TECHNICAL UNIVERSITY OF DENMARK

Planlægningsværktøj til PROKA

IMM-B. Eng-2007-69

Malik Haznadar

2007

DTU IMM

Planlægningsværktøj til PROKA

IMM-B.Eng-2007-69

Denne rapport er udarbejdet som mit eksamensprojekt.

Projektet er udført for NIRAS Informatik i Allerød og i samarbejde med instituttet for Informatik og Matematik Modellering på DTU.

Eksamensprojektet er udført i perioden 20.09.2007 til 19.12.2007.

Allerød, 19. december 2007

Malik Haznadar

Resume

Der skal udvikles en applikation, som en tilføjelse til PROKA, hvor man kan organisere og administrere vedligeholdelsesarbejde der skal udføres på et vandløb. For at gøre det nemt og overskueligt for brugeren skal data vises som et Gantt diagram hvor aktiviteter f.eks. en planlagt vedligeholdelse, vises i forhold til en tidslinje og ved at markere eller klikke på en planlagt vedligeholdelse vil man kunne vise yderligere oplysninger eller redigere den samme.

Ideen er at udvikle denne applikation som en WPF (Windows Presentation Foundation) applikation. WPF er en del af .Net 3.0 framework og er indbygget i Windows Vista. Ideen for dette projekt opstod i den sidste del af min praktikperiode. På nuværende tidspunkt er denne funktion implementeret således at hvis man vil vise en oversigt over alle vedligeholdelser som er planlagt for et vandløb vil disse blive vist i en tabel. Selvom der findes funktioner til at sortere disse data, vil brugeren have svært ved at danne sig et overblik over hvilken type vedligeholdelse udføres på en del af vandløbet og i hvilket tidsinterval. Ved at præsentere disse data for brugeren på en grafisk måde vil brugeren have et bedre overblik over vedligeholdelsesarbejde.
På den anden side vil jeg få noget relevant erfaring med WPF programmering, da det er noget jeg interesserer mig for og som jeg kunne tænke mig at arbejde videre med.

Applikationens brugerinterface er opdelt vertikalt i tre sektioner:

- Yderst til højre en statisk sektion indeholdende signaturforklaring.
- Yderst til venstre en dynamisk aktivitetsliste som viser den aktuelle vandløbsstrækning, vedligeholdelsesaktiviteten samt tidsintervallet.
- I midten Gantt diagrammets tidslinje i et dynamisk vindue. Tidslinjen kan varieres.

Indhold

1 Introduktion

1.1 Baggrund

NIRAS Informatik i Allerød beskæftiger sig hovedsagelig med udvikling af GIS[1] baserede løsninger. De har selv udviklet PROKA, en applikation som anvendes hos danske kommuner for at opsamle og behandle data fra vandløbene under deres administration. Senere blev der udviklet et modul som kan integrere PROKA's data i en GIS applikation. Næste skridt er at udvikle et vedligeholdelse administrations modul og det er så her hvor dette projekt kommer ind i billedet.

1.2 Interesse og motivation

Min beslutning om at lave Proka Planlægningsværktøj som en WPF applikation bygger på følgende grundlag:

- WPF er velegnet til denne type applikationer og i det sidste ende vil det resultere i et bedre produkt.
- Det vil give mig noget relevant erfaring med WPF.

1.3 Projektafgrænsning

Med hensyn til projektets størrelse i forhold til projektperioden vil det være nødvendigt at vælge et subset af funktioner, som vi fandt realistisk at kunne implementere på den tid der er tilgængelig, og som vil illustrere hvordan det endelige program kommer til at fungere. Disse funktioner vil blive defineret af projektets kravspecifikation.
Det vil blive lagt mest vægt på udvikling af grafiske funktioner og komponenter der udfører disse funktioner da det er den centrale del af applikationen.

1.4 Rapportens målgruppe

Rapporten er skrevet på sådan en måde at en stor del af rapporten kan læses af alle, dvs. læseren behøver ikke at have noget kendskab til programmering eller erfaring med .Net framework. Derimod er kapitel 6 skrevet med mange tekniske detaljer som kræver kendskab til objektorienteret programmering og .Net framework.

[1] GIS – forkortelse for Geographic Information System.

2 Introduktion til WPF

Da Planlægningsværktøj til Proka er en WPF applikation synes jeg at det vil være absolut nødvendigt at introducere WPF for læseren med hensyn til forståelse af resten af dokumentet. Derfor gives i dette afsnit en kort gennemgang af WPF.

WPF eller Windows Presentation Foundation er Microsofts seneste framework til udvikling af grafiske brugeflader(GUI). Den kommer som en del af .Net 3.0, som er indbygget i Windows Vista. Udover Vista er det kun muligt at afvikle WPF applikationer under Windows XP SP2 og Windows Server 2003 SP1, hvor der er installeret .Net framework 3.0.

WPF er betydeligt anderledes end Windows Forms og den introducerer mange nye features og tekniker. Her er nogle af de mest betydelige og som også blev brugt til udvikling af Planlægnings-værktøj til Proka:

1. XAML - Ekstensible Application Markup Language. Giver mulighed for at definere opbygning af hele brugerfladen vha. XML. Denne er ikke kun begrænset til WPF men kan også anvendes til udvikling af Windows Workflow Foundations applikationer, hvor man vha. XAML kan definere workflow relateret aktivitet.
2. Dependency properties (DP) - .Net properties med udvidet funktionalitet. DP har bl.a. indbygget funktionalitet til at underrette om ændringerne(*change notification)* eller hvis det er tale om databinding, slutpunktet for denne binding skal være en dependency property.
3. Data templates - Bruges til at definere indholdet af komponenter som *ListBox*, *ListView*, *ComboBox* osv.
4. Logical resources - Anvendes mest af *Styles* og *DataTemplates*. Ved hjælp af *Style* er der muligt at definere værdier for nogle egenskaber som er fælles for en række komponenter. *DataTemplates* er beskrevet under punkt 3.
5. Hardware acceleration - Bygget oven på Direct3D. Det betyder at animationer og visuelle effekter udføres af GPU[2].

I takt med at Windows Vista bliver mere og mere udbredt, vil WPF også blive mere udbredt. Det afspejles også i Microsofts strategi i at ikke bruge flere ressourcer på udvikling af Windows Forms, men vil udelukkende fokusere på WPF. Så hvis der ikke er et krav for et produkt til at kunne afvikles under Windows 2000 eller Win98 vil WPF være det foretrukne værktøj til udvikling af nye GUI.

2.1 Værktøjer

Der blev anvendt Visual Studio 2008 til udvikling af Planlægningsværktøj til Proka. Det er også muligt at anvende Visual Studio 2005 til at udvikle WPF applikationer, men når der gælder WPF applikationer indeholder Visual Studio 2008 er række forbedringer i forhold til Visual Studio 2005. Ikke mindst XAML editor i VS 2008 fungerer langt bedre end den der er tilgængeligt til VS 2005.

For at holde styr på kildekoden blev der anvendt versionsstyring system Subversion.

[2] GPU - Graphics Processing Unit

3 Kravspecifikation

Der skal udvikles en Windows applikation til administration af vedligeholdelse arbejde for en række vandløb. Informationer skal præsenteres for brugeren på en grafisk måde i form af en Gantt diagram, således at hun/han kan hurtig danne sig et overblik over det planlagte arbejde. Programmet kan i høj grad minde om MS Project, mht. layout og arbejdsgang. Der vil være to paneler, et panel hvor data for en vedligeholdelse er vist som i en tabel og et grafikpanel.

Brugeren skal have mulighed for at:

- Oprette en ny vedligeholdelse.
- Slette en vedligeholdelse.
- Tilpasse visning vha. zoom.

Der skal være mulighed for at vise data på to måder.

- I forhold til tid - der skal, som reference, bruges en tidslinje hvor man kan se starttidspunktet for en vedligeholdelse samt varighed, i lighed med en Gantt diagram i MS Project.
- I forhold til sted - der skal, som reference, bruges en liste med alle stationerne som er defineret for et givet vandløb, hvor en vedligeholdelse vil vises i forhold til start og slut stationer.

Da en vedligeholdelse altid er defineret således at den har en bestemt type, skal enhver type vises med en specifik farve.

Ved hjælp af zoom funktionen brugeren vil kunne tilpasse størrelsen af elementerne således, at der vil vises oversigt over vedligeholdelsesarbejde for længere periode eller et større område. Der skal være mindst fire foruddefinerede zoom faktorer: ”Dage”, ”Uger”, ”Måneder” og ”År”.

Programmet skal være kompatibelt med Access og SQL Server databaser. Forbindelse til databasen skal være let at konfigurere så programmet kan bruges med begge typer.

Der skal være mulighed for at sortere eller filtrere data efter diverse kriterier. Som det mindste krav skal programmet kunne udføre følgende funktioner:

- Sortering efter vandløbsnavn.
- Filtrering efter vedligeholdelsestypen.

4 Arkitektur

4.1 Introduktion

I bogen ” Applying UML and Patterns” introducerer Craig Larman begrebet den logiske arkitektur (the logical architecture) og definerer nogle generelle retningslinjer vedrørende applikations design og struktur. Ifølge Craig Larman, den logiske arkitektur definerer hvordan software klasser er opdelt i pakker (eller namespaces), moduler eller lag.
Et modul kan indeholde flere namespaces og enhver namespace kan indeholde mange klasser. Opdeling i lag sker efter den overordnede ”ansvar” det pågældende lag har.

Planlægningsværktøj til PROKA består af følgende lag:

1.	Brugerfladen	(også kaldt *Presentation layer)*.
2.	Logik	(også kaldt *Business logic layer*).
3.	Data	(også kaldt *Data access layer*).

Dette er også et klassisk eksempel på en tre lags arkitektur som er en meget udbredt software model. I denne model er lagene organiseret således at der kun er et højere lag der må kalde funktioner i det nedenstående lag og ikke omvendt.
Ved at dele en applikation op på denne måde kan følgende undgås:

1. At ændringerne i kildekoden går igennem hele systemet.
2. At applikations logik er sammenflettet med brugerfladen så den kan ikke genbruges med en anden brugerflade.

En anden slags arkitektur er system arkitektur[3] og fra dette synspunkt kan Planlægningsværktøj til Proka beskrives som en monolitisk applikation[4]. Den er monolitisk fordi brugerfladen, logik og data er samlet i en enkelt eksekverbar fil. Denne type er karakteristisk for en Windows desktop applikation.

4.2 Namespaces

Der er defineret følgende namespaces i Planlægningsværktøj til Proka:

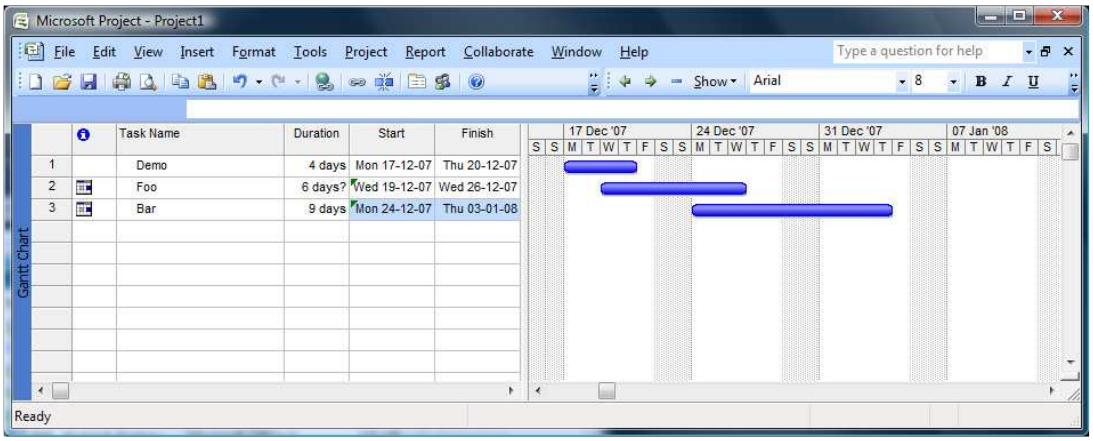
• ProkaApp.	Indeholder diverse hjælpeklasser, ligesom specielle datastrukturer.
• ProkaApp.UI.	Indeholder klasser som har UI[5] relateret funktionalitet.
• ProkaApp.BusinessLogic.	Indeholder klasser der repræsenterer domæne modellen.
• ProkaApp.Data.	Indeholder klasser der tager sig af kommunikation med databasen.

[3] System arkitektur. http://en.wikipedia.org/wiki/Systems_architecture
[4] Monolitisk applikation. http://msdn2.microsoft.com/en-us/library/aa480455.aspx eller http://en.wikipedia.org/wiki/Monolithic_application.
[5] UI – forkortelse for User Interface (brugerfladen).

5 Design

5.1 Introduktion

Som beskrevet tidligere er dette program et planlægningsværktøj og dets formål er at hjælpe den (eller de) miljøansvarlige i en kommune med at planlægge og administrere vandløbs vedligeholdelse. For at hente inspiration til mit projekt har jeg undersøgt hvad der findes på markedet af denne type programmer. Udover MS Project som er nok det mest kendte og udbredte planlægningsværktøj, findes der nogle programmer ligesom SmartDraw og ConceptDraw, som i princippet kan det samme og deres layout minder om layoutet i MS Project. Dette var også min oprindelige idé, da jeg synes at denne layouttype giver det bedste overblik. Derfor har jeg besluttet at holde mig til det "klassiske" layout hvor data præsenteres i form af en Gantt diagram suppleret med en form for tabel hvor yderligere oplysninger vil vises.



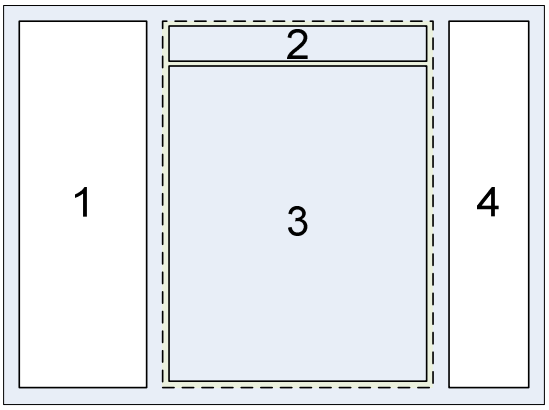
Figur 1: Gantt diagram i MS Project

Kan MS Project anvendes til dette formål?
Et kort svar vil være ja. Og ved hjælp af VBA[6] vil man kunne skabe den ønskede funktionalitet. På den anden side har man mulighed for at bygge en WPF applikation (se afsnit 2 Introduktion til WPF) fra bunden, som er bygget til at være kompatibel med PROKA's data model og kan arbejde med PROKA's data.

[6] VBA – Visual Basic for Applications. Se mere på http://en.wikipedia.org/wiki/Visual_Basic_for_Applications.

5.2 Layout

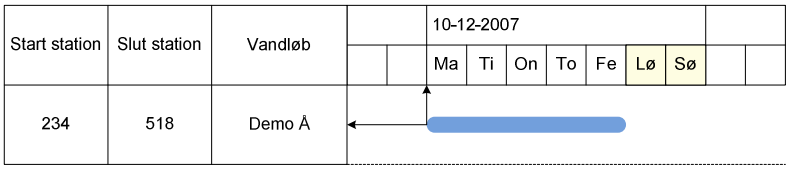
Den overordnede brugerflades struktur er illustreret på følgende diagram.



Figur 2: Layout

Panel 2 skal indeholde ***reference*** komponenten. Denne komponent skal enten være en tidslinje eller en komponent der viser alle stationer for et givet vandløb. Den hedder reference komponent da alle elementer i panel 3 er placeret i forhold til denne.

Panel 3 skal vise indeholde ***grafik*** komponenten. Denne komponent skal kunne tegne et Gantt diagram ud fra vedligeholdelsesdata. For at kunne tegne et enkelt vedligeholdelseselement skal følgende oplysninger være tilgængelige: startdatoen og varighed for en vedligeholdelse, samt typen som vises i form af en bestemt farve. Disse elementer skal placeres korrekt i forhold til reference komponenten i panel 2. Dvs. hvis en vedligeholdelse skal starte d. 10.12.2007 skal denne være placeret nøjagtigt i forhold til 10.12.2007 på tidslinjen. Endvidere skal den placeres korrekt i forhold til det element i listen der indeholder yderligere informationer for denne vedligeholdelse (panel 1). Dette er illustreret på nedenstående diagram.



Figur 3: Placering af elementerne

Panel 1 skal vise informationer om vedligeholdelser i form af en tabel. Der skal vises informationer om vandløbet som ikke fremgår af den grafiske visning (panel 3). På den ovenstående figur vil kolonnerne ”Start station”, ”Slut station” og ”Vandløb” være indholdet af dette panel.

Panel 4 viser alle vedligeholdelsestyper samt farven for disse. Dette panel skal have en filtrerings funktion hvor brugeren kan markere en bestemt type og det vil medføre at kun oplysninger om vedligeholdelser af den valgte vedligeholdelsestype vil vises i panelerne 1 og 3. Endvidere kunne man kombinere dette med muligheden at vælge en bestemt vedligeholdelseskategori der vil resultere i at kun vedligeholdelsestyper som tilhører denne kategori vil blive vist.

Der skal tages hensyn til følgende faktorer:

- Panel 3 skal have samme bredde som panel 2.
- Panel 1 skal have en højde der svarer til sammenlagt højde for panelerne 1 og 2.
- Panelerne 2 og 3 kan have en bredde og højde der er meget større end den tilgængelige plads i vinduet. Derfor skal der være mulighed for scrolling i begge retninger. Ved horisontal scrolling skal indholdet af panelerne 2 og 3 flyttes samtidig. Ved vertikal scrolling skal indholdet af panel 1 og 3 flyttes samtidig.

6 Implementering

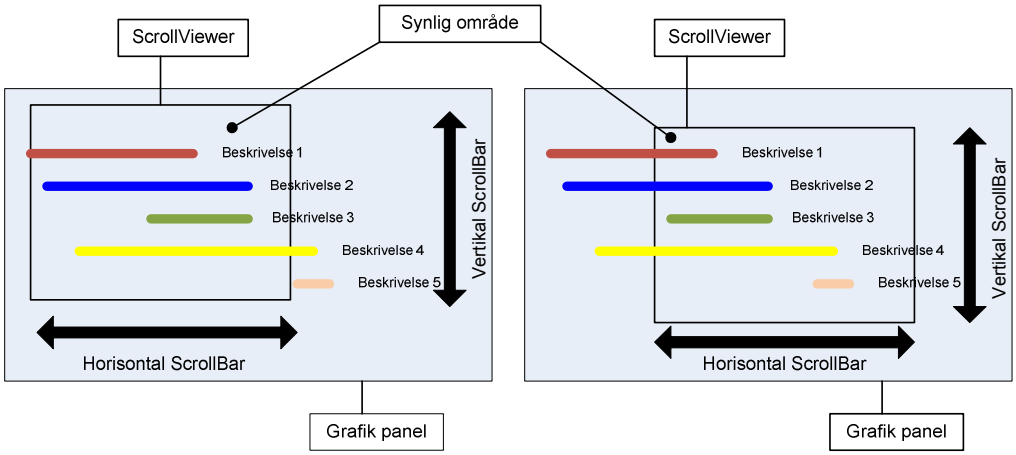
6.1 Brugerfladen

6.1.1 Introduktion

Dette afsnit giver en detaljeret gennemgang af konkrete byggeblokke i denne applikation. Læseren vil blive introduceret til komponenter som *NView*, *TimeLine* og *ContentListItem*, samt samspil mellem disse komponenter. Bilag indeholder et komplet diagram over brugerfladens opbygning, se 11.5 Struktur diagram.

6.1.2 NView

Et NView objekt har en rolle som på en måde minder om den rolle ***controller*** har i *Model View Controller*[7] pattern. Det er ikke i sig selv en visuel komponent, men indeholder reference til visuelle komponenter som brugerfladen består af, samt en reference til det underliggende logik lag. Det er NView's ansvar at instantiere *reference* og *grafik* objekter som skal være indholdet af panel 2 og panel 3 (der henvises til foregående afsnit og Figur 3). Både panel 2 og 3 er implementeret som to *ScrollViewer*[8] objekter. Årsagen til at der bruges netop *ScrollViewer* er at den kan indeholde en komponent der er større end den selv og ved hjælp af scrollbars kan brugeren flytte indholdet i begge retninger.



Figur 4: Scrolling funktion

I denne tilfælde er højden af grafik komponenten afhængigt af antallet af vedligeholdelser den skal vise mens bredden er afhængigt af bredden for reference komponenten.

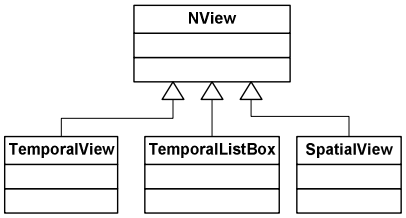
Som beskrevet i afsnit 5.2 Layout, skal indholdet af paneler 2 og 3 henholdsvis være en reference komponent og en grafik komponent. En reference komponent kan være enten en *TimeLine* eller en *SpatialLine* objekt. Disse vil blive beskrevet i efterfølgende afsnit.

Reference komponenten og grafik komponenten skal placeres i separate paneler af en enkelt grund:

- Referencekomponenten skal altid være synlig også mens indholdet af grafik komponenten scrolles ned.
  Den første udgave af programmet bød på en løsning hvor reference og grafik komponenter var samlet i en komponent. Dette har vist sig at være en løsning med (unødvendig) komplekst og hyppigt fejlende (*error prone*) logik der ikke har nogle (ydelsesmæssige) fordele i forhold til den aktuelle implementering.

[7] Model View Controller eller MVC – se "Ordforklaring" afsnittet.
[8] ScrollViewer klassen findes i System.Windows.Controls namespace.

Altså en NView objekt indeholder komponenter som definerer i virkeligheden måden data skal præsenteres for brugeren. Der er tit behov for at kunne vise samme data på flere måder.
Med en modificeret udgave af NView der indeholder specielle reference og grafik komponenter er det muligt at lave skræddersyede løsninger uden at modificere resten af applikationen.
Derfor er *NView* implementeret som en abstrakt klasse i ProkaApp.UI namespace, og den anvendes som base klasse til følgende klasser: *TemporalView, TemporalListBox* og *SpatialView*.



Figur 5: *Klassehierarkiet*

Alt dette er gjort med hensigten at skabe en struktur som er i høj grad *generic*, dvs. der vil være let at tilføje nye typer visninger til applikationen uden at modificere den eksisterende kode. I starten af projektet har jeg også leget med tanken at skabe en plugin baseret applikation som kunne bruges til en hvilken som helst opgavetype. Den har jeg dog opgivet da dette ikke var et krav til denne applikation.

6.1.2.1 TemporalView

TemporalView skal kun beskrives i korte træk da den er blevet erstattet af *TemporalListBox* som er den forbedrede udgave af *TemporalView.* Denne implementering består af en *TimeLine* som reference komponenten og *Content* som grafik komponenten. *Content* komponenten er også forældet og den skal ikke bruges længere. Årsagen at *TemporalView* og *Content* overhovedet bliver nævnt her er at de er endnu ikke fjernet fra koden.

6.1.2.2 TemporalListBox

Her består reference komponenten af en *TimeLine* mens grafik komponenten er en almindelig WPF *ListBox*. Ved hjælp af en *DataTemplate* er det muligt at definere indholdet af en *ListBox*. I forhold til *TemporalView* er denne implementering mere avanceret og den kræver et solidt kendskab til WPF framework. På den anden side er den også mere elegant da logikken bag denne model er mere simpelt end i *TemporalView*. Det skal forstås på den måde at alt den logik det har med håndtering af listeelementer af gøre udføres af selve *ListBox* komponenten. Dette kommer til at spille en væsentlig rolle når vedligeholdelser skal sorteres, filtreres eller når et enkelt element bliver slettet. Alle af de lige nævnte funktioner udføres af nogle funktioner der er indbygget i *ListBox* komponenten. I tilfælde hvor brugeren skifter tegnemåden eller hvilken som helst indstilling der vil resultere i at *TimeLine* ændres skal alle listeelementer opdateres. Dette skal dog gøres manuelt ved at løbe hele elementlisten for ListBox igennem og opdaterer hver eneste element.

6.1.2.3 SpatialView

En visnings type som ikke indeholder en tidslinje komponent men en tilsvarende komponent med alle stationer for et vandløb. Denne er dog ikke implementeret på nuværende tidspunkt.

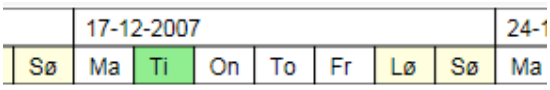
6.1.3 Visuelle komponenter

6.1.3.1 TimeLine

TimeLine komponenten, i det tidligere afsnit kaldt ”*reference*” komponenten er fuldstændig defineret ved hjælp af tre parametre: startdato, slutdato og tegnemåden. Tegnemåden er tilføjet for at opfylde kravet om zoom funktion. Der er defineret fire tegnemåder:

- Dage — Når tegnemåden dage er valgt, har tidslinjen samme form som på billedet nedenunder. På den øverste del er alle mandage angivet med dato. På den nederste del er alle ugedage angivet. Den aktuelle dato markeres med et

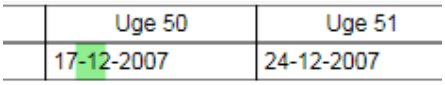
grønt felt, dog ikke hvis det er lørdag eller søndag da disse er markeret med gul farve.



Figur 6: Dage

- Uger

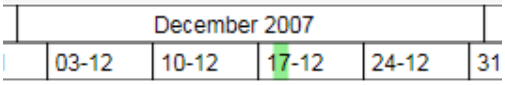
På den øverste del er alle uge nummereret og på den nederste er alle mandage angivet med datoen. Den aktuelle dato markeres med et grønt felt.



Figur 7: Uger

- Måneder

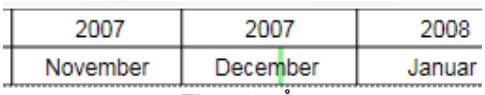
På den øverste del er alle måneder markeret med navn og årstal. På den nederste del er alle mandage markeret med datoen. Den aktuelle dato markeres med et grønt felt.



Figur 8: Måneder

- År

Her er tidslinjen opdelt i en række felter der indeholder en kombination af navn for en måned og årstal. Den aktuelle dato markeres med et grønt felt.



Figur 9: År

TimeLine komponenten er implementeret i *TimeLine* klassen, fra ProkaApp.UI namespace, som nedarver fra *FrameworkElement*[9] klassen, og dermed har *OnRender* funktion. Ved at override denne funktion er det muligt at definere hvordan et element skal tegnes på skærmen. Denne kaldes hver gang brugeren ændrer på start eller slutdato, eller tegnemåden. De tre parametre som var omtalt i starten af afsnittet er implementeret som tre *dependency properties*[10].
En dependency property kan angivet som en der automatisk kalder *OnRender* hver gang den tildeles en ny værdi. Det sker selvfølgelig kun hvis tilhørende klasse indeholder en OnRender funktion. Denne komponent vil altid have samme højde men variabel bredde. Bredden er afhængig af alle tre parametre, dvs. startdato, slutdato og tegnemåden.

TimeLine bliver initialiseret med følgende værdier:

- Den første dato bliver sat til en dato der er en måned før den aktuelle dato.
- Den sidste dato bliver sat til en dato der er tre måneder efter den aktuelle dato.
- Tegnemåden bliver sat til ”Dage”.

Tegne algoritmen

Algoritmen udføres hver gang *OnRender* funktionen bliver kaldt. Derefter, afhængigt af tegnemåden, bliver en af de fire tegnefunktioner kaldt, da der findes en funktion for hver tegnemåde. Fælles for alle tegnemåder er at der oprettes en liste over alle datoer mellem start og slutdatoen. Derefter skal enkelte elementer tegnes i en bestemt rækkefølge:

1. Der tegnes baggrunds farver for weekender og den aktuelle dag.
2. Afhængigt af tegnemåden skal følgende datoer findes og gemmes i en separat liste:
   a. Alle den 1. i en måned hvis tegnemåden er sat til ”År”.
   b. Alle mandage for alle andre tegnemåder.
3. Linjer som danner ”rammen”.

