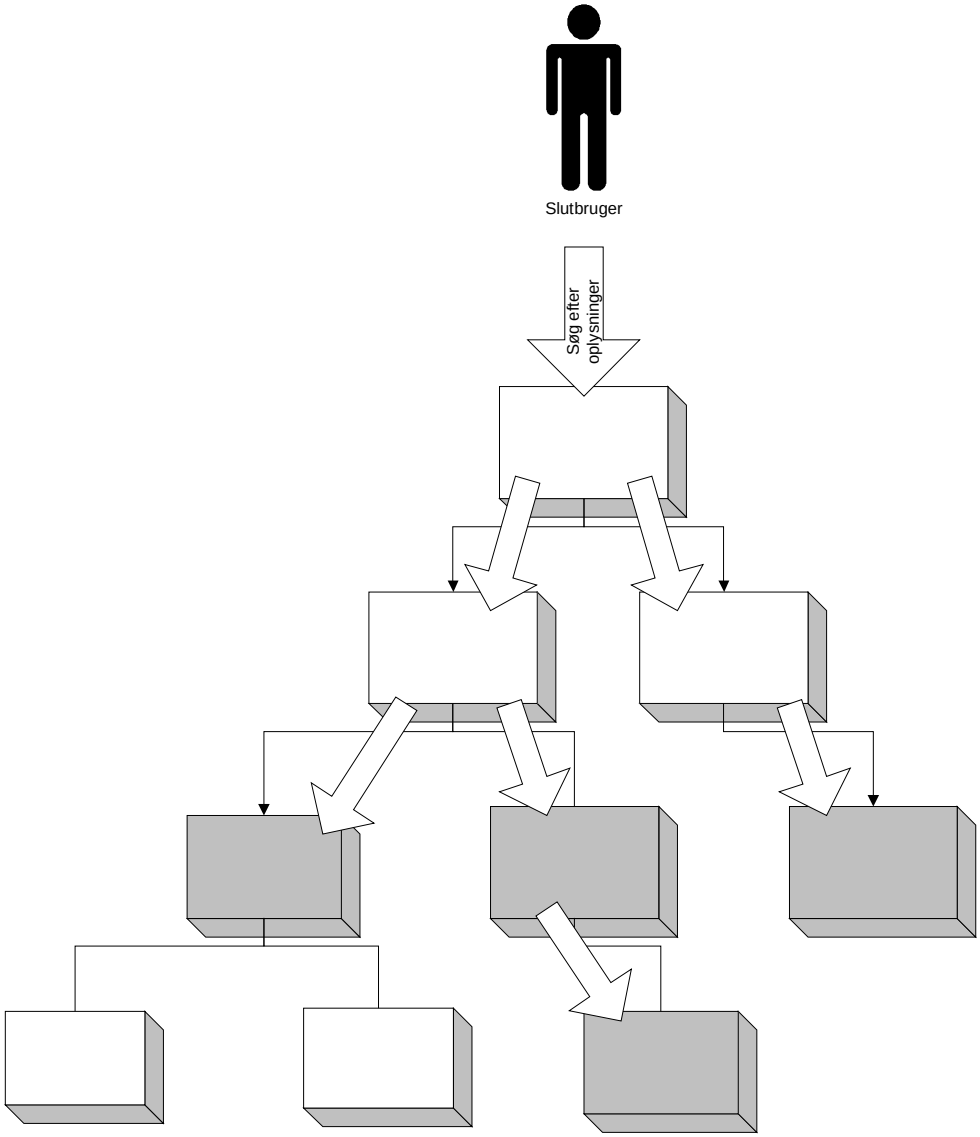
Når man konfigurerer et slutbrugersystem kan man definere, at det også skal hente data fra et andet system, og at det andet system i øvrigt overholder en bestemt kontrakt. Ligeledes kan slutbrugersystemet implementere kontrakter, således at andre systemer kan kommunikere med det.



Figur 1: Datamængden for et system er summen af datamængden for de systemer systemet bruger som datakilde. Der tilføjes et abstraktionslag for brugeren således at data fremgår som del af systemet.

Integration af data fra flere systemer gør slutbrugersystemerne fleksible. Data kan placeres hvor det ejermæssigt hører hjemme og behøver ikke at blive replikeret rundt imellem systemerne[1]. At opbygge Slutbrugersystemerne vha. kontrakter vil hjælpe den enkelte organisation til at kortlægge dataflow.

[1] Se 5.1.8 vedrørende normalisering af data



Figur 2: Viser distribuering af data. Når brugeren søger bliver informationer fundet i forskellige databaser. Grå kasse svarer til systemer der indeholder data som slutbrugeren søger efter.

De enkelte systemer i organisationen kan tilbyde tjenester som en service på netværket. De systemer der er interesserede i tjenesten kan så selv koble sig på og udnytte funktionaliteterne. I en organisation kan der f.eks. sættes et system op, der indeholder telefonbogsoplysninger. Alle andre applikationer der har brug for sådanne informationer kan derefter koble sig på telefonbogstjenesten. Telefonbogen ligger kun ét sted i organisationen og skal derfor kun administreres ét sted. På denne måde kommer data til at ligge ét sted i organisationen, men alle udvalgte afdelinger kan få tilgang til data. Systemet hjælper til at definere de enkelte afdelingers funktion og skaber informationsflow på tværs af afdelingerne.

4.3 Effekt

Hvad er så det endelige resultat af at have et generisk system, der skal kunne det hele?

- Reduceret udviklingstid på fremtidige systemer
- Færre fejl på fremtidige systemer.
- Større overensstemmelse imellem slutbrugerens forventninger til det der bliver leveret og det han rent faktisk får leveret
- Slutbrugeren kan have flere systemer bygget på samme platform og vil derfor kunne opnå større ensartethed.
- Slutbrugeren vil kunne opnå nemmere interaktion imellem systemer.
- Slutbrugeren vil skulle have mere datakraft til at afvikle systemerne på.
- Nemmere opdatering til nye teknologier.

Organisationer opnår flere fordele ved at bruge et sådant produkt. Produktet kan bruges i flere sammenhænge, og organisationen får derfor en mere ensartet måde at håndtere data på.

At slutbrugersystemerne kan bruge hinandens ressourcer gør at større datamængder kan ligge til grund for søgninger i systemet [1], samt at data kun ligger ét sted i organisationen. Dette gør at data bliver nemmere at vedligeholde.

Systemet understøtter distribution af data og automatiseret kommunikation imellem flere systemer. Det betyder at data kan placeres, hvor de ejermæssigt hører til. Andre afdelinger kan gøre brug af disse data ved at forbinde deres egne systemer dertil.

Hvis systemet bruges til at implementere mere end ét slutbrugersystem, vil dets simplicitets begrænsninger gøre at slutbrugersystemerne kommer til at fremstå ens. Dette er med til at slutbrugeren føler sig hjemme i alle slutbrugersystemerne, der er implementeret. Slutbrugeren vil umiddelbart kunne bruge andre systemer i organisationen, forudsat at slutbrugeren kender datastrukturen.

Da slutbrugersystemerne kan kombineres, så data grundlaget kan tilpasses, kan systemet medvirke til, at den enkelte medarbejder kommer i kontakt med færre systemer. Dermed får medarbejderen en mere ren adgang til data, og indlæring af brug af slutbrugersystemet skal kun igennem én gang for hver medarbejder.

[1] se afsnit 5.2 Databehandling

Software udvikleren får færre produkter at vedligeholde. Slutbrugerne vil til en vis grænse rette deres arbejdsgange ind efter hvad systemet umiddelbart kan håndtere, og det samlede antal krav som udvikleren skal implementere vil derfor være reduceret.

Ved at definere hvad der kommer ind i et slutbrugersystem, og hvad der kommer ud, kan rollen for slutbrugersystemet beskrives. Hvis et slutbrugersystem bliver mappet direkte til en afdeling i organisationen, kan hele organisationens funktion kortlægges, og organisationen kan ende med et diagram der viser informationsstrømme i organisationen.

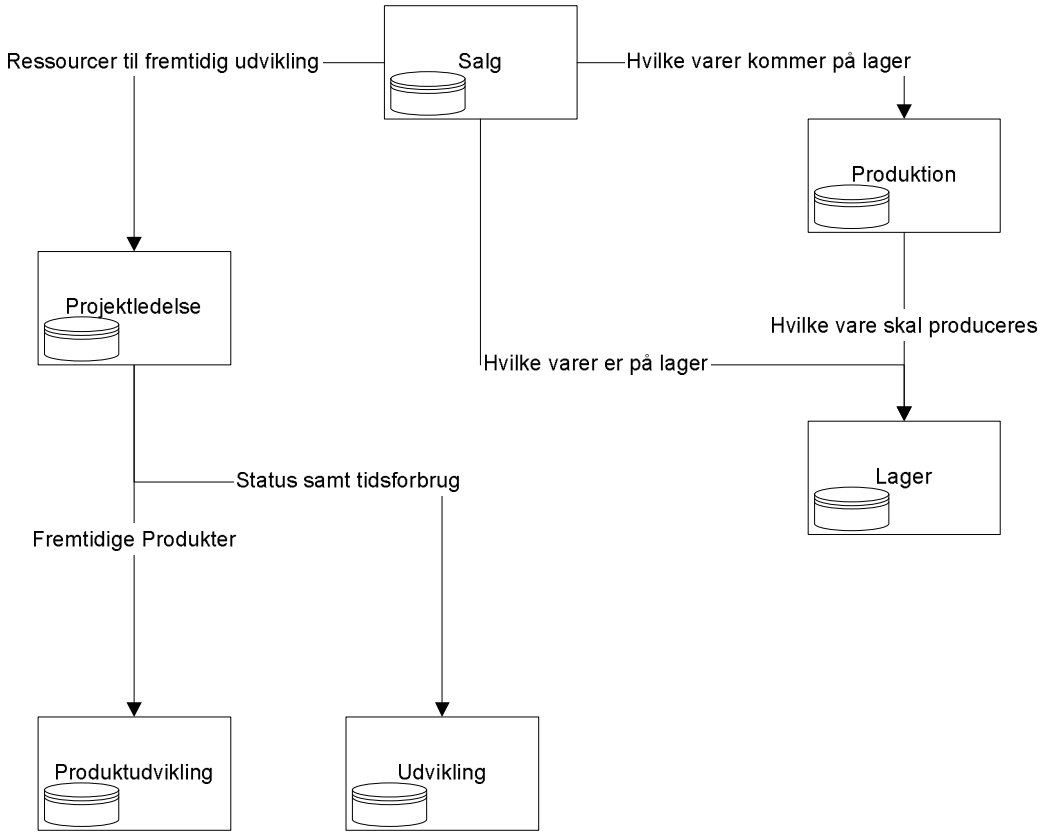
- Organisationen:
  Opnår ensartethed i software systemerne, samt en måde at analysere og regulere dataflowet i organisationen. Derudover opnås en normaliseret og ensartet distribution af data i organisationen.

- Udvikleren:
  Kommer i kontakt med færre systemer og kan koncentrere sig om gøre disse bedre. Når et system skal konfigureres til et bestemt formål, er udviklerens rolle begrænset til at definere datamodellen samt definere slutbrugerens arbejdsgang.

- Administratoren
  Får kun én grænseflade til administration. Brugere og brugergrupper vil i fremtiden kunne deles imellem flere systemer.

- Slutbrugerne
  Får kun én grænseflade til data. Potentielt kan alle informationer den pågældende slutbruger skal bruge i sin hverdag integreres i en enkelt brugergrænseflade.

Hvis der er behov for informationer fra anden kilde end af det system der her bliver udviklet, kan de integreres ved at overholde xml grænsefladen for kommunikation til eksterne systemer, kan der kommunikeres på tværs af implementationer og platforme.



Figur 3: Eksempel på udveksling af data i en virksomhed. Her er det salgsafdelingen der kan forespørge andre afdelinger om deres status. Læg mærke til at systemerne kan kobles sammen i flere niveauer.

Sammenkoblingen af systemerne tænkes ske på baggrund af kontrakter afdelingerne imellem. I eksemplet i Figur 3 kan kontrakten imellem salgsafdelingen og produktionsafdelingen være at produktionsafdelingen skal stille oplysninger til rådighed om hvilke produkter der er planlagt at blive produceret. Derudover skal der udveksles information om hvilken kapacitet afdelingen har for at starte ny produktion op. Til sidst skal salgsafdelingen kunne booke produktion af en vare.

Da information kan tage flere veje igennem organisationen er det vigtigt at der i det enkelte system kun offentliggør eget data, eller forarbejdet data fra ekstern kilde. Et eksempel i systemet kan være at produktionsafdelingen videresender information om lager statusen. I salgs afdelingen vil denne information nu komme fra to kilder. Fra lageret vil informationen at der er 2 af en pågældende vare på lager. Salgsafdelingen vil få det samme at vide fra produktionsafdelingen, og kan meget vel være af den opfattelse, at der er 2 enheder på lageret og 2 enheder i produktionsafdelingen. Det er meget nemt at overskue i det aktuelle eksempel, men hvis virksomheden indeholder flere lagre, og produktionsafdelingen i sig selv indeholder en lagerbeholdning bliver det svært at overskue hvor mange enheder der i virkeligheden er på lager. Derfor er det ikke en god idé at have et system, der videresender data ubehandlet fra en ekstern datakilde til en klient.

Når systemer kobles sammen på kryds og tværs er der flere hensyn at tage. Brugerrettigheder vil ofte muliggøre at mere nuanceret information kan videregives til de brugere der har rettigheder til at se informationerne. Derudover er det vigtigt at kortlægge hvilke systemer der bruger hvad data. Der vil kunne opstå situationer hvor informationer distribueres ud til flere systemer. Hvis der så er et andet system som bruger disse systemer, kan samme oplysning komme fra flere kilder. Især når det gælder tal værdier, der senere skal bearbejdes vil dette kunne give problemer. Summeringer af tal vil give forkerte værdier, fordi de enkelte tal fremkommer flere gange.

Når man definerer en grænseflade til et eksternt system, vil det være en god ide at sørge for at det eksterne system kun videresender behandlet data. Systemet der forespørger data må så i stedet have en direkte forbindelse til den originale datakilde.

5 Begreber

Projektet forsøger at løse en række datatekniske problemstillinger. For at gennemskue et problem kræver det indsigt og der for vil alle begreberne der ligger til grund for dette projekt først blive kortlagt.

Dette afsnit er med til at gøre det muligt for personer uden teknisk kendskab at læse og forstå hoved idéen med dette projekt.

5.1 Databaser

Systemet kan på mange måder sammenlignes med en database. En database er et program der kan bruges til at gemme data struktureret på forskellige måder. Databasers fleksibilitet har gjort dem meget udbredte.

Der er overordnet 3 forskellige typer database : relationel, xml og objekt databaser.

XML databaser og objektorienterede databaser har en grænseflader der er specialiseret imod databasens brug. Der er lagt vægt på at måden at tilgå data på. XML databasen er velegnet til at gemme XML i og objekt databasen er velegnet til at gemme objekter i[1].

Ofte er de to sidste typer databaser bygget oven på en relationel database.

Relationelle databasers data tilgås via tabeller.

Den relationelle databases force er dens hurtighed og standardiserede grænseflader. Når man definerer en datastruktur i en relationel database, skal man beskrive hvordan databasen fysisk skal gemme strukturen. Dette er med til at øge effektiviteten, hvis det gøres rigtigt, men gør samtidigt tilgang til data mere besværlig. Det er ikke altid at data findes på tabel form i de applikationer der skal bruge databasen.
Det betyder at der skal programmeres noget konvertering for at de kan snakke sammen.

Fordelen ved en relationel database er at der er fokus på hvordan data lagres fysisk, og programmøren kan derfor selv være med til at optimere datastrukturen. En anden fordel er at der er standarder for hvorledes man kommunikere til en relationel database. Det vil sige at hvis man beslutter sig for at bruge en relationel database, bliver man ikke fastlåst til en producent.

Der er en række begreber tilknyttet relationelle databaser.

5.1.1 SQL

Structured Query Language er en standard for hvorledes man forespørger information i en relationel database. SQL er en tekst streng hvor man beskriver hvilken handling man vil foretage på databasen, det kunne f.eks. være hvad data man gerne vil have ud og hvordan det skal filtreres.

I SQL er det muligt at beskrive hvordan data skal håndteres og kombineres for at give det ønskede resultat. Resultatet fra en søgning foretaget med SQL kan altså godt vise en datastruktur der er anderledes en den databasen indeholder. Hvordan SQL'en formuleres er

[1] Se kap. 5.5 OO desig

meget vigtigt både for resultatet og for performance på databasen. Dårlig SQL på en god datastruktur vil resultere i et system der performer dårligt.

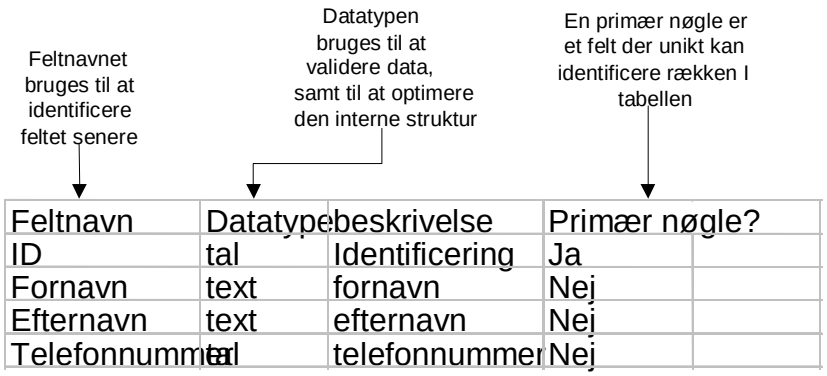
De fleste relationelle databaser understøtter SQL. Flere database produkter udvider SQL standarden for at kunne tilbyde deres kunder et bedre produkt. Hvis SQL skal bruges på tværs af flere database produkter, er det en god idé at overholde standarden.

SQL kan bruges til at hente data fra databasen, men kan også bruges til at opdatere og indsætte data i databasen.

5.1.2 Tabeller

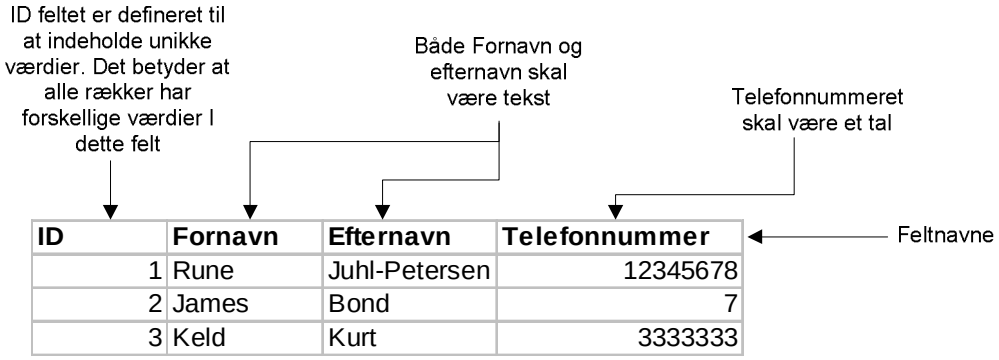
I en relationel database er data placeret i tabeller. I en tabel er data opdelt i rækker og kolonner. Hver gang der sættes data ind i en tabel oprettes en række i tabellen. Hver række indeholder nogle felter defineret af tabellens kolonner.

Når en database oprettes bliver det defineret hvilke tabeller der skal være samt hvilke koloner hver tabel skal have. De enkelte koloner kan gives forskellige egenskaber som f.eks. angiver hvilken datatype kolonnen kan indeholde. På denne måde kan data indeholdt i en tabel valideres til at overholde nogle krav.



Figur 4: Eksempel på definition af en tabel i en database.

I Figur 4 ses et eksempel på en definition af en tabel. Tabellen indeholder 4 kolonner. ID er et tal og defineret som primærnøgle. Hver række indsat i tabellen vil indeholde 4 felter.