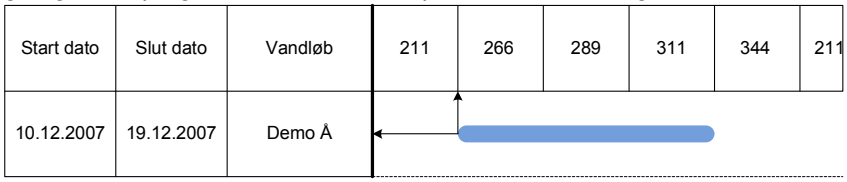
[9] FrameworkElement er defineret i System.Windows namespace.
[10] Kapitel 2, ”Introduktion til WPF”, giver en kort gennemgang af dependency properties.

4. Tekstelementer.

Hvis brugeren har ændret start eller slut dato for tidslinjen skal følgende ske. Der udføres simpelt validering af den nye værdi (det er ikke tilladt at have en startdato som er større end slutdato) og der beregnes bredde(længden) af tidslinjen. Når størrelsen for tidslinjen bliver ændret medfører dette at placering af elementerne på grafik komponenten ikke er længere gyldigt. Derfor skal *TimeLine* komponenten underrette grafik komponenten om ændringerne. Det skal igen medføre at grafik komponenten skal slette alt indhold og gentegne alle vedligeholdelseselementer med de nye koordinater.

6.1.3.2 SpatialLine

Den bruges af *SpatialView* komponenten og skal vise alle stationer der findes for et specificeret vandløb. Derefter skal vedligeholdelses elementer tegnes på grafik komponenten i forhold til start og slut stationer. Denne komponent er ikke færdigimplementeret endnu. Hvordan dette vil se ud i en fremtidig udgave af programmet er illustreret på nedenstående figur.



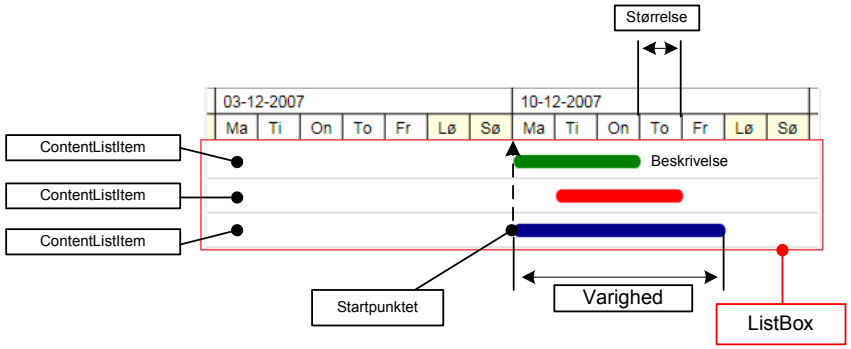
Figur 10: SpatialLine komponenten

6.1.3.3 ContentListItem

Denne komponent bruges som indholdet af en *ListBox*, som grafik komponenten. Den har følgende *dependency properties*:

- **Startpunktet**. Dette startpunkt er ikke en dato men et punkt med koordinater der svarer til startdatoen for den vedligeholdelse den repræsenterer.
- **Varighed**. Den svarer til vedligeholdelsens varighed.
- **Størrelsen**. Navnet for denne *dependency property* er lidt misvisende da den er afhængig af tidslinjens tegnemåde. Men i virkeligheden størrelsen for det grafiske element der bliver tegnet på skærmen fås ved at multiplicere denne værdi med vedligeholdelsens varighed.
- **Farven**. Angiver den farve der skal bruges til at tegne elementet på skærmen. Svarer til vedligeholdelsens type.
- **Beskrivelse**. Beskrivelse for den vedligeholdelse den repræsenterer.

Det nedenstående diagram illustrerer sammenhæng mellem disse *dependency properties* og komponentens udseende.



Disse *dependency properties* gør det muligt at:

- Gentegne enkelte elementer bare ved at ændre en af disse *dependency properties*.
- Anvende denne komponent i en model hvor der bruges databinding.

6.2 Logik

Dette afsnit omhandler elementer der repræsenterer domæne modellen for denne applikation.

Der blev afprøvet to modeller. Fælles for begge modeller er at der anvendes datastrukturer som kan bruges som databindings "*kilder*". Dette var nødvendigt da næsten alle elementerne på brugerfladen er forbundet med underliggende datastrukturer ved hjælp af data bindings. Dette giver et meget fleksibelt design og en pæn adskillelse mellem brugerfladen og logiklaget. Bemærk at WPF introducerer en ny databinding model og den må ikke forveksles med Windows Forms databinding.

Den første model er baseret på et *DataSe*t og tabellen "VedligeholdelseTabel" der indeholder alle vedligeholdelse oplysninger.

Den anden model består af en liste af typen *ObservableCollection*<T>[11]. Denne liste består af en række *Maintenance* objekter. *Maintenance* klassen indeholder alle oplysninger om en vedligeholdelse (se Klassediagram afsnit).

Set fra funktionsmæssig side er disse to modeller næsten ens. Der findes funktioner til at udføre avanceret sortering og filtrering med både *ObservableCollection<T>* og *DataTable*.

Den nuværende implementering er udført med en *ObservableCollection* af *Maintenance* objekter. Dette er gjort kun fordi at der er en grund til at tro at den er "*lettere*" end *DataSet*, dvs. den vil bruge mindre hukommelse. Den eneste måde at bevise dette vil være ved hjælp af en test. Hvis det viser sig at forskellen er minimal så vil en løsning bygget op omkring et *DataSet* være nemmere, da den vil kræve mindre logik.

Udover de lige omtalte elementer findes der følgende strukturer:

- En hashtable der består af vedligeholdelsestype og farve par. Indholdet af denne hashtable bruges blandt andet til at definere indholdet af panelet yderst til højre (panel 4 i afsnit 5.2 Layout).
- En hashtable der består af vedligeholdelsestype og id par. Denne bruges ved oprettelse af en ny vedligeholdelse.
- En hashtable der består af vedligeholdelseskategori og id par. Denne bruges også ved oprettelse af en ny vedligeholdelse.

6.3 Data

6.3.1 Generelt om datalaget

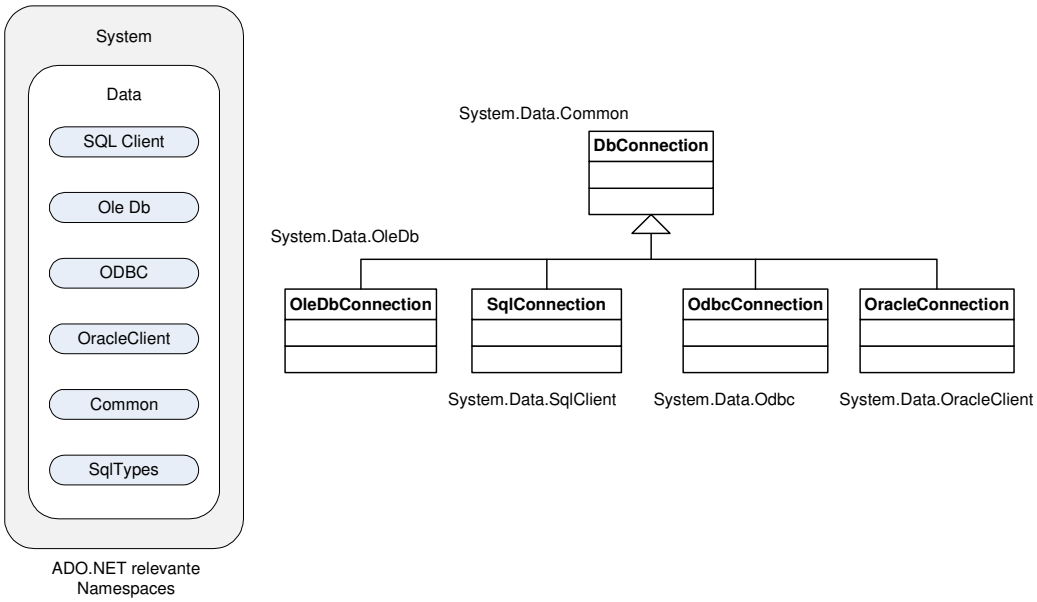
I henhold til kravspecifikation skal Planlægningsværktøj til Proka kunne anvendes med to database typer: Access og MS SQL Server. Med ADO.NET[12] er det muligt at anvende OLE DB data provider for kommunikation med både Access og SQL Server databaser.
Ifølge bogen "Programming Microsoft ADO.NET 2.0 Core Reference" (af David Sceppa), for kommunikation med SQL Serverer database vil SQL Client .Net data provider være hurtigere end OLE DB data provider. Det er fordi SQL Client .Net data provider er optimeret til kommunikation med SQL Server databaser. Derimod hvis man arbejder med en Office Access database, anbefaler forfatteren at bruge OLE DB data provider. Disse retningslinjer blev fulgt i design af datalaget i Planlægningsværktøj til Proka.

6.3.2 Design

For at undgå at skrive det samme kode to gange, en for Access og en for SQL Server, var det nødvendigt at udvikle en model som kan bruges til begge database typer. For at kunne forstå

[11] ObservableCollection<Of T> er defineret i System.Collections.ObjectModel namespace.
[12] ADO.NET – se afsnit "Ordforklaring" for definition.

denne model skal man forstå hvordan ADO.NET er opbygget. Dette er delvist illustreret på det nedenstående diagram.



Figur 11: ADO.NET klasser og namespaces

På det ovenstående diagram kan man se f.eks. at alle *connection* typer nedarver fra *DbConnection* klassen. Dvs. hvis applikationens dataklasse[13] indeholder en reference til en *DbConnection* objekt, gælder det blot at instantiere et objekt af enten *OleDbConnection* eller *SqlConnection* klasserne som denne reference peger på. (Det samme gælder også for *DbCommand* klassen, da der findes en provider specifik implementering for denne klasse). Dette kan let gøres ved hjælp af *DbProviderFactory*[14] klassen. Der kan anvendes følgende fremgangsmåde:

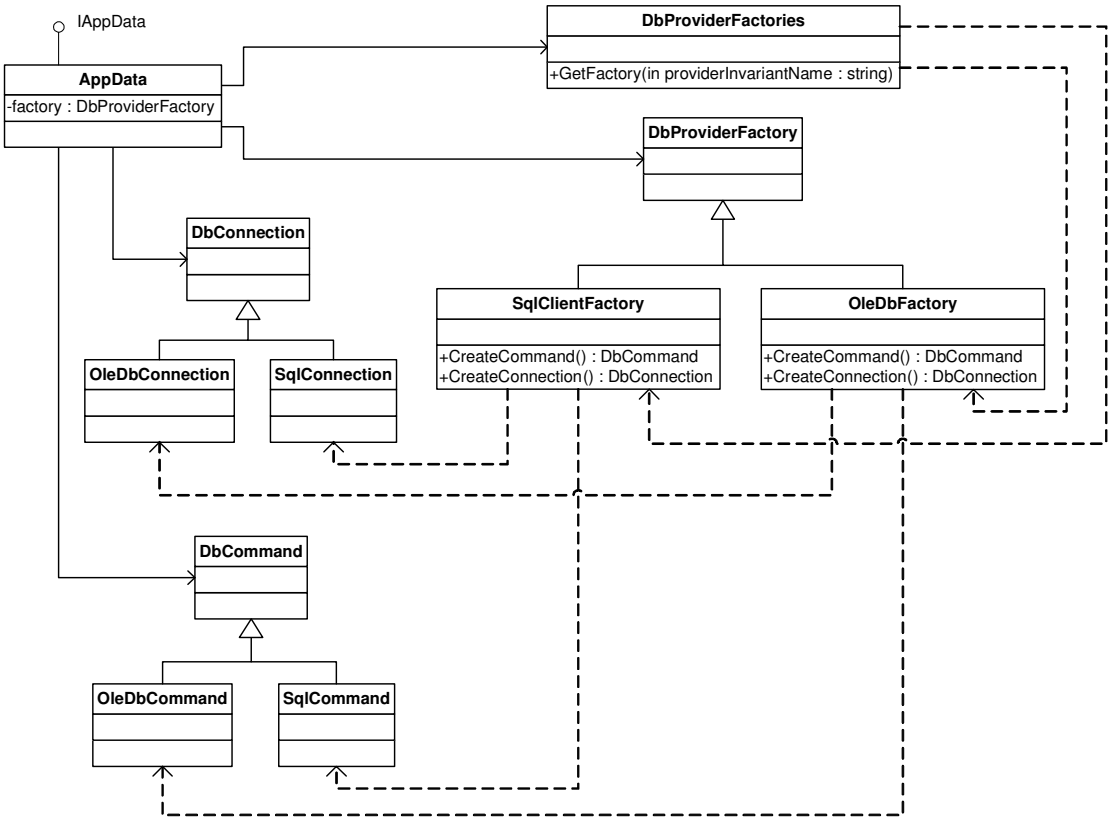
1. Angiv provider navn som en string. Følgende navne kan bruges som gyldige provider navne:
   - "System.Data.OleDb" for OLE DB data provider.
   - "System.Data.SqlClient" for SQL Server
   - "System.Data.Odbc" og "System.Data.OracleClient" kunne i princippet også bruges men det spiller ingen rolle da Planlægningsværktøj til Proka skal bruges som en udvidelse af PROKA og den arbejder kun med Access og SQL Server databaser.

   Et oplagt sted at gemme alle oplysninger om en data forbindelse vil være <connectionString> sektion i konfigurations filen, og dette er også gjort i denne applikation.

2. Brug denne string som argument til *DbProviderFactories.GetFactory*("ProviderName")[15]. Denne returnerer en reference til et objekt som er instans af en klasse som nedarver fra *DbProviderFactory*, dvs. enten *SqlClientFactory* eller *OleDbFactory*, afhængigt at hvilket data provider har man specificeret i argumentet. Derefter kan dette objekt bruges til at instantiere de aktuelle *connection* og *command* objekter.

Hvordan dette er opbyget er illustreret på nedenstående diagram.

[13] Denne klasse tager sig af alt databaserelateret aktivitet. Dette kan være indhentning af data, opdatering eller sletning af indholdet i en database tabel. I denne applikation klassen AppData fra ProkaApp.Data namespace har denne funktion.
[14] Klassen DbProviderFactory er defineret i System.Data.Common namespace.
[15] Klassen DbProviderFactories er defineret i System.Data.Common namespace.



Figur 12: Data factory

Det ovenstående diagram er også et klassisk eksempel på Abstract Factory[16] pattern.

6.3.3 Initialisering og setup

Her skal det tages hensyn til følgende:

- Brugeren skal have mulighed for at konfigurere en forbindelse på en let måde. En god måde vil være ved hjælp af en wizard hvor brugeren guides igennem denne proces.
- Der skal være mulighed for at definere flere forbindelser og det skal være let at skifte fra en forbindelse til en anden.
- Hvis der kun er defineret en forbindelse, skal der automatisk udføres setup for denne forbindelse.

I initialiserings fasen af applikationen indlæses alle "connectionStrings" fra konfigurations filen. Navne for de enkelte forbindelser skal vises som indholdet af en *ComboBox* i applikationens hovedvindue. Når brugeren har valgt en forbindelse skal der udføres setup for denne. Det betyder at der skal aflæses informationer om den valgte forbindelse. Disse informationer består af *connectionString* og *providerName* elementerne. *ProviderName* elementet skal bruges til at instantiere en konkret *DbProviderFactory* objekt ved hjælp af *GetFactory* funktionen i *DbProviderFactories klassen.* Her er et eksempel på dette.

```
DbProviderFactory factory = DbProviderFactories.GetFactory(ConnectionProvider);
```

Senere skal denne *DbProviderFactory* objekt bruges til at instantiere respektive *DbConnection* og *DbCommand* objekter. Dette er illustreret med følgende eksempel:

```
DbConnection connection       = factory.CreateConnection());
connection.ConnectionString   = ConnectionString;
```

[16] Abstract Factory. Se mere på http://www.dofactory.com/Patterns/PatternAbstract.aspx.

```
DbCommand cmd                          = factory.CreateCommand();
```

Derefter udføres en simpel test. Denne test består blot af at åbne og lukke for denne forbindelse med de aktuelle indstillinger.

Det nuværende implementering afviger fra dette da der ikke findes en konfigurations wizard i øjeblikket og den eneste måde at tilføje en ny forbindelse er ved at manuelt redigere ProkaApp.exe.config filen. Af samme grund er det heller ikke muligt at skifte forbindelse. Der er i øjeblikket kun defineret en forbindelse og der udføres setup og initialisering med det samme.

6.3.4 Insert model

Når data for en vedligeholdelse skal skrives i databasen, kan det gøres på følgende måde. Brugeren skal benytte dialogboksen til oprettelse af en ny vedligeholdelse. Dette er beskrevet i detaljer i afsnit 9.3.2, Opret vedligeholdelse.
Oplysninger som brugeren har indtastet skal først skrives til databasen. Funktionen der udfører dette skal returnere id for den nye vedligeholdelse, hvis den blev udført korrekt. Denne id skal bruges til at opdatere den interne liste af vedligeholdelser. Da denne liste er "bundet" til den komponent på brugerfladen der viser vedligeholdelser på skærmen, bliver denne komponent automatisk opdateret.
Dette er en meget simpel løsningsmodel der ikke tager højder for følgende scenarier:

- Data for nogle af vedligeholdelser der er repræsenteret intern af applikationen er blevet opdateret i databasen.
- Der blev tilføjet nye vedligeholdelser til databasen.

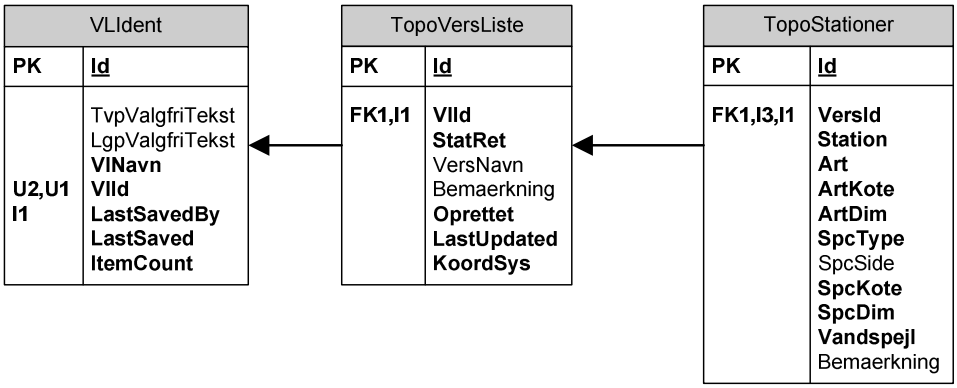
Den aktuelle løsningsmodel er godt nok i tilfælde hvor der er en enkelt person der administrer vedligeholdelser. I tilfælde hvor der er flere eller mange brugere vil der være nødvendigt at udvikle en mere avanceret model.
Fordelen ved denne løsningsmodel er at den er meget effektiv da den mængde data der skal behandles er begrænset til minimum.

6.3.5 PROKA's data model

Planlægningsværktøj til Proka anvender en del af PROKA's datamodel, med introducerer også nogle nye elementer. Da PROKA's datamodel indeholder nogle elementer som har ingen relevans for dette projekt, ligesom integration med tredje parts produkter, her vil blive beskrevet kun den del af datamodellen der indgår i Planlægningsværktøj til Proka datamodel.

Kernen i denne datamodel består af følgende tabeller:
1. **VIIdent** - denne tabel indeholder data som definerer et vandløb.
2. **TopoVersListe** - denne tabel indeholder data som definerer opmålinger relateret til et vandløb.
3. **TopoStationer** - indeholder data som definerer stationer som er relateret til en opmåling.

Dette er illustreret på nedenstående diagram (diagrammet er "*reverse engineered*" ved hjælp af Visio).



Figur 13: PROKA's datamodel

Ud fra det ovenstående diagram fremgår det at der kan være defineret flere opmålinger for et vandløb og der kan være defineret flere / mange stationer for en hver opmåling.

I en fuldstændig beskrivelse af en station indgår følgende elementer:

- ID for det aktuelle vandløb
- ID for den aktuelle opmåling
- ID for stationen

Disse oplysninger skal bruges når der skal tilføjes stationer til en vedligeholdelsesstrækning. Selvom disse oplysninger er nok til at identificere en station for et vilkårligt vandløb der indgår i en vedligeholdelsesstrækning, er det nødvendigt at gemme yderligere oplysninger om vandløbet. Årsagen til det er igen at denne applikation skal dele samme data med PROKA, som kan i nogle tilfælde overskrive alt indholdet i databasen så alle elementerne tildeles nye id-er.

Derfor, for at kunne identificere et station med 100 % sikkerhed, skal der tilføjes yderligere oplysninger om vandløbet som er ikke autogenereret som id-er. Ved at analysere opbygningen af de tre tabeller som er illustreret på Figur 13, kom det frem at tabellen ”VIIdent” indeholder et felt, "VIId", som er velegnet til dette formål da den er både påkrævet og unik. Feltet "VIId" af typen "Text" fungerer som en tekst baseret id for et vandløb. Feltet ”VINavn” er en anden mulighed, den er bare ikke unik. (Men det er også usandsynligt at have to vandløb der heder det samme og under samme administration.)

Det lige nævnte scenario stiller også et krav om at kontrollere data i vedligeholdelse relaterede tabeller for konsistens. Dette kan udføres på en let måde ved at tage en id for et vilkårligt vandløb som er gemt i vedligeholdelse tabellen og sammenligne den med indholdet af ”VIIdent” tabellen. Hvis den findes og har den samme ”Id” så betyder det at data ikke er blevet ændret i mellemtiden. Hvis den ikke findes så skal der udføres et grundigt element for element opdatering af hele vedligeholdelse strukturen. I dette tilfælde vil følgende punkter gøre sig gældende:

- At det kan være en stor mængde data der skal behandles. Dette betyder at der skal tages nogle designmæssige beslutninger hvordan denne databehandling skal håndteres. Her skal man tænke på den tid det kan tage at udføre denne funktion, og hvis den kommer til at afvikles på Dispatcher tråd vil det resultere i en frossen GUI for den tid. Derfor vil det være nødvendigt at det afvikles på en baggrunds tråd. Dvs. når data er indlæst skal applikation underrette GUI'en at den kan gå i gang med at rendere data, og samtidigt på et anden tråd vil der udføres data validering.
- I tilfælde af at en station er blevet slettet fra ”TopoStationer” tabellen, skal brugeren informeres om dette og det er brugeren som skal enten vælge at slette den pågældende station i alle vedligeholdelser der indeholder denne station eller udpege en ny station. I tilfælde af at det er en overstået vedligeholdelse skal der besluttes om dette kan ignoreres, da det ikke er defineret p.t.

Sammenhæng mellem valideringsprocessen og kravet om data sortering og filtrering.

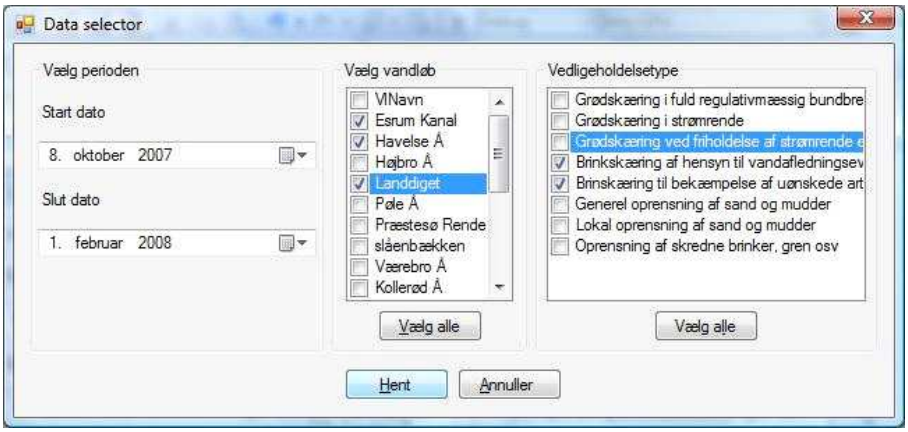
Det skal forstås på den måde at når programmet starter op sker det med nogle "default" indstillinger. En af disse indstillinger er at den skal hente alle vedligeholdelser fra databasen. Dvs.

sortering og filtrering udføres af applikationens logik efter data er indlæst. Hvis mængden af data begrænses før det bliver indlæst vil det give en ydelsesmæssig boost, da på grund af det reducerede mængde data vil både validerings og renderings proces afvikles hurtigere.

Dette kan gøres ved at definere følgende sorterings og filtrerings indstillinger:

- Der er ikke nødvendigt at hente vedligeholdelser som ikke kan vises, dvs. der skal hentes data kun for den "synlige" periode (den periode som er defineret af tidslinjens start og slutdato).
- Brugeren skal kunne vælge at hente data kun for en eller flere vandløb.
- Brugeren skal kunne vælge en eller flere typer af vedligeholdelse.
- Brugeren skal have mulighed at definere hvordan data skal sorteres af databasen selv.
- Der skal stadigvæk være mulighed for at sortere og filtrere data i selve applikationen.

Den mest oplagte måde at udføre dette vil være ved hjælp af en dialogbox der præsenteres for brugeren før der sendes en forespørgsel til databasen. Et eksempel på en passende dialogbox er illustreret nedunder.



I en situation hvor der blev indlæst kun en delmængde af data, vil det alligevel være nødvendigt at opdatere alt indhold (og ikke kun det data man arbejder med) i vedligeholdelse relaterede tabeller i tilfælde af at id'er i "VIIdent", "TopoVersListe" og "TopoStationer" tabellerne blev opdateret. Hvordan det foregår, vil der blive beskrevet om lidt efter en introduktion til vedligeholdelse datamodellen (se afsnit 6.3.7).

6.3.6 Vedligeholdelse datamodellen

Denne datamodel består af tre hovedelementer: vedligeholdelse, vedligeholdelsesstrækning og vedligeholdelsestype.

For at definere en vedligeholdelse skal man som det mindste definere følgende:

- Et vandløb med navn og id.
- To punkter(stationer) på et vandløb som start og slutpunkt for en vedligeholdelse.
- Perioden for vedligeholdelse, dvs. start og slutdato for den aktuelle vedligeholdelse.
- Typen af vedligeholdelse.

Udover lige nævnte kan der tilføjes følgende oplysninger:

- Varighed for en vedligeholdelse. Denne værdi beregnes ved oprettelsen og kan spare noget tid ved sortering af vedligeholdelsesarbejde efter varighed.
- En kort beskrivelse af vedligeholdelsen. Dette vil blive vist på skærmen lige ved siden af det grafiske element som repræsenterer en vedligeholdelse.
- Procent af udført arbejde. Dette er en feature der også findes MS Project og bruges for at følge med i fremgangen af vedligeholdelsesarbejdet. I en mere avanceret model

denne information kunne bruges for at udtrække data som f. eks. effektivitet af enkelte entreprenører ved forskellige typer af vedligeholdelse.

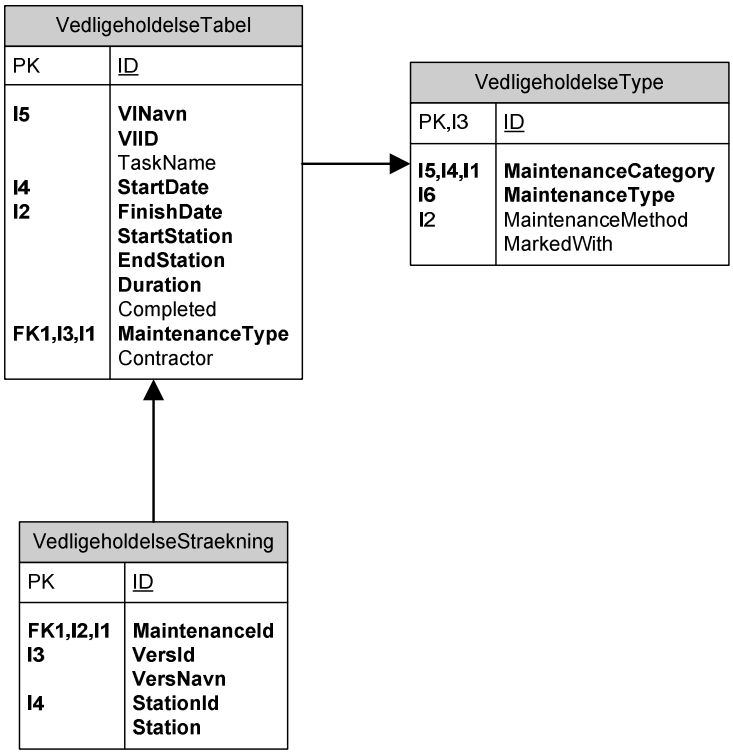
- Oplysninger om enkelte entreprenører som kan hyres til en vedligeholdelse opgave.

Det er også de samme attributter som "VedligeholdelseTabel" tabellen består af.

Udover start og slut stationer der skal gemmes alle "mellem stationer". For at undgå redundante data i " VedligeholdelseTabel" tabellen er det nødvendigt at gemme tilhørende stationer i en separat tabel. Derfor tabellen "VedligeholdelseStraekning" indeholder alle stationer med tilhørende opmålingsdata, der udgør en vedligeholdelsestrækning.

Typen af vedligeholdelse er karakteriseret ved at den tilhører en bestemt kategori.

Den nedenstående figur viser den aktuelle opbygning.



Figur 14: Vedligeholdelse datamodel

6.3.7 Løsningsmodel for data validering

Den tidligere omtalte situation hvor data i "VIIdent", "TopoVersListe" og "TopoStationer" tabellerne bliver overskrevet, medfører at data i "VedligeholdelseTabel" og "VedligeholdelseStraekning" tabellerne indeholder ugyldige / ikke eksisterende værdier. Når disse værdier skal opdateres skal dette udføres i følgende rækkefølge. Der skal bemærkes at alt indholdet af disse to tabeller skal være tilgængeligt lokalt i en DataSet, da dette vil resultere i den bedste ydelse.
Samtidig vil det reducere pres på serveren i tilfælde hvor der anvendes SQL server.

1. Som det første skal "VIId" i "VedligeholdelseTabel" opdateres.
2. Derefter skal alle "VersId" i "VedligholdelseStraekning" opdateres.
3. Og til sidst skal alle "StationId" i "VedligeholdelseStraekning" opdateres.

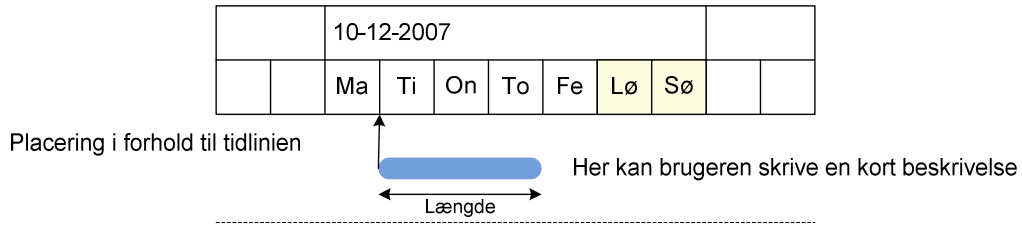
7 Test

Der blev udført en række test løbende, mens programmet var stadigvæk under udviklingen. Disse test kan beskrives som debugging sessioner som ikke er dokumenteret. For at udføre *Renderings* test som er omtalt i næste afsnit, skal der anvendes Visual Studio.

7.1 Renderings test

Formålet med denne test er at bevise at den grafiske element der repræsenterer en vedligeholdelse tegnes korrekt. Et element er tegnet korrekt når:

- Elementets startdato svarer til dato på reference elementet, dvs. tidslinjen.
- Elementets længde svarer til varighed af vedligeholdelsen.
- Elementets farve svarer til typen af vedligeholdelse den repræsenterer.
- Tekst der følger efter elementet svarer til beskrivelse for den vedligeholdelse den repræsenterer.



For denne test skal der defineres følgende test parametre:

- Start og slut dato for tidslinjen.
- Tegnemåden mode.

Testforløb

Når man kender start og slut datoen for en vedligeholdelse kan der beregnes startpunktet og størrelsen for et enkelt element ud fra de aktuelle indstillinger for tidslinjen. Disse værdier skal sammenlignes med aktuelle værdier mens programmet kører. Den letteste måde at aflæse disse værdier er at stoppe programmet efter behov ved at indsætte *breakpoints* de rigtige steder i programmet og aflæse værdier for de aktuelle variabler.

7.2 Funktionel test

Følgende funktioner kan testes:

- Opret en ny vedligeholdelse.
- Sorter vedligeholdelser.
- Filtrerer vedligeholdelse.

For at teste at en vedligeholdelse er oprettet korrekt skal følgende kontrolleres. At data blev gemt korrekt i databasen og at elementet er tegnet korrekt på skærmen. Det sidstnævnte er beskrevet i foregående afsnit.

Det er muligt at kontrollere at data er gemt korrekt på følgende måde. Til denne test skal der bruges en database hvor der ikke er defineret nogen vedligeholdelser. Derefter skal der oprettes 3 – 5 vedligeholdelser og deres egenskaber skal noteres. Programmet skal lukkes nu. Når programmet startes op igen og vedligeholdelsesdata fra databasen er indlæst skal værdier for disse vedligeholdelser nu sammenlignes med værdierne der blev noteret da vedligeholdelsen blev oprettet.

Sorterings og filtrerings test skal udføres på tilsvarende måde. Det går ud på at oprette nogle test vedligeholdelser og derefter observerer hvordan programmet opfører sig når f. eks. der vælges kun en vedligeholdelsestype.

8 Iterationer

Programmet har været igennem flere iterationer og her vil blive nævnt de vigtigste milepæle i forløbet.

Først blev de grafiske funktioner og layout implementeret.

Iteration	Resultat	Beskrivelse
1	Applikations layout	Det overordnede layout blev fastsat.
2	*DataTemplated ListView*	En *ListView* komponent hvor data vises i flere koloner ligesom i en tabel.
3	Tidslinje ver. 1	Det første udgave at *TimeLine* komponenten. Denne var en statisk kontrol hvor ingen af parametrene kunne ikke modificeres.
4	Grafik komponenten ver. 1	Implementeret *Content*, komponenten der tager sig af rendering i sin første udgave.
5	Grafik komponenten ver. 2	*Content* komponenten har nu opdaterings funktioner. F. eks når en vedligeholdelse bliver slettet skal det også slettes fra *Content*.
6	Tidslinje version 2	Forbedret udgave af *TimeLine* komponenten. Alle parametre kan modificeres mens programmet kører.
7	Grafik komponenten ver. 3	En *ListBox* erstatter *Content* koponenten.

Derefter blev data funktioner implementeret.

8	Funktioner i data laget	Database setup og funktioner til at hente og skrive data til databasen.

I det sidste iteration blev data modellen modificeret således at databehandling skal foregå på et baggrunds tråd.

9 Det færdige produkt

Programmet i sin nuværende form er kun en prototype. Funktioner der er implementeret udføres korrekt og programmets ydelse er ganske pæn. Selv med en forholdsvis stor datamængde (1000 autogenererede vedligeholdelser) og på en ældre maskine, vil det ikke tage ret lang tid at behandle og rendere data. Brugeren vil blot bemærke en kort forsinkelse på 1-2 sekunder. På en nyere PC med en dual-core CPU, vil data blive vist med det samme. Sortering og filtrerings-funktioner fungerer også pænt.

9.1 System krav (System requirements)

Følgende krav skal være opfyldt for at kunne afvikle Planlægningsværktøj til PROKA.

- Styresystem:
  - Windows Vista, Windows XP SP2 eller Windows 2003 Server SP1
- .Net Framework 3.5 skal være installeret.
- Hardware
  - Der gælder samme krav som for Windows XP SP2.
    Se mere på: http://www.microsoft.com/windowsxp/sp2/sysreqs.mspx.
  - Den anbefalede konfiguration (som minimum):
    - CPU: 1.2 GHz.
    - RAM: 512 MB
    - 64 MB video hukommelse.

9.2 Funktioner (Features)

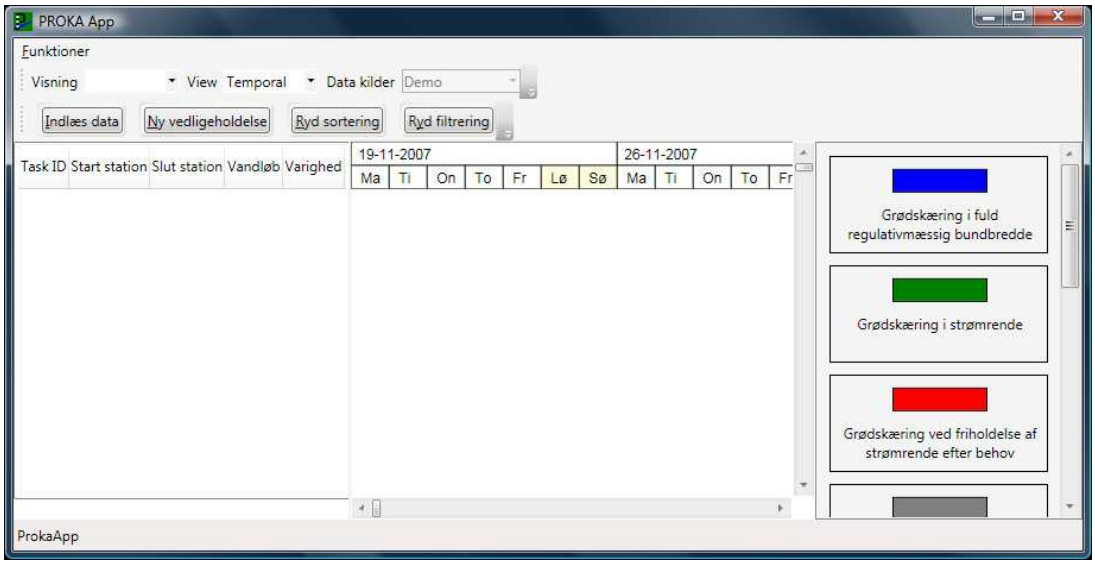
Nedenstående tabel giver en oversigt over implementerede vs. funktioner der mangler.

Funktion	Implementeret	Skal implementeres
Skift visningstype	☑	
Skift start eller slut datoen for den valgte visning	☑	
Tegn vedligeholdelse efter typen i en bestemt farve	☑	
Opret vedligeholdelse	☑	
Slet vedligeholdelse	☑	
Læs data	☑	
Læs data selektivt		☑
Sorter efter vandløbsnavn	☑	
Filtrer efter typen	☑	
Filtrer efter kategori		☑
Opret ny data forbindelse		☑
Rediger data forbindelse		☑
Skift data forbindelsen mens programmet kører		☑
Opret en ny vedligeholdelsestype		☑
Opret en ny vedligeholdelseskategori		☑

På det nuværende tidspunkt hvor applikationens struktur er fastsat og en stor del af funktionerne implementeret og testet, kan nye funktioner implementeres på kort tid.

9.3 Brugervejledning

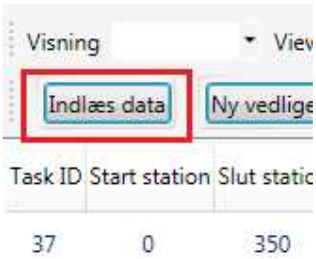
Figur 15: Applikationens hovedvindue viser programmets hovedvindue. Der skal bemærkes at i dette tilfælde er der allerede oprettet en forbindelse til databasen da indholdet af panelet yderst til højre er initialiseret med alle vedligeholdelsestyper der findes i databasen.



Figur 15: Applikationens hovedvindue

9.3.1 Indlæs data fra databasen

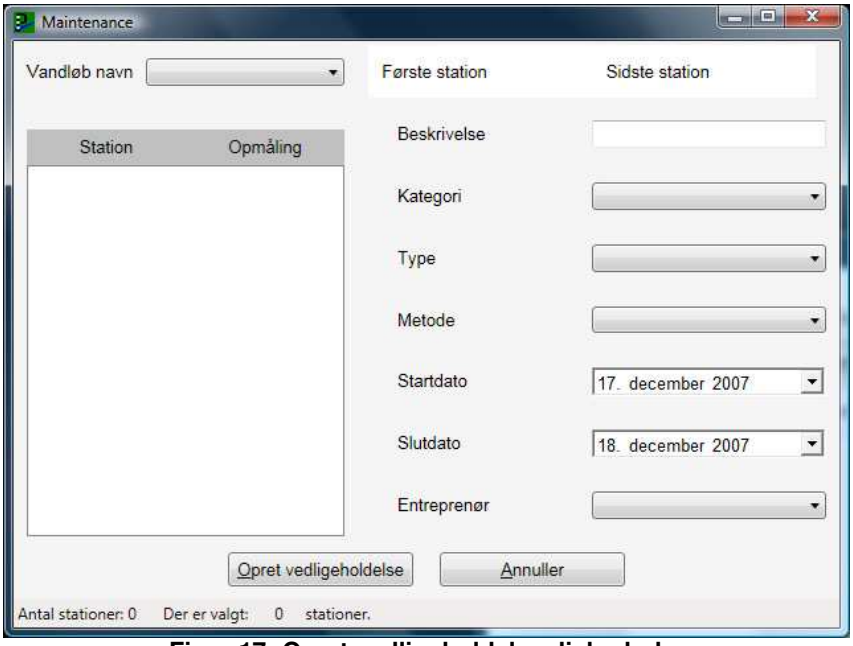
Ved at klikke på knappen ”Indlæs data” bliver alle vedligeholdelser fra databasen indlæst.



Figur 16: Indlæs data

9.3.2 Opret vedligeholdelse

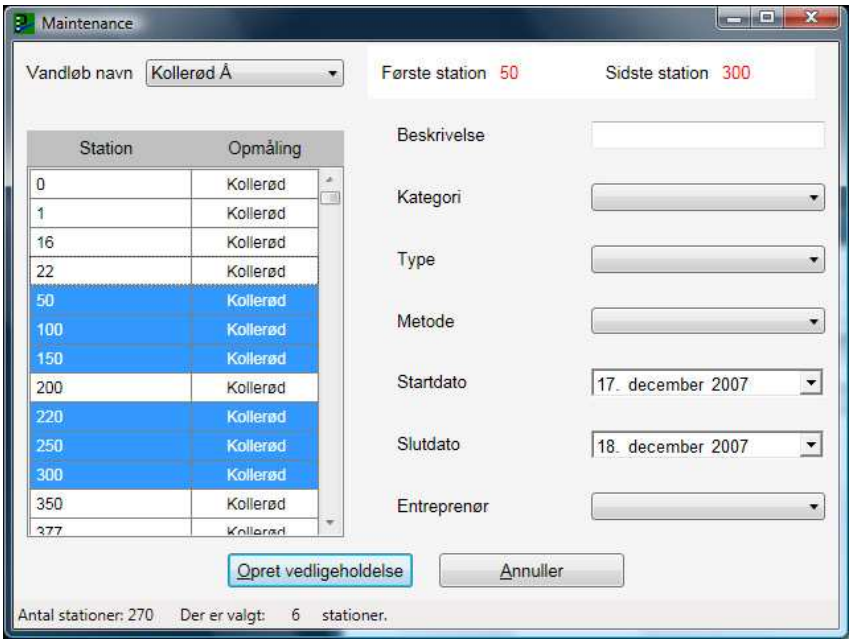
Figur 17 viser den dialogboks der skal benyttes for at oprette en ny vedligeholdelse. Dialogboksen bliver vist når brugeren klikker på ”Ny vedligeholdelse” knappen.



Figur 17: Opret vedligeholdelse dialogboks

For at oprette en ny vedligeholdelse skal følgende felter udfyldes:

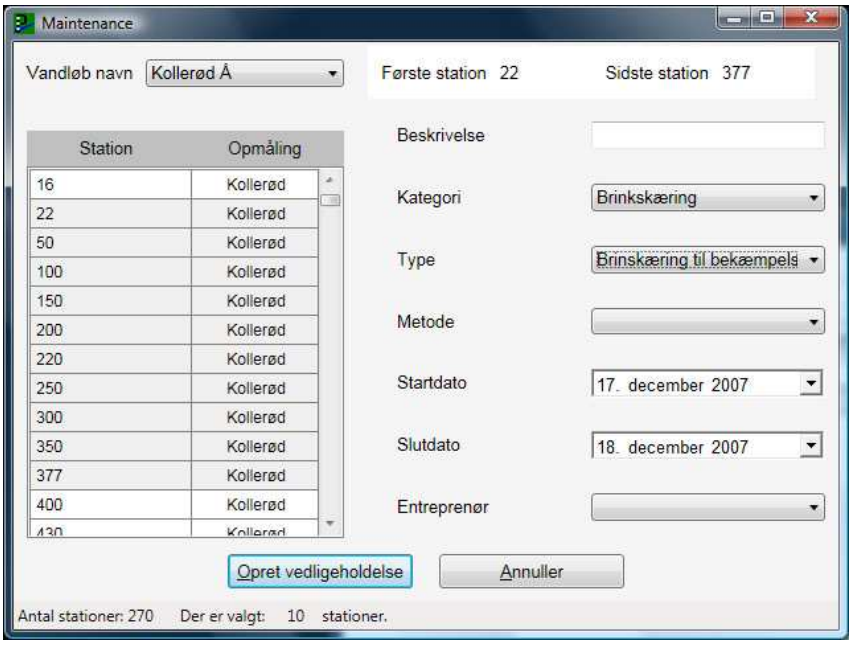
- Der skal vælges et vandløb. Alle vandløbene der findes i databasen kan også findes i indholdet af comboboksen i den venstre side, ved siden af "Vandløb navn" mærket.



Figur 18: Ugyldig vedligeholdelsesstrækning

- Når brugeren har valgt et vandløb vil listekomponenten i den venstre side vise alle stationer der findes for det valgte vandløb. Derefter kan brugeren vælge start og slut stationer. En let måde at gøre dette på er at markere den første station og derefter ved at holde "SHIFT" knappen og markere den sidste station vil alle mellem stationer blive inkluderet. Dette er også et krav for en vedligeholdelsesstrækning, at der ikke er "huller" mellem start og slut stationer. Hvis der er et hul mellem stationer vil tallene ved siden af "Første station" og "Sidste station" mærkerne være røde. Dette er illustreret på Figur 18. Hvis brugeren har defineret en gyldig vedligeholdelsesstrækning vil disse tal være sorte.

- Der skal defineres en kategori og en type af vedligeholdelse. Indholdet af vedligeholdelsestype comboboksen er afhængigt af den valgte vedligeholdelseskategori. Derfor er det også et krav at vælge en kategori først.



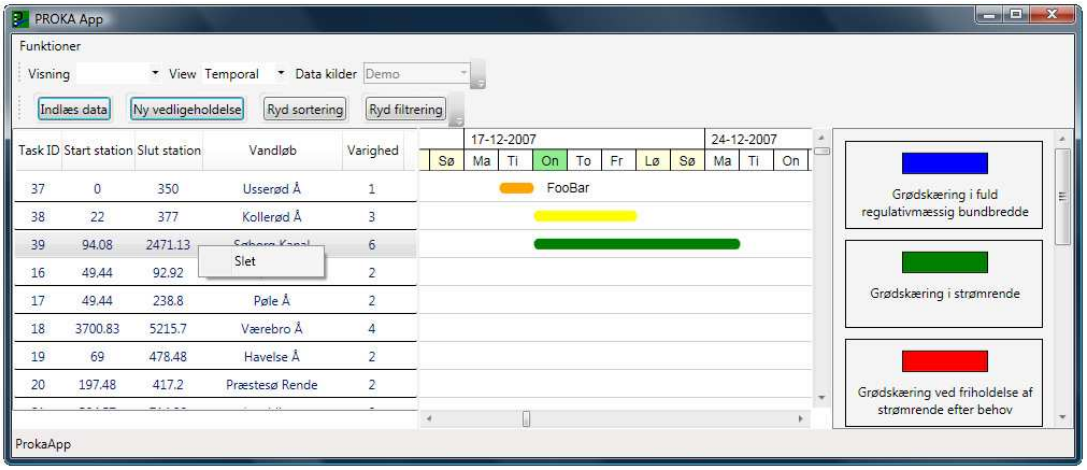
- Det sidste brugeren skal gøre er at definere vedligeholdelsesperioden. Hun / han kan vælge en vilkårlig periode, ved at definere start og slutdato.

Der kan tilføjes en beskrivelse for en vedligeholdelse men det er ikke et krav. Hvis brugeren vil tilføje en beskrivelse til vedligeholdelsen, kan hun / han skrive det i tekstboksen. Der skal bemærkes at denne beskrivelse kan være på max. 255 tegn.

Comboboksene ”Metode” og ”Entreprenør” har ingen funktion i øjeblikket.

9.3.3 Slet vedligeholdelse

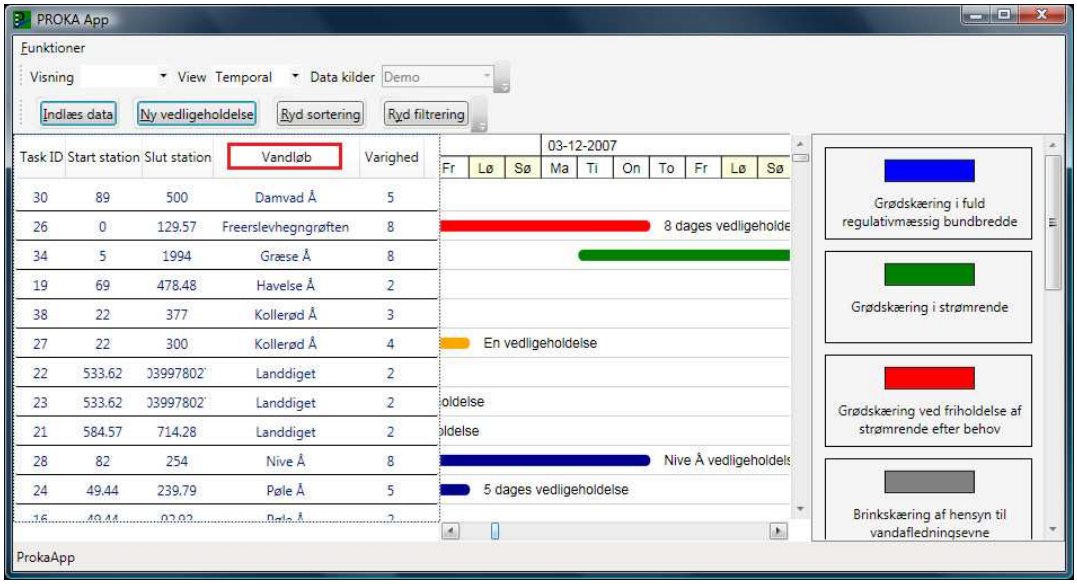
Brugeren kan slette en vedligeholdelse ved at højreklikke på et vilkårligt element i listekomponenten i højre side. Der vil vises en menu med kun et element, ”Slet”. Ved at klikke på denne bliver den valgte vedligeholdelse slettet. Dette er illustreret på Figur 19.



Figur 19: Slet vedligeholdelse

9.3.4 Sorter vedligeholdelser

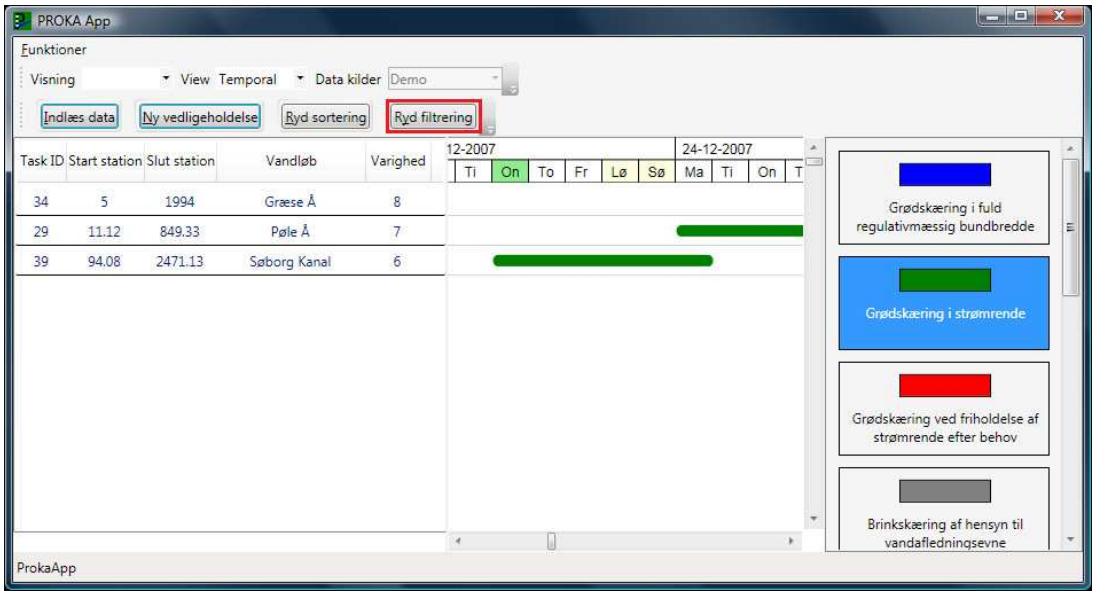
For at sortere vedligeholdelser skal brugeren klikke på ”*header*” elementet mærket ”Vandløb”, på listekomponenten i den venstre side. Dette vil resultere i at vedligeholdelser bliver sorteret i alfabetisk rækkefølge. For at vise vedligeholdelser i den oprindelige rækkefølge kan knappen ”Ryd sortering” benyttes.
I øjeblikket er det muligt at sortere vedligeholdelser kun efter vandløbsnavn. Andre ”*header*” elementer har ingen funktion.



Figur 20: Sorterede vedligeholdelser

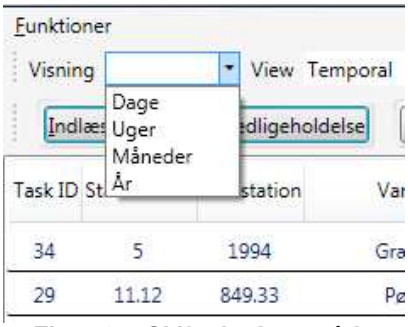
9.3.5 Filtrer vedligeholdelser

Dette foregår på den måde at brugeren vælger et af elementerne i listen yderst til højre. Dette medfører at kun vedligeholdelser der svarer til den valgte type vil blive vist. For at vise alle vedligeholdelser igen skal brugeren klikke på knappen ”Ryd filtrering”. Denne knap er markeret med et rødt rektangel på det viste billede.



9.3.6 Skift visningsmåde

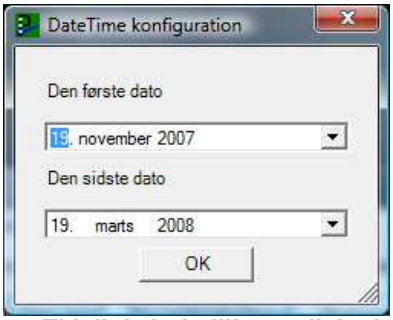
For at skifte visningsmåde skal brugeren vælge en af valgmulighederne i comboboksen ved siden af "Visning" mærket. Hvordan dette ser ud er illustreret på nedenstående diagram.



Figur 21: Skift visningsmåde

9.3.7 Skift indstillinger for tidslinjen

For at skifte indstillinger for tidslinjen skal brugeren dobbeltklikke på tidslinjen. Dialogboksen der er vist på det nedenstående billede kommer frem. Brugeren kan nu vælge en vilkårlig periode for tidslinjen.