ID	Fornavn	Efternavn	Telefonnummer
1	Rune	Juhl-Petersen	12345678
2	James	Bond	7
3	Keld	Kurt	3333333

Figur 5: Eksempel på data i en tabel i en database

5.1.3 Alias

Et alias i en database bruges i forbindelse med udtræk af data. Et alias er når et felt eller en tabel i en database gives et anden navn under behandling.
Der kan være flere grunde til at man vil bruge aliaser i sine udtræk. Enten kan det være fordi at felter i forskellige tabeller hedder det samme om man godt vil undgå at skrive det fulde navn på feltet, eller også kan det være fordi man skal have trukket samme felt ud i forskellige sammenhænge.

Eks.
En rute er defineret af et startpunkt og et slutpunkt. Der er to tabeller en tabel til ruten og en tabel til punkterne. Både startpunktet og slutpunktet vil ligge i samme tabel, men vil i udtrækket have meget forskellig betydning.

5.1.4 Primær nøgle

Når et felt er defineret som værende en primær nøgle, vil det sige at der kan refereres til rækken ved hjælp af feltværdien. Værdien vil med andre ord ikke forekomme i andre rækker i tabellen. En primær nøgle kan antage en sigende værdi som f.eks. kombinationen af fornavn og efternavn i telefonlisten. Hvis data ikke opfylder kravet om at være unikt for alle rækker, kan der i stedet programmatisk generes en nøgle.

En primærnøgle kan godt defineres til at være flere felter samtidigt. Så vil felternes kombinerede værdi udgøre en unik nøgle.

5.1.5 Relation

En relation beskriver hvorledes tabeller i en database er relateret til hinanden. En relation fremkommer når et felt i en tabel antager værdien af et felt fra en eller flere rækker i en anden tabel.

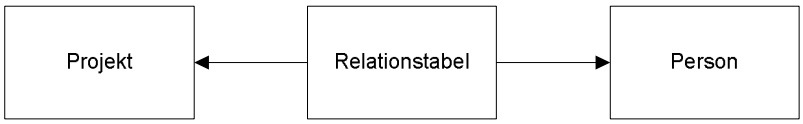
Der findes to forskellige slags relationer. Èn til mange og mange til mange. Èn til mange er den simpleste relation. Når et felt antager værdien fra et unikt felt i en anden tabel er det en én til mange relation. Flere rækker kan pege på samme feltværdi.

Mange til mange relationer laves ved at sætte 2 én til mange relationer sammen. For at gøre dette laves der en relationstabel der har to kolonner der hver i sær kan pege på rækker i forskellige tabeller.

Figur 6: Én til mange relation. Et felt i person tabellen kan indeholde primærnøglen for et projekt. En person kan deltage i ét projekt, mens et projekt kan indeholde mange personer.

Det er meget normalt at bruge primærnøgler når tabeller refereres til hinanden. Dette bliver gjort fordi primærnøgler har den egenskab at de er unikke for hver tabel.

Når en kolonne i en tabel kan indeholde en reference til en række i en anden tabel, kaldes feltet for en fremmednøgle.



Figur 7: Mange til mange relation. Relationstabellen indeholder både en fremmednøgle til et projekt og en person. En person kan deltage i mange projekter, og der kan være mange personer på hvert projekt.

5.1.6 Join

En *join* er når en SQL forespørgsel bruger en relation til at kombinere tabeller i databasen. Flere join kan kombineres og resultere i komplekse udtræk. Der er to forskellige slags joins; inner joins og outer joins. I en inner join skal begge tabeller være med i udtrækket, i en outer join behøver kun den ene at være med.

Nogle databaser har en decideret syntaks til at beskrive joins, mens andre bruger at sætte de to felter lig hinanden i filteret i SQLen.

5.1.7 View

Et View bruges til at simplificere udtræk fra databasen samt til at optimere hastigheden på udtræk. Et View er ofte en predefineret SQL forespørgsel.

Et view er en abstraktion oven på den reelle datastruktur. Et View kan bruges til at skjule den reelle datastrukturs kompleksitet fra den applikation der skal bruge den. Kompliceret logik for udtræk fra databasen kan derfor fjernes fra applikationen og implementeres i databasen.

Nogle bryder sig ikke om at bruge Views når de designer applikationer. Grunden til dette er at et View indeholder information om brug af data, og dette efter deres mening ikke skal ligge i databasen.

Da Views ligner tabeller så meget, er det også muligt at hente data fra andre Views. Det giver genbrug af kode og simplificerer database adgangen ved komplekse forespørgsler.

Views bliver gemt på databasen, og nogle databaser kan optimere disse så der opnås en bedre performance i forhold til at kalde databasen med tilsvarende SQL.

5.1.8 Normalisering af data

Normalisering af data går ud på at optimere datastrukturer.

Ved at normalisere data undgås redundant data, og vedligeholdelse bliver derfor nemmere. Datamodeller kan kategoriseres i forskellige normaliseringsgrader. 3. normaliseringsform anses oftest som den mest optimale.
I 3. normaliseringsform bliver alle informationer kun gemt én gang i systemet. Normaliseret data er struktureret efter hvordan det hænger sammen med andet data, og ikke efter hvordan data skal bruges. På denne måde bliver det nemmere at genbruge data på tværs af funktionalitet.

Eks.
En normaliseret adresse vil bl.a. bestå af en fremmednøgle til en vej. En denormaliseret adresse vil i sig selv indeholde vejnavnet.
Hvis der er ændringer til vejnavnet, skal det i en normaliseret datastruktur kun ske ét sted. Alle adresser der refererer til den vej vil automatisk blive 'opdateret'. Hvis ikke strukturen var normaliseret, ville alle adresserne skulle opdateres.
En anden fordel er at adresser kun kan ligge på gyldige veje. Det er kun muligt at lave en reference til en vej der rent faktisk eksisterer. Hvis vejnavnet lå som en tekststreng på adressen, vil det være meget svært at validere om adressen nu også er gyldig.

Fordelen ved at have normaliseret data er at ændringer kun skal foretages ét sted, samt at det kan valideres at det er gyldigt data.

5.1.9 Indeksering

Indeksering er en slags denormalisering af data. At indeksere går ud på at analysere felterne i en kolonne for hvilke informationer felterne indeholder, for på den måde at undgå at skulle læse alle felterne igennem ved en evt. søgning. Hvor normalisering går ud på at optimere den måde data lagres i forhold til hvad det repræsenterer, så går indeksering ud på at optimere søgninger, uden at ødelægge normaliseringen.

For at oprette et effektivt indeks, skal man vide hvilke slags søgninger der bliver foretaget på databasen. Ofte er primærnøglerne indekseret.

Når der bliver lavet et indeks på noget data, laver databasen internt en tabel med metadata om indholdet i den rigtige tabel. Når noget data er indekseret vil det sige at informationerne findes flere steder i databasen. Det betyder at der er flere steder der skal opdateres hver gang et felt i databasen bliver opdateret. Det vil sige at det tager længere tid at sætte data i en indekseret tabel. Til gengæld er det meget hurtigere at finde data igen.

Der findes forskellige indekstyper alt efter hvad data der skal indekseres og hvordan indekset skal bruges. Et meget brugt indeks er fulltekst indekseringen. Et fulltekst indeks bruges til at indeksere tekster med. Alle ord i teksten bliver gemt i indekset, og søgninger efter ord i databasen bliver derfor meget hurtige.
Mere komplekse indekse hvor det er det forståelsesmæssige indhold bliver indekserer kan også forekomme, men er ikke så udbredte.

Eks. Store datamængder
En database indeholder en tabel med mere end 1.000.000 rækker. En af kolonnerne kan indeholde heltal.
Alle rækker hvor feltværdien er 8 skal udtrækkes.
Hvis kolonnen ikke er indekseret vil en søgning resultere i at alle rækker i tabellen skal søges igennem. Med 1.000.000 rækker vil dette tage tid.
Hvis kolonnen er indekseret vil der i indekset være en liste over hvilke rækker der har værdien 8. Alle rækker i tabellen skal altså ikke søges igennem.

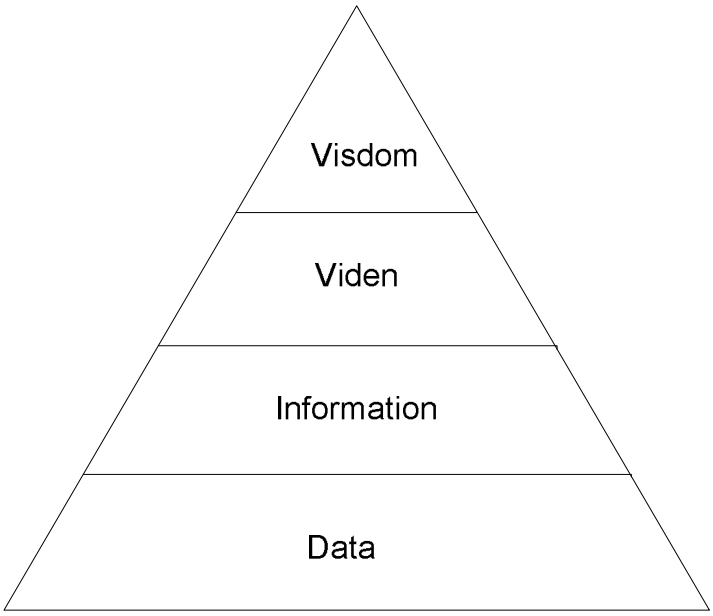
Eks. Kompleks struktur
En database indeholder en kategoristruktur.
Kategoristrukturen er lavet ved at kategori tabellen har en fremmednøgle der peger på en primærnøgle i samme tabel. En moder/barn struktur. Der ønskes en funktionalitet der for 2 kategorier kan give svar på om de er i er i 'familie'. Hvis strukturen ikke er indekseret, vil dette spørgsmål kunne resultere i flere rekursive kald til databasen. Hvis strukturen derimod er indekseret vil oplysningen kunne slå op umiddelbart.

5.2 Databehandling

I afsnittet om databaser er det beskrevet hvorledes data kan opbevares, så det senere kan hentes frem og bruges. Det er dog sjældent data der i sig selv er hovedgrundlaget for at få udviklet et nyt system. Når der er gemt en tekst i en database er det sjældent vigtigt hvilke tegn der er gemt i database, eller hvordan. Oftest er det de oplysninger som teksten beskriver. Den mindste enhed af data er sjældent interessant, men kombineret med andet data kan informationsværdien øges.

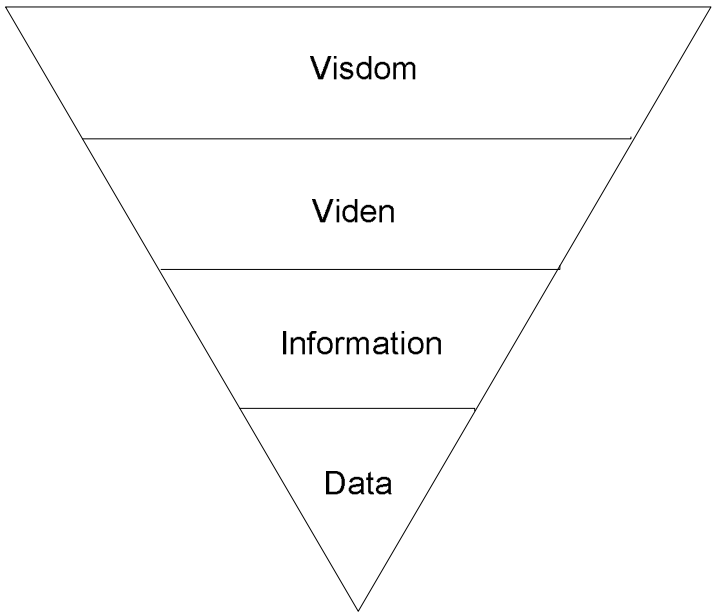
Efterhånden som information bliver behandlet, bliver der mindre og mindre af den. Til gengæld bliver information mere dækkende.

I bogen ”Hitchiker’s guide to the Galaxy” bygges der en computer der skal finde et svar til det ultimative spørgsmål: *Hvad er meningen med livet og det hele* (frit oversat). Til at beregne dette skabes en ny verden som skal være datagrundlaget for beregningen. Behandlingen af al den information der er tilgængelig tager lang tid, men systemet returnerer et heltal der skulle besvare hvad meningen med livet er. Bogen er science fiction og ikke realistisk, men princippet er det samme som vi stræber efter, og da overdrivelse fremmer forståelsen er det et godt eksempel til at anskueliggøre hvad det er vi prøver at opnå. For at opnå et kort og præcist svar, kræves et stort datagrundlag.



Figur 8: Informationspyramiden: Kvantitet. Når data bliver behandlet og forfinet bliver der mindre af det.

For at opnå gode, forfinede oplysninger, er det vigtigt at datagrundlaget er så stort som muligt.



Figur 9: Informationspyramiden: Kvalitet. Når data bliver behandlet og forfinet bliver dets værdi større.

Figur 9 og 10 viser hvad der sker når data bliver behandlet. Figur 9 viser hvad der sker med mængden af informationer. Jo mere data bliver behandlet, jo mindre bliver der af det. Figur 9 viser værdien af data efterhånden som det bliver behandlet. Ubehandlet data er ikke meget værd, mens behandlet data kan være meget værd.

Eks.
I en virksomhed er direktøren, meget firkantet, kun interesseret i ét spørgsmål: Går det godt eller skidt for virksomheden? Svaret er enten godt, eller skidt. Mængden af information direktøren er interesseret i er begrænset, men værdien af informationen er kolossal. Informationen er udledt af alle parametre som virksomheden består af. I princippet ligger alle parametre i virksomheden til grund for svaret, da parametrene i modsat fald ikke har betydning for firmaet, og derfor burde fjernes. F.eks. er kvaliteten af maden i kantinen en parameter. Den påvirker ikke direkte svaret som direktøren stiller, men det påvirker medarbejdernes tilfredshed, som igen påvirker deres produktivitet, som igen påvirker hvor meget virksomheden kan producere, som påvirker økonomien.

5.3 Serviceorienteret arkitektur

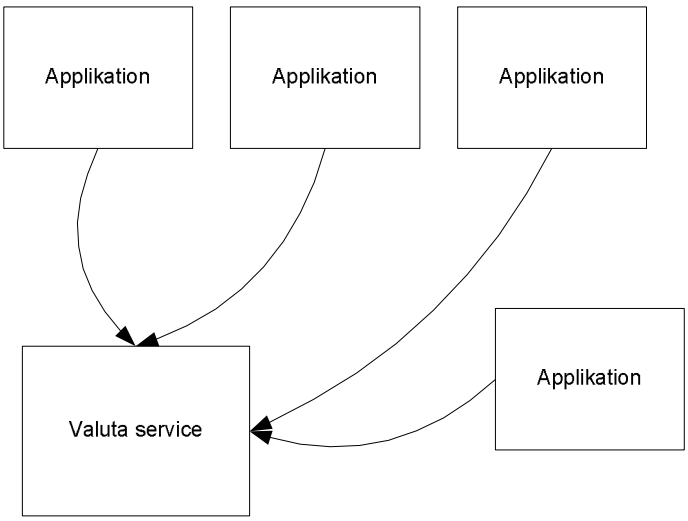
En måde at lave et distribueret system er at programmere det op fra bunden. Designe og programmere alle de forbindelser der skal være imellem delsystemerne. Dette giver en meget låst struktur da alle delsystemerne er afhængige af hinanden.

Serviceorienteret arkitektur er et design pattern[1] der bruges til at designe kommunikation imellem forskellige systemer. Ideen er at det enkelte system giver sig til kender over for de andre systemer på netværket, og beskriver hvilken service det stiller til rådighed.
Der kan således laves lister over hvilke systemer der tilbyder hvilke services. Når fremtidige applikationer skal designes er det let at genbruge funktionalitet, allerede tilgængelig.

Fordelen ved at bruge en service orienteret arkitektur er at netværkskoblingerne i systemet bliver løsere og at funktioner i organisationen kan centraliseres i nogle enkelte installationer, og vedligeholdelsen derfor bliver simplere. Funktionalitet der bliver brugt mange steder kan placeres ét sted i organisationen og skal derefter kun vedligeholdes der. Derudover bliver grænsefladerne imellem systemerne standardiserede og simplificerede.

Eks.

Flere applikationer skal kunne konvertere imellem forskellige valutaer.



Figur 10: Eksempel på brug af service orienteret arkitektur

Valutaservice er en service der kender kurserne. Alle de andre applikationer kan benytte servicen til at konvertere beløb, eller til at få kurserne at vide.

[1] Et godt og anerkendt råd fra nogen der har prøvet det før.

Fordelen ved denne konstruktion er at kurserne kun skal opdateres i én applikation. Derudover simplificeres de andre applikationer i og med at valutakonvertering ikke skal implementeres i disse.

Applikationen der kan konvertere valutaer kender ikke til de andre applikationer.

Der er defineret standarder for hvorledes der kan kommunikeres med services. Den mest brugte standard er SOAP[1]. SOAP er oprindeligt udviklet af Microsoft og bruger XML[2] som kommunikationsmedie.

[1] Simple Object Access Protocol
[2] Se afsnit 5.6 XML

5.4 CMS – Content Management System

Content management betyder noget i retning af indholdsstyring. Et CMS er et værktøj til at styre informationer struktureret på en bestemt måde.

CMS løsninger er blevet meget udbredte. CMS bruges af virksomheder til at håndtere informationer. Ofte vil disse informationer antage en kategoriseret struktur.
Ved hjælp af mappe strukturer kan informationer sættes i relation til hinanden. I nogle CMS løsninger kan den enkelte bruger selv oprette dokumenttyper.
CMS løsninger bliver ofte brugt som intranet eller extranet.

Fordele:

- ingen udviklings tid
- ofte mange ekstra features

Ulemper:

- Begrænset mulighed for at modellere data.
- Kan være svært at tilpasse, hvis der er specielle krav.

5.5 OO design

Designet af systemet laves som objektorienteret. Objektorienteret programmering tilbyder et abstraktionslag til programmøren. I stedet for at betragte et program som en serie af kommandoer, kan et program betragtes som en række objekter der kan kommunikere med hinanden. Det giver mere genbrug af kode og gør vedligeholdelse af kode nemmere.

Begreberne brugt kommer fra Java og er som følger:

- Objekt: Samling af data og funktionalitet.
- Klasse: Definering af objekter.
- Abstrakt klasse: Klasse der ikke kan oprettes sig selv. En abstrakt klasse skal nedarves af en anden klasse før der kan laves et objekt med klassens funktionalitet.
- Interface: Beskrivelse af klassers metoder. Alle klasser der implementerer et interface skal implementere de metoder der er beskrevet i interfacet.
- Pakke: Samling af klasser og objekter som logisk hænger sammen.

5.6 XML

XML står for Extensible Markup Langauge. Det er et format til at opbevare data i. XML er et tekstformat og kan indeholde brugerdefinerede datastrukturer. Da XML er tekstbaseret kan det overføres over serielle forbindelser, og bruges ofte som databærer imellem systemer på forskellige platforme.

XML udvikler sig konstant og kan bruges til mange ting.

XML har den egenskab at det er selvbeskrivende. Hvis en node i et XML dokument skal indeholde værdien af et navn, kan noden gives navnet 'navn'. På den måde kan indholdet i et XML dokument oftest læses af mennesker.

5.6.1 XSD

XSD står for *XML Schema Definition*. Et XSD bruges til at beskrive en XML struktur, og kan senere bruges til programmatisk at validere selve den XML fil der skal indeholde datastrukturen.

XSD giver mange muligheder for at validere XML. Tekststrenge kan valideres vha. regulære udtryk og referentiel integritet kan valideres vha. Indentity Constraints.

5.6.2 XPath

XPath er et søgningssprog til XML. I XPath er det muligt at søge ned igennem XML strukturen og returnere de XML noder der opfylder kriteriet.

5.6.3 XSLT

XSLT står for *XML StyleSheet Language (Transformation)*. Det er et XML baseret scripting sprog der kan transformere et XML dokument til et andet tekstformat.