Figur 22: Tidslinje indstillinger dialogboksen

9.4 Installation

Da programmet i øjeblikket er konfigureret med en standard indstilling, hvor en database forbindelse indlæses i starten, skal brugeren sørge for at database filen findes. Det er en MS Access fil der hedder "testproka.mdb" og applikationen "forventer" at den findes i mappen **C:\Data**.
Den anden mulighed er at redigere ProkaApp.exe.config filen, som ligger i samme mappe som ProkaApp.exe. Det er en almindelig tekstbaseret fil og kan åbnes med Notepad. Der skal man redigere følgende linje:

```
<add name="Demo" connectionString="Provider=Microsoft.Jet.OLEDB.4.0; Data Source=
      C:\Data\testproka.mdb" providerName="System.Data.OleDb" />
```

Den del af teksten der er markeret med gul farve angiver stien til database filen. Ved at redigere denne sti kan brugeren placere database filen efter sit ønske.

9.5 Kendte mangler

Der er dog to "huller" i den nuværende implementering der skal lappes inden man kan stemple programmet som færdigt.

Det ene er at programmet har udvist mærkelig opførsel når det afvikles under Windows XP. Denne mærkelige opførsel manifesteres på følgende måde. Normalt er liste elementer i *ListView* komponenten i panelet yderst til venstre og liste elementer i *ListBox* komponenten på grafik panelet i midten sat til at have samme højde. Når programmet afvikles under Windows XP med denne indstilling, opstår der forskydning mellem liste elementer i disse to komponenter. Derfor skal højden for et liste element i grafik panelet sættes til en værdi der er en mindre end højden af listeelementer i panelet yderst til venstre, når programmet afvikles under XP. Dette skal gøres hver gang der oprettes et nyt liste element på grafik panelet, da det er muligt at bestemme styresystemets version mens programmet kører.
Yderligere er det kommet frem at programmet er specielt "allergisk" over for Windows XP Themes. Når programmet afvikles under Windows XP hvor der anvendes XP Theme bliver headeren for listen i panelet yderst til venstre mindre end når det samme program afvikles under Vista eller Windows XP med Classic Theme. Det medfører at der opstår forskydning mellem liste elementer i panelet yderst til venstre og liste elementer i grafik komponenter.
Derfor skal der udvikles en metode til at detektere hvilket theme der anvendes når programmet afvikles under XP. Og hvis det er XP Theme skal liste indhold for denne komponent flyttes længere ned så den passer med indholdet af den anden komponent.
Ingen af disse problemer eksisterer under Windows Vista.

Den anden sag er vertikal scrolling. Der kan opstå forskydning mellem liste elementerne i to paneler når indholdet scrolles helt i bund. Det samme forskydning forsvinder i det samme øjeblik indholdet flyttes længere op. Dette skyldes manglende præcision i den algoritme der skal beregne hvor langt indholdet skal scrolles. En mere præcis algoritme vil løse dette problem.

9.6 Forbedringer

I en kommerciel udgave af programmet vil det være nødvendigt at tage hensyn til at lørdage og søndage er weekend dage. Sådan som det er implementeret nu er lørdage og søndage almindelige arbejdsdage. Det resulterer i at en vedligeholdelse der starter på onsdag i en hvilken som helst uge og slutter på tirsdag ugen efter, vil have en varighed på 7 dage da applikationen regner lørdag og søndag med som arbejdsdage.

En anden smart funktion vil være at tidslinjen tilpasser sin størrelse i forhold til den tilgængelige plads i vinduet. Denne funktion eksisterer i MS Projekt hvor man kan definere at vise alle opgaver for en bestemt periode, f. eks tre uger. Derefter vil tidslinjen beregne sin størrelse ud fra den ønskede start og slutdato, samt vinduets størrelse, således at hele perioden kan vises i vinduet

10 Konklusion

Der er blevet udviklet en løsning som implementerer funktioner defineret i projektets kravspecifikation. Denne løsning er stadigvæk på prototype stadium, men danner et solidt grundlag for videreudvikling. Kodemængden der blev produceret i projektperioden er også pæn, ca. 4700 linjer kode. Testen viste at programmet fungerer efter hensigten, men der er altid plads til forbedringer.

Jeg synes at beslutningen om at lave denne applikation ved hjælp af WPF var 100 % rigtigt. WPF har vist sin styrke og når det gælder udvikling af Windows applikationer der skal have en stærk brugerflade vil WPF være mit foretrukne framework.

Mit kendskab til WPF var minimal forud dette projekt, men efter *learning by doing* princippet gik det ret godt. Det kan dog være at dette har skabt en mindre forsinkelse i projektets startfase som har resulteret i at udviklingsfasen har taget længere tid end planlagt.

11 Bilag

11.1 Litteraturliste

Nathan, A. (2006). *Windows Presentation Foundation Unleashed.* Sams.

Sceppa, D. (2006). *Programming Microsoft® ADO.NET 2.0 Core Reference 2.Ed.* Microsoft Press.

Troelsen, A. (2007). *Pro C# with .NET 3.0, Special Edition.* Apress.

Larman, C. (2004). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition).* Prentice Hall.

11.2 Ordforklaring

- ADO.NET

  ADO.NET er en samling af software komponenter som programmører der udvikler software til .NET platform kan bruge til at tilgå datakilder.

  Gode artikler om ADO.NET kan findes hos Microsoft og Wikipedia.
  http://msdn2.microsoft.com/en-us/library/aa286484.aspx
  http://en.wikipedia.org/wiki/ADO.NET

- Gantt diagram

  En udmærket gennemgang og eksempler kan findes på:

  http://en.wikipedia.org/wiki/Gantt_chart.

  eller

  http://searchsoftwarequality.techtarget.com/sDefinition/0,,sid92_gci331397,00.html.

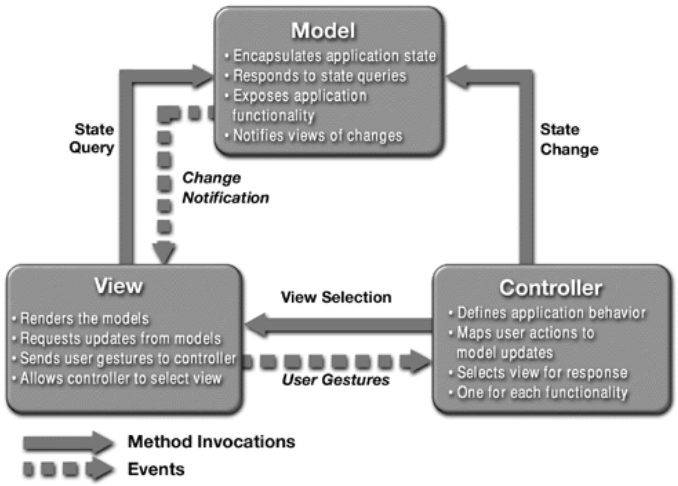
- Model View Controller (MVC)

  Der findes rigtigt mange artikler om MVC på nettet, men her er de bedste:

  http://en.wikipedia.org/wiki/Model-view-controller

  http://msdn2.microsoft.com/en-us/library/ms978748.aspx

  Den nedenstående figur viser strukturen for Model View Controller pattern.



Figur 23: Model View Controller

- Rendering

  I denne rapport bliver begrebet rendering brugt til at beskrive en proces hvor en form for data omdannes til en grafisk struktur på skærmen.

  En artikel om dette emne hos Wikipedia

  http://en.wikipedia.org/wiki/Rendering_(computer_graphics)

11.3 Use cases

11.3.1 Use case ”Opret vedligeholdelse”

USE CASE $0001	Opret vedligeholdelse.	
Formål:	Der er behov for at udføre en bestemt type vedligeholdelsesarbejde på et specificeret vandløb under brugerens administration.	
Primær aktør:	Personen som ansvarlig for vandløbsadministration hos en kommune.	
Forudsætninger:	Ingen.	
Succesforløb:	Brugeren har udfyldt de nødvendige felter i en dialogboks som hun / han skal benytte for at oprette en ny vedligeholdelse. Efterfølgende blev denne vedligeholdelse gemt i databasen og vist på brugerens skærm.	
Hovedforløb:	Step	Action
	1	Brugeren skal vælge et vandløb.
	2	Brugeren skal udpege to stationer der skal danne en sammenhængende vedligeholdelsesstrækning.
	3	Brugeren skal angive vedligeholdelseskategori.
	4	Brugeren skal angive vedligeholdelsestype.
	5	Brugeren skal angive startdatoen for vedligeholdelse.
	6	Brugeren skal angive slutdatoen for vedligeholdelse.
	7	Brugeren kan vælge at skrive en kort beskrivelse.
Prioritet:	Høj.	
Hyppighed:	Meget varierende.	
Specielle krav:	Ingen.	
Referencer:	Ingen	

11.3.2 Use case ”Skift tidslinjeindstillinger”

USE CASE $0002	Skift tidslinjeindstillinger	
Formål:	Brugeren ønsker at vise alle vedligeholdelser for en bestemt periode.	
Primær aktør:	Personen som ansvarlig for vandløbsadministration hos en kommune.	
Forudsætninger:	Ingen.	
Succesforløb:	Brugeren har skiftet indstillinger for tidslinjen efter sit ønske. Tidslinjen blev opdateret og alle vedligeholdelser for denne periode blev vist.	
Hovedforløb:	Step	Action
	1	Brugeren dobbeltklikker på tidslinjen.
	2	Den dialogboks der bruges til at justere tidslinjens indstillinger kommer frem.
	3	Brugeren kan nu angive nye værdier for start og slut.
	4	Når brugeren har klikket på ”OK” knappen skal dialogboksen lukkes og tidslinje skal gentegne sig selv i henhold til de nye indstillinger.
Prioritet:	Høj.	
Hyppighed:	Formodes stor.	
Specielle krav:	Ingen.	
Referencer:	Ingen.	

11.3.3 Use case ”Skift visning”

USE CASE $0003	Skift visning	
Formål:	Brugeren ønsker at vise alle vedligeholdelser for en bestemt periode.	
Primær aktør:	Personen som ansvarlig for vandløbsadministration hos en kommune.	
Forudsætninger:	Ingen.	
Succesforløb:	Brugeren har skiftet visning og det har medført at tidslinjen blev opdateret og alle vedligeholdelser for den aktuelle periode blev vist.	
Hovedforløb:	Step	Action
	1	Brugeren kan vælge mellem fire visninger: ”Dage”, ”Uger”, ”Måneder” og ”År”
	2	Efter brugeren har valgt en af de fire visninger, tidslinjen blev vist i et format der svarer til den valgte visning.
Prioritet:	Høj.	
Hyppighed:	Formodes stor.	
Specielle krav:	Ingen.	
Referencer:	Ingen.	

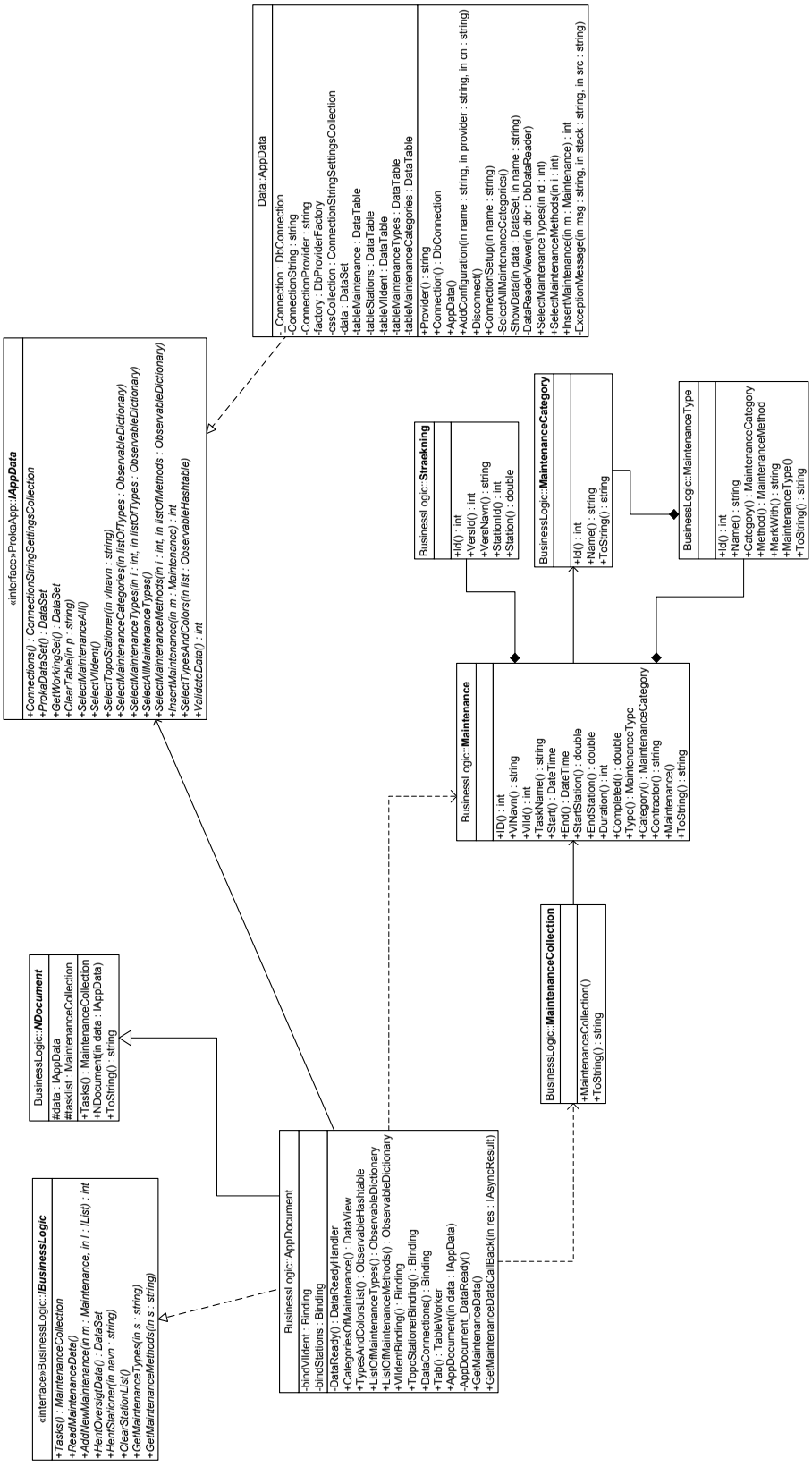
11.3.4 Use case ”Filtrer vedligeholdelse”

USE CASE $0004	Filtrer vedligeholdelse	
Formål:	Brugeren er interesseret i at få vist kun en bestemt vedligeholdelsestype.	
Primær aktør:	Personen som ansvarlig for vandløbsadministration hos en kommune.	
Forudsætninger:	Ingen.	
Succesforløb:	Brugeren har mulighed for at vælge en bestemt vedligeholdelsestype i en liste. Når en bestemt type er markeret vil kun vedligeholdelser det svarer til den valgte type vises.	
Hovedforløb:	Step	Action
	1	Brugeren har valgt en vedligeholdelsestype i en liste.
	2	Der vises kun vedligeholdelser der svarer til den valgte type.
Prioritet:	Høj.	
Hyppighed:	Formodes stor.	
Specielle krav:	Ingen.	
Referencer:	Ingen.	

11.4 Klassediagram

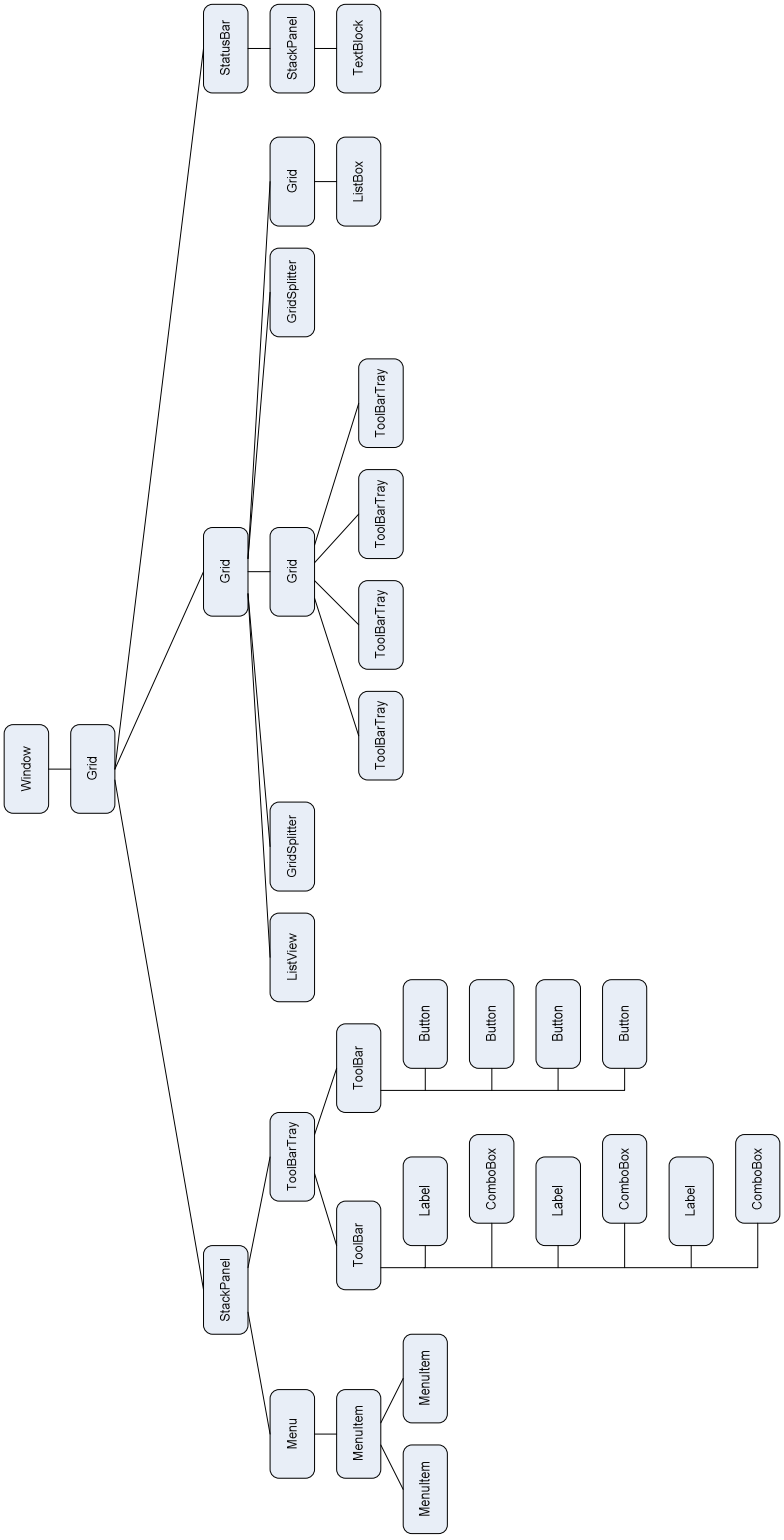
Udvalgte klasser fra brugerflade laget.

Vigtigste klasser fra logik og datalaget.



11.5 Struktur diagram

Dette diagram viser hvilke WPF komponenter blev brugt til at opbygge applikationens brugerflade og komponenternes placering i forhold til hinanden.



Figur 24:Struktur diagram

11.6 CD indhold

På den vedlagte CD findes:

- Kildekoden.
- Flere udgaver (dette bliver afklaret om lidt) af ProkaApp.exe fil.
- PROKA database.
- .NET Framework 3.5 der er nødvendig for at kunne køre applikationen.

I mappen ”Data” ligger PROKA database.

Kildekoden ligger i mappen ”Projekt”. Den indeholder projektets workspace og den kan åbnes med Visual Studio 2008. Der anbefales at kopiere denne mappe til den lokale disk hvis man vil studere koden i Visual Studio eller udføre debugging.

I mappen ”Bin” findes der følgende udgaver af programmet:

- NoTheme.
- Theme.

Disse to udgaver er på den funktionelle plan fuldstændig ens. Den eneste forskel er at udgaven i mappen Theme vil afvikles korrekt kun på Windows XP hvor der anvendes XP Theme. NoTheme udgave vil afvikles korrekt på Windows XP hvor Theme er sat til ”Classic” og Windows Vista. Årsagen til det er beskrevet i afsnit 9.5 Kendte mangler.
Endvidere findes der tre mapper i både NoTheme og Theme: DB, Demo100 og Demo1000.
I mappen DB er der en udgave der kommunikerer med PROKA’s database.
I mappen Demo100 er der en udgave der henter data fra en XML fil hvor der er defineret 100 vedligeholdelser.
I mappen Demo1000 er der en udgave der henter data fra en XML fil hvor der er defineret 1000 vedligeholdelser.

11.7 Kildekode

Indhold

App.xaml

```
<Application x:Class="ProkaApp.App"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Startup="AppStartup"
    Exit="Application_Exit">
    <Application.Resources>
        <ToolTip x:Key="timelineTip">
            <Grid>
                <Grid.RowDefinitions>
                    <RowDefinition />
                    <RowDefinition />
                </Grid.RowDefinitions>
                <Border Background="DarkBlue" Height="20" Margin="2">
                    <TextBlock Foreground="White" Text="TimeLine Tip" TextWrapping="Wrap"
                               HorizontalAlignment="Left" VerticalAlignment="Center"
                               Margin="5, 0, 0, 0"/>
                </Border>
                <Label Grid.Row="1" Content="Klik for at skifte indstillinger"
                       Margin="10, 10, 10, 10" />
            </Grid>
        </ToolTip>
     </Application.Resources>
</Application>
```

App.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Windows;

namespace ProkaApp
{
    /// <summary>
    /// Interaction logic for App.xaml
    /// </summary>
    public partial class App : Application
    {
        private ProkaAppCore AppCore;
        private void Application_Exit(object sender, ExitEventArgs e)
        {
            string cap     = "Application terminated abnormally";
            string errCode = "";
            int s = e.ApplicationExitCode;
            if (s != 0)
            {
                errCode = String.Format("Error code: {0}", s);
                MessageBox.Show(errCode, cap, MessageBoxButton.OK, MessageBoxImage.Warning);
            }
#if DEBUG
            MessageBox.Show("Application exit", "Info",
                MessageBoxButton.OK, MessageBoxImage.Information);
#endif
        }

        private void AppStartup(object sender, StartupEventArgs e)
        {
#if DEBUG
            ShowCapability();
#endif
            // Initialize app structure
            // Application data
            // Aplication logic(data)
            // Application UI (logic)
            AppCore = new ProkaAppCore();
            ProkaApp.MainWindow wnd = new MainWindow { Document = AppCore.Doc };
            wnd.Show();
```

```
        }

        private void ShowCapability()
        {
#if SHOW_CAPABILITY
            int cap = RenderCapability.Tier >> 16;
            MessageBox.Show(String.Format("{0}", cap), "RenderCapability.Tier",
                MessageBoxButton.OK, MessageBoxImage.Information);
#endif
        }
    }
}
```

AppData.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections;
using System.Collections.ObjectModel;
using System.Data;
using System.Data.Common;
using System.Data.OleDb;
using System.Data.SqlClient;
using System.Configuration;
//
namespace ProkaApp.Data
{
   class AppData : IAppData
   {
      private DbConnection _Connection; //
      private String ConnectionString;    // = Provider=StrProvider; Data Source=StrDataSource;
      private String ConnectionProvider;  // = "System.Data.OleDb || System.Data.SqlClient";
      private DbProviderFactory factory;
      private ConnectionStringSettingsCollection cssCollection;
      //
      public String Provider { get { return ConnectionProvider; } }
      public DbConnection Connection { get { return _Connection; } }
      public ConnectionStringSettingsCollection Connections { get { return cssCollection; } }
      //
      public DataSet ProkaDataSet { get { return data; } }
      private DataSet data;
      private DataTable tableMaintenance;
      private DataTable tableStations;
      private DataTable tableVlIdent;
      private DataTable tableMaintenanceTypes;
      private DataTable tableMaintenanceCategories;
        //
      public AppData()
      {
         data = new DataSet("ProkaData");
            tableMaintenance    = new DataTable("VedligeholdelseTabel");
         tableStations      = new DataTable("Topostationer");
         tableVlIdent       = new DataTable("VlIdent");
         tableMaintenanceTypes      = new DataTable("VedligeholdelseType");
         tableMaintenanceCategories = new DataTable("VedligeholdelseKategori");
            data.Tables.Add(tableMaintenance);
         data.Tables.Add(tableStations);
         data.Tables.Add(tableVlIdent);
         data.Tables.Add(tableMaintenanceTypes);
         data.Tables.Add(tableMaintenanceCategories);
         //
         cssCollection = ConfigurationManager.ConnectionStrings;
            /*     */

         ConnectionSetup("Demo");
         if (Connection != null)
         {
            // Load all from VlIdent
            SelectVlIdent();
            SelectAllMaintenanceCategories();
```

```
            }
        }
          //
          public void SelectMaintenanceAll()
          {
#if DEBUG
            int code = System.Threading.Thread.CurrentThread.GetHashCode();
            System.Diagnostics.Debug.WriteLine(String.Format("Starting read on thread {0}: ",
code));
            DateTime start = DateTime.Now;
#endif
            String sqlCmd = "SELECT * FROM VedligeholdelseTabel";
            using (DbConnection con = factory.CreateConnection())
            {
                try
                {
#if PERFORM_LONG_READ
                    System.Threading.Thread.Sleep(10000);
#endif
                    ClearTable("VedligeholdelseTabel");

                    DbCommand cmd = factory.CreateCommand();
                    DbDataAdapter adapter = factory.CreateDataAdapter();
                    con.ConnectionString = this.ConnectionString;
                    cmd.Connection = con;
                    cmd.CommandText = sqlCmd;
                    cmd.CommandType = CommandType.Text;
                    adapter.SelectCommand = cmd;
                    adapter.Fill(tableMaintenance);
                    //data.WriteXml("DataSet.xml");
                }
                catch (Exception ex)
                {
                    throw ex;
                }
                finally
                {
                    con.Close();
                }
#if DEBUG
                DateTime stop = DateTime.Now;
                TimeSpan ts = stop - start;
                System.Diagnostics.Debug.WriteLine(String.Format("Read performed in: {0}",
ts.Seconds));
#endif
            }
          }
      //
      public void AddConfiguration(string name, string provider, string cn)
      {
         try
         {
            ConnectionStringSettings css;
            css = new ConnectionStringSettings(name, cn, provider);
            Configuration cfg =
            ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);
            ConnectionStringsSection csSection = cfg.ConnectionStrings;
            csSection.ConnectionStrings.Add(css);
            cfg.Save(ConfigurationSaveMode.Modified);
         }
         catch (ConfigurationErrorsException e)
         {
            string msg = e.Message;
            AppMessageBox.ShowError(msg, "Fejl");
         }
      }
      //
      public void Disconnect()
      {
         if (_Connection.State == ConnectionState.Open)
            _Connection.Close();
         _Connection.Dispose();
      }
      //
```

```
public void ConnectionSetup(object o)
{
}
//
public void ConnectionSetup(string name)
{
   ConnectionStringSettings css;
   try
   {
      css = ConfigurationManager.ConnectionStrings[name];
      if (css == null)
         return;
      ConnectionString = css.ConnectionString;
      ConnectionProvider = css.ProviderName;
      factory = DbProviderFactories.GetFactory(ConnectionProvider);
      _Connection = factory.CreateConnection();
      _Connection.ConnectionString = ConnectionString;
      // Test connection
      _Connection.Open();
      _Connection.Close();
   }
   catch (NullReferenceException e)
   {
      AppMessageBox.ShowError(e.Message, e.Source);
   }
   catch (ArgumentException e)
   {
      AppMessageBox.ShowError(e.Message, e.Source);
   }
}
//
public void SelectVlIdent()
{
   string Sql = ProkaApp.ProkaResources.SelectVlIdent;
   try
   {
      /**/
      using (DbConnection c = factory.CreateConnection())
      {
         c.ConnectionString = ConnectionString;
         DbCommand cmd = factory.CreateCommand();
         cmd.CommandText = Sql;
         cmd.Connection = c;
         DbDataAdapter adapter = factory.CreateDataAdapter();
         adapter.SelectCommand = cmd;
         adapter.Fill(tableVlIdent);
         //ShowData(data, "VlIdent");
      }
   }
   catch (Exception e)
   {
      AppMessageBox.ShowError(e.StackTrace, e.Message);
   }
}
//
public void SelectTopoStationer(string vlnavn)
{
   string msg;
   try
   {
      tableStations.Clear();
      tableStations.Columns.Clear();
      /**/
      if (this._Connection.State != ConnectionState.Closed)
      {
         msg = ProkaResources.ErrorMsgConnectionNotReady;
         AppMessageBox.ShowError(msg, "Fejl");
         return;
      }

      DbCommand cmd = factory.CreateCommand();
      DbDataAdapter adapter = factory.CreateDataAdapter();
      DbParameter param = factory.CreateParameter();
      param.ParameterName = "VLIdent.VlNavn";
```

```
      param.Value = vlnavn;
      cmd.Connection = this.Connection;
      cmd.CommandType = CommandType.Text;
      cmd.CommandText = ProkaResources.SelectTopostationer;
      cmd.Parameters.Add(param);
      adapter.SelectCommand = cmd;
      adapter.Fill(tableStations);
      //
      if (tableStations.IsInitialized && tableStations.Columns.Contains("TopoStationer.Id"))
      {
         tableStations.Columns["TopoStationer.Id"].ColumnName = "TopoStationerId";
      }
   }
   catch (Exception e)
   {
      AppMessageBox.ShowError(e.StackTrace, e.Message);
   }
   finally
   {
      this._Connection.Close();
   }
}
//
private void SelectAllMaintenanceCategories()
{
   string sqlCmd = "SELECT VedligeholdelseKategori.CategoryID, " +
      "VedligeholdelseKategori.CategoryName FROM VedligeholdelseKategori";
   using (DbConnection conn = factory.CreateConnection())
   {
      try
      {
         // clear old results
         this.ClearTable("VedligeholdelseKategori");
         conn.ConnectionString = this.ConnectionString;
         DbCommand cmd = factory.CreateCommand();
         cmd.CommandText = sqlCmd;
         cmd.CommandType = CommandType.Text;
         cmd.Connection = conn;
         DbDataAdapter adapter = factory.CreateDataAdapter();
         adapter.SelectCommand = cmd;
         adapter.Fill(tableMaintenanceCategories);
      }
      catch (Exception ex)
      {
         AppMessageBox.ShowError(ex.StackTrace, ex.Message);
      }
   }
}
//
public void SelectAllMaintenanceTypes()
{
      string sqlCmd = "SELECT VedligeholdelseType.ID, " +
          " VedligeholdelseType.MaintenanceType FROM VedligeholdelseType";
   using (DbConnection conn = factory.CreateConnection())
   {
      try
      {
         // clear old results
         this.ClearTable("VedligeholdelseType");
         conn.ConnectionString = this.ConnectionString;
         DbCommand cmd = factory.CreateCommand();
         cmd.CommandText = sqlCmd;
         cmd.CommandType = CommandType.Text;
         cmd.Connection = conn;
         DbDataAdapter adapter = factory.CreateDataAdapter();
         adapter.SelectCommand = cmd;
         adapter.Fill(tableMaintenanceTypes);
      }
      catch (Exception ex)
      {
         AppMessageBox.ShowError(ex.StackTrace, ex.Message);
      }
   }
}
```

```
//
public void ClearTable(string p)
{
   try
   {
      if (data.Tables.Contains(p))
      {
         data.Tables[p].Clear();
         data.Tables[p].Columns.Clear();
      }
      else
         AppMessageBox.ShowWarning(p + " findes ikke!", "Advarsel");
   }
   catch (Exception e)
   {
      AppMessageBox.ShowError(e.Message, e.Source);
   }
}
  //
  public int ValidateData()
  {
      return 0;
  }
//

  //
#region INFO_FUNCTIONS
//
private void ShowData(DataSet data, string name)
{
   //
   DataViewer frm = new DataViewer();
   frm.Tag = name;
   frm.Data = data;
   frm.ShowDialog();
}
//
private void DataReaderViewer(DbDataReader dbr)
{
   int length;
   StringBuilder sb = new StringBuilder();
   while (dbr.Read())
   {
      length = dbr.FieldCount;
      for (int i = 0; i < length; i++)
      {
         sb.AppendFormat("{0}\t{1}\n", dbr.GetValue(i).GetType(), dbr.GetValue(i));
      }
      System.Windows.MessageBoxResult res =
         System.Windows.MessageBox.Show(sb.ToString(), "Info",
         System.Windows.MessageBoxButton.OKCancel,
         System.Windows.MessageBoxImage.Information);
      if (res == System.Windows.MessageBoxResult.Cancel)
         break;
   }
}
#endregion

#region IAppData Members

public DataSet GetWorkingSet()
{
   return this.data;
}
  //
 public void SelectMaintenanceTypes(int i, ObservableDictionary listOfTypes)
  {
   string SqlCmd = ProkaResources.SelectMaintenanceTypeByID;
      using (DbConnection conn = factory.CreateConnection())
      {
          try
          {
              conn.ConnectionString = this.ConnectionString;
              DbCommand cmd = factory.CreateCommand();
```

```
                    cmd.CommandType = CommandType.Text;
                    cmd.CommandText = SqlCmd;
                    cmd.Connection = conn;
                    DbParameter param = factory.CreateParameter();
                    param.ParameterName = "VedligeholdelseKategori.CategoryID";
                    param.Value = i;
                    cmd.Parameters.Add(param);
                    conn.Open();
                    DbDataReader reader = cmd.ExecuteReader();
                    while (reader.Read())
                    {
                        listOfTypes.Add(new KeyValuePair<string,int>(reader.GetString(1),
reader.GetInt32(0)));
                    }
                }
                catch (Exception ex)
                {
#if DEBUG
                    AppMessageBox.ShowError(ex.StackTrace, ex.Message);
#else
                    AppMessageBox.ShowError(ex.Message, ex.Source);
#endif
                }
                finally
                {
                    conn.Close();
                }
            }
        }
        //
      public void SelectMaintenanceTypes(int id)
      {
         string SqlCmd = ProkaResources.SelectMaintenanceTypeByID;
         using (DbConnection conn = factory.CreateConnection())
         {
            try
            {
               ClearTable("VedligeholdelseType");
               conn.ConnectionString = this.ConnectionString;
               DbCommand cmd = factory.CreateCommand();
               cmd.CommandType = CommandType.Text;
               cmd.CommandText = SqlCmd;
               cmd.Connection  = conn;
               DbParameter param = factory.CreateParameter();
               param.ParameterName = "VedligeholdelseKategori.CategoryID";
               param.Value = id;
               cmd.Parameters.Add(param);
               DbDataAdapter adapter = factory.CreateDataAdapter();
               adapter.SelectCommand = cmd;
               adapter.Fill(tableMaintenanceTypes);
            }
            catch (Exception ex)
            {
               AppMessageBox.ShowError(ex.StackTrace, ex.Message);
            }
         }
      }
        //
        public void SelectMaintenanceMethods(int i, ObservableDictionary listOfMethods)
        {
         throw new NotImplementedException();
        }
        //
        public void SelectMaintenanceMethods(int i)
        {
            throw new NotImplementedException();
        }
        //
        public void SelectMaintenanceCategories(ObservableDictionary listOfTypes)
        {
            throw new NotImplementedException();
        }
        //
        public void SelectTypesAndColors(ObservableHashtable list)
```

```
  {
      string sqlcmd = "SELECT VedligeholdelseType.MaintenanceType," +
          " VedligeholdelseType.MarkedWith FROM VedligeholdelseType";
      using (DbConnection con = factory.CreateConnection())
      {
          try
          {
              con.ConnectionString = this.ConnectionString;
              DbCommand cmd = factory.CreateCommand();
              cmd.Connection = con;
              cmd.CommandText = sqlcmd;
              cmd.CommandType = CommandType.Text;
              con.Open();
              DbDataReader reader = cmd.ExecuteReader();
              while (reader.Read())
              {
                  if (reader.IsDBNull(1))
                  {
                      list.Add(new DictionaryEntry(reader.GetString(0), String.Empty));
                  }
                  else
                  {
                      list.Add(new DictionaryEntry(reader.GetString(0), reader.GetString(1)));
                  }
              }
          }
          catch (Exception ex)
          {
              ExceptionMessage(ex.Message, ex.StackTrace, ex.Source);
          }
          finally
          {
              con.Close();
          }
      }
  }
  //
public int InsertMaintenance(ProkaApp.BusinessLogic.Maintenance m)
{
      int result = -1;
      string sqlCmd = "INSERT INTO VedligeholdelseTabel ( VlNavn, VlID, TaskName," +
          " StartDate, FinishDate, StartStation, EndStation, Duration, Completed," +
          " MaintenanceType, Contractor ) VALUES (?, ?, ?, ?, ?, ?, ?," +
          " ?, ?, ?, ?)";
      using (DbConnection con = factory.CreateConnection())
      {
          try
          {
              con.ConnectionString = this.ConnectionString;
              DbCommand cmd   = factory.CreateCommand();
              cmd.Connection  = con;
              cmd.CommandType = CommandType.Text;
              cmd.CommandText = sqlCmd;
              //
              DbParameter par = factory.CreateParameter();
              par.Value = m.VlNavn;
              par.ParameterName = "VlNavn";
              cmd.Parameters.Add(par);
              //
              par = factory.CreateParameter();
              par.Value = m.VlId;
              par.ParameterName = "VlId";
              cmd.Parameters.Add(par);
              //
              par = factory.CreateParameter();
              par.Value = m.TaskName;
              par.ParameterName = "TaskName";
              cmd.Parameters.Add(par);
              //
              par = factory.CreateParameter();
              par.Value = m.Start;
              par.ParameterName = "StartDate";
              cmd.Parameters.Add(par);
              //
```

```
                par = factory.CreateParameter();
                par.Value = m.End;
                par.ParameterName = "FinishDate";
                cmd.Parameters.Add(par);
                //
                par = factory.CreateParameter();
                par.Value = m.StartStation;
                par.ParameterName = "StartStation";
                cmd.Parameters.Add(par);
                //
                par = factory.CreateParameter();
                par.Value = m.EndStation;
                par.ParameterName = "EndStation";
                cmd.Parameters.Add(par);
                //
                par = factory.CreateParameter();
                par.Value = m.Duration;
                par.ParameterName = "Duration";
                cmd.Parameters.Add(par);
                //
                par = factory.CreateParameter();
                par.DbType = DbType.Double;
                par.Value = m.Completed;
                par.ParameterName = "Completed";
                cmd.Parameters.Add(par);

                par = factory.CreateParameter();
                par.Value = m.Type.Id;
                par.ParameterName = "MaintenanceType";
                cmd.Parameters.Add(par);
                //
                par = factory.CreateParameter();
                par.Value = 100;
                par.ParameterName = "Contractor";
                cmd.Parameters.Add(par);
                //
                DbCommand cmdid = factory.CreateCommand();
                cmdid.Connection  = con;
                cmdid.CommandType = CommandType.Text;
                if (cmdid is OleDbCommand)
                    cmdid.CommandText = "SELECT @@IDENTITY";
                else if (cmdid is SqlCommand)
                    cmdid.CommandText =
      "SELECT * FROM VedligeholdelseTabel WHERE ID = IDENT_CURRENT('VedligeholdelseTabel')";
                else
                    throw new InvalidOperationException(cmdid.ToString() +
                                                        " is illegal connection type");
                con.Open();
                int cnt = cmd.ExecuteNonQuery();
                result  = (int)cmdid.ExecuteScalar();
                con.Close();
            }
            catch (Exception ex)
            {
                ExceptionMessage(ex.Message, ex.StackTrace, ex.Source);
            }
            finally
            {
                con.Close();
            }
        }
    return result;
}
#endregion
  //
public int DeleteMaintenance(ProkaApp.BusinessLogic.Maintenance maintenance)
{
    int res = 0;
    string sqlcmd = "DELETE VedligeholdelseTabel.*, VedligeholdelseTabel.ID " +
       "FROM VedligeholdelseTabel WHERE ((VedligeholdelseTabel.ID)=?)";
    using (DbConnection con = factory.CreateConnection())
    {
       try
       {
```

```
                con.ConnectionString = this.ConnectionString;
                DbCommand cmd = factory.CreateCommand();
                cmd.CommandText = sqlcmd;
                cmd.CommandType = CommandType.Text;
                cmd.Connection = con;
                DbParameter param = factory.CreateParameter();
                param.ParameterName = "ID";
                param.Value = maintenance.ID;
                cmd.Parameters.Add(param);
                con.Open();
                res = cmd.ExecuteNonQuery();
                con.Close();
            }
            catch (Exception exc)
            {
                ExceptionMessage(exc.Message, exc.StackTrace, exc.Source);
            }
         }
         return res;
      }
      //
        private void ExceptionMessage(string msg, string stack, string src)
        {
#if DEBUG
            AppMessageBox.ShowError(stack, msg);
#else
            AppMessageBox.ShowError(msg, src);
#endif
        }
   }
}
```

AppDocument.cs

```
//#define TEST_SET
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections;
using System.Collections.ObjectModel;
using System.Collections.Specialized;
using System.Data;
using System.Windows.Data;
using ProkaApp.Data;
using System.Windows;
//
namespace ProkaApp.BusinessLogic
{
    public interface IBusinessLogic
    {
        MaintenanceCollection Tasks { get; }
        void ReadMaintenanceData();
        int AddNewMaintenance(Maintenance m, IList l);
        void DeleteMaintenance(Maintenance maintenance);
        DataSet HentOversigtData();
        void HentStationer(string navn);
        void ClearStationList();
        void GetMaintenanceTypes(string s);
        void GetMaintenanceMethods(string s);
        void DataSetup(object o);
   }

    public abstract class NDocument
    {
        protected IAppData data;
        protected MaintenanceCollection tasklist;
        public    MaintenanceCollection Tasks { get { return tasklist; } }
        public NDocument(IAppData data)
        {
            this.data  = data;
            tasklist = new MaintenanceCollection();
        }
```

```
        public override string ToString()
        {
            String msg = "NDocument.data: " + data.ToString();
            return msg;
        }
    }
    /// <summary>
    ///
    /// </summary>
    public delegate void AsyncDataReader();
    //
    class AppDocument : NDocument, IBusinessLogic
    {
        private delegate void DataReadyHandler();
        private event DataReadyHandler DataReady;
        //
      public DataView CategoriesOfMaintenance { get; private set; }
        public ObservableHashtable  TypesAndColorsList       { get; private set; }
        public ObservableDictionary ListOfMaintenanceTypes   { get; private set; }
        public ObservableDictionary ListOfMaintenanceMethods { get; private set; }
      //
        private Binding bindVlIdent;
        public  Binding VlIdentBinding { get { return bindVlIdent; } }
        private Binding bindStations;
        public  Binding TopoStationerBinding { get { return bindStations; } }
        //
        public Binding DataConnections { get; private set; }
        /**/
        public TableWorker Tab { get; private set; }
        /**/
        public AppDocument(IAppData data)
            : base(data)
        {
            CategoriesOfMaintenance  =
                            data.ProkaDataSet.Tables["VedligeholdelseKategori"].DefaultView;
            ListOfMaintenanceTypes   = new ObservableDictionary();
            ListOfMaintenanceMethods = new ObservableDictionary();
            TypesAndColorsList       = new ObservableHashtable();
            //
            bindVlIdent  = new Binding("VlIdent");
            bindVlIdent.Source = data.ProkaDataSet;
            bindStations = new Binding("Topostationer");
            bindStations.Source = data.ProkaDataSet;
            DataConnections = new Binding();
            DataConnections.Source = data.Connections;
            //
            data.SelectTypesAndColors(TypesAndColorsList);
            data.SelectAllMaintenanceTypes();
            DataReady += new DataReadyHandler(AppDocument_DataReady);
            //
            Tab = new TableWorker();
            Tab.AData  = data;
            Tab.ATasks = Tasks;
        }
        //
        void AppDocument_DataReady()
        {
#if DEBUG
            int code = System.Threading.Thread.CurrentThread.GetHashCode();
            System.Diagnostics.Debug.WriteLine(String.Format("AppDocument_DataReady: {0}", code));
#endif
        }
        //
        public void ReadMaintenanceData()
        {
#if DEBUG
            int code = System.Threading.Thread.CurrentThread.GetHashCode();
            System.Diagnostics.Debug.WriteLine(String.Format("Call on thread: {0}", code));
#endif
            AsyncDataReader aread = new AsyncDataReader(GetMaintenanceData);
            IAsyncResult res = aread.BeginInvoke(new AsyncCallback(GetMaintenanceDataCallBack),
null);
        }
        //
```

```
        public void GetMaintenanceData()
        {
#if DEBUG
            int code = System.Threading.Thread.CurrentThread.GetHashCode();
            System.Diagnostics.Debug.WriteLine(String.Format("Read on thread: {0}", code));
#endif
            data.SelectMaintenanceAll();
            int tcode = data.ProkaDataSet.Tables["VedligeholdelseTabel"].GetHashCode();
            Tab.Dispatcher.BeginInvoke(System.Windows.Threading.DispatcherPriority.Normal,
                new TableWorker.RunOnDispatcherDelegate(Tab.RunOnDispatcher), tcode);
        }
        //
        public void GetMaintenanceDataCallBack(IAsyncResult res)
        {
#if DEBUG
            int code = System.Threading.Thread.CurrentThread.GetHashCode();
            System.Diagnostics.Debug.WriteLine(String.Format("CallBack on thread: {0}", code));
#endif
            if (DataReady != null)
                DataReady();
        }
        //
        public DataSet HentOversigtData()
        {
            return this.data.GetWorkingSet();
        }
        //
        public void HentStationer(string navn)
        {
            data.SelectTopoStationer(navn);
        }
        //
        public int AddNewMaintenance(Maintenance m, IList dlist)
        {
            // Get typeid
            int id = data.InsertMaintenance(m);
            m.ID   = id;
            tasklist.Add(m);
            return id;
        }
      //
      public void DeleteMaintenance(Maintenance maintenance)
      {
#if TEST_SET
         this.Tasks.Remove(maintenance);
#else
         int res = this.data.DeleteMaintenance(maintenance);
         if (res == 1)
         {
            this.Tasks.Remove(maintenance);
         }
         else
         {
            string msg = "Vedligeholdelsen kunne ikke slettes!";
            AppMessageBox.ShowWarning(msg, "Fejl ved sletning");
         }
#endif
      }
      //
      public void ClearStationList()
      {
         data.ClearTable("Topostationer");
      }
      //
        #region IBusinessLogic Members

        public void GetMaintenanceTypes(string s)
        {
            DataTable table = data.ProkaDataSet.Tables["VedligeholdelseKategori"];
            var v = from db in table.AsEnumerable()
               where db.Field<string>("CategoryName") == s
               select db.Field<int>("CategoryID");
            int count = v.Count();
            int id = v.ElementAt(0);
```

```
            ListOfMaintenanceTypes.Clear();
            data.SelectMaintenanceTypes(id, ListOfMaintenanceTypes);
        }

      public void GetMaintenanceMethods(string s)
      {
         //
      }

      public void DataSetup(object o)
      {
         data.ConnectionSetup(o);
      }

      #endregion
   }

    class TableWorker : DependencyObject
    {
        public MaintenanceCollection ATasks { get; set; }
        public IAppData AData { get; set; }
        public delegate void RunOnDispatcherDelegate(int code);

        public int HashCode
        {
            get { return (int)GetValue(HashCodeProperty); }
            set { SetValue(HashCodeProperty, value); }
        }

        public static readonly DependencyProperty HashCodeProperty =
            DependencyProperty.Register("HashCode", typeof(int), typeof(TableWorker),
            new FrameworkPropertyMetadata(0, new PropertyChangedCallback(OnChange)));

        public static void OnChange(DependencyObject o, DependencyPropertyChangedEventArgs de)
        {
#if DEBUG
            System.Threading.Thread tc = System.Threading.Thread.CurrentThread;
            System.Windows.Threading.Dispatcher td;
            td = (o as TableWorker).Dispatcher;
            PrintDebugInfo(String.Format("OnChange on thread: {0}", tc.GetHashCode()));
            PrintDebugInfo(String.Format("OnChange Dispatcher: {0}", td.Thread.GetHashCode()));
#endif
        }

        public void RunOnDispatcher(int xcode)
        {
#if DEBUG
            System.Threading.Thread tc =
                System.Threading.Thread.CurrentThread;
            System.Diagnostics.Debug.WriteLine(
               String.Format("RunOnDispatcher on thread: {0}", tc.GetHashCode()));
            this.HashCode = xcode;
#endif
            RebuilList(ATasks, AData);
        }

        private void RebuilList(Collection<Maintenance> Tasks, IAppData data)
        {
            Hashtable tab = new Hashtable();
            foreach (var typ in data.ProkaDataSet.Tables["VedligeholdelseType"].AsEnumerable())
            {
                tab.Add((int)typ.ItemArray.ElementAt(0), (string)typ.ItemArray.ElementAt(1));
            }
            try
            {
                Tasks.Clear();
                DataTable table;
#if TEST_SET
                DataSet test = new DataSet();
                test.ReadXml("vedligeholdelse_x100.xml");
                table = test.Tables["VedligeholdelseTabel"];
#else
                table = data.ProkaDataSet.Tables["VedligeholdelseTabel"];
#endif
```

```
                foreach (var item in table.AsEnumerable())
                {
                    Maintenance m = new Maintenance();
#if TEST_SET
                    m.ID            = Int32.Parse((string)item.ItemArray.ElementAt(0));
                    m.VlNavn        = (string)item.ItemArray.ElementAt(1);
                    m.VlId          = Int32.Parse((string)item.ItemArray.ElementAt(2));
                    m.TaskName      = (string)item.ItemArray.ElementAt(3);
                    m.Start         = DateTime.Parse((string)item.ItemArray.ElementAt(4));
                    m.End           = DateTime.Parse((string)item.ItemArray.ElementAt(5));
                    m.StartStation  = Double.Parse((string)item.ItemArray.ElementAt(6));
                    m.EndStation    = Double.Parse((string)item.ItemArray.ElementAt(7));
                    m.Duration      = Int32.Parse((string)item.ItemArray.ElementAt(8));
                    m.Completed     = Double.Parse((string)item.ItemArray.ElementAt(9));
                    int tid = Int32.Parse((string)item.ItemArray.ElementAt(10));
                    if (tid == 0)
                        tid = 8;
                    m.Type.Id = tid;
                    m.Contractor    = String.Format("{0}", item.ItemArray.ElementAt(11));
#else
                    m.ID        = (int)item.ItemArray.ElementAt(0);
                    m.VlNavn    = (string)item.ItemArray.ElementAt(1);
                    m.VlId      = (int)item.ItemArray.ElementAt(2);
                    m.TaskName  = (string)item.ItemArray.ElementAt(3);
                    m.Start     = (DateTime)item.ItemArray.ElementAt(4);
                    m.End       = (DateTime)item.ItemArray.ElementAt(5);
                    m.StartStation  = (double)item.ItemArray.ElementAt(6);
                    m.EndStation    = (double)item.ItemArray.ElementAt(7);
                    m.Duration  = (int)item.ItemArray.ElementAt(8);
                    m.Completed = (double)item.ItemArray.ElementAt(9);
                    m.Type.Id   = (int)item.ItemArray.ElementAt(10);
                    m.Contractor    = String.Format("{0}", item.ItemArray.ElementAt(11));
                    //
#endif
                    if (tab.ContainsKey(m.Type.Id))
                        m.Type.Name = (string)tab[m.Type.Id];
                    else
                    {
                        m.Type.Name = "";
                        String msg = "Kan ikke finde type med Id: " +
                            String.Format("{0}", m.Type.Id);
                        AppMessageBox.ShowWarning(msg, "Advarsel");
                    }
                    Tasks.Add(m);
                }
            }
            catch (Exception ex)
            {
                AppMessageBox.ShowError(ex.StackTrace, ex.Message);
            }
        }

        private static void PrintDebugInfo(string p)
        {
            System.Diagnostics.Debug.WriteLine(p);
        }
    }
 }
```

AppMessageBox.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ProkaApp
{
   class AppMessageBox
   {
      public static void ShowError(string msg, string cap)
      {
         System.Windows.MessageBox.Show(msg, cap,
            System.Windows.MessageBoxButton.OK,
```