5.7 HTML

HTML står for *Hyper Text Markup Language*. HTML er et tekst format der bruges til at overføre brugergrænseflader på en seriel forbindelse. HTML kan minde om XML, men i modsætning til XML er HTML ikke dataorienteret men præsentations orienteret.

HTML er udbredt på internettet, og der er meget få maskiner der ikke kan læse HTML i en internet browser.

5.8 .Net

.Net er den nyeste udviklingsplatform fra Microsoft. .Net bygger videre på en del andre platforme og værktøjer.

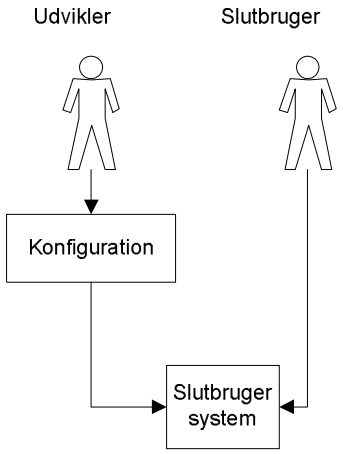
De vigtigste ting om .Net:

- Programmer bliver eksekveret af en virtuel maskine der håndterer ressourcer. Det giver robuste programmer. I fremtiden kan det gøre .Net programmer platformsuafhængige.
- .Net indeholder er række standard biblioteker som gør at programmøren ikke skal opfinde den dybe tallerken for hver projekt.
- Er objektorienteret.
- Integrere godt med flere teknologier. F.eks. er der indbygget support for XML, og .Net indeholder et godt framework til at tilgå databaser igennem.
- .Net er stadig en ny teknologi og udvikler sig konstant. .Net har på nuværende tidspunkt flere børnesygdomme.
- Der er fokuseret på brugervenlighed i .Net. Det betyder at der er lavet mange små detaljer både i udviklingsværktøjerne og i selve platformen. Dette kan være med til at få nybegyndere med, men er ikke nødvendigvis en fordel i et professionelt arbejdsmiljø.

5.9 Projektorienterede begreber

De foregående afsnit har afklaret datatekniske begreber. Dette afsnit omhandler de begreber der er opfundet til projektet og som beskriver delene i projektet. Der er flere roller i projektet, og systemet udviklet kan antage flere forskellige statuser.

Begreb	Beskrivelse
System	Fællesbetegnelse for en selvstændig software enhed. Vil ofte have kommunikationsgrænseflader
Ukonfigureret system	Produktet udviklet i løbet af denne rapport, før det konfigures til at indeholde data.
Slutbrugersystem	System konfigureret med datastruktur og grænseflader, der umiddelbart kan bruges af en slutbruger
Server system	Er et system der stiller services til rådighed overfor andre.
Klient system	Et system der kobler sig på server systemer og udnytter disses ressourcer.
Administrator	Skal vedligeholde systemet.
Udvikler	Person ansat til at udvikle database systemer. Vil med dette produkt være den person der laver en konfigurations fil og derved konverterer systemet fra at være et ukonfigureret system til at være et slutbrugersystem.
Definition af administrationsgrænseflade	XML schema der definerer hvorledes en datastruktur og brugergrænseflade kan defineres vha XML.
Definition af datastruktur	Det dokument der ligger til grund for hvilken datastruktur systemet implementerer.
Slutbruger	En bruger der skal bruge systemet i sidste ende. Slutbrugeren er brugeren af den datastruktur, systemet implementerer.
Skærmbillede	Præsentation af data til brugeren



Figur 11: Overblik over begreber i projektet.

6 Tilsvarende produkter

6.1.1 Struts.

http://jakarta.apache.org/struts/userGuide/index.html

Struts er et framework til at adskille navigering fra resten af brugergrænsefladen. Struts er helt sikkert et framework der er kommet for at blive. Det gør det muligt at programmere sine brugergrænseflader, og senere hæfte dem sammen i en logisk række følge. Struts er henvendt til webprogrammering.

Fordele:

- Adskiller brugergrænseflade fra navigeringslogik
- Stor fleksibilitet

Ulemper:

- Kræver programmering af brugergrænseflader samt logik til databehandling

6.1.2 Oracle Portal

http://www.orafaq.com/faqwebdb.htm#PORTAL
Oracles portal er et add-on til deres applikationsserver. En portalløsning fungerer ved at der er en central portal der har referencer til en masse decentrale løsninger. Applikationen kan på den måde blive decentraliseret.

Fordele:

- Stor integration til Oracle (formegentlig nemt for Oracle superbrugere at konfigurere)

Ulempe:

- Meget tæt tilknytning til oracle platformen.
- Mulige integrationsproblemer med andre platforme.

6.1.3 WPS

http://www.wpsnet.dk
Web Publishing System leverer et produkt der muliggør deling af dokumenter. Det er et standard CMS system det tilbyder brugerne af gemme dokumenter i en kategoristruktur. Det er muligt at lave flere forskellige dokumenttyper der kan gemmes i systemet.
Det er desuden muligt at bruge systemet til at levere indhold til en hjemmeside.

Fordele:

- Hurtig implementering

Ulemper:

- Begænsede muligheder i definering af datastruktur.

6.1.4 Intrasuite 4.0

Intrasuite er et CMS system udviklet af Dansk Internet Selskab AS.
Intrasuiten er et dokumenthåndteringssystem hvor det er muligt at definere egne dokumenttyper. Dokumenterne gemmes i en kategori struktur. Intrasuite har et kompliceret rettighedssystem der gør det muligt at give rettigheder på hvert enkelt dokument.
Dokumenter kan importeres og eksporteres vha. XML.

Fordele:

- Intrasuiten udmærker sig ved konstant at udvikle sig. Mange nye ønsker fra kunderne bliver tilføjet til standard applikationen.
- Mange rettighedsmuligheder.
- Mulighed for at tilføje flere applikationer oven på samme datagundlag , f.eks. kan der laves en hjemmeside der henter informationer fra et intranet.
- Mulighed for selv at lave dokumenttyper (kræver tilkøbsprodukt)

Ulemper:

- Kræver licens
- Er låst fast i kategoristrukturen. Det er meget svært og kræver tilretninger for at kunne relatere dokumenter ud over kategorierne.
- Udokumenterede grænseflader. Dette er ikke en ulempe så længe det bruges alene, men så snart andre applikationer skal kommunikere med det bliver det et problem.

Alt i alt er Intrasuiten et godt produkt. Dets funktion er dog begrænset til at have data placeret i en kategoristruktur og ikke have behov for features ud over dem der er understøttet ved levering.

7 Design

Systemets formål er at kunne implementere udviklerens konfiguration.

Der er 2 områder for design i dette projekt. Den første er hvordan grænsefladerne skal være udformet. Hvordan bliver det muligt at konfigurere slutbrugersystemet, og hvordan skal slutbrugersystemet så se ud? Anden del er selve programmeringsdelen. Første del er en kravspecifikation på hvilke anden del skal overholde.

Følgende ting skal der tages stilling til under designet.

- Konfiguration
- Opbevaring af data
- Brugergrænseflade
- Kommunikation imellem brugergrænsefladen og data.
- Navigering.
- Kommunikation imellem slutbrugersystemer.

Det er ikke muligt at forudsige alt på forhånd, og det er derfor ønskeligt at produktet kan udvides senere hen.

7.1 Valg af teknologier

Produktet ønskes udviklet ved brug af standarder. Derudover skal projektet simplificere den fremtidige udviklingsproces af slutbrugersystemer.

Simple grænseflader opnås bla. ved at bruge XML som det primære kommunkationsformat.

Internt tænkes data lagres i en relationel database. Slutbrugersystemerne tænkes at skulle håndtere struktureret data og en relationel database leverer både kræfterne og standarderne til at gøre dette.

Brugergrænsefladen opbygges ved hjælp af XSLT, som kan bruges til at formatere HTML ud fra XML.

Der skal kommunikeres data over netværk. Til dette bruges webservices. Webservices er en simpel måde at stille ressourcer til rådighed på tværs af et netværk.

- **XML -** Der bruges XML version 1.0
  Bruges til grænsefladerne i systemet. XML'ens struktur kan valideres, og XML kan derfor bruges til såvel dataoverførsler som til konfiguration og beskrivelser i systemet

- **WebServices** - Der bruges protokollerne SOAP og WSDL.

- **HTML -** Optimeret til Internet Explorer 6.0
  Dette bruges til at vise brugergrænseflader til slutbrugeren. HTML er valgt fordi det er tekstbaseret og derfor meget nemt, dynamisk at ændre. HTML er i stor udstrækning platformsuafhængigt.

- **.Net** – version 1.1
  Fordelen ved at bruge .Net er alle de standard funktionaliteter der er inkluderet. En anden fordel ved .Net er at Visual Studio .Net er et udviklings værktøj der at specialfremstillet til .Net's funktionaliteter.

- **XSLT**
  Dette er et scriptsprog der kan konvertere en xml struktur til en anden struktur. I projektet er det XSLT der genererer HTML ud fra den konfiguration og den data der er tilgængelig.

- **SQL Server 2000**
  Relationen database. Er valgt pga. den gode integrering i .Net og dens generelt meget gode brugergrænseflader.

Produktet udvikles vha. en række værktøjer.

- **XMLSpy** er et fantastisk værktøj til at arbejde med XML i. Det indeholder flere måder at få visualiseret XML strukturer. XML er en træstruktur, og xmlSpy kan bl.a. visualiserer denne vha. kasser.

- **Visual Studio .Net** er et udviklings værktøj der er udviklet sideløbende med selve .Net platformen. Visual Studio er fuld a wizards der gør opgaver, ukendte for programmøren, til en leg.

- **Together** bruges til at tegne UML[1]. Together er målrettet mod Java, og bruges her derfor kun i kraft af dets visuelle egenskaber.

[1] *Unified Modeling Language*. Standard til at beskrive software strukturer.

7.2 Grænseflader

Der er overordnet set 2 grænseflader på produktet. Den ene grænseflade er der hvor udvikleren kan konfigurere. Den anden er den som leverer den konfigurerede funktionalitet til slutbrugeren. Grænsefladen til udvikleren vil være fast defineret, mens grænsefladen til slutbrugeren vil være afhængig af den konfiguration som udvikleren har foretaget.

Selvom slutbrugergrænsefladen er afhængig af udviklerens konfiguration, er det vigtigt at den overholder nogle retningslinier. Dette fordi systemernes skal kunne udnytte hinandens datagrundlag til søgninger.

7.2.1 Konkret implementering ved brug af XML

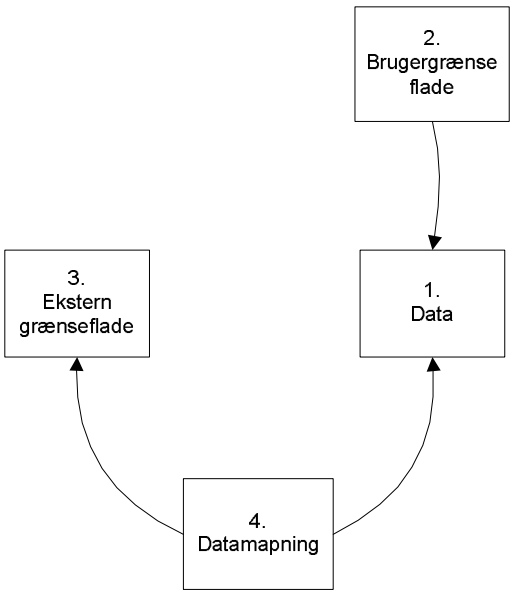
Slutbrugersystemet skal kun kunne indeholde struktureret data. Der er derfor ikke behov for at udnytte alle XML's funktionaliteter når der skal kommunikeres data. Derimod er der behov for det i konfigurationsfilerne. Derfor er XSD blevet fravalgt som værende formatetet til konfigureringerne, og XML er valgt i stedet. Det giver også den fordel at det strukturen for konfigurationerne kan begrænses til at kunne indeholde gyldige konfigurationer.

Slutbrugersystemerne skal kun kommunikere struktureret data, og der laves en XML struktur der kan indeholde noget sådant.

7.2.2 Konfigureringsgrænseflade

Konfigurationsgrænsefladen er der hvor slutbrugersystemet får sine egenskaber. Det er i konfigurationen at datastruktur og eksterne grænseflader samt brugergrænsefladen defineres.

Konfigurationerne i systemet er delt op efter type. Der vil være en konfiguration der beskriver hvorledes datagrundlaget hænger sammen og en der beskriver hvorledes brugergrænsefladen hænger sammen. Disse konfigurationer er afhængige af hinanden og dette er de efter diagrammet i Figur 12. Datastrukturen ligger til grund for alle de andre konfigurationer og er derfor ikke afhængige af andre. Brugergrænsefladen er direkte afhængig af datastrukturen. Den eksterne grænseflades konfiguration er ikke direkte afhængig af datamodellen. I stedet er der en mapning imellem datamodellen og den eksterne grænseflade.



Definitioner:
Systemet bliver defineret i forskellige dele. De enkelte dele kan på den måde genbruges I andre sammenhænge.

1. Datadefinitionen er den mest grundlæggende beskrivelse I systemet. Den skal modellere det problem systemet skal løse.

2. Brugergrænsefladedefinitionen bygger på datadefinitionen og beskriver hvorledes brugeren får præsenteret og navigerer igennem data.

3. Den eksterne grænseflade definition er en beskrivelse af den service applikationen stiller til rådighed. Da den eksterne grænseflade ønskes at kunne udveksles imellem systemer samt at kunne implementeres af forskellige systemer, er definitionen af den eksterne grænseflade ikke koblet direkte til datastrukturen. Koblingen bliver foretaget I mapnings definitionen.

4. Mapningen kobler den eksterne grænseflade til selve datastrukturen.

Figur 12: Konfigurationernes afhængigheder

Når slutbrugersystemet skal beskrives, startes der med datastrukturen. Datastrukturen bestemmer hvad data der kan gemmes i slutbrugersystemet, og hvordan data kan søges og manipuleres.
Når datastrukturen er på plads, kan der defineres en brugergrænseflade oven på datastrukturen. Brugergrænsefladen refererer direkte til den datastruktur der er defineret. Brugergrænsefladen kan defineres både i udseende og i funktion. I brugergrænsefladen kan der refereres til elementer i datastrukturen, og navigation for slutbrugeren kan defineres. Det

er på den måde muligt at specialtilpasse brugergrænsefladen til det specielle behov, og sågar lave flere brugergrænseflader bygget på samme datastruktur.

De eksterne grænseflader defineres uafhængigt af datastrukturen i system. Dette gøres fordi den eksterne grænseflade så kan bruges af flere systemer, samt at beskrivelsen af den eksterne grænseflade, direkte kan bruges af mulige klienter til at hive information ud om hvordan der kan kommunikeres med systemet. I praksis vil det sige at en organisation kan have en række standard ”eksterne grænseflader”, og hvis de enkelte systemer skal kommunikere, skal det ske igennem disse.
Da den eksterne grænseflade ikke har noget at gøre med den interne datastruktur, bliver der nød til at være en mapning imellem de to.

Eks.
Det vil således være muligt i organisationen at bestemme at der skal kunne udveksles kunde oplysninger imellem afdelingerne. Ved at lave en definition af hvordan kundeoplysninger skal udveksles, bliver det i fremtiden meget nemt at lave nye systemer der enten udbyder eller tager brug af andre systemer der kan udveksle kundeoplysninger.
Det vil være nemt at omorganisere organisationen, hvis der ikke er nogen systemer internt der er afhængige af hinanden. Alle systemerne er kun afhængige at kunne koble på et system der levere data i et bestemt format. Hvor det system er placeret, eller hvad det ellers kan er ikke relevant for klient programmet. Systemerne bliver ikke afhængige af hinandens implementeringer, men af hvad de leverer.

Krav til konfigurationen.

- Det skal være muligt at oprette en datastruktur.
- Det skal være muligt at se hvorledes systemet er konfigureret.
- Det skal være muligt at oprette en brugergrænseflade der udnytter datastrukturen.
- Det skal være muligt at definere eksterne grænseflader.
- Det skal være muligt at bruge data fra andre systemer i datastrukturen.
- Det skal være muligt at definere interaktion med data (søgning, manipulation etc.)

Al konfiguration sker vha. XML. På den måde er det muligt at lave konfigurationerne semistrukturerede. Det vil sige at brugergrænseflader kan konfigureres så de indeholder rekursive elementer, og at referencer i konfigurationen kan valideres for faktisk at pege på noget gyldigt.
Valideringen af strukturen forgår vha. XSD som muliggør at hver eneste felt kan valideres for korrekthed.

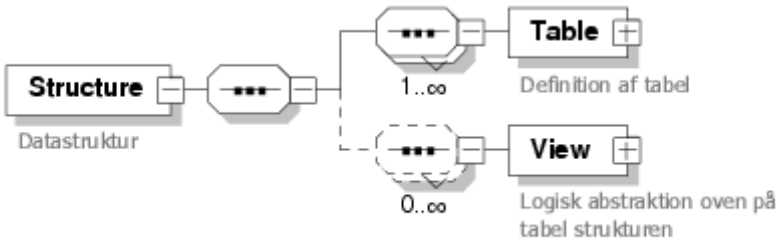
Der er valgt at udvikle en egen XML struktur til konfigureringen. Alternativt kunne der bruges XSD til at definere datastrukturen, men da det ikke vil gavne simpliciteten at give udvikleren en masse unødvendige valg er dette fravalgt.

7.2.3 Datastruktur

Hovedformålet med slutbrugersystemerne er at kunne behandle data. Det er derfor nødvendigt at det i systemet er nemt at konfigurere datastrukturer. Der er taget udgangspunkt i hvad der kan defineres i en relationel database. Specifikationen på grænsefladen til at konfigurere datastrukturen kan ses i kap. 12 Appendix – Systemkonfiguration.

Følgende skal kunne defineres i datastrukturen:

- Tabeller, og disse skal kunne gives navne
- Integritetsvalidering på tabelnivaeu (unikke værdier etc.)
- Relationer imellem tabeller
- Abstraktioner på data i form af views
- Valideringsregler på feltniveau
- Søgningsregler
- Forbindelser til eksternt data



Figur 13: Konfigurationsgrænsefladen som XMLSpy visualiserer den.

Konfigurationen er delt op i to dele. Den ene del omhandler definition af tabeller med felter og valideringsregler mm. Det er altså selve strukturen, og hvordan data lagres og valideres.

Den anden del omhandler hvorledes data kan hives ind og ud. Det er Views der bliver defineret hvor der både bliver påført information om hvorledes tabellerne hænger sammen og hvilke eksterne ressourcer der skal inkluderes. Det er i Views at søgninger bliver defineret.

I Figur 13 er den overordnede datakonfigurationsgrænsefladen vist. Yderst til venstre ses hovednoden for al datadefinering. *Structure* noden indeholder mindst én *Table* node, 0 til mange View noder. Det er ikke muligt at se attributter og afhængigheder.

7.2.3.1 Tabeller

Det er påkrævet i en konfiguration at definere en tabel.

En definition af en tabel består af en liste på mindst en feltdefinition.