```
            System.Windows.MessageBoxImage.Error);
      }

      public static void ShowWarning(string msg, string cap)
      {
         System.Windows.MessageBox.Show(msg, cap,
            System.Windows.MessageBoxButton.OK,
            System.Windows.MessageBoxImage.Warning);
      }

      public static void ShowInfo(string msg, string cap)
      {
         System.Windows.MessageBox.Show(msg, cap,
            System.Windows.MessageBoxButton.OK,
            System.Windows.MessageBoxImage.Information);
      }
   }
}
```

AppView.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Controls;
using System.Linq;
using System.Collections;
using System.Collections.Specialized;
using ProkaApp.BusinessLogic;

namespace ProkaApp.UI
{
   #region VIEWS

    public class AppViewChangedArgs
    {
        public double Value { get; private set; }
        public Orientation AOrientation { get; private set; }

        public AppViewChangedArgs(Orientation arg, double value)
        {
            this.Value        = value;
            this.AOrientation = arg;
        }
    }

    public delegate void AppViewChangedHandler(object sender, AppViewChangedArgs arg);

   #region BASE
   public abstract class NView : INView
   {
      protected FrameworkElement top;
      protected NPanel        content;
      protected INotifyCollectionChanged tasklist;
      protected ColorConverter FColorColnverter;
      public ObservableHashtable TypesAndColorsList { get; set; }
      public FrameworkElement TopPanel { get { return top; } }
      public NPanel ContentPanel       { get { return content; } }
      public INotifyCollectionChanged Tasks { get { return tasklist; } }
      public abstract event AppViewChangedHandler AppViewChanged;
      public NView(INotifyCollectionChanged data)
      {
         this.tasklist = data;
            FColorColnverter = new ColorConverter();
            DateToPointConverterUtility.AView = this;
            TypeToColorConverterUtility.AView = this;
      }
        //
        public Color ConvertToColor(string p)
        {
            Color c = this.ColorFromType(p);
```

```
            return c;
        }
        //
        public Color ColorFromType(string p)
        {
            Color res = Colors.Black;
            try
            {
                if (TypesAndColorsList == null)
                    return Colors.Black;
                var result = from dc in TypesAndColorsList
                             where (string)dc.Key == p
                             select dc;
                if (result == null)
                    throw new ArgumentException("Kan ikke finde " + p);
                int count = result.Count();
                if (count > 0)
                {
                    var item = result.ToList()[0];
                    string value = String.Format("{0}", item.Value);

                    res = (Color)FColorColnverter.ConvertFrom(value);
                }
            }
            catch (Exception ex)
            {
                AppMessageBox.ShowError(ex.StackTrace, ex.Message);
            }
            return res;
        }
        //
        public virtual void ChangeTimeUnit(DrawingMode unit)
        {
            TimeLine tline = top as TimeLine;
            switch (unit)
            {
                case DrawingMode.Days:
                    tline.TimeUnit = DrawingMode.Days;
                    break;
                case DrawingMode.Months:
                    tline.TimeUnit = DrawingMode.Months;
                    break;
                case DrawingMode.Weeks:
                    tline.TimeUnit = DrawingMode.Weeks;
                    break;
                case DrawingMode.Year:
                    tline.TimeUnit = DrawingMode.Year;
                    break;
                default:
                    break;
            }

            if (content != null)
            {
                ContentPanel cp = content as ContentPanel;
                cp.Width = tline.Width;
            }
        }
        //
        public Point ConvertDateToPoint(DateTime d)
        {
            try
            {
                return (top as TimeLine).GetPoint(d);
            }
            catch (Exception ex)
            {
                System.Diagnostics.Debug.WriteLine(ex.StackTrace);
            }
            return new Point(0,0);
        }
        public abstract void RedrawAll();
    }
    #endregion
```

```
/*************************************************************************/
#region TEMPORAL
class TemporalView : NView
{
     public override event AppViewChangedHandler AppViewChanged;
     //
   public TemporalView(INotifyCollectionChanged data)
      : base(data)
   {
      tasklist = data;
      tasklist.CollectionChanged +=
               new NotifyCollectionChangedEventHandler(Tasks_CollectionChanged);
      this.top     = new TimeLine();
      this.content    = new ContentPanel();
      TimeLine tline  = top as TimeLine;
         tline.TimeSpanChanged += new TimeSpanChangedHandler(TimeLine_TimeSpanChanged);
      ContentPanel cp = content as ContentPanel;
      cp.HorizontalAlignment = HorizontalAlignment.Left;
      cp.VerticalAlignment   = VerticalAlignment.Top;
      cp.Background   = Brushes.AliceBlue;
      cp.Width        = tline.Width;
      cp.Height       = cp.DrawingElementHeight;
      int index = 0;
      MaintenanceCollection taskitems = tasklist as MaintenanceCollection;
      foreach (var item in taskitems)
      {
         Point p       = tline.GetPoint(item.Start);
         double length = tline.GetLength(item.Duration);
         Color c       = this.ConvertToColor(item.Type.Name);
         cp.InsertElement(index++, p.X, length, c, item.TaskName);
      }
   }

     void TimeLine_TimeSpanChanged(object sender, object arg)
     {
         this.RedrawAll();
         ContentPanel cp = content as ContentPanel;
         double value = (sender as TimeLine).Width;
         cp.Width = value;

         if (AppViewChanged != null)
         {
             AppViewChanged(this, new AppViewChangedArgs(Orientation.Horizontal, (int)value));
         }
     }

   void Tasks_CollectionChanged(object sender, NotifyCollectionChangedEventArgs e)
   {
      IList items;
      DateTime date;
      ContentPanel cp = content as ContentPanel;
      double xref = 0;  // Task startdate
      double length = 0;   // Task length
      Color cl = Colors.Brown;

         //
         Point p = new Point();
         string caption = "";
         //
         items = e.NewItems;
         if (items != null)
         {
             Maintenance data = items[0] as Maintenance;
             // find coordinates
             date = data.Start;
             int dur = data.Duration;
             caption = data.TaskName;
             p = (this.top as TimeLine).GetPoint(date);
             length = (this.top as TimeLine).GetLength(dur);
         cl = this.ConvertToColor(data.Type.Name);
         }
      // draw new item on content panel
      switch(e.Action)
      {
```

```
            case NotifyCollectionChangedAction.Add:
               // Custom type converters
               xref = p.X;
               cp.InsertElement(e.NewStartingIndex, xref, length, cl, caption);
               break;
            case NotifyCollectionChangedAction.Move:
               break;
            case NotifyCollectionChangedAction.Remove:
               break;
            case NotifyCollectionChangedAction.Replace:
               break;
            case NotifyCollectionChangedAction.Reset:
                cp.Erase();
               break;
            default: break;
         }
      }
        //
        public override void  RedrawAll()
        {
            ContentPanel cp = content as ContentPanel;
            cp.Erase();
            MaintenanceCollection taskitems = tasklist as MaintenanceCollection;
            TimeLine tline = top as TimeLine;
            int index = 0;
            foreach (var item in taskitems)
            {
                Point p = tline.GetPoint(item.Start);
                double length = tline.GetLength(item.Duration);
                Color c = this.ConvertToColor(item.Type.Name);
                cp.InsertElement(index++, p.X, length, c, item.TaskName);
            }
        }
      //
      public override string ToString()
      {
            return "ProkaApp.UI.TemporalView";
      }
   }
#endregion
/*************************************************************************/
#region SPATIAL
class SpatialView : NView
{
   public SpatialView(INotifyCollectionChanged data)
      : base(data)
   {
      this.top    = new SpatialLine();
      this.content   = null;
         // only used to remove compile warning
         if (AppViewChanged != null)
         {
             return;
         }
   }

   public override void RedrawAll()
   {
   }

   public override string ToString()
   {
      return base.ToString();
   }

   public override event AppViewChangedHandler AppViewChanged;
}
#endregion
 /*************************************************************************/
 #region TEMPORAL_LISTBOX
 class TemporalListBox : NView
 {
     public ContentListBox ContentBox { get; private set; }
     public override event AppViewChangedHandler AppViewChanged;
```

```
//
public TemporalListBox(INotifyCollectionChanged data)
    : base(data)
{
    object o = Application.Current.MainWindow.FindResource("ContentListView");
    if (o == null)
    {
        return;
    }
    this.ContentBox = o as ContentListBox;
    // if Windows XP
    if (System.Environment.OSVersion.Version.Major < 6)
    {
        this.ContentBox.Padding = new Thickness(0, 2, 0, 0);
    }
    this.ContentBox.SizeChanged += new SizeChangedEventHandler(ContentBox_SizeChanged);
    //
    this.top        = new TimeLine();
    TimeLine tline  = top as TimeLine;
    tline.TimeSpanChanged += new TimeSpanChangedHandler(TimeLine_TimeSpanChanged);
    this.ContentBox.Width = tline.Width;
    //
    System.Windows.Data.Binding bd =
        new System.Windows.Data.Binding();
    bd.Source = data;
    ContentBox.SetBinding(ListBox.ItemsSourceProperty, bd);
    //
    System.Windows.Data.Binding contentWidthBinding =
        new System.Windows.Data.Binding("TimeUnit");
    contentWidthBinding.Source = tline;
    contentWidthBinding.Converter = new TimeUnitToIntegerConverter();
    ContentBox.SetBinding(ContentListBox.ReferenceWidthProperty, contentWidthBinding);
    //
}

void ContentBox_SizeChanged(object sender, SizeChangedEventArgs e)
{
    if (AppViewChanged != null)
    {
        AppViewChanged(this, new AppViewChangedArgs(Orientation.Vertical,
                        e.NewSize.Height));
    }
}
//
void TimeLine_TimeSpanChanged(object sender, object arg)
{
    double value = (sender as TimeLine).Width;
    this.ContentBox.Width = value;
    this.RedrawAll();
    // Must be defined here for horisontal values
    if (AppViewChanged != null)
    {
        AppViewChanged(this, new AppViewChangedArgs(Orientation.Horizontal, value));
    }
}
//
public override void ChangeTimeUnit(DrawingMode unit)
{
    base.ChangeTimeUnit(unit);
    ContentBox.Width = top.Width;
    this.RedrawAll();
    // Must be defined here for horisontal values
    if (AppViewChanged != null)
    {
        AppViewChanged(this, new AppViewChangedArgs(Orientation.Horizontal, top.Width));
    }
}
//
public override void RedrawAll()
{
    int cnt = ContentBox.Items.Count;
    int itemwidth = (int)(top as TimeLine).ItemWidth;
    //ContentListItem.SetReferenceWidth(ContentBox, itemwidth);
    //
```

```
            for (int i = 0; i < cnt; i++)
            {
                // Getting the currently selected ListBoxItem
                // Note that the ListBox must have
                // IsSynchronizedWithCurrentItem set to True for this to work
                ListBoxItem myListBoxItem =(ListBoxItem)
                  (ContentBox.ItemContainerGenerator.ContainerFromItem(ContentBox.Items[i]));
                // Getting the ContentPresenter of myListBoxItem
                ContentPresenter myContentPresenter =
                  FindVisualChild<ContentPresenter>(myListBoxItem);
                // Finding textBlock from the DataTemplate that is set on that ContentPresenter
                DataTemplate myDataTemplate = myContentPresenter.ContentTemplate;
                ContentListItem item = (ContentListItem)
                  myDataTemplate.FindName("AListItem", myContentPresenter);
                item.ItemSize = itemwidth;
                item.Start = this.ConvertDateToPoint((ContentBox.Items[i] as Maintenance).Start);
            }
        }
        // helper funktion som bliver ikke kald ved normal brug
        private childItem FindVisualChild<childItem>(DependencyObject obj)
            where childItem : DependencyObject
        {
            for (int i = 0; i < VisualTreeHelper.GetChildrenCount(obj); i++)
            {
                DependencyObject child = VisualTreeHelper.GetChild(obj, i);
                if (child != null && child is childItem)
                    return (childItem)child;
                else
                {
                    childItem childOfChild = FindVisualChild<childItem>(child);
                    if (childOfChild != null)
                        return childOfChild;
                }
            }
            return null;
        }
        //
        public override string ToString()
        {
            return "ProkaApp.UI.TemporalListBox";
        }
    }
    #endregion
    #endregion

    #region PANELS
    #region BASE
    public abstract class NPanel : Panel
    {
        //
        public int DataCount
        {
            get { return (int)GetValue(DataCountProperty); }
            set { SetValue(DataCountProperty, value); }
        }

        public static readonly DependencyProperty DataCountProperty =
            DependencyProperty.Register("DataCount",
            typeof(int),
            typeof(NPanel),
            new FrameworkPropertyMetadata(0,
                FrameworkPropertyMetadataOptions.AffectsMeasure |
                FrameworkPropertyMetadataOptions.AffectsRender,
                new PropertyChangedCallback(ItemCountChanged)));
        //
        private static void ItemCountChanged(DependencyObject o,
                                             DependencyPropertyChangedEventArgs de)
        {
            Type t = o.GetType();
            DependencyProperty dp = de.Property;
        }

        public NPanel()
        {
```

```
    }
}
#endregion
/*************************************************************************/
#region CONTENT
class ContentPanel : NPanel
{
   private const double DrawingHeight = 10;
   private const double DrawingOffset = DrawingHeight / 2;
   //private const double ElementHeight = 25;  // XP
     private const double ElementHeight = 26;    // Vista
     public  double DrawingElementHeight { get { return ElementHeight; } }
   private const double ElementOffset = ElementHeight / 2;
   //private const double TopMargin = 4;       // XP
     private const double TopMargin = 3;         // Vista
   private NContent element;
   public FrameworkElement ContentElement { get { return element; } }

   public ContentPanel()
   {
      element = new NContent();
      this.Children.Add(element);
      this.Margin = new Thickness(0, TopMargin, 0, 0);
   }
     //
     public void Erase()
     {
         element.InvalidateContent();
     }
     //
   public void InsertElement(double yref, double xref, double length, Color cl, string s)
   {
         this.Height = (yref + 1) * ElementHeight;
         double Ypos = (yref * ElementHeight) + (ElementOffset-DrawingOffset);
      element.DrawElement(new Point(xref, Ypos), new Size(length, DrawingHeight), cl, s);
   }
     //
   protected override Size ArrangeOverride(Size finalSize)
   {
      return base.ArrangeOverride(finalSize);
   }
     //
   protected override Size MeasureOverride(Size availableSize)
   {
      Size s = base.MeasureOverride(availableSize);
      foreach (UIElement ui in Children)
      {
         ui.Measure(availableSize);
      }
         // Return width of the timeline and height of the itemslist
         if (availableSize.Width == Double.PositiveInfinity)
             s.Width  = 0;
         else
             s.Width = availableSize.Width;
         //
         if (availableSize.Height == Double.PositiveInfinity)
             s.Height = 0;
         else
             s.Height = availableSize.Height;
         return s;
   }
     //
   protected override void OnRender(DrawingContext dc)
   {
      base.OnRender(dc);

   }
}
#endregion
 /*************************************************************************/
#endregion

 #region UTILITIES
 public class TimeUnitToIntegerConverter : System.Windows.Data.IValueConverter
```

```
    {
        #region IValueConverter Members

        public object Convert(object value, Type targetType, object parameter,
                              System.Globalization.CultureInfo culture)
        {
            int res = 0;
            if (targetType != typeof(Int32))
                throw new ArgumentException("TargetType skal være Int32!");
            if (value.GetType() != typeof(DrawingMode))
                throw new ArgumentException("Value skal være DrawingMode");
            DrawingMode dm = (DrawingMode)value;
            switch (dm)
            {
                case DrawingMode.Days:   res = 32; break;
                case DrawingMode.Weeks:  res = 16; break;
                case DrawingMode.Months: res = 8;  break;
                case DrawingMode.Year:   res = 3;  break;
                default: throw new ArgumentException("Value af DrawingMode findes ikke!");
            }
            return res;
        }

        public object ConvertBack(object value, Type targetType, object parameter,
                                  System.Globalization.CultureInfo culture)
        {
            throw new NotImplementedException();
        }

        #endregion
    }
    #endregion
}
```

ColorsCollection.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Media;
using System.Windows.Data;
//
namespace ProkaApp
{
   public class ColorsCollection : List<Color>
    {
     public ColorsCollection()
      {
         this.Add(Colors.AliceBlue);
         this.Add(Colors.AntiqueWhite);
         this.Add(Colors.Aqua);
         this.Add(Colors.Aquamarine);
         this.Add(Colors.Azure);
         this.Add(Colors.Beige);
         this.Add(Colors.Bisque);
         this.Add(Colors.Black);
         this.Add(Colors.BlanchedAlmond);
         this.Add(Colors.Blue);
         this.Add(Colors.BlueViolet);
         this.Add(Colors.Brown);
         this.Add(Colors.BurlyWood);
         this.Add(Colors.CadetBlue);
         this.Add(Colors.Chartreuse);
         this.Add(Colors.Chocolate);
         this.Add(Colors.Coral);
         this.Add(Colors.CornflowerBlue);
         this.Add(Colors.Cornsilk);
         this.Add(Colors.Crimson);
         this.Add(Colors.Cyan);
         this.Add(Colors.DarkBlue);
         this.Add(Colors.DarkCyan);
         this.Add(Colors.DarkGoldenrod);
         this.Add(Colors.DarkGray);
```

```
this.Add(Colors.DarkGreen);
this.Add(Colors.DarkKhaki);
this.Add(Colors.DarkMagenta);
this.Add(Colors.DarkOliveGreen);
this.Add(Colors.DarkOrange);
this.Add(Colors.DarkOrchid);
this.Add(Colors.DarkRed);
this.Add(Colors.DarkSalmon);
this.Add(Colors.DarkSeaGreen);
this.Add(Colors.DarkSlateBlue);
this.Add(Colors.DarkSlateGray);
this.Add(Colors.DarkTurquoise);
this.Add(Colors.DarkViolet);
this.Add(Colors.DeepPink);
this.Add(Colors.DeepSkyBlue);
this.Add(Colors.DimGray);
this.Add(Colors.DodgerBlue);
this.Add(Colors.Firebrick);
this.Add(Colors.FloralWhite);
this.Add(Colors.ForestGreen);
this.Add(Colors.Fuchsia);
this.Add(Colors.Gainsboro);
this.Add(Colors.GhostWhite);
this.Add(Colors.Gold);
this.Add(Colors.Goldenrod);
this.Add(Colors.Gray);
this.Add(Colors.Green);
this.Add(Colors.GreenYellow);
this.Add(Colors.Honeydew);
this.Add(Colors.HotPink);
this.Add(Colors.IndianRed);
this.Add(Colors.Indigo);
this.Add(Colors.Ivory);
this.Add(Colors.Khaki);
this.Add(Colors.Lavender);
this.Add(Colors.LavenderBlush);
this.Add(Colors.LawnGreen);
this.Add(Colors.LemonChiffon);
this.Add(Colors.LightBlue);
this.Add(Colors.LightCoral);
this.Add(Colors.LightCyan);
this.Add(Colors.LightGoldenrodYellow);
this.Add(Colors.LightGray);
this.Add(Colors.LightGreen);
this.Add(Colors.LightPink);
this.Add(Colors.LightSalmon);
this.Add(Colors.LightSeaGreen);
this.Add(Colors.LightSkyBlue);
this.Add(Colors.LightSlateGray);
this.Add(Colors.LightSteelBlue);
this.Add(Colors.LightYellow);
this.Add(Colors.Lime);
this.Add(Colors.LimeGreen);
this.Add(Colors.Linen);
this.Add(Colors.Magenta);
this.Add(Colors.Maroon);
this.Add(Colors.MediumAquamarine);
this.Add(Colors.MediumBlue);
this.Add(Colors.MediumOrchid);
this.Add(Colors.MediumPurple);
this.Add(Colors.MediumSeaGreen);
this.Add(Colors.MediumSlateBlue);
this.Add(Colors.MediumSpringGreen);
this.Add(Colors.MediumTurquoise);
this.Add(Colors.MediumVioletRed);
this.Add(Colors.MidnightBlue);
this.Add(Colors.MintCream);
this.Add(Colors.MistyRose);
this.Add(Colors.Moccasin);
this.Add(Colors.NavajoWhite);
this.Add(Colors.Navy);
this.Add(Colors.OldLace);
this.Add(Colors.Olive);
this.Add(Colors.OliveDrab);
```

```
            this.Add(Colors.Orange);
            this.Add(Colors.OrangeRed);
            this.Add(Colors.Orchid);
            this.Add(Colors.PaleGoldenrod);
            this.Add(Colors.PaleGreen);
            this.Add(Colors.PaleTurquoise);
            this.Add(Colors.PaleVioletRed);
            this.Add(Colors.PapayaWhip);
            this.Add(Colors.PeachPuff);
            this.Add(Colors.Peru);
            this.Add(Colors.Pink);
            this.Add(Colors.Plum);
            this.Add(Colors.PowderBlue);
            this.Add(Colors.Purple);
            this.Add(Colors.Red);
            this.Add(Colors.RosyBrown);
            this.Add(Colors.RoyalBlue);
            this.Add(Colors.SaddleBrown);
            this.Add(Colors.Salmon);
            this.Add(Colors.SandyBrown);
            this.Add(Colors.SeaGreen);
            this.Add(Colors.SeaShell);
            this.Add(Colors.Sienna);
            this.Add(Colors.Silver);
            this.Add(Colors.SkyBlue);
            this.Add(Colors.SlateBlue);
            this.Add(Colors.SlateGray);
            this.Add(Colors.Snow);
            this.Add(Colors.SpringGreen);
            this.Add(Colors.SteelBlue);
            this.Add(Colors.Tan);
            this.Add(Colors.Teal);
            this.Add(Colors.Thistle);
            this.Add(Colors.Tomato);
            this.Add(Colors.Transparent);
            this.Add(Colors.Turquoise);
            this.Add(Colors.Violet);
            this.Add(Colors.Wheat);
            this.Add(Colors.White);
            this.Add(Colors.WhiteSmoke);
            this.Add(Colors.Yellow);
            this.Add(Colors.YellowGreen);
        }
    }
}
```

ConnectionFactory.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
//
using System.Data.Common;  // DbConnection
using System.Configuration;   // ConfigurationManager

namespace ProkaApp.Data
{
   class ConnectionFactory
   {
      private String ConnectionName;

      public ConnectionFactory()
      {

      }

      internal void Initialize(DbConnection Connection)
      {
         try
         {
            if (LoadDefaultConnection() == false)
            {
               LoadLastConnection();
```

```
            }
            if (!String.IsNullOrEmpty(ConnectionName))
            {
               System.Windows.MessageBox.Show(ConnectionName, "ConnectionName");
               ConnectionStringSettings cns =
                     ConfigurationManager.ConnectionStrings[ConnectionName];
               DbProviderFactory factory = DbProviderFactories.GetFactory(cns.ProviderName);
               DbConnection con       = factory.CreateConnection();
               con.ConnectionString   = cns.ConnectionString;
               // Test
               con.Open();
               System.Windows.MessageBox.Show(con.Database + "\n" + con.ToString(), "Info");
               con.Close();
            }
         }
         catch (NullReferenceException e)
         {
            string msg = "Der er en fejl i konfigurations filen.";
            string cap = e.Source;
#if DEBUG
            msg += "\n" + e.Message + "\n" + e.StackTrace;
#endif
            AppMessageBox.ShowError(msg, cap);
         }
         catch (ArgumentException e)
         {
            AppMessageBox.ShowError(e.StackTrace, e.Message);
         }
         catch (Exception e)
         {
            AppMessageBox.ShowError(e.Message + "\n****\n" + e.StackTrace, e.Source);
         }
      }

      private bool LoadDefaultConnection()
      {
         bool result = false;
         String LoadDefault;
         String DefaultConnection;
         try
         {
            LoadDefault = ConfigurationManager.AppSettings["LoadDefault"];
            if (String.IsNullOrEmpty(LoadDefault))
               return false;
            //if (LoadDefault.Length != 1) return false
            if (LoadDefault.Trim().Equals("0"))
               result = false;
            else if (LoadDefault.Trim().Equals("1"))
            {
               DefaultConnection = ConfigurationManager.AppSettings["DefaultConnection"];
               if (!String.IsNullOrEmpty(DefaultConnection.Trim()))
               {
                  ConnectionName = DefaultConnection;
                  result = true;
               }
            }
            else
            {
               // Error in reading configuration file
               // LoadDefault is set to true but there is no corresponding
               AppMessageBox.ShowError("ErrorMsg", "Fejl");
            }
            return result;
         }
         catch (Exception e)
         {
            e.Message.Trim();
            throw;
         }
      }

      private bool LoadLastConnection()
      {
         bool result = false;
```

```
            String LoadLast;
            String LastConnection;
            try
            {
               LoadLast = ConfigurationManager.AppSettings["LoadLast"];
               if (String.IsNullOrEmpty(LoadLast))
                  return false;
               //if (LoadLast.Length != 1) return false;
               if (LoadLast.Trim().Equals("0"))
                  result = false;
               else if (LoadLast.Trim().Equals("1"))
               {
                  LastConnection = ConfigurationManager.AppSettings["LastConnection"];
                  if (!String.IsNullOrEmpty(LastConnection.Trim()))
                  {
                     ConnectionName = LastConnection;
                     result = true;
                  }
               }
               else
               {
                  // Error in reading configuration file
                  // LoadDefault is set to true but there is no corresponding
                  AppMessageBox.ShowError("ErrorMsg", "Fejl");
               }
               return result;
            }
            catch (Exception e)
            {
               e.Message.Trim();
               throw;
            }
         }
      }
   }
```

ContentListBlocks.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Media;
//
using ProkaApp.BusinessLogic;

namespace ProkaApp.UI
{
    public class ContentListBox : ListBox
    {
        private DateTime start;
        private DateTime end;
        #region ReferenceWidthProperty
        public int ReferenceWidth
        {
            get { return (int)GetValue(ReferenceWidthProperty); }
            set { SetValue(ReferenceWidthProperty, value); }
        }
        //
        public static readonly DependencyProperty ReferenceWidthProperty =
            DependencyProperty.Register("ReferenceWidth", typeof(int), typeof(ContentListBox),
                new FrameworkPropertyMetadata(0, FrameworkPropertyMetadataOptions.AffectsRender,
                    new PropertyChangedCallback(OnReferenceWidthChanged)));
        //
        private static void OnReferenceWidthChanged(DependencyObject d,
            DependencyPropertyChangedEventArgs de)
        {
            return;
        }
        //
        #endregion
```

```
    //
    public ContentListBox()
    {
    }
    //
    protected override void OnRender(DrawingContext drawingContext)
    {
        base.OnRender(drawingContext);
        end = DateTime.Now;
        string s = String.Format("Render time: {0}", (end - start).Milliseconds);
        System.Diagnostics.Debug.WriteLine(s);
    }
    //
    public void Redraw()
    {
        return;
    }

    protected override void OnItemsChanged(
                          System.Collections.Specialized.NotifyCollectionChangedEventArgs e)
    {
        base.OnItemsChanged(e);
        if (e.NewStartingIndex == 0)
        {
            start = DateTime.Now;
        }
    }
}

public class ContentListItem : FrameworkElement
{
    #region DependencyProperties
    //
    #region StartDependencyProperty
    public Point Start
    {
        get { return (Point)GetValue(StartProperty); }
        set { SetValue(StartProperty, value); }
    }
    //
    public static readonly DependencyProperty StartProperty =
        DependencyProperty.Register("Start", typeof(Point), typeof(ContentListItem),
        new FrameworkPropertyMetadata(new Point(0, 0),
            FrameworkPropertyMetadataOptions.AffectsRender |
            FrameworkPropertyMetadataOptions.AffectsArrange,
            new PropertyChangedCallback(OnStartChanged)));
    //
    private static void OnStartChanged(DependencyObject d,
                                       DependencyPropertyChangedEventArgs de)
    {
        return;
    }
    #endregion

    #region DurationProperty
    public int Duration
    {
        get { return (int)GetValue(DurationProperty); }
        set { SetValue(DurationProperty, value); }
    }
    //
    public static readonly DependencyProperty DurationProperty =
        DependencyProperty.Register("Duration", typeof(int), typeof(ContentListItem),
        new FrameworkPropertyMetadata(0, FrameworkPropertyMetadataOptions.AffectsRender,
            new PropertyChangedCallback(OnDurationChanged)));

    private static void OnDurationChanged(DependencyObject d,
                                          DependencyPropertyChangedEventArgs de)
    {
        return;
    }
    #endregion

    #region ItemSizeDependencyProperty
```

```
public int ItemSize
{
    get { return (int)GetValue(ItemSizeProperty); }
    set { SetValue(ItemSizeProperty, value); }
}

public static readonly DependencyProperty ItemSizeProperty =
    DependencyProperty.Register("ItemSize", typeof(int), typeof(ContentListItem),
    new FrameworkPropertyMetadata(32,
        FrameworkPropertyMetadataOptions.AffectsRender |
        FrameworkPropertyMetadataOptions.AffectsMeasure,
        new PropertyChangedCallback(OnItemSizeChanged)));

private static void OnItemSizeChanged(DependencyObject d,
                                      DependencyPropertyChangedEventArgs de)
{
    return;
}
#endregion

#region ItemColorDependencyProperty
//
public Color ItemColor
{
    get { return (Color)GetValue(ItemColorProperty); }
    set { SetValue(ItemColorProperty, value); }
}
//
public static readonly DependencyProperty ItemColorProperty =
    DependencyProperty.Register("ItemColor", typeof(Color), typeof(ContentListItem),
        new FrameworkPropertyMetadata(Colors.White,
            FrameworkPropertyMetadataOptions.AffectsRender,
            new PropertyChangedCallback(OnItemColorChange)));
//
private static void OnItemColorChange(DependencyObject d,
                                      DependencyPropertyChangedEventArgs de)
{
    return;
}
#endregion

#region DescriptionProperty
public string Description
{
    get { return (string)GetValue(DescriptionProperty); }
    set { SetValue(DescriptionProperty, value); }
}
//
public static readonly DependencyProperty DescriptionProperty =
    DependencyProperty.Register("Description", typeof(string), typeof(ContentListItem),
    new FrameworkPropertyMetadata("", FrameworkPropertyMetadataOptions.AffectsRender,
        new PropertyChangedCallback(OnDescriptionChanged)));

private static void OnDescriptionChanged(DependencyObject d,
                                         DependencyPropertyChangedEventArgs de)
{
    return;
}
#endregion

//

//
#endregion
//
private Visual[] items;
private DrawingVisual itemShape;
private DrawingVisual itemText;
private System.Globalization.CultureInfo cultureDK;
//
public ContentListItem()
    : base()
{
    try
```

```
    {
        cultureDK = new System.Globalization.CultureInfo("da-DK");
        items     = new Visual[2];
        itemShape = new DrawingVisual();
        itemText  = new DrawingVisual();
        items[0]  = itemShape;
        items[1]  = itemText;
        AddLogicalChild(itemShape);
        AddVisualChild(itemShape);
        AddLogicalChild(itemText);
        AddVisualChild(itemText);
        this.SizeChanged += new SizeChangedEventHandler(ContentListItem_SizeChanged);
    if (System.Environment.OSVersion.Version.Major >= 6)
       this.Height = 25;
    else
       this.Height = 24;
    }
    catch (Exception ex)
    {
        String s = ex.StackTrace;
    }
}

void ContentListItem_SizeChanged(object sender, SizeChangedEventArgs e)
{
    return;
}

protected override int VisualChildrenCount
{
    get
    {
        return items.Count();
    }
}

protected override Visual GetVisualChild(int index)
{
    if (index > 1)
        throw new ArgumentException("Value > 1", "index");
    return items[index];
}

protected override void OnRender(DrawingContext drawingContext)
{
    base.OnRender(drawingContext);
    UpdateItem();
}
//
private void UpdateItem()
{
    // What do you need to draw an item
    // Point, Size, Color and Caption
    // Point = Start DP
    // Size  = (Int)Duration * (Int)ItemSize
    // Color = ItemColor
    // Description = Caption
    using (DrawingContext dc = itemShape.RenderOpen())
    {
        Pen ptransparent = new Pen(Brushes.Transparent, 0);
        Point p = this.Start;
        p.X -= 2;
        p.Y  = 8;
        dc.DrawRoundedRectangle(new SolidColorBrush(ItemColor), ptransparent,
            new Rect(p, new Size(this.Duration * this.ItemSize, 10)), 4, 8);
    }
    //
    using (DrawingContext dc = itemText.RenderOpen())
    {
        FlowDirection dir = FlowDirection.LeftToRight;
        Typeface txtFace  = new Typeface("Arial");
        FormattedText ft  = new FormattedText(this.Description, cultureDK, dir,
                                              txtFace, 12, Brushes.Black);
        Point p = this.Start;
```

```
            p.X += (double)(this.Duration * this.ItemSize) + 10;
            p.Y = (26 / 2) - (ft.Height / 2);
            dc.DrawText(ft, p);
        }
    }
    //
    protected override void OnMouseLeftButtonDown(System.Windows.Input.MouseButtonEventArgs e)
    {
        base.OnMouseLeftButtonDown(e);
        Point loc = e.GetPosition(this);
        HitTestResult res = VisualTreeHelper.HitTest(this, loc);
        if (res.VisualHit.GetType() == typeof(DrawingVisual))
        {
            DrawingVisual dv = res.VisualHit as DrawingVisual;
            object o = this.VisualParent;
            if (o is ContentPresenter)
            {
                ContentPresenter cp = o as ContentPresenter;
                object p = cp.Content;
                if (p is Maintenance)
                {
                    Maintenance mt = p as Maintenance;
                    StringBuilder sb = new StringBuilder();
                    sb.AppendLine(String.Format("Vedligeholdelse information"));
                    sb.AppendLine(String.Format("ID:\t{0}", mt.ID));
                    sb.AppendLine(String.Format("Vandløbnavn: {0}", mt.VlNavn));
                    sb.AppendLine(String.Format("Vandløb ID: {0}", mt.VlId));
                    sb.AppendLine("Start dato:\t" + mt.Start.ToShortDateString());
                    sb.AppendLine("Slut dato:\t" + mt.End.ToShortDateString());
                    sb.AppendLine(String.Format("Type:\t{0}", mt.Type.Name));
                    MessageBox.Show(sb.ToString(), "Vedligeholdelse information",
                        MessageBoxButton.OK, MessageBoxImage.Information);
                }
            }
        }
    }
}

public class DateToPointConverter : IValueConverter
{
    #region IValueConverter Members

    public object Convert(object value, Type targetType, object parameter,
                          System.Globalization.CultureInfo culture)
    {
        if (targetType != typeof(Point))
            return null;
        Point p = DateToPointConverterUtility.GetPoint((DateTime)value);
        int hc = this.GetHashCode();
        return p;
    }

    public object ConvertBack(object value, Type targetType, object parameter,
                              System.Globalization.CultureInfo culture)
    {
        throw new NotImplementedException();
    }

    #endregion
}
//
public class DateToPointConverterUtility
{
    public static NView AView { get; set; }
    //
    public DateToPointConverterUtility()
    {
    }
    //
    public static Point GetPoint(DateTime d)
    {
        if (AView == null)
        {
            throw new NullReferenceException("Reference to the NView is null!");
```

```
            }
            Point p = AView.ConvertDateToPoint(d);
            return p;
        }
    }
    //
    public class TypeToColorConverter : IValueConverter
    {
        private int hashcode;
        public TypeToColorConverter()
        {
            hashcode = this.GetHashCode();
        }
        #region IValueConverter Members
        public object Convert(object value, Type targetType, object parameter,
                              System.Globalization.CultureInfo culture)
        {
            if (targetType != typeof(Color))
                throw new ArgumentException("Target type must be a Color!");
            Color c = TypeToColorConverterUtility.GetColor((string)value);

            return c;
        }

        public object ConvertBack(object value, Type targetType, object parameter,
                                  System.Globalization.CultureInfo culture)
        {
            throw new NotImplementedException();
        }
        #endregion
    }
    //
    public class TypeToColorConverterUtility
    {
        public static NView AView { get; set; }
        public static Color GetColor(string s)
        {
            if (AView == null)
            {
                throw new ArgumentException("Reference to the NView is null!");
            }
            Color c = AView.ConvertToColor(s);
            return c;
        }
    }
}
```

DataViewer.cs

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

namespace ProkaApp
{
   public partial class DataViewer : Form
   {
      private DataSet _data;
      public DataSet Data
      {
         get { return _data; }
         set
         {
            _data = value;
            this.dataGridView1.DataSource = new DataView(_data.Tables[String.Format("{0}",
               this.Tag)]);
```

```
         }
      }
      public DataViewer()
      {
         InitializeComponent();
         this.dataGridView1.AllowUserToDeleteRows = false;
         this.dataGridView1.AllowUserToAddRows = false;
      }
   }
}
```

DataViewer.Designer.cs

```
namespace ProkaApp
{
   partial class DataViewer
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }

      #region Windows Form Designer generated code

      /// <summary>
      /// Required method for Designer support - do not modify
      /// the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
      {
         this.dataGridView1 = new System.Windows.Forms.DataGridView();
         ((System.ComponentModel.ISupportInitialize)(this.dataGridView1)).BeginInit();
         this.SuspendLayout();
         //
         // dataGridView1
         //
         this.dataGridView1.ColumnHeadersHeightSizeMode =
            System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize;
         this.dataGridView1.Dock = System.Windows.Forms.DockStyle.Fill;
         this.dataGridView1.Location = new System.Drawing.Point(0, 0);
         this.dataGridView1.Name = "dataGridView1";
         this.dataGridView1.Size = new System.Drawing.Size(456, 324);
         this.dataGridView1.TabIndex = 0;
         //
         // DataViewer
         //
         this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
         this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
         this.ClientSize = new System.Drawing.Size(456, 324);
         this.Controls.Add(this.dataGridView1);
         this.Name = "DataViewer";
         this.Text = "Data Viewer";
         ((System.ComponentModel.ISupportInitialize)(this.dataGridView1)).EndInit();
         this.ResumeLayout(false);

      }

      #endregion
```

```
        private System.Windows.Forms.DataGridView dataGridView1;
    }
}
```

DateTimeConfig.cs

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

namespace ProkaApp
{
    public partial class DateTimeConfig : Form
    {
        public DateTime FirstDate { get; set; }
        public DateTime LastDate  { get; set; }

        public DateTimeConfig()
        {
            InitializeComponent();
            this.Icon = ProkaResources.Vandloebsbog;
            this.Load += new EventHandler(DateTimeConfig_Load);
        }

        void DateTimeConfig_Load(object sender, EventArgs e)
        {
            this.dateTimeSpan1.InitialValues(this.FirstDate, this.LastDate);
        }

        private void button1_Click(object sender, EventArgs e)
        {
            DateTime first = dateTimeSpan1.FirstDate;
            DateTime last = dateTimeSpan1.LastDate;
            int res = DateTime.Compare(first, last);
            if (res < 0)
            {
                FirstDate = first;
                LastDate  = last;
                this.DialogResult = DialogResult.OK;
                this.Close();
            }
            else
            {
                MessageBox.Show("Den sidste dato må ikke være mindre end den første",
                    "Information", MessageBoxButtons.OK, MessageBoxIcon.Information);
                return;
            }
        }
    }
}
```

DateTimeConfig.Designer.cs

```
namespace ProkaApp
{
    partial class DateTimeConfig
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;
        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise,
            false.</param>
        protected override void Dispose(bool disposing)
        {
```

```
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.dateTimeSpan1 = new UtilityClasses.DateTimeSpan();
            this.button1 = new System.Windows.Forms.Button();
            this.SuspendLayout();
            //
            // dateTimeSpan1
            //
            this.dateTimeSpan1.Location = new System.Drawing.Point(9, 12);
            this.dateTimeSpan1.Name = "dateTimeSpan1";
            this.dateTimeSpan1.Size = new System.Drawing.Size(218, 113);
            this.dateTimeSpan1.TabIndex = 0;
            //
            // button1
            //
            this.button1.Location = new System.Drawing.Point(81, 131);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(75, 23);
            this.button1.TabIndex = 1;
            this.button1.Text = "OK";
            this.button1.UseVisualStyleBackColor = true;
            this.button1.Click += new System.EventHandler(this.button1_Click);
            //
            // DateTimeConfig
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(237, 168);
            this.Controls.Add(this.button1);
            this.Controls.Add(this.dateTimeSpan1);
            this.MaximizeBox = false;
            this.MinimizeBox = false;
            this.Name = "DateTimeConfig";
            this.Text = "DateTime konfiguration";
            this.ResumeLayout(false);

        }

        #endregion

        private UtilityClasses.DateTimeSpan dateTimeSpan1;
        private System.Windows.Forms.Button button1;
    }
}
```

DateTimeHelper.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ProkaApp
{
    class DateTimeHelper
    {
        public TimeSpan TSpan { get; private set; }
        public DateTime First { get; set; }
        public DateTime Last { get; set; }
        public List<DateTime> AllDays { get; private set; }
```

```
        public DateTimeHelper()
        {
            AllDays = new List<DateTime>();
        }
        public int WeekNumber(DateTime day)
        {
            int year = day.Year;
            First = new DateTime(year, 1, 1);
            Last = new DateTime(year, 12, 31);
            TSpan = Last - First;
            DateTime Current = First;
            AllDays.Clear();
            for (int i = 0; i < TSpan.Days + 1; i++)
            {
                AllDays.Add(Current);
                Current = Current.AddDays(1);
            }
            IEnumerable<DateTime> dummy = from list in AllDays
                  where list.DayOfWeek == day.DayOfWeek select list;
            List<DateTime> foo = new List<DateTime>(dummy);
            if (foo[0].DayOfYear == 1)
                return foo.IndexOf(day);
            else
                return (foo.IndexOf(day) + 2);
        }
    }
}
```

Maintenance.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections.ObjectModel; // ObservableCollection<T>
using System.Collections.Specialized; // NotifyCollectionChangedEventHandler

namespace ProkaApp.BusinessLogic
{
   #region MaintenanceCollection Class
   public class MaintenanceCollection : ObservableCollection<Maintenance>
   {
      public MaintenanceCollection()
         : base()
        {
        }

      public override string ToString()
      {
         return base.ToString();
      }
   }
   #endregion

   #region Maintenance Class

    public class Maintenance
   {
      public int ID        { get; set; }
      public string VlNavn { get; set; }
      public int VlId         { get; set; }
      public string TaskName  { get; set; }
      public DateTime Start   { get; set; }
      public DateTime End     { get; set; }
      public double StartStation { get; set; }
      public double EndStation   { get; set; }
      public int Duration     { get; set; }
      public double Completed { get; set; }
      public MaintenanceType Type { get; set; }
      public MaintenanceCategory Category
        {
            get { return Type.Category; }
            // Set by Type.Category
        }
```

```
        public String Contractor { get; set; }

        public Maintenance()
        {
            VlNavn    = "";
            TaskName = "";
            Start = DateTime.Today;
            End   = DateTime.Today.AddDays(1);
            Duration = 1;
            this.Type       = new MaintenanceType();
            this.Contractor = "";
        }

        public override string ToString()
        {
            return String.Format("Maintenance ID: {0}", ID);
        }
    }
    #endregion

    #region Straekning Class
    public class Straekning
    {
        public int Id          { get; set; }
        public int VersId      { get; set; }
        public string VersNavn { get; set; }
        public int StationId   { get; set; }
        public double Station  { get; set; }
    }
    #endregion

    #region MaintenanceType Class
    public class MaintenanceType
    {
        public int Id                       { get; set; }
        public string Name                  { get; set; }
        public MaintenanceCategory Category { get; set; }
        public MaintenanceMethod Method     { get; set; }
        public string MarkWith              { get; set; }

        public MaintenanceType()
        {
            Category = new MaintenanceCategory();
               Method   = new MaintenanceMethod();
        }

        public override string ToString()
        {
            return String.Format("Vedligeholdelse type: {0}({1})\t{2}",
               Name, Id, Category.ToString());
        }
    }
    #endregion

    #region MaintenanceCategory Class
    public class MaintenanceCategory
    {
        public int Id { get; set; }
        public string Name { get; set; }

        public override string ToString()
        {
            return String.Format("Vedligeholdelse kategori: {0}({1})", Name, Id);
        }
    }
    #endregion

     #region MaintenanceMethod
     public class MaintenanceMethod
     {
         public string Name { get; set; }
     }
     #endregion
}
```

MaintenanceWnd.xaml

```
<Window x:Class="ProkaApp.MaintenanceWnd"

    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:my="clr-namespace:System.Windows.Forms.Integration;assembly=WindowsFormsIntegration"
    Title="Maintenance" Height="480" Width="640" Loaded="Window_Loaded"
    FontSize="13" FontFamily="Arial" SnapsToDevicePixels="True">
    <Window.Resources>
        <Style x:Key="textBlockStyle" TargetType="TextBlock">
            <Setter Property="VerticalAlignment" Value="Center" />
            <Setter Property="Margin" Value="40, 0, 0, 0" />
        </Style>
        <Style x:Key="comboStyle" TargetType="ComboBox">
            <Setter Property="VerticalAlignment" Value="Center" />
            <Setter Property="Margin" Value="0, 0, 10, 0" />
        </Style>
        <DataTemplate x:Key="stationListTemplate">
            <Grid>
                <Grid.RowDefinitions>
                    <RowDefinition Height="25" />
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="120"/>
                    <ColumnDefinition Width="95"/>
                </Grid.ColumnDefinitions>
                <Border Grid.Column="0" BorderBrush="Gray" BorderThickness="0, 0, 1, 0" >
                    <TextBlock Text="{Binding Path = Station}" TextAlignment="Left"
                               VerticalAlignment="Center" FontSize="12" Margin="2,0,0,2" />
                </Border>
                <Border Grid.Column="1">
                    <TextBlock Text="{Binding Path = VersNavn}" TextAlignment="Center"
                               VerticalAlignment="Center" FontSize="12" Margin="2,0,0,2" />
                </Border>
            </Grid>
        </DataTemplate>
        <Style x:Key="stationListStyle" TargetType="ListBoxItem">
            <Setter Property="Height" Value="22" />
            <Setter Property="BorderBrush" Value="Gray" />
            <Setter Property="BorderThickness" Value="1, 1, 1, 0" />
        </Style>
    </Window.Resources>
    <Window.Background>
        <LinearGradientBrush EndPoint="1,0.5" StartPoint="0,0.5">
            <GradientStop Color="WhiteSmoke" Offset="0.5"/>
        </LinearGradientBrush>
    </Window.Background>
    <Grid >
        <Grid.RowDefinitions>
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition  Width="250"/>
            <ColumnDefinition />
            <ColumnDefinition />
        </Grid.ColumnDefinitions>
        <Grid Grid.Row="0" Grid.Column="0" >
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="100" />
                <ColumnDefinition />
            </Grid.ColumnDefinitions>
```

```
    <TextBlock Text="Vandløb navn" Margin="10, 0, 0, 0" VerticalAlignment="Center"/>
    <ComboBox Grid.Column="1" Name="comboVlIdent" VerticalAlignment="Center"
              DisplayMemberPath="VlNavn" SelectionChanged="comboVlIdent_SelectionChanged"/>
</Grid>
<Grid Grid.Column="1" Grid.ColumnSpan="2" Grid.RowSpan="1">
    <Grid.ColumnDefinitions>
        <ColumnDefinition />
        <ColumnDefinition Width="Auto"/>
        <ColumnDefinition />
    </Grid.ColumnDefinitions>
    <StackPanel Grid.Column="1" Orientation="Horizontal" VerticalAlignment="Center"
                Height="40" Background="White">
        <TextBlock Text="Første station" VerticalAlignment="Center" Margin="10,0,0,0"/>
        <TextBlock Name="txtFirstStation" Width="60" Margin="10, 0, 0, 0"
                   VerticalAlignment="Center"/>
        <TextBlock Margin="20, 0, 0, 0" VerticalAlignment="Center" Text="Sidste station" />
        <TextBlock Name="txtLastStation" Width="60" Margin="10, 0, 10, 0"
                    VerticalAlignment="Center" />
    </StackPanel>
</Grid>

<Grid Grid.Row="1" Margin="10, 0, 0, 0">
    <Grid.ColumnDefinitions>
        <ColumnDefinition  Width="120"/>
        <ColumnDefinition />
    </Grid.ColumnDefinitions>
    <Border Background="Silver" Margin="0, 20, 0, 0">
        <TextBlock Text="Station" TextAlignment="Center" VerticalAlignment="Center"/>
    </Border>
    <Border Grid.Column="1" Background="Silver" Margin="0, 20, 0, 0">
        <TextBlock Text="Opmåling" TextAlignment="Center" VerticalAlignment="Center" />
    </Border>
</Grid>

<ListBox Grid.Row="2" Grid.RowSpan="6" Name="listStationer" Margin="10, 0 , 0, 0"
         ItemTemplate="{StaticResource stationListTemplate}"
         ItemContainerStyle="{StaticResource stationListStyle}"
         SelectionMode="Extended" SelectionChanged="listStationer_SelectionChanged" />
<TextBox Grid.Column="2" Grid.Row="1" VerticalAlignment="Center" Margin="0, 0, 10, 0"
          Name="txtboxCaption"/>
<ComboBox Grid.Column="2" Grid.Row="2" Name="comboCategory"
          Style="{StaticResource comboStyle}"
          SelectionChanged="comboCategory_SelectionChanged"
          DisplayMemberPath="CategoryName">
</ComboBox>
<ComboBox Grid.Column="2" Grid.Row="3" Name="comboType" Style="{StaticResource comboStyle}"
          SelectionChanged="comboType_SelectionChanged" DisplayMemberPath="Key" />
<ComboBox Grid.Column="2" Grid.Row="4" Name="comboMethod"
          Style="{StaticResource comboStyle}"
          ItemsSource="{Binding Path = Keys}"
          SelectionChanged="comboMethod_SelectionChanged" />
<my:WindowsFormsHost Grid.Column="2" Grid.Row="5" Name="windowsFormsHost1"
                     Height="25" Margin="0, 0, 10, 0" />
<my:WindowsFormsHost Grid.Column="2" Grid.Row="6" Name="windowsFormsHost2"
                     Height="25" Margin="0, 0, 10, 0" />
<ComboBox Grid.Column="2" Grid.Row="7" Height="21" Name="comboContractor"
          Style="{StaticResource comboStyle}"
          ItemsSource="{Binding Path = Keys}"
          SelectionChanged="comboContractor_SelectionChanged" />
<TextBlock Grid.Column="1" Grid.Row="1" Name="txtCaption" Text="Beskrivelse"
          Style="{StaticResource textBlockStyle}"/>
<TextBlock Grid.Column="1" Grid.Row="2" Name="txtCategory" Text="Kategori"
          Style="{StaticResource textBlockStyle}"/>
<TextBlock Grid.Column="1" Grid.Row="3" Name="txtType" Text="Type"
          Style="{StaticResource textBlockStyle}" />
<TextBlock Grid.Column="1" Grid.Row="4" Name="txtMethod" Text="Metode"
          Style="{StaticResource textBlockStyle}"/>
<TextBlock Grid.Column="1" Grid.Row="5" Name="txtFirst" Text="Slut dato"
          Style="{StaticResource textBlockStyle}"/>
<TextBlock Grid.Column="1" Grid.Row="6" Name="txtLast" Text="Start dato"
          Style="{StaticResource textBlockStyle}" />
<TextBlock Grid.Column="1" Grid.Row="7" Name="txtContractor" Text="Entreprenør"
          Style="{StaticResource textBlockStyle}" />
<StackPanel Grid.Row="8" Grid.Column="0" Grid.ColumnSpan="3">
```

```
            <Grid>
                <Grid.RowDefinitions>
                    <RowDefinition Height="50" />
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition />
                    <ColumnDefinition />
                    <ColumnDefinition />
                    <ColumnDefinition />
                </Grid.ColumnDefinitions>
                <Button Grid.Column="1" IsDefault="True" VerticalAlignment="Center" Height="26"
                        Margin="0, 0, 10, 0" HorizontalAlignment="Right" Width="140"
                        Click="Button_Opret">
                    <AccessText>_Opret vedligeholdelse</AccessText>
                </Button>
                <Button Grid.Column="2" IsCancel="True" VerticalAlignment="Center" Height="26"
                        Margin="10, 0, 0, 0" HorizontalAlignment="Left" Width="140"
                        Click="Button_Cancel">
                    <AccessText>
                        _Annuller
                    </AccessText>
                </Button>
            </Grid>
        </StackPanel>
        <StatusBar Grid.Row="9" Grid.Column="0" Grid.ColumnSpan="3">
            <StackPanel Orientation="Horizontal">
                <TextBlock Text="Antal stationer: " />
                <TextBlock Text="{Binding Items.Count, ElementName=listStationer}"
                           Margin="0, 0, 20, 0" />
                <TextBlock Text="Der er valgt: "/>
                <TextBlock Name="txtStationCount"
                           Text="{Binding SelectedItems.Count, ElementName=listStationer}"
                           TextAlignment="Center" Width ="40" />
                <TextBlock Text="stationer." />
            </StackPanel>
        </StatusBar>
    </Grid>
</Window>
```

MaintenanceWnd.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
//
using ProkaApp.BusinessLogic;

namespace ProkaApp
{
    /// <summary>
    /// Interaction logic for Maintenance.xaml
    /// </summary>
    public partial class MaintenanceWnd : Window
    {
      private bool IsSelectionValid = false;
        private double firstStation;
        private double lastStation;
        private System.Collections.Hashtable SelectedStations;
        private System.Collections.ArrayList SelectedIndexes;
        // Windows Forms controls
        private System.Windows.Forms.DateTimePicker dateStart;
        private System.Windows.Forms.DateTimePicker dateFinish;
        //
        public IBusinessLogic BusinessLogic { get; set; }
        public MaintenanceWnd()
```

```
{
    InitializeComponent();
    dateStart = new System.Windows.Forms.DateTimePicker { Value = DateTime.Today };
    dateFinish = new System.Windows.Forms.DateTimePicker
                        { Value = DateTime.Today.AddDays(1) };
    dateStart.ValueChanged += new EventHandler(dateStart_ValueChanged);
    dateFinish.ValueChanged += new EventHandler(dateFinish_ValueChanged);
    windowsFormsHost1.Child = dateStart;
    windowsFormsHost2.Child = dateFinish;
 SelectedStations = new System.Collections.Hashtable();
 SelectedIndexes  = new System.Collections.ArrayList();
}
//
void dateStart_ValueChanged(object sender, EventArgs e)
{
    return;
}
//
void dateFinish_ValueChanged(object sender, EventArgs e)
{
    return;
}
//
private void Button_Opret(object sender, RoutedEventArgs e)
{
 // Validation
    if (!IsSelectionValid)
    {
        string msg = "En vedligeholdelse strækning skal bestå af ";
        msg += "en række sammenhængende stationer";
        AppMessageBox.ShowWarning(msg, "Advarsel");
        return;
    }
 if (this.comboCategory.SelectedIndex < 0)
 {
    AppMessageBox.ShowWarning("Kategori skal defineres!", "Advarsel");
    return;
 }
 if (this.comboType.SelectedIndex < 0)
 {
    AppMessageBox.ShowWarning("Type skal defineres!", "Advarsel");
    return;
 }
 if (dateStart.Value.Date.CompareTo(dateFinish.Value.Date) > 0)
 {
    AppMessageBox.ShowWarning("Ugyldig dato kombination!", "Advarsel");
    return;
 }
 Maintenance m = this.Tag as Maintenance;
    System.Data.DataRowView d = comboVlIdent.SelectedItem as System.Data.DataRowView;
    m.VlNavn = String.Format("{0}", d.Row[1]);  // VlNavn
    m.VlId   = (int)d.Row[0];                   // VlId
    m.TaskName = txtboxCaption.Text;
    m.Start  = dateStart.Value.Date;
    m.End    = dateFinish.Value.Date;
    m.Duration = (dateFinish.Value.Date - dateStart.Value.Date).Days + 1;
    m.StartStation = firstStation;
    m.EndStation   = lastStation;
    m.Completed = 0;
    m.Type.Name = String.Format("{0}", comboType.Text);
    m.Type.Id   =
        (int)((KeyValuePair<string, int>)comboType.SelectedItem).Value;
    m.Category.Name = String.Format("{0}",
        (comboCategory.SelectedItem as System.Data.DataRowView).Row[1]);
    m.Category.Id = (int)(comboCategory.SelectedItem as System.Data.DataRowView).Row[0];
    // Rebuild list of stations <TopoStationerId, Station>
    System.Collections.ArrayList datalist = new System.Collections.ArrayList();
    foreach (int i in SelectedIndexes)
    {
        System.Data.DataRowView data;
        data = this.listStationer.Items[i] as System.Data.DataRowView;
        datalist.Add(data.Row.ItemArray);
        /* [Topostationer] = items
         * ---------------
```

```
            * 0 - MaintenanceID
            * 1 - TopoVersListe.Id
            * 2 - VersNavn
            * 3 - TopoStationerId
            * 4 - Station
            */
        }
        int id = BusinessLogic.AddNewMaintenance(m, datalist);
        this.Close();
    }
    //
    private void Button_Cancel(object sender, RoutedEventArgs e)
    {
        this.Close();
    }
    //
    private void Window_Loaded(object sender, RoutedEventArgs e)
    {
        AppDocument doc = BusinessLogic as AppDocument;
     Binding bindcategories = new Binding();
     bindcategories.Source  = doc.CategoriesOfMaintenance;
     this.comboCategory.SetBinding(ComboBox.ItemsSourceProperty, bindcategories);
        //
        Binding bindingTypes = new Binding();
        bindingTypes.Source = doc.ListOfMaintenanceTypes;
        this.comboType.SetBinding(ComboBox.ItemsSourceProperty, bindingTypes);
        this.comboVlIdent.SetBinding(ComboBox.ItemsSourceProperty, doc.VlIdentBinding);
        this.listStationer.SetBinding(ListBox.ItemsSourceProperty, doc.TopoStationerBinding);
    }
    //
    private void comboVlIdent_SelectionChanged(object sender, SelectionChangedEventArgs e)
    {
     object o = (sender as ComboBox).SelectedItem;
        System.Data.DataRowView item = o as System.Data.DataRowView;
        object s = item.Row["VlNavn"];
        BusinessLogic.HentStationer(String.Format("{0}", s));
     // Clean up previous results
     SelectedIndexes.Clear();
     SelectedStations.Clear();
     txtFirstStation.Text = "";
     txtLastStation.Text  = "";
    }

private void listStationer_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    try
    {
        this.IsSelectionValid = true;
        //
        int index;
        System.Collections.IList items = e.AddedItems;
        System.Data.DataRowView rowItem;
        // Add Items
        foreach (object item in items)
        {
            rowItem = item as System.Data.DataRowView;
            if (listStationer.Items.Contains(item))
            {
                index = listStationer.Items.IndexOf(item);
                SelectedStations.Add(index, rowItem.Row["Station"]);
                SelectedIndexes.Add(index);
            }
        }
        // Remove Items
        items = e.RemovedItems;
        foreach (var item in items)
        {
            rowItem = item as System.Data.DataRowView;
            if (listStationer.Items.Contains(item))
            {
                index = listStationer.Items.IndexOf(item);
                SelectedStations.Remove(index);
                SelectedIndexes.Remove(index);
            }
```

```
            }
            // Handle selected items
            SelectedIndexes.Sort();
            for (int i = 0; i < SelectedIndexes.Count - 1; i++)
            {
               int first  = (int)SelectedIndexes[i];
               int second = (int)(SelectedIndexes[i + 1]);
               bool res = (first + 1 == (second)) ? true : false;
               if (!res)
               {
                  IsSelectionValid = false;
                  break;
               }
            }
            //
            int cnt = SelectedIndexes.Count;
            if (cnt == 1)
            {
               object foo = SelectedStations[SelectedIndexes[0]];
               txtFirstStation.Text = String.Format("{0}", foo);
               txtLastStation.Text  = "";
               IsSelectionValid   = false;
            }
            else if (cnt > 1)
            {
                    firstStation = (double)SelectedStations[SelectedIndexes[0]];
                    lastStation  = (double)SelectedStations[SelectedIndexes[cnt - 1]];

                    txtFirstStation.Text = String.Format("{0}", firstStation);
               txtLastStation.Text  = String.Format("{0}", lastStation);
            }
            //
            if (!IsSelectionValid)
            {
               txtFirstStation.Foreground = Brushes.Red;
               txtLastStation.Foreground = Brushes.Red;
            }
            else
            {
               txtFirstStation.Foreground = Brushes.Black;
               txtLastStation.Foreground = Brushes.Black;
            }

         }
         catch (NotSupportedException ex)
         {
#if DEBUG
            AppMessageBox.ShowError(ex.StackTrace, ex.Message);
#else
            AppMessageBox.ShowError(ex.Message, ex.Source);
#endif
         }
         catch (ArgumentException ex)
         {
            AppMessageBox.ShowError(ex.Message, ex.Source);
         }
      }

      private void comboCategory_SelectionChanged(object sender, SelectionChangedEventArgs e)
      {
            System.Collections.IList l = e.AddedItems;
            int count = l.Count;
            if (count > 0)
            {
                System.Data.DataRowView data = l[0] as System.Data.DataRowView;
                string s = String.Format("{0}", data.Row[1]);
                this.BusinessLogic.GetMaintenanceTypes(s);
            }
      }

      private void comboType_SelectionChanged(object sender, SelectionChangedEventArgs e)
      {
            System.Collections.IList l = e.AddedItems;
            int count = l.Count;
```

```
            if (count > 0)
            {
                KeyValuePair<String, int> data = (KeyValuePair<String, int>)l[0];
                string s = String.Format("{0}", data.Key);
                this.BusinessLogic.GetMaintenanceMethods(s);
            }
        }

        private void comboMethod_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            System.Collections.IList l = e.AddedItems;
        }

        private void comboContractor_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            System.Collections.IList l = e.AddedItems;
          }
    }
}
```