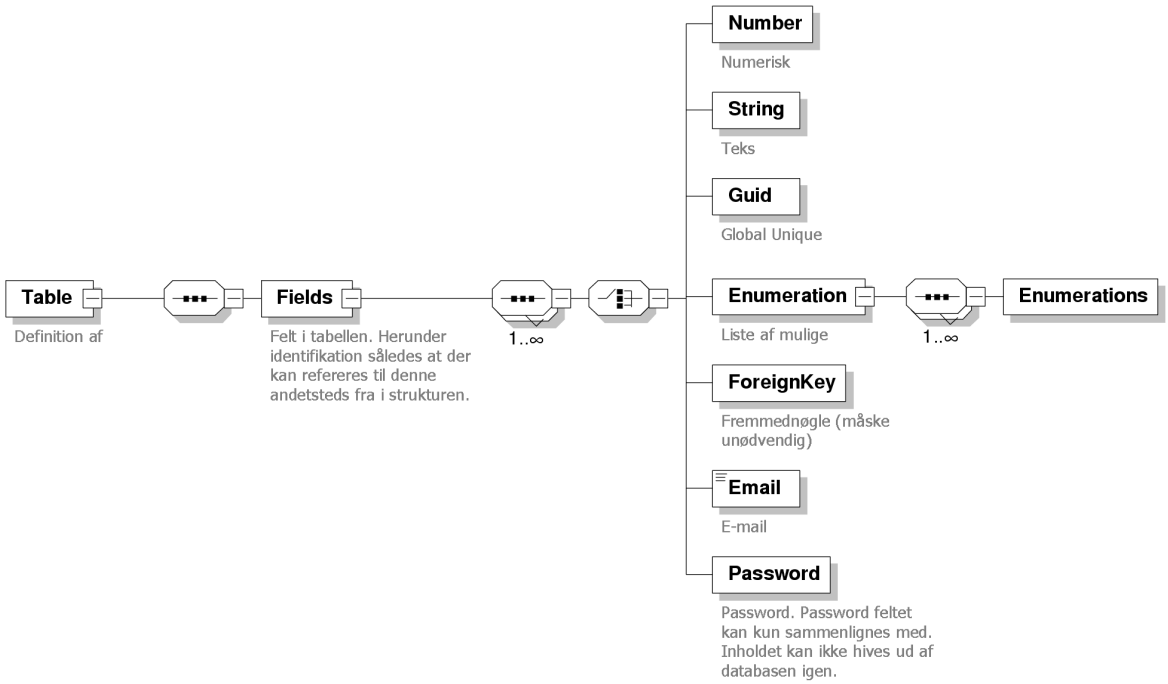
Number	Numerisk værdi
String	Tekst
Guid	GUID: Globaly Unique ID. Unikt id genereret ud fra mac adressen på maskinen samt et timestamp og en tæller.
Enumeration	Kan antage værdien af én af en prædefineret liste.
ForeignKey	Fremmednøgle. Det valideres at feltet nøglen peger på eksisterer
Email	Tekst indeholdende email adresse.
Password	Tekst der ikke kan trækkes ud af slutbrugersystemet. Der er kun muligt at sammenligne med dette.

Figur 14: Felttyper i tabeldefinition

En tabel vil bestå af flere felter der hver især kan være af en af de prædefinerede typer. Data vil blive valideret op imod typedefinitionen på feltet.

Tabellernes rolle i slutbrugersystemet er at her opbevares alle data. Det vil derfor ofte være her at udvikleren starter med at udvikle et nyt slutbrugersystem. Hele grundlaget for resten af slutbrugersystemet laves her.

For at kunne referere til data i slutbrugersystemet skal alle tabeller og felter kunne identificeres unikt. Dette bliver valideret vha. et XSD. Både *Table* og felt elementer skal have et navn. Tabellens navn bliver valideret på system niveau, mens felternes navne bare skal være unikke inden for den tabel de er defineret i.



Figur 15: Definering af data

For hver tabel bliver der lavet noget XML der overholder strukturen vist i Figur 15. en *Tabel* definition vil altid have et element under sig der hedder *Fields*. Under *Fields* er der en række elementer. Disse elementer repræsenterer hvilke felter der er i selve tabellen. Til hver slags felt er der en elementtype der skal overholdes. F.eks. vil man når man laver et felt af typen *Guid* skal man udfylde attributten *ISROWGUID* som fortæller systemet om det er den primære nøgle for tabellen.

Eksempel på definition af tabel.

```
<Table name="Personer" description="Personer i virksomheden">
    <Fields>
        <Guid name="personId" description="Identifier" defaultValue="newid()" required="true" isKey="true"/>
        <String name="Fornavn" description="Fornavn" required="true"/>
        <String name="Efternavn" description="Efternavn" defaultValue="String" required="true"/>
        <String name="Personnummer" description="personen offentlige personnummer" required="false"/>
    </Fields>
</Table>
```

Figur 16: Eksempel på definering af tabel

7.2.3.2 View

Et View er en forespørgsel af data.

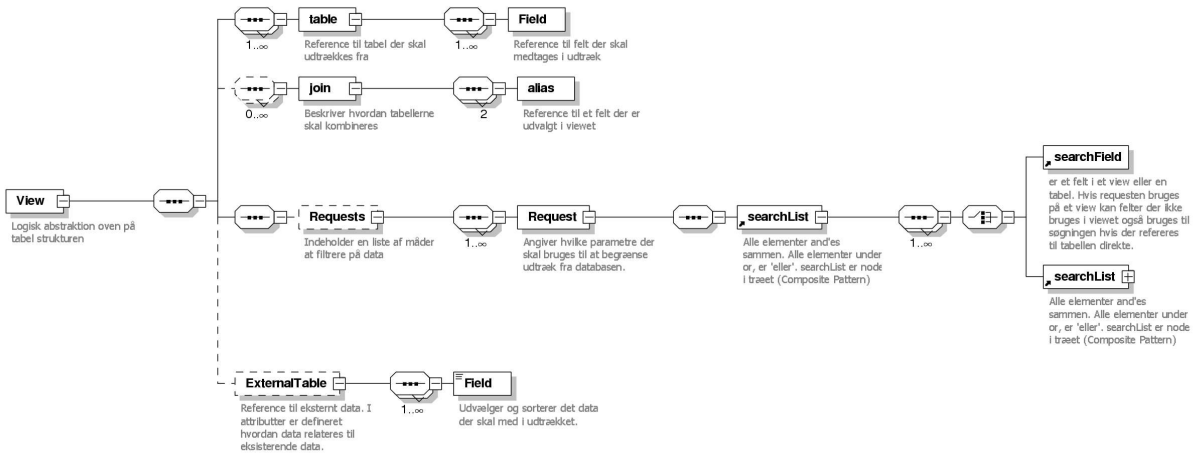
En forespørgsel kan deles op i tre ting. Information om hvad der ønskes returneret. Information om hvorledes data er relateret og til sidst information om hvilke kriterier forespørgslen skal foretages på grundlag af.

En visualisering af strukturen for en definition af et view er vist på Figur 17.

Hvilke data der ønskes returneret er en simpel liste af tabeller og de felter der skal med fra hver tabel. Hvis det er alle felter i en enkel tabel kan der bare refereres til tabellen.

Informationer gemt som relationelt data er ofte gemt i flere relaterede tabeller med relationer defineret ved hjælp af fremmednøgler. Dette bliver understøttet vha. join elementet. Da det skal være muligt at have flere relationer imellem to tabeller, skal systemet kunne referere til tabeller vha. aliaser. Således kan samme tabel tilgås forskelligt i samme udtræk, alt efter hvilken relation den indgår i.

Requests er selve kriterierne der skal bruges i søgningen. Det er en liste over hvilke felter der skal indgå i filtreringen, og i hvilken sammenhæng.



Figur 17: Udtræk af data.

Figur 17 viser strukturen for hvordan et udtræk kan defineres. *View* elementet kan indeholde flere elementer. *View* elementet skal indeholde et *Table* element. Et udtræk giver ikke nogen mening hvis ikke der refereres til noget data.

Det er muligt at vælge data fra flere forskellige tabeller, og derfor er det og nødvendigt at kunne beskrive hvorledes disse tabeller skal relateres. I *join* elementet er det muligt at beskrive hvordan de to tabeller skal joines, og hvilke felter der skal samenlignes.

ExternalTable refererer til data på et andet slutbrugersystem. Som attribut til elementet er der en reference til et felt på den eksterne ressource og en på den interne. At kunne joine udtræk med eksterne ressourcer giver meget stor fleksibilitet, men giver samtidigt mulighed for at lave meget tunge udtræk. Hvis data ikke bliver frasorteret det rigtige sted ender man med at overføre meget store mængder data. Det gælder om at frasortere så meget som muligt, før det sendes over en netværksforbindelse. En almindelig måde at filtrere sit datamængde på en database er at filtrere på relationen. I et distribueret system vil det dog være en alvorlig performance faktor. Dette fordi datamængden ikke kan filtreres før det sendes over nettet, og man ender så med at sende data fra en hel tabel over nettet.

Requests elementet er samling af *Request* elementer. En *Request* er en beskrivelse af hvordan data filtreres. Altså hvordan det data man ikke skal bruge bliver sorteret fra. En *Request* indeholder en *searchList*. En *searchList* er et rekursivt element. En *searchList* kan således indeholde en eller flere *searchList*s. De er rekursive for at det er muligt at selv definere komplekse filtre.
En searchList kan indeholde searchField som svarer til at brugeren kan filtrere på et felt.

Eksempel:

Et udtræk skal kun returnere de poster der overholder følgende:
(**BLÅ** eller **RØD**) og **4 hjul**

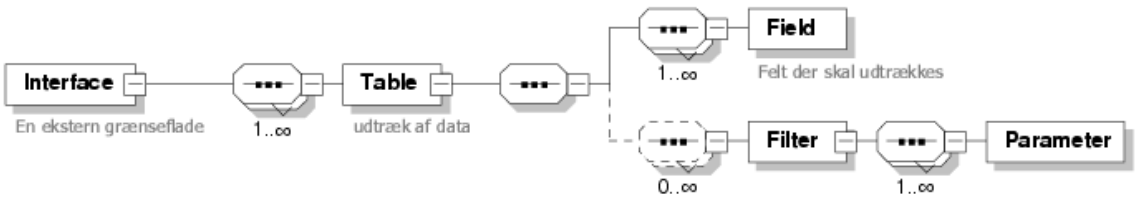
I et sådant tilfælde er det ikke nok at liste de kriterier der skal overholdes. Det skal være BLÅ eller RØD.

```
<Request>
    <searchList operator="and">
        <searchList operator="or">
            <searchField nameref="farve" type="equal" name="BLÅ"/>
            <searchField nameref="farve" type="equal" name="RØD"/>
        </searchList>
        <searchField nameref="hjul" type="equal" name="4"/>
    </searchList>
</Request>
```

Figur 18: Eksempel på søgekriterier

7.2.3.3 Ekstern grænseflade

Den eksterne grænseflade i slutbrugersystemet skal kunne bruges til at kommunikere data imellem flere forskellige systemer. Grænsefladen skal kun understøtte udveksling af data, og definitionen vil derfor være meget lig definitionen af datastruktur.



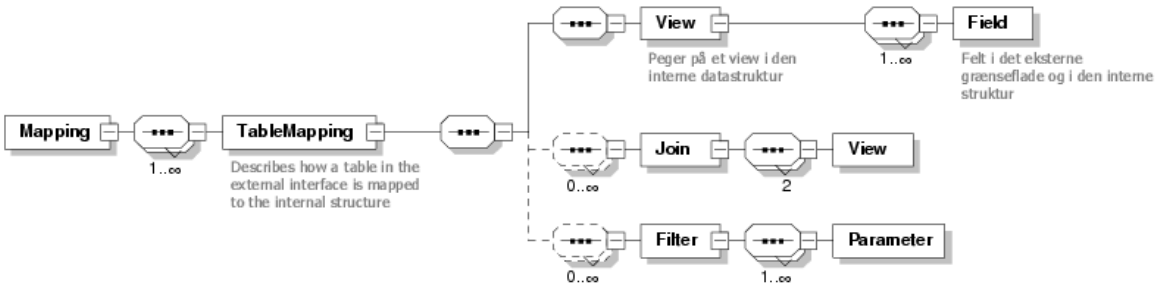
Figur 19: ekstern grænseflade

Til at starte med vil den eksterne grænseflade kun understøtte udstræk af data. I Figur 19 er vist definitionen på en konfiguration af en ekstern grænseflade. Det er kun selve forespørgslen der er defineret. Data bliver overført i det XML format som .Net's DataSet objekt understøtter.

De eksterne grænseflader vil kunne defineres før resten af systemet defineres. Det muliggør at den overordnede struktur kan implementeres først. Således kan hele organisationens struktur først implementeres kun ved hjælp af de eksterne grænseflader. Senere kan de enkelte systemer så defineres med hver deres specifikke funktionalitet.

7.2.3.4 Mapnings grænseflade

Mapningen forgår lidt ligesom et View. Eneste forskel er at der ikke bruges tabeller som datagrundlag, men View. Alle felter i den eksterne grænsefladedefinition skal mappes til et internt felt. Løsningen giver i sin nuværende form ikke mulighed for at lave meget komplekse mapninger.



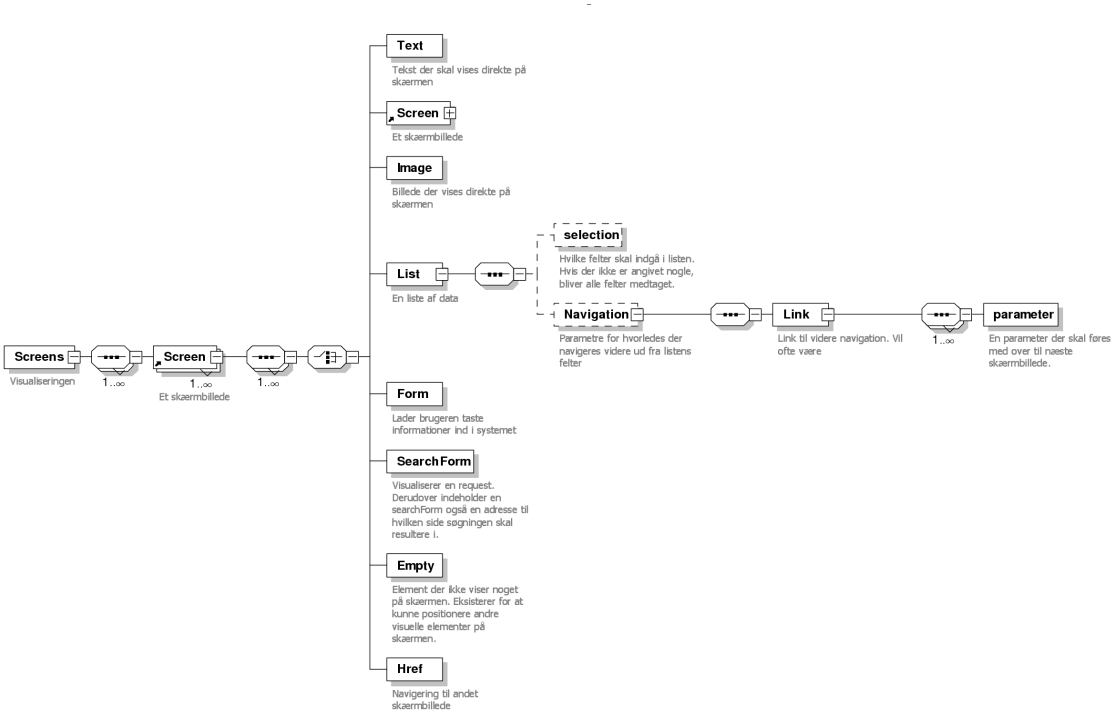
Figur 20: Mapning

7.2.4 Konfiguration af Brugergrænseflade

Brugergrænsefladen defineres på grundlag af datastrukturen. Definitionen af brugergrænsefladen indeholder både informationer om de enkelte skærmbilleders visuelle udseende og deres funktion. Det er altså også her at den enkelte bruger arbejdsgang bliver defineret.

Brugergrænsefladen skal kunne defineres med følgende parametre:

- Udseende af det enkelte skærmbillede (Med udseende forstås hvilke data der skal være på den enkelte side samt hvordan de er placeret i forhold til hinanden)
- Navigering imellem skærmbilleder
- Hvordan håndteres data (søgning og opdatering af data)



Figur 21: Brugergrænseflade definition

7.2.4.1 Layout

Det skal være muligt at placere de visuelle elementer på skærmen. Til dette bruges et element der repræsenterer et skærmbillede.

Brugergrænsefladen er defineret ved en liste af skærmbilleder. Et skærmbillede er et visuelt element der i sig selv kan indeholde andre visuelle elementer. Alle skærmbilleder skal kunne unikt identificeres således at der kan refereres til skærmbilleder med henblik på navigeringsforløb etc. Et skærmbillede er tænkt som værende en container, som brugeren kan bruge til vise flere visuelle elementer på samme skærmbillede og derved selv skabe et overblik over datasammenhænge i datastrukturen. Et skærmbillede indeholder en ren visuel attribut. *Split*, som beskriver hvordan listen af visuelle elementer skal tegnes på skærmen. Da et skærmbillede selv er et visuelt element, kan andre skærmbilleder godt indgå i et skærmbilledes liste af visuelle elementer. Ved at inkludere skærmbilleder i hinanden, og opdele disse skærmbilleder på forskellige leder, er det muligt at positionere elementer overalt inden for skærmbilledet.

```
<Screen id="456" bgcolor="blue" name="Automatik til system oprettelse">
    <Text value="Velkommen"/>
    <Empty bgcolor="grey" heigth="400" width="300"/>
    <Screen split="vertical" bgcolor="grey">
        <Screen >
            <Href target="4813e3f8-f929-473e-a230-a7da117d967a" value="Søg"/>
            <Href target="d9828c7b-cf21-4dcc-88ff-6da14840e856" value="Kontakt"/>
        </Screen>
        <Image src="logo2.jpg" type="URL"/>
    </Screen>
    <Text value="Slutbrugersystemet"/>
</Screen>
```

Ønsket resultat:

Figur 22: Eksempel på definition af skærmbillede

Skærmbillede definitionen indeholder ikke i sig selv noget visuelt der vises på skærmen. Den bruges til at positionere andre visuelle elementer. De andre visuelle elementer vises for slutbrugeren. Det er elementer som f.eks. Image og Text.

Visuelle elementer indeholder informationer om hvordan systemet præsenterer data til brugeren. Derfor indeholder alle visuelle elementer en række fælles parametre. Højde, bredde samt baggrundsfarve. Det visuelle element Empty indeholder kun disse standard parametre, og kan derfor kun bruges til at skabe luft imellem andre visuelle elementer.

Screen	Skærmbilled elementet, der indeholder andre visuelle elementer og bruges til at skabe sammenhæng imellem de visuelle elementer.
Text	Statisk element der skriver en statisk tekst ud på skærmen.

Image	Statisk element, der indeholder en reference til et billede der vil blive vist på skærmbilledet.
List	En liste af data. Data vil komme fra et udtræk fra systemet, og List kan betragtes som værende en visualisering af et af datastrukturens Views. Foruden et view vil en Liste også have en reference til en request, samt evt. nogle parametre vedrørende videre navigering fra Listen.
searchForm	Søgeformularens formål er at få en request udfyldt med slutbrugerens indtastede data.
Empty	Indeholder ingen informationer ud over de givet af at det er et visuelt element. Elementet bruges at skabe afstand imellem andre visuelle elementer.

Figur 23: Visuelle elementer i en brugergrænseflade konfiguration

Nogle af de visuelle elementer er statiske og vil altid vise det samme til slutbrugeren. Andre er dynamiske og ændre sig alt efter hvilken handling brugeren har foretaget, eller hvilke data der er i databasen

7.2.4.2 Databehandling

For at slutbrugeren kan bruge slutbrugersystemet til noget, skal der være mulighed for at foretage søgninger. Der skal være en måde at slutbrugeren kan taste oplysninger ind, og derefter få vist data fra databasen der opfylder disse kriterier. Datastrukturen indeholder allerede en specifikation på hvordan man kan søge, og brugergrænsefladen skal kunne udnytte disse.

Søgninger bliver defineret ved hjælp af et visuelt element der hedder searchForm. På searchForm kan der henvises til et Request element fra et View. En searchForm vil vise de felter der er i Reqest'en. Det er en visualisering af Request'en

searchFormen gør det muligt for slutbrugeren at taste data ind i de felter der er defineret i requesten. Disse data vil slutbrugersystemet bruge disse data til at lave en søgning. På searchForm er der en attribut der refererer til det skærmbillede der kan vise søgeresultatet.

Når søgningen bliver foretaget vil der blive skiftet til skærmbilledet med listevisningen. Listevisningen får den Request struktur med der skal bruges til at begrænse hvad data der bliver vist i listen. Informationerne fra View'et om hvilke felter der skal vises, og i hvilken rækkefølge vil blive brugt når søgeresultatet vises på skærmen.

En liste skal have et default Request objekt der skal bruges, hvis der ikke bliver sendt et Request objekt med.

7.2.4.3 Navigering

Brugeren skal kunne navigere ud fra de informationer der vises på skærmen. Når der vises data i brugergrænsefladen skal det være muligt at definere navigations forløb. Et eksempel er at brugeren ved at udfylde en søgeformular skal havne på et søgeresultat.

Den eneste mulighed der for at lave navigerings muligheder for slutbrugeren er ved at bruge Link elementet. Link elementet tager en tekst og en reference til et skærmbillede. På brugergrænsefladen vil dette resultere i en knap som vil bringe brugeren til det refererede skærmbillede.

7.2.5 Grænseflade på slutbrugersystem

Dette er en beskrivelse af hvorledes der kan kommunikeres med slutbrugersystemet.
Der er tre kommunikationsformer.

- Igennem brugergrænseflade
- Igennem objektstrukturen beskrevet i kap. 7.3
- Kommunikation igennem ekstern grænseflade

7.2.5.1 Brugergrænseflade

Når der implementeres en brugergrænseflade, vil den modtage selve definitionen på brugergrænsefladen plus data til den pågældende brugergrænseflade af systemet.

Brugergrænsefladeimplementationen generer ud fra konfigurationen, selve brugergrænsefladen og udfylde den med data. En brugergrænsefladeimplementation kan ses i afsnit 16 .

7.2.5.2 Objektstruktur

Objektstrukturen er forklaret i kap. 7.3

Denne grænseflade understøtter al kommunikation ind og ud af systemet. Det er denne grænseflade der kan bruges, når produktet skal udvides med ny funktionalitet.

7.2.5.3 Ekstern grænseflade

Den eksterne grænseflade er den grænseflade andre slutbrugersystemer skal kunne kommunikere med systemet igennem.

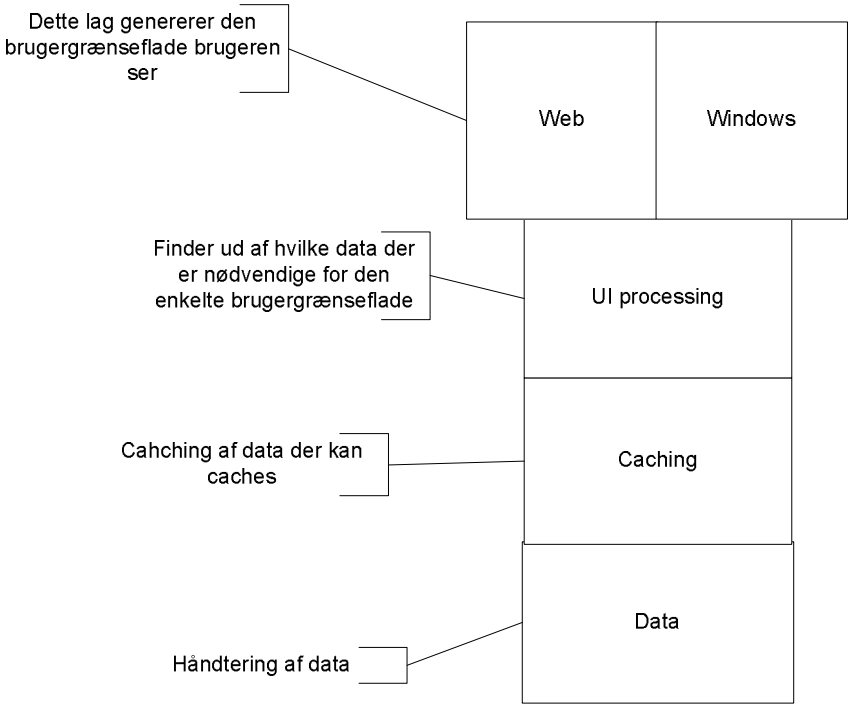
Den eksterne grænseflade understøtter kun søgning på data.

7.3 Klassestruktur

Der lægges der vægt på at det skal være grundlæggende funktionaliteter der implementeres, samt at nye funktionaliteter senere kan tilføjes. Desuden vil integration og overskuelighed blive prioriteret højere end performance.

Designet af koden til dette projekt sker Objektorienteret[1].

Systemet designes således at alle objekter både skal fundere ved læsning og implementering af konfigurationer, og eksistere i slutbrugersystemet.



Figur 24: Implementering af systemet lag.

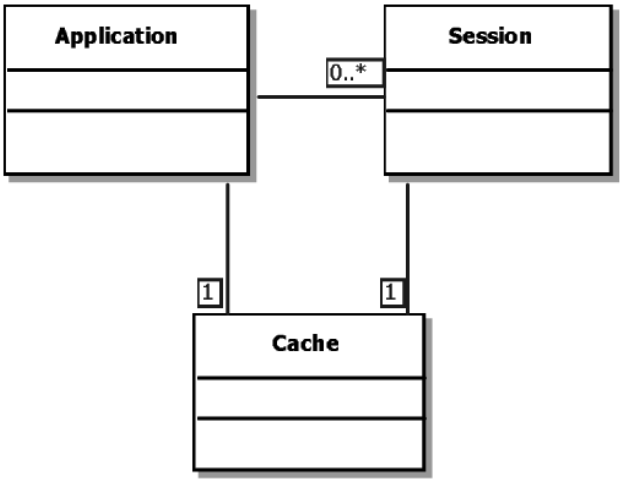
Programmet deles op i lag. Disse lag afspejler også pakke strukturen. En pakke vil tage sig af al kommunikation med databasen. Alle objekterne i en anden pakke vil tage sig af håndtering af brugergrænsefladen.

[1] Se 5.5 OO desig

7.3.1 Overordnet

Yderste pakke er den overordnede. Herfra er der adgang til funktionalitet implementeret i de andre pakker.

Application	I denne klasse styrer de overordnede ting i applikationen. Den har forbindelsen til databasen, og den initierer når slutbrugersystemet skal omkonfigureres. Application er det centrale objekt i systemet, og det er derfor også her at data kan caches.
Session	Hver gang en ny forbindelse bliver lavet til systemet, bliver et session objekt lavet. Session objektets formål er at håndtere brugespecifikke informationer og at cache brugerens data.
Cache	Denne klasse bliver både brugt af Application og af Session klasserne. Cache objekteter kan gemme søgeresultater sammen med søgeparametrene. Når samme søgning forekommer igen, kan resultatet hentes i her.



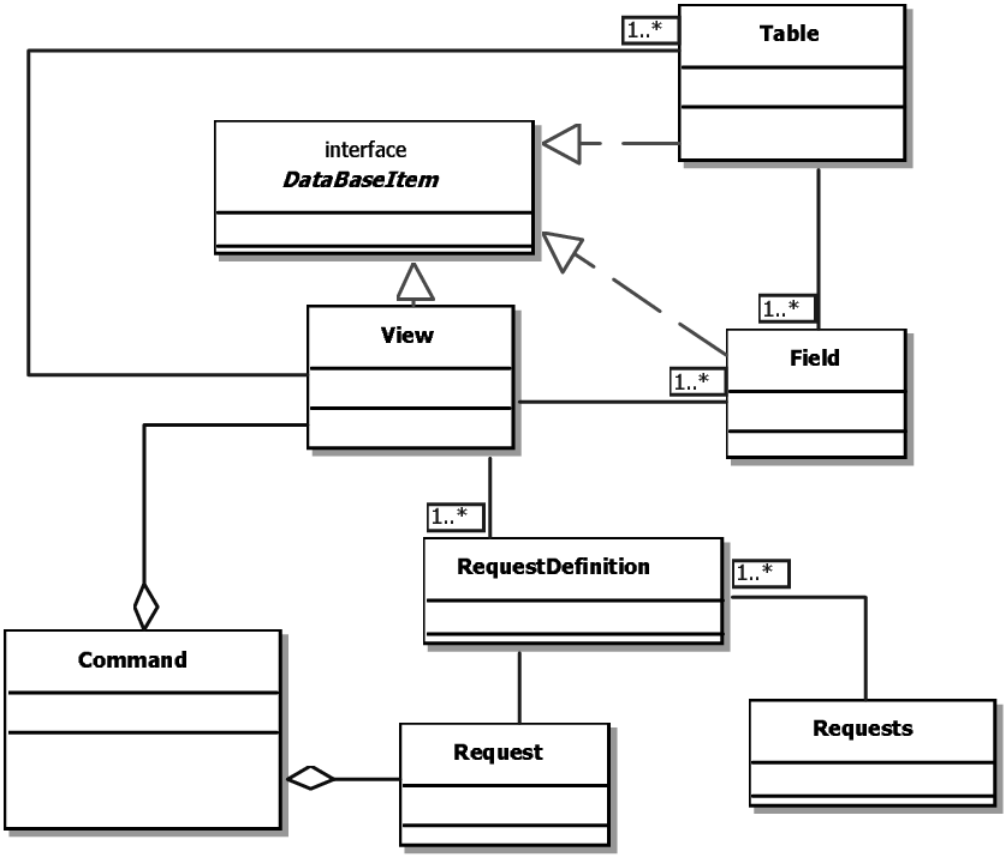
Figur 25: Klassestruktur, overordnet

Figur 25 viser strukturen af de tre klasser. Der er altid kun et Application objekt. Der kan være flere Session objekter knyttet til Application. Både Application og Session har ét Cache objekt

7.3.2 Data

Data pakken skal både kunne oprette datastrukturer i en database, samt hente og gemme data i denne struktur.

DatabaseItem	Dette er et interface der repræsenterer et objekt i databasen. Alle objekter der skal i databasen skal implementere dette interface. Det har metoder til at generere SQL til at oprette og søge i datastrukturer.
Table	Implementerer DataBaseItem. Table skal konfigureres med en datastruktur før den kan tages i brug. Når en tabel er konfigureret vil den indeholde information om hvordan dens struktur kan oprettes i databasen samt hvordan man søger.
View	Implementerer DatabaseItem Et view skal konfigureres før det kan tages i brug. Et view indeholder en liste af felter fra tabeller. Et view kan indeholde RequestDefinition objekter.
Field	Implementerer DatabaseItem. dette er felter i databasen. Field klassen er en hjælpe klasse, som Table og View klasserne kan bruge til at få genereret sql til felterne.
Request Definition	RequestDefinition objekter svarer til Request elementer i konfigurations grænsefladen. Et RequestDefinition indeholder information om hvorledes søgninger kan filtreres
Request	Dette er en liste af data fra brugeren. Ved hjælp af RequestDefinition bliver værdierne i Request objektet omdannet til et filter der kan bruges i databasen.
Command	Dette objekt indeholder information om hvilken søgning der skal foretages. Et command objekt indeholder en reference til et View samt et Request objekt. Command objekter skal kunne sammenlignes, for at de senere kan bruges til caching.
DataSet	Indeholder resultat fra databasen. I Cachen gemmes dette objekt sammen med. Et Command objekt.
DataDefinition	Dette objekt bruges til at finde DataBaseItem objekter.



Figur 26: Klassestruktur for datatilgang

7.3.3 Brugergrænseflader

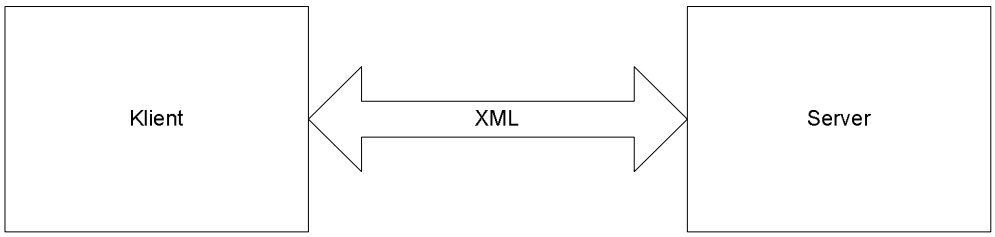
I UI pakken er alle klasser der har noget med brugergrænsefladen at gøre. Disse er hjælpeklasser der bruges til at generere brugergrænsefladen samt navigere og til at finde ud af hvad for noget data der skal hentes fra databasen for at skærmbilledet kan vises.

Under UI pakken er der en pakke der hedder WEB. Denne er lavet for at implementere de forhold der gør sig gældende når systemet skal bruges på en web-grænseflade. På samme måde ville andre medier kunne implementeres i tilsvarende pakker.

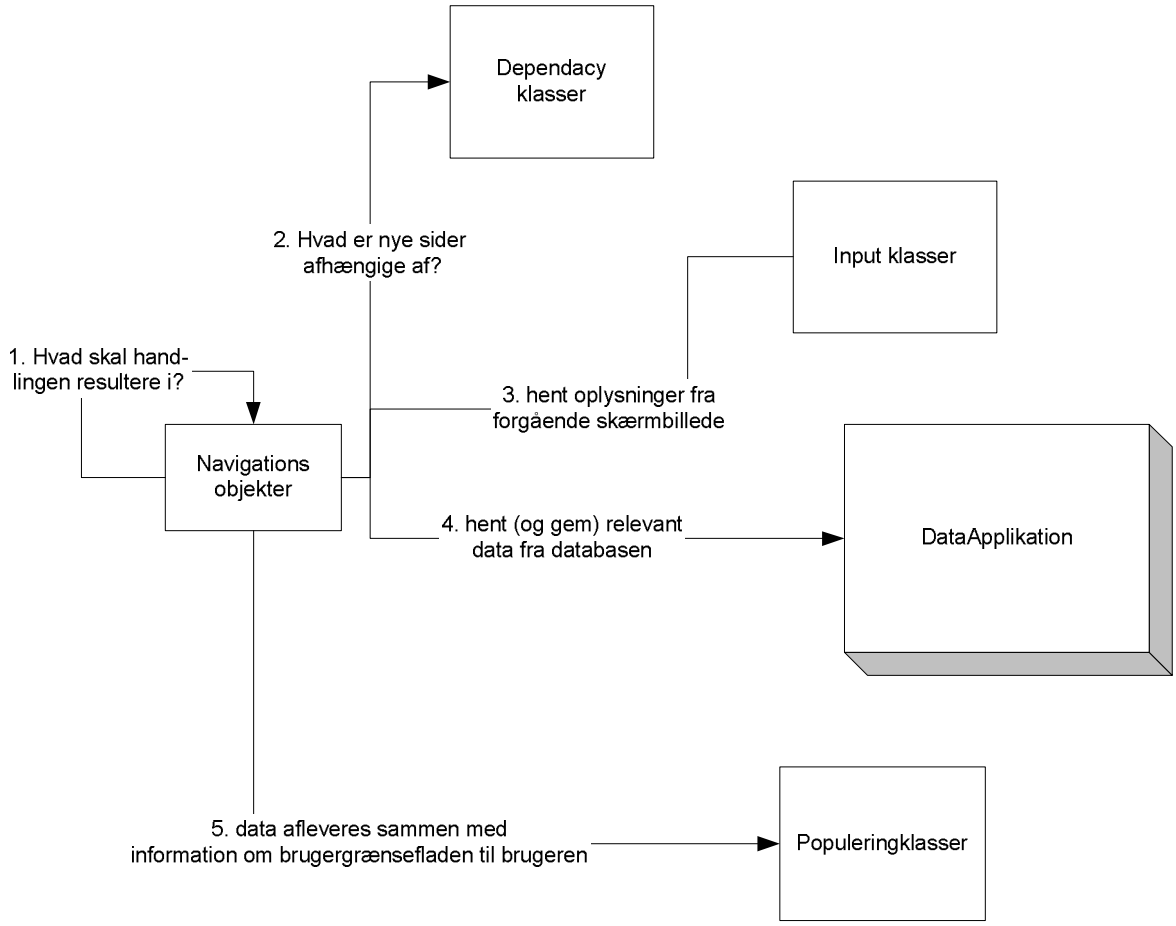
Skærmgrænsefladen er i første omgang tænkt rettet mod web, og forløbet for at kommunikere med systemet vil derfor være request / response orienteret. Hver handling vil derfor resultere i en forespørgsel til servere.

Brugergrænsefladen er designet til at serveren genererer HTML til klienten, som kan få det vist vha. en browser. Med mere avancerede klienter, kunne kommunikationen reduceres til at være data. Det vil reducere arbejdsbyrden på serveren, og vil også mindske mængden af data overført.

Figur 27: Webbaseret klient med minimal belastning af brugerens computer. Skærmbillederne samt brugerens input bliver håndteret på serveren



Figur 28: Klient der selv kan generere skærmbillederne. Brugeren input bliver håndteret på klienten.



Brugeren har foretaget en handling fra brugergrænsefladen

1. Det er navigeringsobjektet der først håndterer brugerens handling. Når en handling bliver foretaget, skal navigeringsmodulet finde til hvad der skal returneres til brugeren.

Dette sker ved at bruge forskellige andre moduler.

2. Dependencies bliver fundet I dependency modulet. Dependency modulet holder styr på hvad der er krævet for de forskellige skærmbilleder.

3. Input modulet kan tage brugerens input og konvertere disse til brugbar information I dataapplikationen.

4. Dataapplikationen er beskrevet andet steds. Den hovedfunktionalitet er at tilbyde databehandlings funktionalitet til andre moduler.

5. Når det er undet ud af hvilket skærmbillede handlingen skal resultere i, skal det sendes tilbage til brugeren.

Til aller sidst skal brugeren præsenteres for skærmbilledet. Skærmbilledet bliver genereret andetsteds.

Figur 29: Flowet ved generering af en brugergrænseflade.

7.3.4 Eksterne Grænseflader

DataService er en pakke der håndterer den eksterne grænseflader. Hvis man laver en webservice der skal kunne hente data fra systemet skal man bare sende forespørgslerne direkte videre til klasserne i DataService.

De eksterne grænseflader vil være implementeret vha. webservices der benytter SOAP protokollen som kommunikations medie. SOAP bruger i sig selv XML som transport lag. SOAP er en fleksibel protokol der kan tilpasses til formålet, og som sågar kan ændres dynamisk i løbet af programmets forløb.

7.4 Datamodel

Slutbrugersystemet bruger en database til at lagre alt. Alle konfigurationsparametre ligger som XML, og dette kan derfor gemmes som en enkelt streng. XML'en og andre eventuelle parametre der er behov for at gemme bliver lagt i en key / value pair tabel. Den ene kolonne hedder key og indeholder primærnøgler for tabellen. Den enden hedder value, og den indeholder den værdi der passer til primærnøglen. Tabellen vil indeholde en række hvor primærnøglen er 'konfiguration' og det andet felt indeholder hele XML strukturen der definerer slutbrugersystemet.

Alle tabellerne i slutbrugersystemets datastruktur, bliver oprettet som tabeller i databasen. Databasens egenskaber i form af performance kan udnyttes til det maksimale. View bliver også oprettet i databasen.

For at holde slutbrugersystemets datastruktur adskilt fra andre tabeller, navngives alle slutbrugersystemets tabeller med et prefix.

8 Implementering

Selve udviklingen er foregået iterativt. Slutproduktets funktionalitet er blevet delt op i mindre dele og disse er så blevet implementeret. Produktets funktionalitet kan ikke afprøves hvis ikke der også er defineret et slutbrugersystem. Der er der lavet et simpelt person kartotek med adresser og telefonnumre på personer.

Systemet er ikke tænkt som værende et realistisk scenario for produktet, men skal fungere som værende et mål for implementeringen. Da kravene til produktet er meget generelle er det ikke muligt at lave et system der kan teste det hele.

Test scenariet er et personkartotek med telefonnumre og adresser.

Følgende skal slutbrugersystemet kunne

- Lade slutbrugeren finde adressen på en person, givet navnet på personen.
- Lade slutbrugeren finde en persons telefonnummer, givet navnet på personen.
- Lade slutbrugeren selv navigere til en af ovenstående muligheder og lade slutbrugeren skifte imellem de to.

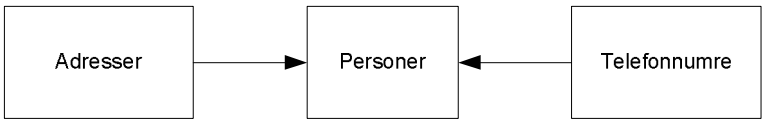
For at udfordre mulighederne i produktet, er informationerne delt op i forskellige tabeller. Og for at teste forbindelsen til eksterne ressourcer vil telefonnumre til søgningerne blive hentet vha. en webservice.

8.1 Datastruktur

Informationerne deles op i tre tabeller.

- Person: Indeholder persondata så som fornavn og efternavn
- Adresse: Indeholder adresser samt fremmednøgler til rækker i persontabellen.
- Telefonnummer: indeholder telefonnummer samt fremmednøgle til rækker i persontabellen.

Persontabellen og adresse tabellen tænkes placeret på samme server, mens telefonnummer placeres på en anden server. De tre tabeller ligger i virkeligheden på samme server, men adgang til telefonnummer tabellen sker kun igennem det eksterne



Figur 30: Datastruktur for testimplemetering. Adresser og telefonnumre er relateret til personnumre ved at indeholde fremmednøgler der peger på rækker i personer.

```
<!-- DataStructure  -->
<Table name="Personer" id="c7e1d7da-a8e5-48e0-886a-748427f18ce8" description="Personer i virksomheden">
    <Fields>
        <Guid name="personId" id="35238b1a-6e3a-445c-a2f6-2f83783aa025" defaultValue="newid()" isKey="true"/>
        <String name="Fornavn" id="36e2cafa-5f2d-4a2c-bd9b-c655f1a7e3eb" description="Fornavn" required="true"/>
        <String name="Efternavn" id="e423814c-5336-4ec9-b71c-a1a36eabc5a4" defaultValue="String" required="true"/>
        <String name="Personnummer" id="5c3578c5-729c-430d-8f9e-b442af6730c5" required="false"/>
    </Fields>
</Table>
<Table name="Adresse" id="7f7820ec-d206-4fd6-897b-e8f80a38dfb4" description="Adresse som personen er tilknyttet">
    <Fields>
        <Guid name="adressseId" defaultValue="newid()" id="acaa4145-b278-4edd-aae4-2b38bb36e321" isKey="true"/>
        <ForeignKey id="31b8c024-e705-4ab7-b65c-731cd8202ec0" name="personId" refId="35238b1a-6e3a-445c-a2f6-
2f83783aa025" required="true"/>
        <String name="Adresse1" id="aaae60dc-0de5-422f-ae9c-936dd0fbddea" description="Første linie i adressen"/>
        <String name="Adresse2" id="0611a904-1edf-4f53-99c7-87adf4684314" description="String" defaultValue="String"
required="true"/>
        <Number name="Postnummer" id="8c3f52e9-946a-49a7-afb4-06700e374d46" description="Postnummer"/>
    </Fields>
</Table>
<Table name="Telefon" id="1a00dcdd-f6de-4fbf-90c3-0276054a9db4" description="String">
    <Fields>
        <Guid name="telefonId" id="14876ba7-ef9d-4253-950b-f7686a98e6dc" defaultValue="newId()" isKey="true" />
        <ForeignKey name="personId" refId="35238b1a-6e3a-445c-a2f6-2f83783aa025" required="true" id="333b6a03-ee5a-
4447-97de-0b691aa5ba24"/>
        <String name="telefonnummer" id="603a4026-e3f7-4e40-8511-6028796a1f22" required="true" />
    </Fields>
</Table>
```

Figur 31: Datadefinition

I Figur 31 ovenfor er definitionen på de tre tabeller. Det ses hvordan indholdet i tabellerne valideres ved at definere forskellige felttyper og angive ekstra parametre. isKey parametren betyder at kolonnen er primærnøgle for tabellen. defaultValue angiver den værdi, felter antager hvis det ikke bliver sat af brugeren.

8.1.1 Ekstern grænseflade

Den eksterne grænseflade skal give adgang til telefonnummer tabellen.

```
<?xml version="1.0" encoding="UTF-8"?>
<Interface>
    <Table name="Telefon">
        <Field>personId</Field>
        <Field>telefonnummer</Field>
    </Table>
</Interface>
```

Figur 32: Definition af ekstern grænseflade

Den eksterne grænseflade som vist på Figur 32 definerer at der kan tilgås en tabel med navnet Telefon. Tabellen består af 2 felter, personId og telefonnummer.

```
<!-- define queries -->
    <View name="personMedAdresse" id="da094b66-68be-4862-93e9-47ee2fdee98f" ">
        <table idref="c7e1d7da-a8e5-48e0-886a-748427f18ce8" alias="person">
            <Field id="35238b1a-6e3a-445c-a2f6-2f83783aa025" name="pk" visible="false" sortorder="6"/>
            <Field id="36e2cafa-5f2d-4a2c-bd9b-c655f1a7e3eb" name="fornavn" visible="true" sortorder="1"/>
            <Field id="e423814c-5336-4ec9-b71c-a1a36eabc5a4" name="efternavn" sortorder="2" visible="true"/>
        </table>
        <table idref="7f7820ec-d206-4fd6-897b-e8f80a38dfb4" alias="adresse">
            <Field id="31b8c024-e705-4ab7-b65c-731cd8202ec0" name="fk" sortorder="5" visible="false"/>
            <Field id="aaae60dc-0de5-422f-ae9c-936dd0fbddea" name="adresse1" sortorder="3" visible="true"/>
            <Field id="0611a904-1edf-4f53-99c7-87adf4684314" name="adresse2" sortorder="4" visible="true"/>
        </table>
        <join>
            <alias required="true" nameref="pk"/>
            <alias required="true" nameref="fk"/>
        </join>
        <Requests>
            <Request id="19076c8c-5dd8-4584-9b58-9922c8153da9">
                <searchList operator="and">
                    <searchList operator="or">
                        <searchField nameref="fornavn" type="startswith" name="fornavn"/>
                        <searchField nameref="fornavn" type="startswith" name="alternativtfornavn"/>
                    </searchList>
                    <searchList operator="or">
                        <searchField nameref="efternavn" type="startswith" name="efternavn1"/>
                        <searchField nameref="efternavn" type="startswith" name="efternavn2"/>
                    </searchList>
                </searchList>
            </Request>
        </Requests>
    </View>
    <!-- følgende skal bruges til at trække data til den eksterne grænseflade -->
    <View name="telefonnumre" id="7729b063-14ab-4d7d-a6a5-fb73e6c59d8d" description="telefonnumrene">
        <table idref="1a00dcdd-f6de-4fbf-90c3-0276054a9db4" alias="telefonnumre">
            <Field id="333b6a03-ee5a-4447-97de-0b691aa5ba24" name="personId" sortorder="1"/>
            <Field id="603a4026-e3f7-4e40-8511-6028796a1f22" name="telefonnummer" sortorder="2"/>
        </table>
        <Requests>
            <Request id="30f293e5-cd44-4992-a2b8-deea48a31c98">
                <searchList>
                    <searchField nameref="telefonnummer" type="startswith" name="telefonnummer"/>
                </searchList>
            </Request>
        </Requests>
    </View>
    <View name="personMedTelefon" id="726891e2-4c43-4817-a904-094a479107d5" description="eksterne data ">
        <table idref="c7e1d7da-a8e5-48e0-886a-748427f18ce8" alias="Person">
            <Field id="35238b1a-6e3a-445c-a2f6-2f83783aa025" name="personId" sortorder="1" visible="false"/>
            <Field id="36e2cafa-5f2d-4a2c-bd9b-c655f1a7e3eb" name="Fornavn" sortorder="2" visible="true"/>
            <Field id="e423814c-5336-4ec9-b71c-a1a36eabc5a4" name="Efternavn" sortorder="3" visible="true"/>
        </table>
        <Requests>
            <Request id="7b055fd9-f2bc-4a14-95ca-c7bd27471f72">
                <searchList operator="or">
                    <searchField nameref="Fornavn" type="startswith" name="Fornavn1"/>
                    <searchField nameref="Fornavn" type="startswith" name="Fornavn2"/>
                </searchList>
            </Request>
        </Requests>
        <ExternalTable externalId="personId" internalId="personId" href="http://localhost/Runex/Service1.asmx">
            <Field name="pId" sortorder="4">personId</Field>
            <Field name="telefonnummer" sortorder="5" visible="true">telefonnummer</Field>
        </ExternalTable>
    </View>
```

Figur 33: Definition af udtræk fra databasen

På Figur 33 ses definitioner på views i testimplementationen. Hvert view er defineret i sin egen træstruktur.

personMedAdresse	• kombinerer data fra person og adresse tabellen. • Joinet på personId • Søgning på : (fornavn eller fornavn) og (efternavn eller efternavn)
telefonnumre	• Hiver data ud fra telefonnummer tabellen • Søgning på: telefonnummer
personMedTelefon	• Kombinerer persondata med telefonnummer hentet fra ekstern kilde. • Søgning på: fornavn eller fornavn

Figur 34: Udtræk

For at den eksterne grænseflade kan levere data, skal der også laves en mapning imellem den interne og den eksterne struktur. For at simplificere tingene en smule er den eksterne grænseflade afpasset efter en interne. Den eksterne grænseflade beskriver en tabel der hedder Telefon. Tabellen indeholder to felter: personId og telefonnummer. Internt er der et View der indeholde samme felter. Figur 35 viser hvorledes mapningen ser ud.

```
<?xml version="1.0" encoding="UTF-8"?>
<Mapping xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\rex\mapping.xsd">
    <TableMapping nameRef="Telefon">
        <View nameRef="7729b063-14ab-4d7d-a6a5-fb73e6c59d8d">
            <Field externalRef="personId" internalRef="personId"/>
            <Field externalRef="telefonnummer" internalRef="telefonnummer"/>
        </View>
    </TableMapping>
</Mapping>
```

Figur 35: Mapning til eksternt grænseflade

Nu er selve datastrukturen på plads. Tabellerne er defineret. Måder at tække data ud af systemet er defineret. Eksterne grænseflader er defineret og de er mapped til den interne datastruktur.

8.2 Skærmbilleder

Skærmbillederne er defineret, så de giver mulighed for at teste slutbrugersystemernes muligheder.

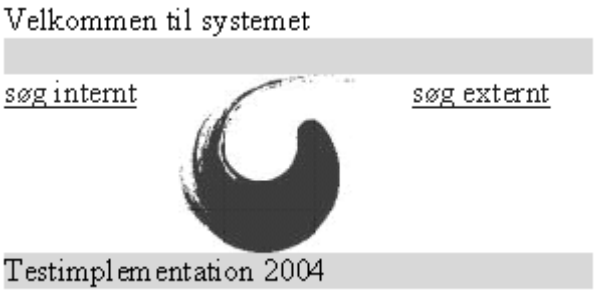
Opstart	Velkomst skærmbillede med mulighed for at skifte til skærmbillede til to andre skærmbilleder. Se Figur 37. Skærmbilledet bruger direkte links til navigering. Der vises både tekst og billede
Søg internt	Søge formular der muliggør søgninger på personer og adresser. Se Figur 38. Navigering sker ved at submitte søgeformularen til serveren eller ved at klikke på linket til forsiden.
Adresseliste	Viser en liste af personer og disses adresser. Se Figur 39
Søg eksternt	Søge formular der muliggør søgning på personer og telefonnumre. Se Figur 40.
Telefonliste	Viser en liste af personer og disses telefonnumre. Se Figur 41

Figur 36 viser selve definitionen af skærmbillederne i systemet. Hvert skærmbillede er et element i strukturen. Læg mærke til at et skærmbillede godt kan indeholde et andet skærmbillede. Hvis et skærmbillede ikke har et id, kan der ikke refereres til det, og det kan ikke indgå som et selvstændigt skærmbillede.

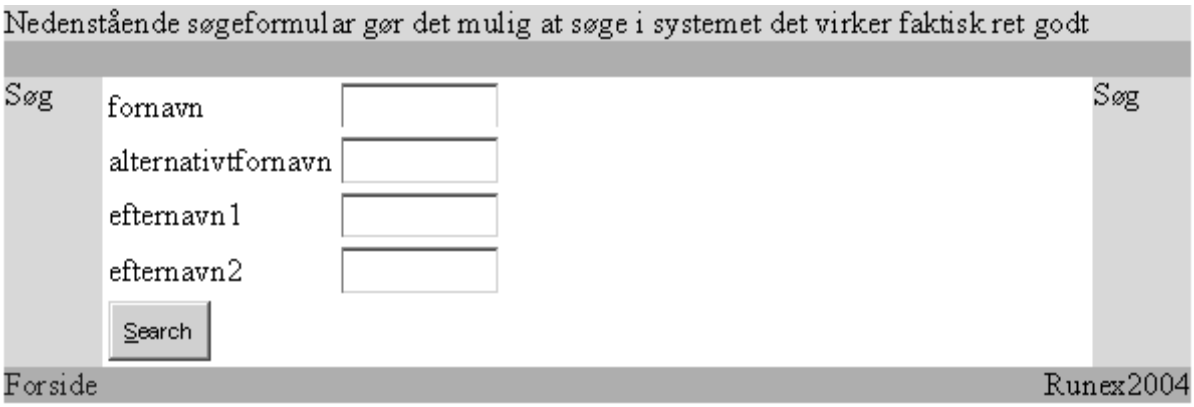
```
<Screens defaultScreen="db3d522d-b119-448b-8a5d-dcc7439f2c0a">
    <Screen id="db3d522d-b119-448b-8a5d-dcc7439f2c0a" bgcolor="#e0e0e0" name="opstart">
        <Text value="Velkommen til systemet"/>
        <Empty bgcolor="#e0d0f0" heigth="400" width="300"/>
        <Screen split="vertical">
            <Href target="4813e3f8-f929-473e-a230-a7da117d967a" value="søg internt"/>
            <Image src="c:\rex\logo2.jpg" type="URL"/>
            <Href target="d9828c7b-cf21-4dcc-88ff-6da14840e856" value="søg externt"/>
        </Screen>
        <Text value="Testimplementation 2004" bgcolor="#e0d0f0"/>
    </Screen>
    <Screen id="d9828c7b-cf21-4dcc-88ff-6da14840e856">
        <Text value="En søgning med eksterne systemer"/>
        <SearchForm request="7b055fd9-f2bc-4a14-95ca-c7bd27471f72" target="66c79246-ef32-477d-8a23-
95836c432f61"/>
        <Href value="Forside" target="db3d522d-b119-448b-8a5d-dcc7439f2c0a"/>
    </Screen>
    <Screen id="66c79246-ef32-477d-8a23-95836c432f61" name="Liste med eksternt data">
        <Text value="Telefonliste"/>
        <List source="726891e2-4c43-4817-a904-094a479107d5" target="fd663f30-b389-4fda-8aa7-
468288d0216d" bgcolor="#d0d0d0"/>
        <Href value="Forside" target="db3d522d-b119-448b-8a5d-dcc7439f2c0a"/>
        <Href value="Søg igen" target="d9828c7b-cf21-4dcc-88ff-6da14840e856"/>
    </Screen>
    <Screen id="4813e3f8-f929-473e-a230-a7da117d967a">
        <Text value="Nedenstående søgeformular gør det mulig at søge i systemet det virker faktisk ret godt"
bgcolor="#e0d0f0"/>
        <Empty bgcolor="#aeaeae" heigth="20"/>
        <Screen split="vertical">
            <Text value="Søg" bgcolor="#e0d0f0" width="50"/>
            <SearchForm request="19076c8c-5dd8-4584-9b58-9922c8153da9" target="bbeb0ee1-4134-4349-
8179-26b70054c3f0"/>
            <Text value="Søg" bgcolor="#e0d0f0" width="50"/>
        </Screen>
        <Screen split="vertical" bgcolor="#aeaeae">
            <Href value="Forside" target="db3d522d-b119-448b-8a5d-dcc7439f2c0a"/>
            <Empty/>
            <Text value="Runex2004" width="50"/>
        </Screen>
    </Screen>
    <Screen id="bbeb0ee1-4134-4349-8179-26b70054c3f0">
        <Text value="Liste af personer og deres adresser" bgcolor="#f0e0d0"/>
        <List source="da094b66-68be-4862-93e9-47ee2fdee98f" target="fd663f30-b389-4fda-8aa7-
468288d0216d" bgcolor="#d0d0d0"/>
        <Image src="c:\rex\runexlogo.jpg" type="URL"/>
        <Screen split="vertical" bgcolor="#eeddff">
            <Href target="4813e3f8-f929-473e-a230-a7da117d967a" value="Ny søgning"/>
            <Href target="db3d522d-b119-448b-8a5d-dcc7439f2c0a" value="Forside"/>
        </Screen>
    </Screen>
    <Screen id="fd663f30-b389-4fda-8aa7-468288d0216d">
        <Text value="Detail Data"/>
        <Form source="726891e2-4c43-4817-a904-094a479107d5" type="view"/>
        <Href target="4813e3f8-f929-473e-a230-a7da117d967a" value="til start"/>
    </Screen>
</Screens>
```

Figur 36: Definition af brugergrænseflader

Som en attribut til elementet Screens er angivet en optartsskærm som brugeren ser når han starter systemet op. Det første skærmbillede viser en liste af muligheder for at navigere videre. Figur 37, Figur 38, Figur 39, Figur 40 og Figur 41 viser skærmbillederne som de bliver vist for slutbrugeren.

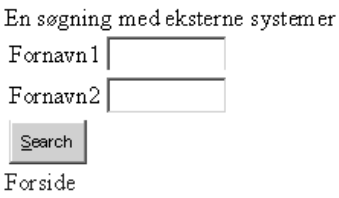


Figur 37: Skærmbillede opstart

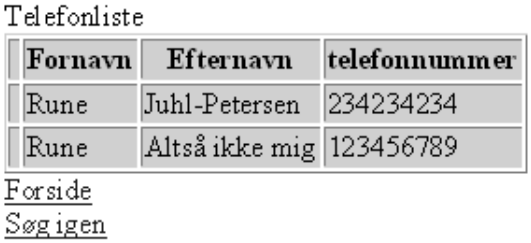


Figur 38:skærmbillede: Søg internt

Figur 39: Skærmbillede, Adresseliste



Figur 40: Skærmbillede, Søg eksternt



Figur 41: Skærmbillede, Telefonliste

8.3 Iterationer

Prototypen er blevet udviklet iterativt. Både grænsefladerne og implementeringen har udviklet sig efterhånden som nye problemer dukkede op.

Oprettelse af datastruktur i databasen	Her blev objektmodellen for felter, tabeller og views defineret. For hver tabel i konfigurationsfilen blev der lavet et tabel objekt der selv kunne læse strukturen ud fra konfigurationen. Ligeledes med felter og views. Objekterne kunne generere den SQL der kunne oprette strukturen i databasen.
Søgning på data i databasen	Den allerede oprettede objektstruktur fra første prototype blev udvidet lidt. Ud over at kunne generere SQL til oprettelse af datastrukturen, kunne den nu også oprette SQL til søgning i datastrukturen.
Brugergrænseflade	Der blev først lavet en brugergrænseflade der kunne genereres ud fra konfigurations filen. Senere kom der også data fra databasen ind i brugegrænsefladen. Til sidst blev navigationen tilføjet.
Implementere ekstern grænseflade	Der blev lavet en webservice. Webservicen oprettede objektstrukturen for slutbrugersystemet, og brugte derefter selv konfigurationen for den ekstern grænseflade og mapningen til at kunne udveksle data.
Bruge eksterne resourcer	Denne del blev mest brugt til at teste den eksterne grænseflade.
Joine eksterne ressourcer.	Her skulle view klassen udvides. Ud over at kunne joine data fra databasen, skulle der også joines med ekstern data. Det er implementeret vha. .Net's DataSet objekt. Et DataSet er et objekt der indeholder et udtræk fra en database. DataSet har funktionalitet til at join sit indhold fra andre DataSets.

8.4 Brugervejledning

Selvom udviklingsprocessen er blevet simplificeret, er den ikke blevet helt fejlsikret. Udvikleren har stadigt mulighed for at lave uhensigtsmæssige datastrukturer og arbejdsgange, og derved gøre slutbrugerproduktet ubrugeligt.

Systemets service orienterede arkitektur er en funktionalitetsmæssig gevinst. Rent performancemæssigt er det ikke særligt godt design, da det potentielt kan betyde kommunikation imellem en masse systemer på netværket. Slutbrugeren vil først få svar når alle systemer involveret i kommunikationen har leveret den information der er blevet forespurgt. Hvis systemerne er kædet sammen i flere niveauer, er det klart, at vente tiden bliver endnu længere.

At skulle udveksle data imellem flere slutbrugeresystemer, imens at slutbrugeren venter på et resultat er kritisk. Dette især hvis der bliver overført data slutbrugeren ikke har bedt om. Dette kan forekomme hvis der bliver joinet på data og selve kriterierne er relationen imellem de to tabeller. For at lave et resultat bliver slutbrugersystemet også nødt til at hente hele den eksterne tabel og derefter filtrere alle de uønskede rækker fra. Det er derfor meget vigtigt at udvikleren er opmærksom på data flowet og prøver at undgå at overføre for meget unødvendigt data.

9 Konklusion

Det er lykkedes at fremstille et system, der beviser muligheden i at simplificere udviklingsproccesen for software. Dette system vil i en videreudviklet udgave kunne erstatte en udvikler på mange trivielle opgaver, og derved kan udvikleren koncentrere sig om mere udfordrende opgaver.

Systemet er generisk og kan klare de fleste datastrukturer. Der er dog grænser for hvor omfattende det kan bliver og hvor mange opgaver det kan antage, uden at det går ud over simpliciteten.

Systemet skal i sin ideelle udformning kunne implementere alt, og samtidigt begrænse udviklerens valg under konfigurationen. Disse to ting kan ikke forenes, og der skal findes et kompromis der giver det optimale resultat. Kompromiset er resultatet af dette projekt!

Systemets fremtidige brugere kan være virksomheder, der gerne vil have struktur og ensartethed i deres computersystemer.

10 Litteraturliste

Walmsley, Priscilla:	Definitive XML Schema
Chaundri:	XML Data Management
Eriksson	UML Toolkit
Troelsen, Andrew	C# and the .Net Platform
Harold & Means	XML in a Nutshell, A Desktop Quick Reference
Sceppa, David	Microsoft ADO.Net
Gamma & Helm & Johnson & Vlissides	Design patterns, Elements of Reusable Object-Oriented Software
Fowler, Martin	Patterns of Enterprise Application Architecture
Adams, Douglas	The Hitchhikers Guide to the Galaxy.

11 Apendix - Tidsplan

Tidsplan for eksamensprojekt af Rune Juhl-Petersen

Projekt start 5. januar 2004.
Projekt afsluttes 2. august 2004 .

Projektet har et forløb på 31 uger

Opdeling af opgaver.

11.1 Kravdefinering

Dette er et afsnit der beskriver hvilke forventninger jeg personligt har til projektet, samt hvad andre forventer af det. Forsvarsstaben ønsker projektet udført som et pilotprojekt med produktet værende en prototype for et fremtidigt produkt. Projektet skal derfor også bruges til at frembringe ideer der ikke nødvendigvis skal implementeres i projektets levetid.

- Kort beskrivelse af hvad er det egentligt Forsvarsstaben ønsker af dette produkt.
- Hvilke krav stilles der for at dette produkt kan bruges til det ønskede formål.
- Beskrivelse af hvad der logisk og pragmatisk ligger til grund for projektet.
- Footprint for rapport

11.2 Grænseflader

Systemet går basalt set ud på at definere nogle grænseflader til at kommunikere data imellem systemer. Grænsefladerne vil udvikle sig igennem forløbet og tilpasse sig til hvad implementeringen stiller af krav eller ønsker.

- Overordnet design
- Hvordan tilgås systemet.
- Hvilke grænseflader er nødvendige og hvorfor

11.3 Administration

Der vil i givet fald være noget administration til systemet. Hvad der skal administreres vil fremgå af projektets forrige faser.

- Administrations modul hvor systemet kan sættes op. Hvad der skal kunne sættes op må fremgå af kravdefineringen.

11.4 Brugergrænseflade

Det er essentielt for systemet at det har en brugergrænseflade. Brugergrænsefladen skal bygge oven på de grænseflader der er beskrevet. I projektet vil brugergrænsefladen være implementeret i html rettet mod Internet Explorer 6.0

- En brugergrænseflade

11.5 Oprettelse af datastruktur ud fra definition

Ud fra en xml definition skal der gøres klart til at data kan håndteres.

- Der skal fysisk kunne oprettes plads i systemet til at håndtere datastrukturen.
- Oprettelse af grænseflader der understøtter den givne datastruktur.

11.6 Udtræk af data

Specifikationen for grænseflade vedr. udtræk af data skal implementeres. Søgninger såvel som udtræk baseret på kombinationer af data skal implementeres.

- Udtræk fra datastrukturen skal implementeres.
- Tilgang til eksterne systemer.

11.7 Opdatering/ import af data

Systemet skal kunne modtage data fra såvel brugeren som andre systemer. Hvis denne funktionalitet ikke bliver integreret i et af de andre punkter skal det håndteres separat.

- Data skal importeres

Tidsforbrug i uger

Kravdefinering	4
grænseflader	5
administration	3
brugergrænseflader	3
udtræk	5
import	3
eksterne systemer	3
	26

12 Appendix – Systemkonfiguration

XSD definitionen på hvordan systemkonfigurationsfilerne skal være struktureret:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:element name="Rex">
        <xs:annotation>
            <xs:documentation>Definering af system til databehandling</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Structure">
                    <xs:annotation>
                        <xs:documentation>Datastruktur</xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                        <xs:sequence>
                            <xs:sequence maxOccurs="unbounded">
                                <xs:element name="Table">
                                    <xs:annotation>
                                        <xs:documentation>Definition af tabel</xs:documentation>
                                    </xs:annotation>
                                    <xs:complexType>
                                        <xs:complexContent>
                                            <xs:extension base="DataField">
                                                <xs:sequence>
                                                    <xs:element name="Fields">
                                                        <xs:annotation>
                                                            <xs:documentation>Felt i tabellen. Herunder identifikation
således at der kan refereres til denne andetsteds fra i strukturen.</xs:documentation>
                                                        </xs:annotation>
                                                        <xs:complexType>
                                                            <xs:sequence maxOccurs="unbounded">
                                                                <xs:choice>
                                                                    <xs:element name="Number" type="DataField">
                                                                        <xs:annotation>
                                                                            <xs:documentation>Numerisk
værdi</xs:documentation>
                                                                        </xs:annotation>
                                                                    </xs:element>
                                                                    <xs:element name="String" type="DataField">
                                                                        <xs:annotation>

    <xs:documentation>Tekst</xs:documentation>
                                                                        </xs:annotation>
                                                                    </xs:element>
                                                                    <xs:element name="Guid">
                                                                        <xs:annotation>
                                                                            <xs:documentation>Global Unique
Identifier</xs:documentation>
                                                                        </xs:annotation>
                                                                        <xs:complexType>
                                                                            <xs:complexContent>
                                                                                <xs:extension base="DataField">
                                                                                    <xs:attribute name="isKey"
type="xs:boolean" use="optional"/>
                                                                                </xs:extension>
                                                                            </xs:complexContent>
                                                                        </xs:complexType>
                                                                    </xs:element>
                                                                    <xs:element name="Enumeration">
                                                                        <xs:annotation>
```

```
                                        <xs:documentation>Liste af mulige
værdier</xs:documentation>
                                    </xs:annotation>
                                    <xs:complexType>
                                        <xs:complexContent>
                                            <xs:extension base="DataField">
                                                <xs:sequence
maxOccurs="unbounded">
                                                    <xs:element
name="Enumerations"/>
                                                </xs:sequence>
                                            </xs:extension>
                                        </xs:complexContent>
                                    </xs:complexType>
                                </xs:element>
                                <xs:element name="ForeignKey">
                                    <xs:annotation>
                                        <xs:documentation>Fremmednøgle (måske
unødvendig)</xs:documentation>
                                    </xs:annotation>
                                    <xs:complexType>
                                        <xs:complexContent>
                                            <xs:extension base="DataField">
                                                <xs:attribute name="refId"
type="guid" use="required"/>
                                            </xs:extension>
                                        </xs:complexContent>
                                    </xs:complexType>
                                </xs:element>
                                <xs:element name="Email">
                                    <xs:annotation>
                                        <xs:documentation>E-mail
adresse</xs:documentation>
                                    </xs:annotation>
                                    <xs:simpleType>
                                        <xs:restriction base="xs:string">
                                            <xs:pattern value="[a-z]*"/>
                                        </xs:restriction>
                                    </xs:simpleType>
                                </xs:element>
                                <xs:element name="Password" type="DataField">
                                    <xs:annotation>
                                        <xs:documentation>Password. Password
feltet kan kun sammenlignes med.  Inholdet kan ikke hives ud af databasen igen.</xs:documentation>
                                    </xs:annotation>
                                </xs:element>
                            </xs:choice>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:unique name="FieldName">
    <xs:selector xpath="Fields/*"/>
    <xs:field xpath="@name"/>
</xs:unique>
</xs:element>
</xs:sequence>
<xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:element name="View">
        <xs:annotation>
            <xs:documentation>Logisk abstraktion oven på tabel
strukturen</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
```

```
                                    <xs:extension base="DataField">
                                        <xs:sequence>
                                            <xs:sequence maxOccurs="unbounded">
                                                <xs:element name="table">
                                                    <xs:annotation>
                                                        <xs:documentation>Reference til tabel der skal
udtrækkes fra</xs:documentation>
                                                    </xs:annotation>
                                                    <xs:complexType>
                                                        <xs:sequence maxOccurs="unbounded">
                                                            <xs:element name="Field">
                                                                <xs:annotation>
                                                                    <xs:documentation>Reference til felt der
skal medtages i udtræk</xs:documentation>
                                                                </xs:annotation>
                                                                <xs:complexType>
                                                                    <xs:attribute name="id" type="guid"
use="required"/>
                                                                    <xs:attribute name="name" type="xs:string"
use="required"/>
                                                                    <xs:attribute name="visible"
type="xs:boolean" use="optional" default="true"/>
                                                                    <xs:attribute name="sortorder" type="xs:int"
use="required"/>
                                                                </xs:complexType>
                                                            </xs:element>
                                                        </xs:sequence>
                                                        <xs:attribute name="idref" type="guid" use="required"/>
                                                        <xs:attribute name="alias" type="xs:string"
use="required"/>
                                                    </xs:complexType>
                                                    <xs:keyref name="fieldname" refer="FieldId">
                                                        <xs:selector xpath="Field"/>
                                                        <xs:field xpath="@id"/>
                                                    </xs:keyref>
                                                </xs:element>
                                            </xs:sequence>
                                            <xs:sequence minOccurs="0" maxOccurs="unbounded">
                                                <xs:element name="join">
                                                    <xs:annotation>
                                                        <xs:documentation>Beskriver hvordan tabellerne skal
kombineres</xs:documentation>
                                                    </xs:annotation>
                                                    <xs:complexType>
                                                        <xs:sequence minOccurs="2" maxOccurs="2">
                                                            <xs:element name="alias">
                                                                <xs:annotation>
                                                                    <xs:documentation>Reference til et felt der
er udvalgt i viewet</xs:documentation>
                                                                </xs:annotation>
                                                                <xs:complexType>
                                                                    <xs:attribute name="required"
type="xs:boolean" use="required"/>
                                                                    <xs:attribute name="nameref"
type="xs:string" use="required"/>
                                                                </xs:complexType>
                                                            </xs:element>
                                                        </xs:sequence>
                                                    </xs:complexType>
                                                </xs:element>
                                            </xs:sequence>
                                            <xs:sequence>
                                                <xs:element name="Requests" minOccurs="0">
                                                    <xs:annotation>
                                                        <xs:documentation>Indeholder en liste af måder at
filtrere på data</xs:documentation>
                                                    </xs:annotation>
                                                    <xs:complexType>
```

```
                                                    <xs:sequence maxOccurs="unbounded">
                                                        <xs:element name="Request">
                                                            <xs:annotation>
                                                                <xs:documentation>Angiver hvilke
parametre der skal bruges til at begrænse udtræk fra databasen.</xs:documentation>
                                                            </xs:annotation>
                                                            <xs:complexType>
                                                                <xs:sequence>
                                                                    <xs:element ref="searchList"/>
                                                                </xs:sequence>
                                                                <xs:attribute name="id" type="guid"
use="required"/>
                                                            </xs:complexType>
                                                            <xs:unique name="searchFieldName">
                                                                <xs:selector xpath=".//searchField"/>
                                                                <xs:field xpath="@name"/>
                                                            </xs:unique>
                                                        </xs:element>
                                                    </xs:sequence>
                                                </xs:complexType>
                                            </xs:element>
                                        </xs:sequence>
                                        <xs:element name="ExternalTable" minOccurs="0">
                                            <xs:annotation>
                                                <xs:documentation>Reference til eksternt data. I attributter
er defineret hvordan data relateres til eksisterende data.</xs:documentation>
                                            </xs:annotation>
                                            <xs:complexType>
                                                <xs:sequence maxOccurs="unbounded">
                                                    <xs:element name="Field">
                                                        <xs:annotation>
                                                            <xs:documentation>Udvælger og sorterer det
data der skal med i udtrækket.</xs:documentation>
                                                        </xs:annotation>
                                                        <xs:complexType>
                                                            <xs:simpleContent>
                                                                <xs:extension base="xs:string">
                                                                    <xs:attribute name="name"
type="xs:string" use="required"/>
                                                                    <xs:attribute name="visible"
type="xs:boolean" use="optional"/>
                                                                    <xs:attribute name="sortorder"
type="xs:int" use="required"/>
                                                                </xs:extension>
                                                            </xs:simpleContent>
                                                        </xs:complexType>
                                                    </xs:element>
                                                </xs:sequence>
                                                <xs:attribute name="externalId" type="xs:string"
use="required"/>
                                                <xs:attribute name="internalId" type="xs:string"
use="required"/>
                                                <xs:attribute name="href" type="xs:string" use="required"/>
                                            </xs:complexType>
                                        </xs:element>
                                    </xs:sequence>
                                </xs:extension>
                            </xs:complexContent>
                        </xs:complexType>
                        <xs:keyref name="tableid" refer="TableId">
                            <xs:selector xpath="table"/>
                            <xs:field xpath="@idref"/>
                        </xs:keyref>
                        <xs:unique name="tablename">
                            <xs:selector xpath="table"/>
                            <xs:field xpath="@name"/>
                        </xs:unique>
                        <xs:key name="aliasname">
```

```
                        <xs:selector xpath="table/Field|ExternalTable/Field"/>
                        <xs:field xpath="@name"/>
                    </xs:key>
                    <xs:keyref name="joinref" refer="aliasname">
                        <xs:selector xpath="join/alias"/>
                        <xs:field xpath="@nameref"/>
                    </xs:keyref>
                    <xs:keyref name="requestField" refer="aliasname">
                        <xs:selector xpath=".//searchField"/>
                        <xs:field xpath="@nameref"/>
                    </xs:keyref>
                    <xs:unique name="sorting">
                        <xs:selector xpath="table/Field|ExternalTable/Field"/>
                        <xs:field xpath="@sortorder"/>
                    </xs:unique>
                    <xs:keyref name="internalId" refer="aliasname">
                        <xs:selector xpath="ExternalTable"/>
                        <xs:field xpath="@internalId"/>
                    </xs:keyref>
                    <xs:key name="externalId">
                        <xs:selector xpath="ExternalTable/Field"/>
                        <xs:field xpath="."/>
                    </xs:key>
                    <xs:keyref name="externalRef" refer="externalId">
                        <xs:selector xpath="ExternalTable"/>
                        <xs:field xpath="@externalId"/>
                    </xs:keyref>
                </xs:element>
            </xs:sequence>
        </xs:sequence>
    </xs:complexType>
    <xs:key name="FieldId">
        <xs:selector xpath="Table/Fields/*"/>
        <xs:field xpath="@id"/>
    </xs:key>
    <xs:key name="TableId">
        <xs:selector xpath="*"/>
        <xs:field xpath="@id"/>
    </xs:key>
    <xs:unique name="TableName">
        <xs:selector xpath="*"/>
        <xs:field xpath="@name"/>
    </xs:unique>
    <xs:key name="PrimaryKey">
        <xs:selector xpath="Table/Fields/Guid"/>
        <xs:field xpath="@id"/>
    </xs:key>
    <xs:keyref name="ForeignKey" refer="PrimaryKey">
        <xs:selector xpath="Table/Fields/ForeignKey"/>
        <xs:field xpath="@refId"/>
    </xs:keyref>
</xs:element>
<xs:element name="Screens" minOccurs="0">
    <xs:annotation>
        <xs:documentation>Visualiseringen</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:sequence maxOccurs="unbounded">
            <xs:element ref="Screen" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="defaultScreen" type="guid" use="required"/>
    </xs:complexType>
    <xs:key name="screenId">
        <xs:selector xpath="Screen"/>
        <xs:field xpath="@id"/>
    </xs:key>
    <xs:keyref name="defaultScreen" refer="screenId">
        <xs:selector xpath="."/>
```

```
                    <xs:field xpath="@defaultScreen"/>
                </xs:keyref>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:key name="requestId">
        <xs:selector xpath="Structure/View/Requests/Request"/>
        <xs:field xpath="@id"/>
    </xs:key>
</xs:element>
<xs:complexType name="Item">
    <xs:annotation>
        <xs:documentation>Visible item</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
        <xs:extension base="IdentifiableObject">
            <xs:attribute name="heigth" type="xs:string" use="optional"/>
            <xs:attribute name="width" type="xs:string" use="optional"/>
            <xs:attribute name="bgcolor" type="xs:string" use="optional"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="DataField">
    <xs:annotation>
        <xs:documentation>general information about field</xs:documentation>
    </xs:annotation>
    <xs:attribute name="name" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:pattern value="\w*"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="id" type="guid" use="optional"/>
    <xs:attribute name="description" type="xs:string" use="optional"/>
    <xs:attribute name="defaultValue" type="xs:string" use="optional"/>
    <xs:attribute name="required" type="xs:boolean" use="optional"/>
</xs:complexType>
<xs:element name="Screen">
    <xs:annotation>
        <xs:documentation>Et skærmbillede</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="Item">
                <xs:sequence maxOccurs="unbounded">
                    <xs:choice>
                        <xs:element name="Text">
                            <xs:annotation>
                                <xs:documentation>Tekst der skal vises direkte på skærmen</xs:documentation>
                            </xs:annotation>
                            <xs:complexType>
                                <xs:complexContent>
                                    <xs:extension base="Item">
                                        <xs:attribute name="value" type="xs:string" use="required"/>
                                    </xs:extension>
                                </xs:complexContent>
                            </xs:complexType>
                        </xs:element>
                        <xs:element ref="Screen"/>
                        <xs:element name="Image">
                            <xs:annotation>
                                <xs:documentation>Billede der vises direkte på skærmen</xs:documentation>
                            </xs:annotation>
                            <xs:complexType>
                                <xs:complexContent>
                                    <xs:extension base="Item">
                                        <xs:attribute name="src" type="xs:string" use="required"/>
```

```
                                        <xs:attribute name="type" use="required">
                                            <xs:simpleType>
                                                <xs:restriction base="xs:string">
                                                    <xs:enumeration value="URL"/>
                                                    <xs:enumeration value="Intern"/>
                                                </xs:restriction>
                                            </xs:simpleType>
                                        </xs:attribute>
                                    </xs:extension>
                                </xs:complexContent>
                            </xs:complexType>
                        </xs:element>
                        <xs:element name="List">
                            <xs:annotation>
                                <xs:documentation>En liste af data</xs:documentation>
                            </xs:annotation>
                            <xs:complexType>
                                <xs:complexContent>
                                    <xs:extension base="Item">
                                        <xs:sequence>
                                            <xs:element name="selection" minOccurs="0">
                                                <xs:annotation>
                                                    <xs:documentation>Hvilke felter skal indgå i listen. Hvis der ikke
er angivet nogle, bliver alle felter medtaget.</xs:documentation>
                                                </xs:annotation>
                                                <xs:complexType>
                                                    <xs:attribute name="type" use="required">
                                                        <xs:simpleType>
                                                            <xs:restriction base="xs:string">
                                                                <xs:enumeration value="exclude"/>
                                                                <xs:enumeration value="include"/>
                                                            </xs:restriction>
                                                        </xs:simpleType>
                                                    </xs:attribute>
                                                </xs:complexType>
                                            </xs:element>
                                            <xs:element name="Navigation" minOccurs="0">
                                                <xs:annotation>
                                                    <xs:documentation>Parametre for hvorledes der navigeres
videre ud fra listens felter</xs:documentation>
                                                </xs:annotation>
                                                <xs:complexType>
                                                    <xs:sequence>
                                                        <xs:element name="Link">
                                                            <xs:annotation>
                                                                <xs:documentation>Link til videre navigation. Vil ofte
være </xs:documentation>
                                                            </xs:annotation>
                                                            <xs:complexType>
                                                                <xs:sequence maxOccurs="unbounded">
                                                                    <xs:element name="parameter">
                                                                        <xs:annotation>
                                                                            <xs:documentation>En parameter der
skal føres med over til næste skærmbillede.</xs:documentation>
                                                                        </xs:annotation>
                                                                        <xs:complexType>
                                                                            <xs:attribute name="name"
type="xs:string" use="required"/>
                                                                            <xs:attribute name="value"
type="xs:string" use="required"/>
                                                                            <xs:attribute name="field"
type="xs:string" use="required"/>
                                                                        </xs:complexType>
                                                                    </xs:element>
                                                                </xs:sequence>
                                                                <xs:attribute name="screen" type="xs:string"
use="required"/>
                                                            </xs:complexType>
```

```
                                        </xs:element>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                        <xs:attribute name="style" use="optional">
                            <xs:simpleType>
                                <xs:restriction base="xs:string">
                                    <xs:enumeration value="noHeader"/>
                                    <xs:enumeration value="default"/>
                                </xs:restriction>
                            </xs:simpleType>
                        </xs:attribute>
                        <xs:attribute name="source" type="guid" use="required"/>
                        <xs:attribute name="target" type="guid" use="optional"/>
                        <xs:attribute name="targetkey" type="xs:string" use="optional"/>
                    </xs:extension>
                </xs:complexContent>
            </xs:complexType>
            <xs:keyref name="viewId" refer="TableId">
                <xs:selector xpath="."/>
                <xs:field xpath="source"/>
            </xs:keyref>
        </xs:element>
        <xs:element name="Form">
            <xs:annotation>
                <xs:documentation>Lader brugeren taste informationer ind i
systemet</xs:documentation>
            </xs:annotation>
            <xs:complexType>
                <xs:complexContent>
                    <xs:extension base="Item">
                        <xs:attribute name="source" type="guid" use="required"/>
                        <xs:attribute name="target" type="xs:string" use="optional"/>
                        <xs:attribute name="type" use="required">
                            <xs:simpleType>
                                <xs:restriction base="xs:string">
                                    <xs:pattern value="search"/>
                                    <xs:pattern value="edit"/>
                                    <xs:pattern value="view"/>
                                    <xs:enumeration value="view"/>
                                    <xs:enumeration value="edit"/>
                                </xs:restriction>
                            </xs:simpleType>
                        </xs:attribute>
                    </xs:extension>
                </xs:complexContent>
            </xs:complexType>
        </xs:element>
        <xs:element name="SearchForm">
            <xs:annotation>
                <xs:documentation>Visualiserer en request. Derudover indeholder en searchForm
også en adresse til hvilken side søgningen skal resultere i.</xs:documentation>
            </xs:annotation>
            <xs:complexType>
                <xs:attribute name="request" type="guid" use="required"/>
                <xs:attribute name="target" type="guid" use="required"/>
            </xs:complexType>
        </xs:element>
        <xs:element name="Empty" type="Item">
            <xs:annotation>
                <xs:documentation>Element der ikke viser noget på skærmen. Eksisterer for at
kunne positionere andre visuelle elementer på skærmen.</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="Href">
            <xs:annotation>
                <xs:documentation>Navigering til andet skærmbillede</xs:documentation>
```

```
                                </xs:annotation>
                                <xs:complexType>
                                    <xs:complexContent>
                                        <xs:extension base="Item">
                                            <xs:attribute name="target" type="guid" use="required"/>
                                            <xs:attribute name="value" type="xs:string" use="required"/>
                                        </xs:extension>
                                    </xs:complexContent>
                                </xs:complexType>
                            </xs:element>
                        </xs:choice>
                    </xs:sequence>
                    <xs:attribute name="split" use="optional">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:enumeration value="horisontal"/>
                                <xs:enumeration value="vertical"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:attribute>
                    <xs:attribute name="name" type="xs:string" use="optional"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:keyref name="request" refer="requestId">
            <xs:selector xpath="SearchForm"/>
            <xs:field xpath="@request"/>
        </xs:keyref>
        <xs:keyref name="target" refer="screenId">
            <xs:selector xpath="*"/>
            <xs:field xpath="@target"/>
        </xs:keyref>
        <xs:keyref name="formsource" refer="TableId">
            <xs:selector xpath="Form"/>
            <xs:field xpath="@source"/>
        </xs:keyref>
    </xs:element>
    <xs:complexType name="IdentifiableObject">
        <xs:annotation>
            <xs:documentation>Et element der unikt kan refereres til </xs:documentation>
        </xs:annotation>
        <xs:attribute name="id" type="guid" use="optional"/>
    </xs:complexType>
    <xs:element name="searchField">
        <xs:annotation>
            <xs:documentation>er et felt i et view eller en tabel. Hvis requesten bruges på et view kan felter der ikke
bruges i viewet også bruges til søgningen hvis der refereres til tabellen direkte.</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:attribute name="nameref" type="xs:string" use="required"/>
            <xs:attribute name="type" use="required">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="equal"/>
                        <xs:enumeration value="lessthan"/>
                        <xs:enumeration value="greaterthan"/>
                        <xs:enumeration value="startswith"/>
                        <xs:enumeration value="endswith"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="name" type="xs:string" use="required"/>
        </xs:complexType>
    </xs:element>
    <xs:simpleType name="guid">
        <xs:restriction base="xs:string">
            <xs:pattern value="[\da-fA-F]{8}\-[\da-fA-F]{4}\-[\da-fA-F]{4}\-[\da-fA-F]{4}\-[\da-fA-F]{12}">
                <xs:annotation>
```

```
                    <xs:documentation source="sfsdfsadfsaf"/>
                </xs:annotation>
            </xs:pattern>
        </xs:restriction>
    </xs:simpleType>
    <xs:element name="searchList">
        <xs:annotation>
            <xs:documentation>Alle elementer and'es sammen. Alle elementer under or, er 'eller'. searchList er node i
træet (Composite Pattern)</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence maxOccurs="unbounded">
                <xs:choice>
                    <xs:element ref="searchField"/>
                    <xs:element ref="searchList"/>
                </xs:choice>
            </xs:sequence>
            <xs:attribute name="operator">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="and"/>
                        <xs:enumeration value="or"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

13 Appendix – struktur for konfigurering af ekstern grænseflade

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:element name="Interface">
        <xs:annotation>
            <xs:documentation>En ekstern grænseflade</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence maxOccurs="unbounded">
                <xs:element name="Table">
                    <xs:annotation>
                        <xs:documentation>udtræk af data</xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                        <xs:sequence>
                            <xs:sequence maxOccurs="unbounded">
                                <xs:element name="Field">
                                    <xs:annotation>
                                        <xs:documentation>Felt der skal udtrækkes</xs:documentation>
                                    </xs:annotation>
                                </xs:element>
                            </xs:sequence>
                            <xs:sequence minOccurs="0" maxOccurs="unbounded">
                                <xs:element name="Filter">
                                    <xs:complexType>
                                        <xs:sequence maxOccurs="unbounded">
                                            <xs:element name="Parameter"/>
                                        </xs:sequence>
                                        <xs:attribute name="name" type="xs:string" use="required"/>
                                    </xs:complexType>
                                    <xs:unique name="param">
                                        <xs:selector xpath="Parameter"/>
                                        <xs:field xpath="."/>
                                    </xs:unique>
                                </xs:element>
                            </xs:sequence>
                        </xs:sequence>
                        <xs:attribute name="name" type="xs:string" use="required"/>
                    </xs:complexType>
                    <xs:unique name="fieldName">
                        <xs:selector xpath="Field"/>
                        <xs:field xpath="."/>
                    </xs:unique>
                    <xs:unique name="filterName">
                        <xs:selector xpath="Filter"/>
                        <xs:field xpath="@name"/>
                    </xs:unique>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
        <xs:unique name="tableName">
            <xs:selector xpath="Table"/>
            <xs:field xpath="@name"/>
        </xs:unique>
    </xs:element>
</xs:schema>
```

14 Appendix – Struktur for mapning af ekstern grænseflade

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:element name="Mapping">
        <xs:complexType>
            <xs:sequence maxOccurs="unbounded">
                <xs:element name="TableMapping">
                    <xs:annotation>
                        <xs:documentation>Describes how a table in the external interface is mapped to the internal
structure</xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                        <xs:sequence>
                            <xs:sequence>
                                <xs:element name="View">
                                    <xs:annotation>
                                        <xs:documentation>Peger på et view i den interne
datastruktur</xs:documentation>
                                    </xs:annotation>
                                    <xs:complexType>
                                        <xs:sequence maxOccurs="unbounded">
                                            <xs:element name="Field">
                                                <xs:annotation>
                                                    <xs:documentation>Felt i det eksterne grænseflade og i den interne
struktur</xs:documentation>
                                                </xs:annotation>
                                                <xs:complexType>
                                                    <xs:attribute name="externalRef" type="xs:string" use="required"/>
                                                    <xs:attribute name="internalRef" type="xs:string" use="required"/>
                                                </xs:complexType>
                                            </xs:element>
                                        </xs:sequence>
                                        <xs:attribute name="nameRef"/>
                                    </xs:complexType>
                                </xs:element>
                            </xs:sequence>
                            <xs:sequence minOccurs="0" maxOccurs="unbounded">
                                <xs:element name="Join">
                                    <xs:complexType>
                                        <xs:sequence minOccurs="2" maxOccurs="2">
                                            <xs:element name="View">
                                                <xs:complexType>
                                                    <xs:attribute name="viewRef" type="xs:string" use="required"/>
                                                    <xs:attribute name="fieldRef" type="xs:string" use="required"/>
                                                </xs:complexType>
                                            </xs:element>
                                        </xs:sequence>
                                    </xs:complexType>
                                </xs:element>
                            </xs:sequence>
                            <xs:sequence minOccurs="0" maxOccurs="unbounded">
                                <xs:element name="Filter">
                                    <xs:complexType>
                                        <xs:sequence maxOccurs="unbounded">
                                            <xs:element name="Parameter">
                                                <xs:complexType>
                                                    <xs:attribute name="internalRef" type="xs:string" use="required"/>
                                                    <xs:attribute name="externalRef" type="xs:string" use="required"/>
                                                </xs:complexType>
                                            </xs:element>
                                        </xs:sequence>
                                        <xs:attribute name="nameRef"/>
                                        <xs:attribute name="requestRef" type="xs:string" use="required"/>
```

```
                                    </xs:complexType>
                                </xs:element>
                            </xs:sequence>
                        </xs:sequence>
                        <xs:attribute name="nameRef"/>
                    </xs:complexType>
                    <xs:key name="viewId">
                        <xs:selector xpath="View"/>
                        <xs:field xpath="@nameRef"/>
                    </xs:key>
                    <xs:keyref name="viewRef" refer="viewId">
                        <xs:selector xpath="Join/View"/>
                        <xs:field xpath="@viewRef"/>
                    </xs:keyref>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

15 Appendix - Grænseflade til kommunikation af data

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:element name="Request">
        <xs:annotation>
            <xs:documentation>Comment describing your root element</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence maxOccurs="unbounded">
                <xs:element name="Get">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="Table"/>
                            <xs:element name="Filter" minOccurs="0">
                                <xs:complexType>
                                    <xs:sequence maxOccurs="unbounded">
                                        <xs:element name="Parameter">
                                            <xs:complexType>
                                                <xs:simpleContent>
                                                    <xs:extension base="xs:string">
                                                        <xs:attribute name="nameRef" type="xs:string" use="required"/>
                                                    </xs:extension>
                                                </xs:simpleContent>
                                            </xs:complexType>
                                        </xs:element>
                                    </xs:sequence>
                                    <xs:attribute name="nameRef"/>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

16 Apendix – generering af brugergrænseflade

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <xsl:param name="ScreenId">66c79246-ef32-477d-8a23-95836c432f61</xsl:param>
    <xsl:output method="html" encoding="UTF-8"/>
    <xsl:template match="/">
        <xsl:for-each select="//Screen[@id=$ScreenId]">
            <xsl:variable name="screenName">
                <xsl:value-of select="@name"/>
            </xsl:variable>
            <html>
                <head>
                    <title>
                        <xsl:value-of select="$screenName"/>
                    </title>
                </head>
                <body>
                    <table>
                        <tbody>
                            <tr>
                                <td>
                                    <xsl:call-template name="renderScreen"/>
                                </td>
                            </tr>
                        </tbody>
                    </table>
                </body>
            </html>
        </xsl:for-each>
    </xsl:template>
    <!-- Render af screen - Will split screen into multiple parts dependant on the @split value  -->
    <xsl:template name="renderScreen">
        <table border="0" width="100%" cellpadding="0" cellspacing="0">
            <xsl:choose>
                <!-- horisontal or vertical split of screen -->
                <xsl:when test="@split='vertical'">
                    <tr>
                        <xsl:for-each select="./*">
                            <xsl:call-template name="renderItem"/>
                        </xsl:for-each>
                    </tr>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:for-each select="./*">
                        <tr>
                            <xsl:call-template name="renderItem"/>
                        </tr>
                    </xsl:for-each>
                </xsl:otherwise>
            </xsl:choose>
        </table>
    </xsl:template>
    <!--Default template for rendering an item-->
    <xsl:template name="renderItem">
        <td>
            <xsl:attribute name="valign"><xsl:value-of select="top"/></xsl:attribute>
            <xsl:attribute name="bgcolor"><xsl:value-of select="@bgcolor"/></xsl:attribute>
            <xsl:attribute name="height"><xsl:value-of select="@height"/></xsl:attribute>
            <xsl:attribute name="width"><xsl:value-of select="@width"/></xsl:attribute>
            <!-- one level down - alternative to foreach must be implemented (only one item exists) -->
            <xsl:choose>
                <xsl:when test="name()='Text'">
                    <xsl:call-template name="renderText"/>
```

```
                </xsl:when>
                <xsl:when test="name()='Screen'">
                    <xsl:call-template name="renderScreen"/>
                </xsl:when>
                <xsl:when test="name()='List'">
                    <xsl:call-template name="renderList"/>
                </xsl:when>
                <xsl:when test="name()='Image'">
                    <xsl:call-template name="renderImage"/>
                </xsl:when>
                <xsl:when test="name()='SearchForm'">
                    <xsl:call-template name="renderSearchForm"/>
                </xsl:when>
                <xsl:when test="name()='Form'">
                    <xsl:call-template name="renderForm"/>
                </xsl:when>
                <xsl:when test="name()='Empty'">
                    <xsl:text disable-output-escaping="yes">&amp;nbsp;</xsl:text>
                </xsl:when>
                <xsl:when test="name()='Href'">
                    <xsl:call-template name="renderHref"/>
                </xsl:when>
                <xsl:otherwise>not a valid node</xsl:otherwise>
            </xsl:choose>
        </td>
    </xsl:template>
    <!-- Display a view data will be -->
    <xsl:template name="renderList">
        <xsl:variable name="listId">
            <xsl:value-of select="@source"/>
        </xsl:variable>
        <table border="1">
            <tr>
                <th> </th>
                <xsl:for-each select="//Structure/View[@id=$listId]/table/Field[@visible='true']">
                    <xsl:sort select="@sortorder"/>
                    <!--<xsl:attribute name="select"><xsl:value-of select="@sortorder"/></xsl:attribute>-->
                    <th>
                        <xsl:value-of select="@name"/>
                    </th>
                </xsl:for-each>
                <xsl:for-each select="//Structure/View[@id=$listId]/ExternalTable/Field[@visible='true']">
                    <th>
                        <xsl:value-of select="."/>
                    </th>
                </xsl:for-each>
            </tr>
            <xsl:variable name="viewname">
                <xsl:value-of select="//Structure/View[@id=$listId]/@name"/>
            </xsl:variable>
            <!-- <xsl:value-of select="$listId"/><br/>
              <xsl:value-of select="$viewname"/>-->
            <xsl:for-each select="//DataSet/*[name()=$viewname]">
                <tr>
                    <td/>
                    <xsl:for-each select="*">
                        <xsl:sort data-type="number"
select="//Structure/View[@id=$listId]/table/Field[@name=name(current())]/@sortorder|//Structure/View[@id=$listId]/Exte
rnalTable/Field[@name=name(current())]/@sortorder"/>
                        <!--<xsl:value-of select="name()"/>-->
                        <xsl:variable name="name">
                            <xsl:value-of select="@name"/>
                        </xsl:variable>
                        <xsl:choose>
                            <xsl:when
test="(//Structure/View[@id=$listId]/table/Field[@name=name(current())]/@visible) = 'true'">
                                <td>
                                    <xsl:value-of select="."/>
```

```
                                    </td>
                                </xsl:when>
                                <xsl:when
test="(//Structure/View[@id=$listId]/ExternalTable/Field[@name=name(current())]/@visible) = 'true'">
                                    <td>
                                        <xsl:value-of select="."/>
                                    </td>
                                </xsl:when>
                            </xsl:choose>
                        </xsl:for-each>
                    </tr>
                </xsl:for-each>
            </table>
        </xsl:template>
        <!-- this will show a text value on screen -->
        <xsl:template name="renderText">
            <xsl:value-of select="@value"/>
        </xsl:template>
        <!-- this will show a text value on screen -->
        <xsl:template name="renderHref">
            <a>
                <xsl:attribute name="href">?targetId=<xsl:value-of select="@target" disable-output-
escaping="yes"/></xsl:attribute>
                <xsl:value-of select="@value"/>
            </a>
        </xsl:template>
        <!-- Image -->
        <xsl:template name="renderImage">
            <img>
                <xsl:attribute name="src"><xsl:value-of select="@src"/></xsl:attribute>
            </img>
        </xsl:template>
        <xsl:template name="renderForm">
            <i>Form</i>
        </xsl:template>
        <!-- Search Form -->
        <xsl:template name="renderSearchForm">
            <table>
                <form method="post" action="runex.aspx">
                    <input type="hidden" name="requestId">
                        <xsl:attribute name="value"><xsl:value-of select="@request"/></xsl:attribute>
                    </input>
                    <xsl:variable name="rId">
                        <xsl:value-of select="@request"/>
                    </xsl:variable>
                    <input type="hidden" name="targetId">
                        <xsl:attribute name="value"><xsl:value-of select="@target"/></xsl:attribute>
                    </input>
                    <xsl:for-each select="//Request[@id=$rId]//searchField">
                        <tr>
                            <td>
                                <xsl:value-of select="@name"/>
                            </td>
                            <td>
                                <input>
                                    <xsl:attribute name="name"><xsl:value-of select="@name"/></xsl:attribute>
                                </input>
                            </td>
                        </tr>
                    </xsl:for-each>
                    <tr>
                        <td colspan="2">
                            <input type="submit" value="Search"/>
                        </td>
                    </tr>
                </form>
            </table>
        </xsl:template>
```

```
</xsl:stylesheet>
```