MainWindow.xaml

```
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:sys="clr-namespace:System;assembly=mscorlib"
    xmlns:proka="clr-namespace:ProkaApp"
    xmlns:prokaUI="clr-namespace:ProkaApp.UI"
    xmlns:prokaLogic="clr-namespace:ProkaApp.BusinessLogic"
    xmlns:AppTheme="clr-namespace:Microsoft.Windows.Themes;assembly=PresentationFramework.Luna"
    x:Class="ProkaApp.MainWindow"
    Loaded="WndLoaded"
    Title="PROKA App" Height="768" Width="1024">
    <Window.Resources>
        <!---->
        <Style x:Key="TaskListStyle" TargetType="{x:Type ListViewItem}">
            <Setter Property="BorderBrush" Value="Black" />
            <Setter Property="BorderThickness" Value="0,0,0,1" />
            <Setter Property="HorizontalContentAlignment" Value="Center" />
            <Setter Property="Height" Value="25" />
        </Style>
        <!---->
        <Style x:Key="myHeaderStyle" TargetType="{x:Type GridViewColumnHeader}">
            <Setter Property="Height" Value="38" />
            <Setter Property="Margin" Value="0, 0, 0, 4" />
        </Style>
        <Style x:Key="styleMaintenaceTypesList" TargetType="{x:Type ListBoxItem}">
            <Setter Property="Margin" Value="10, 10, 10, 0" />
            <Setter Property="BorderBrush" Value="Black" />
            <Setter Property="BorderThickness" Value="1" />
        </Style>
        <!-- Content List style and datatemplate-->
        <Style x:Key="ContentListItemStyle" TargetType="{x:Type ListBoxItem}">
            <Setter Property="BorderBrush" Value="LightGray" />
            <Setter Property="BorderThickness" Value="0,0,0,1" />
            <Setter Property="HorizontalContentAlignment" Value="Center" />
            <Setter Property="Focusable" Value="False" />
            <Setter Property="VerticalContentAlignment" Value="Top" />
            <Setter Property="HorizontalContentAlignment" Value="Left" />
        </Style>
        <prokaUI:DateToPointConverter x:Key="convertDateToPoint"
                                      x:Name="ContentListItemConvertet" />
        <prokaUI:TypeToColorConverter x:Key="convertTypeToColor"
                                      x:Name="ContentListItemColorConverter" />
        <DataTemplate x:Key="ContentListTemplate">
            <prokaUI:ContentListItem Name="AListItem"
                Start="{Binding Path = Start, Converter={StaticResource convertDateToPoint}}"
                Duration="{Binding Path = Duration}"
                Description="{Binding Path = TaskName}"
                ItemColor = "{Binding Path = Type.Name,
                Converter={StaticResource convertTypeToColor}}"
                ItemSize = "{Binding RelativeSource={RelativeSource FindAncestor,
```

```
                AncestorType={x:Type prokaUI:ContentListBox}}, Path = ReferenceWidth}">
        </prokaUI:ContentListItem>
    </DataTemplate>
    <prokaUI:ContentListBox x:Name="ListBoxContent" x:Key="ContentListView" Margin="0, 2, 0, 0"
             BorderThickness="0" IsSynchronizedWithCurrentItem="True"
             HorizontalAlignment="Left" VerticalAlignment="Top"
             ItemContainerStyle="{StaticResource ContentListItemStyle}"
             ItemTemplate="{StaticResource ContentListTemplate}">

    </prokaUI:ContentListBox>
    <ContextMenu x:Key="ItemListContextMenu">
        <MenuItem Header="Slet" Click="MenuItem_Click" />
    </ContextMenu>
</Window.Resources>
<Grid x:Name="TopGrid">
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto" />
        <RowDefinition Height="*"/>
        <RowDefinition Height="Auto"/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition />
    </Grid.ColumnDefinitions>
    <StackPanel Grid.Row="0" x:Name="PanelTools" Orientation="Vertical">
        <Menu x:Name="AppMenu">
            <Menu.Background>
                <LinearGradientBrush EndPoint="1, 0" StartPoint="0,0">
                    <GradientStop Color="WhiteSmoke" Offset="0.5"/>
                    <GradientStop Color="WhiteSmoke" Offset="0.7"/>
                </LinearGradientBrush>
            </Menu.Background>
            <MenuItem>
                <MenuItem.Header>
                    <AccessText>
                        _Funktioner
                    </AccessText>
                </MenuItem.Header>
                <MenuItem Header="_Vis oversigt" Click="VisVlOversig"></MenuItem>
                <MenuItem Header="_Exit" Click="MenuExit_Click"></MenuItem>
            </MenuItem>
        </Menu>
        <!--<ToolBarOverflowPanel Grid.Row="0">-->
        <ToolBarTray>
            <ToolBarTray.Background>
                <LinearGradientBrush EndPoint="1, 0" StartPoint="0,0">
                    <GradientStop Color="WhiteSmoke" Offset="0.5"/>
                    <GradientStop Color="WhiteSmoke" Offset="0.7"/>
                </LinearGradientBrush>
            </ToolBarTray.Background>
            <ToolBar Band="1" Background="Transparent">
                <Label VerticalAlignment="Center">Visning</Label>
                <ComboBox Name="ComboSelectTimeUnit" MinWidth="80" SelectedIndex="-1"
                      SelectionChanged="ComboSelectTimeUnit_SelectionChanged">
                    <ComboBoxItem>Dage</ComboBoxItem>
                    <ComboBoxItem>Uger</ComboBoxItem>
                    <ComboBoxItem>Måneder</ComboBoxItem>
                    <ComboBoxItem>År</ComboBoxItem>
                </ComboBox>
                <Label VerticalAlignment="Center">View</Label>
                <ComboBox Name="ComboView" MinWidth="80" SelectedIndex="0"
                      SelectionChanged="ComboBox_SelectionChanged">
                    <ComboBoxItem>Temporal</ComboBoxItem>
                    <ComboBoxItem>Spatial</ComboBoxItem>
                </ComboBox>
                <Label Name="LbConnection" VerticalAlignment="Center">Data kilder</Label>
                <ComboBox Name="ComboConnections" MinWidth="100" DisplayMemberPath="Name"
                      SelectionChanged="ComboConnections_SelectionChanged" IsEnabled="False"
                      SelectedIndex="1">
                </ComboBox>
            </ToolBar>
            <ToolBar Band="2" Background="Transparent">
                <AppTheme:ButtonChrome Margin="10, 0, 0, 0">
                    <Button x:Name="BtnReadData" Width="Auto" Click="BtnReadData_Click">
                        <AccessText>
```

```
                        _Indlæs data
                    </AccessText>
                </Button>
            </AppTheme:ButtonChrome>
            <AppTheme:ButtonChrome Margin="10, 0, 0, 0">
                <Button x:Name="BtnNyVedligeholdelse" Width="Auto"
                        Click="NewMaintenace_Click">
                    <AccessText>
                        _Ny vedligeholdelse
                    </AccessText>
                </Button>
            </AppTheme:ButtonChrome>
            <AppTheme:ButtonChrome Margin="10, 0, 0, 0">
                <Button x:Name="BtnRydSortering" Width="Auto" Click="BtnRydSortering_Click">
                    <AccessText>
                        _Ryd sortering
                    </AccessText>
                </Button>
            </AppTheme:ButtonChrome>
            <AppTheme:ButtonChrome Margin="10, 0, 0, 0">
                <Button x:Name="BtnRydFiltrering" Width="Auto"
                        Click="BtnRydFiltrering_Click">
                    <AccessText>
                        R_yd filtrering
                    </AccessText>
                </Button>
            </AppTheme:ButtonChrome>
        </ToolBar>
    </ToolBarTray>
</StackPanel>
<Grid Grid.Row="1" Name="PanelData" ShowGridLines="False">
    <Grid.RowDefinitions>
        <RowDefinition />
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="Auto"></ColumnDefinition>
        <ColumnDefinition Width="*"></ColumnDefinition>
        <ColumnDefinition Width="Auto"></ColumnDefinition>
    </Grid.ColumnDefinitions>
    <ListView Grid.Column="0" Name="AppListView" SelectionChanged="MntView_SelectionChanged"
              ItemContainerStyle="{StaticResource TaskListStyle}"
              ContextMenu="{StaticResource ItemListContextMenu}"
              GridViewColumnHeader.Click = "ColumnHeaderClick"
              ScrollViewer.VerticalScrollBarVisibility="Hidden"
              ScrollViewer.ScrollChanged="AppListView_ScrollChanged"
              SelectionMode="Single" Margin="0, 0, 0, 17">
        <ListView.View>
            <GridView ColumnHeaderContainerStyle="{StaticResource myHeaderStyle}">
                <GridViewColumn Header="Task ID" Width="Auto"
                                DisplayMemberBinding="{Binding Path=ID}"/>
                <GridViewColumn Header="Start station" Width="Auto"
                                DisplayMemberBinding="{Binding Path=StartStation}" />
                <GridViewColumn Header="Slut station" Width="Auto"
                                DisplayMemberBinding="{Binding Path=EndStation}"/>
                <GridViewColumn Header="Vandløb" Width="Auto"
                                DisplayMemberBinding="{Binding Path=VlNavn}" />
                <GridViewColumn Header="Varighed" Width="Auto"
                                DisplayMemberBinding="{Binding Path=Duration}" />
            </GridView>
        </ListView.View>
        <!-- List items. -->
    </ListView>
    <GridSplitter Grid.Column="0" Margin="0" Width="2" Name="Split1" />
    <Grid x:Name="ContentGrid" Grid.Column="1" ShowGridLines="False">
        <Grid.RowDefinitions>
            <RowDefinition Height="40" />
            <RowDefinition />
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition />
            <ColumnDefinition Width="Auto"/>
        </Grid.ColumnDefinitions>
        <ScrollViewer x:Name="ReferencePane" Grid.Row="0" Grid.Column="0"
```

```
                                HorizontalAlignment="Left"
                                HorizontalScrollBarVisibility="Hidden"
                                VerticalScrollBarVisibility="Hidden"
                                CanContentScroll="False">
                        <!-- Content = CustomPanel -->
                    </ScrollViewer>
                    <ScrollViewer x:Name="ViewPane" Grid.Row="1" Grid.Column="0"
                                Margin="0"
                                HorizontalScrollBarVisibility="Hidden"
                                VerticalScrollBarVisibility="Hidden"
                                CanContentScroll="False">
                        <!-- Content -->
                    </ScrollViewer>
                    <!---->
                    <ScrollBar x:Name="VScroll" Grid.Row="0" Grid.Column="1" Grid.RowSpan="2" Margin="1"
                               Scroll="VScroll_Scroll"> </ScrollBar>
                    <ScrollBar x:Name="HScroll" Grid.Row="2" Grid.Column="0" Orientation="Horizontal"
                                Margin="1"
                               Scroll="HScroll_Scroll"></ScrollBar>
                </Grid>
                <GridSplitter Grid.Column="2" Width="2" HorizontalAlignment="Left" />
                <Grid Grid.Column="2">
                    <ListBox Name="mTypesList" Grid.Column="2" Background="WhiteSmoke"
                             ItemContainerStyle="{StaticResource styleMaintenaceTypesList}"
                             SelectionChanged="mTypesList_SelectionChanged">
                        <ListBox.ItemTemplate>
                            <DataTemplate>
                                <Grid>
                                    <Grid.ColumnDefinitions>
                                        <ColumnDefinition Width="40" />
                                        <ColumnDefinition Width="100"/>
                                        <ColumnDefinition Width="40" />
                                    </Grid.ColumnDefinitions>
                                    <Grid.RowDefinitions>
                                        <RowDefinition Height="40"/>
                                        <RowDefinition Height="40"/>
                                    </Grid.RowDefinitions>
                                    <Border Grid.Column="1" Margin="10"
                                            Background="{Binding Path = Value}"
                                           BorderThickness="1" BorderBrush="Black" />
                                    <TextBlock Grid.Row="1" Grid.Column="0" Grid.ColumnSpan="3"
                                               TextAlignment="Center"
                                               Text="{Binding Path = Key}" TextWrapping="Wrap" />
                                </Grid>
                            </DataTemplate>
                        </ListBox.ItemTemplate>
                    </ListBox>
                </Grid>
            </Grid>
            <StatusBar x:Name="ProkaAppStatus" Grid.Row="2" Padding="0" Margin="0">
                <StackPanel Orientation="Horizontal" Height="20">
                    <TextBlock HorizontalAlignment="Center"
                         VerticalAlignment="Center">ProkaApp</TextBlock>
                </StackPanel>
            </StatusBar>
        </Grid>
    </Window>
```

MainWindow.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
```

```
//
using System.Collections;
using System.Windows.Controls.Primitives; // ScrollBar
using System.ComponentModel;
using ProkaApp.UI;
using ProkaApp.BusinessLogic;
//
namespace ProkaApp
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        private double VItemHeight = 0; // Vista = 26 and XP = 25
        private double ViewPaneContentHeight = 0;
        private double ViewPaneContentWidth  = 0;
        private ScrollViewer sv;
        public IBusinessLogic Document { get; set; }
        public NView TView { get; set; }
        public NView SView { get; set; }

        public MainWindow()
        {
            InitializeComponent();
        }

        private void WndLoaded(object sender, RoutedEventArgs e)
        {
            try
            {
#if DEBUG
                System.Diagnostics.Debug.Assert(Document != null);
#endif
                AppDocument AppDoc = Document as AppDocument;
                // Initialize custom components

                TView = new TemporalListBox(Document.Tasks);
                //
                TView.TypesAndColorsList = (Document as AppDocument).TypesAndColorsList;
                TView.AppViewChanged += new AppViewChangedHandler(TView_AppViewChanged);

                //
                this.ReferencePane.Content = TView.TopPanel;

                this.ViewPane.Content = (TView as TemporalListBox).ContentBox;
                //
                Binding taskbinding = new Binding();
                taskbinding.Source = Document.Tasks;
                AppListView.SetBinding(ListView.ItemsSourceProperty, taskbinding);
                Binding bindTaskColor = new Binding();
                bindTaskColor.Source = AppDoc.TypesAndColorsList;
                mTypesList.SetBinding(ListBox.ItemsSourceProperty, bindTaskColor);
                //
                ComboConnections.SetBinding(ComboBox.ItemsSourceProperty, AppDoc.DataConnections);
                //
                if (System.Environment.OSVersion.Version.Major < 6)
                    VItemHeight = 25; // WinXP
                else
                    VItemHeight = 26; // Vista
                ViewPaneContentWidth = TView.TopPanel.Width;
                HScroll.SizeChanged += new SizeChangedEventHandler(HScroll_SizeChanged);
                VScroll.SizeChanged += new SizeChangedEventHandler(VScroll_SizeChanged);
                //
                double val = Math.Round(this.TView.TopPanel.Width / 100);
                this.HScroll.Maximum = this.TView.TopPanel.Width - val;
                this.HScroll.SmallChange = val;
                this.HScroll.LargeChange = val *4;
                // PrintVisualTree(0, AppListView);
                DependencyObject themeobj= VisualTreeHelper.GetChild(AppListView, 0);
                DependencyObject scrollobj = VisualTreeHelper.GetChild(themeobj, 0);
                sv = scrollobj as ScrollViewer;
                sv.ScrollChanged += new ScrollChangedEventHandler(sv_ScrollChanged);
            }
```

```
    catch (Exception ex)
    {
        MessageBox.Show(ex.StackTrace, ex.Message, MessageBoxButton.OK,
                                                        MessageBoxImage.Error);
    }
}

void VScroll_SizeChanged(object sender, SizeChangedEventArgs e)
{
    double max = 0;
    double val = 0;
    if (ViewPane.ActualHeight == e.NewSize.Height)
        max = 0; // dummy;
    if (Double.IsNaN(ViewPane.ActualHeight))
        max = ViewPaneContentHeight;
    else
        max = ViewPaneContentHeight - ViewPane.ActualHeight;
    if (max < 0)
        max = ViewPaneContentHeight;
    this.VScroll.Maximum = Math.Round(max);
    val = Math.Round(max / 256);
    this.VScroll.SmallChange = val;
    this.VScroll.LargeChange = val * 8;
}
//
void HScroll_SizeChanged(object sender, SizeChangedEventArgs e)
{
    double max = 0;
    double val = 0;
    if (ViewPane.ActualWidth == e.NewSize.Width)
        max = 0; // dummy
    if (Double.IsNaN(ViewPane.ActualWidth))
        max = ViewPaneContentWidth;
    else
        max = ViewPaneContentWidth - ViewPane.ActualWidth;
    if (max < 0)
        max = ViewPaneContentWidth;
    this.HScroll.Maximum = max;
    val = Math.Round(max / 128);
    this.HScroll.SmallChange = val;
    this.HScroll.LargeChange = val * 4;
}

void TView_AppViewChanged(object sender, AppViewChangedArgs arg)
{
    double val = 0;
    double max = 0;
    switch (arg.AOrientation)
    {
        case Orientation.Horizontal:
            ViewPaneContentWidth = arg.Value;
            if (Double.IsNaN(ViewPane.ActualWidth))
                max = arg.Value;
            else
                max = arg.Value - ViewPane.ActualWidth;
            if (max < 0)
                max = arg.Value;
            this.HScroll.Maximum = max;
            val = Math.Round(max / 128);
            this.HScroll.SmallChange = val;
            this.HScroll.LargeChange = val * 4;
            break;
        case Orientation.Vertical:
            ViewPaneContentHeight = arg.Value;
            if (Double.IsNaN(ViewPane.ActualHeight))
                max = arg.Value;
            else
                max = arg.Value - ViewPane.ActualHeight;
            if (max < 0)
                max = arg.Value;
            this.VScroll.Maximum = Math.Round(max);
            val = Math.Round(max / 256);
            this.VScroll.SmallChange = val;
            this.VScroll.LargeChange = val * 8;
```

```
            break;
        default: break;
    }
}

void sv_ScrollChanged(object sender, ScrollChangedEventArgs e)
{
    return;
}

void PrintVisualTree(int depth, DependencyObject obj)
{
    System.Diagnostics.Debug.WriteLine(String.Format("{0}, {1}", depth, obj));
    int count = VisualTreeHelper.GetChildrenCount(obj);
    for (int i = 0; i < count; i++)
    {
        PrintVisualTree(depth + 1, VisualTreeHelper.GetChild(obj, i));
    }
}

private void MntView_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    return;
}

private void VisVlOversig(object sender, RoutedEventArgs e)
{
    VandloebView wnd = new VandloebView();
    wnd.Data = Document.HentOversigtData();
    wnd.Tag  = Document;
    wnd.ShowDialog();
}

private void MenuExit_Click(object sender, RoutedEventArgs e)
{
    Application.Current.Shutdown();
}

private void ComboBox_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
 //
    if (this.ComboView.SelectedIndex < 0 )
        return;
    object sel = (this.ComboView.SelectedItem as ComboBoxItem).Content;
    string s = sel.ToString().Trim();
    if (this.ComboView.SelectedIndex == 1)
        AppMessageBox.ShowInfo("Spatial view er ikke implementeret i denne version",
            "Information");
}

private void NewMaintenace_Click(object sender, RoutedEventArgs e)
{
 Maintenance AMaintenance = new Maintenance();
    MaintenanceWnd wnd = new MaintenanceWnd();
 wnd.Tag = AMaintenance;
    wnd.BusinessLogic = this.Document;
 object res = wnd.ShowDialog();
 if (res == null)
    return;
 else
 {
    if ((bool)res)
    {
       //Document.Tasks.Add(new ProkaApp.BusinessLogic.Maintenance());
    }
 }
 Document.ClearStationList();
}

private void BtnTypes_Click(object sender, RoutedEventArgs e)
{
}

private void ComboSelectTimeUnit_SelectionChanged(object sender,
```

```
                                                    SelectionChangedEventArgs e)
{
    ComboBox cb = sender as ComboBox;
    int i    = cb.SelectedIndex;
    switch (i)
    {
        case 0: TView.ChangeTimeUnit(DrawingMode.Days); break;
        case 1: TView.ChangeTimeUnit(DrawingMode.Weeks); break;
        case 2: TView.ChangeTimeUnit(DrawingMode.Months); break;
        case 3: TView.ChangeTimeUnit(DrawingMode.Year); break;
        default: break;
    }
    TView.RedrawAll();
}
//
private void BtnReadData_Click(object sender, RoutedEventArgs e)
{
    this.Document.ReadMaintenanceData();
}
//
private void ColumnHeaderClick(object sender, RoutedEventArgs e)
{
    object o = e.OriginalSource;
    GridViewColumnHeader head = o as GridViewColumnHeader;
    if (head != null)
    {
        string str = (string)head.Column.Header;
        if (str.Equals("Vandløb"))
        {
            ICollectionView colview = CollectionViewSource.GetDefaultView(Document.Tasks);
            if (colview.SortDescriptions.Count > 0 &&
                colview.SortDescriptions[0].PropertyName == "VlNavn" &&
                colview.SortDescriptions[0].Direction == ListSortDirection.Ascending)
            {
                colview.SortDescriptions.Clear();
                colview.SortDescriptions.Add(new SortDescription("VlNavn",
                    ListSortDirection.Descending));
            }
            else
            {
                colview.SortDescriptions.Clear();
                colview.SortDescriptions.Add(new SortDescription("VlNavn",
                    ListSortDirection.Ascending));
            }
        }
        else if (str.Equals("Beskrivelse"))
        {
            ICollectionView colview = CollectionViewSource.GetDefaultView(Document.Tasks);
            if (colview.SortDescriptions.Count > 0 &&
                colview.SortDescriptions[0].PropertyName == "TaskName" &&
                colview.SortDescriptions[0].Direction == ListSortDirection.Ascending)
            {
                colview.SortDescriptions.Clear();
                colview.SortDescriptions.Add(new SortDescription("TaskName",
                    ListSortDirection.Descending));
            }
            else
            {
                colview.SortDescriptions.Clear();
                colview.SortDescriptions.Add(new SortDescription("TaskName",
                    ListSortDirection.Ascending));
            }
        }
    }
}
//
private void BtnRydSortering_Click(object sender, RoutedEventArgs e)
{
    ICollectionView colview = CollectionViewSource.GetDefaultView(Document.Tasks);
    colview.SortDescriptions.Clear();
}
//
private void mTypesList_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
```

```
            int cnt = e.AddedItems.Count;
            if (cnt > 0)
            {
                DictionaryEntry de = (DictionaryEntry)e.AddedItems[0];
                string key = (string)de.Key;
                ICollectionView colView = CollectionViewSource.GetDefaultView(Document.Tasks);
                colView.Filter = delegate(object o)
                {
                    return ((o as Maintenance).Type.Name) == key;
                };
            }
        }
        //
        private void BtnRydFiltrering_Click(object sender, RoutedEventArgs e)
        {
            ICollectionView colView = CollectionViewSource.GetDefaultView(Document.Tasks);
            colView.Filter = null;
            // Remove selection from the item in the list
            mTypesList.SelectedIndex = -1;
        }

        private void AppListView_ScrollChanged(object sender, ScrollChangedEventArgs e)
        {
            return;
        }

        private void VScroll_Scroll(object sender, ScrollEventArgs e)
        {
            double value = Math.Round(e.NewValue / VItemHeight);
            this.ViewPane.ScrollToVerticalOffset(value * VItemHeight);
            this.sv.ScrollToVerticalOffset(value);
        }

        private void HScroll_Scroll(object sender, ScrollEventArgs e)
        {
            this.ReferencePane.ScrollToHorizontalOffset(e.NewValue);
            this.ViewPane.ScrollToHorizontalOffset(e.NewValue);
        }
        //
        private void MenuItem_Click(object sender, RoutedEventArgs e)
        {
            object o = AppListView.SelectedItem;
            if (o != null)
            {
               this.Document.DeleteMaintenance(o as Maintenance);
            }
        }

      private void ComboConnections_SelectionChanged(object sender, SelectionChangedEventArgs e)
      {
         ComboBox cb = sender as ComboBox;
         object o = cb.SelectedItem;
         Document.DataSetup(o);
      }
    }
}
```

ObservableDictionary.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
//
using System.Collections.ObjectModel;
using System.Collections.Specialized;
using System.ComponentModel;

namespace ProkaApp
{
    public class ObservableDictionary : ObservableCollection<KeyValuePair<String, Int32>>
```

```
    {
    }

    public class ObservableHashtable: ObservableCollection<DictionaryEntry>
    {
    }
}
```

ProkaAppCore.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Media;
using System.Configuration;
using ProkaApp.Data;
using ProkaApp.BusinessLogic;
using ProkaApp.UI;
//
namespace ProkaApp
{

    public interface IAppCore
    {
        IAppData Data { get; set; }
        IBusinessLogic Doc { get; set; }
    }
    //
    public interface INView
    {
    }
    //
    public interface IAppData
    {
        ConnectionStringSettingsCollection Connections { get; }
        System.Data.DataSet ProkaDataSet { get; }
        System.Data.DataSet GetWorkingSet();
        void ClearTable(string p);
        void SelectMaintenanceAll();
        void SelectVlIdent();
        void SelectTopoStationer(string vlnavn);
        void SelectMaintenanceCategories(ObservableDictionary listOfTypes);
        void SelectMaintenanceTypes(int i, ObservableDictionary listOfTypes);
        void SelectAllMaintenanceTypes();
        void SelectMaintenanceMethods(int i, ObservableDictionary listOfMethods);
        int InsertMaintenance(Maintenance m);
        void SelectTypesAndColors(ObservableHashtable list);
        int ValidateData();
        void ConnectionSetup(object o);
        int DeleteMaintenance(Maintenance maintenance);
   }
    //
    class ProkaAppCore : IAppCore
    {
        public IAppData  Data     { get; set; }
        public IBusinessLogic Doc { get; set; }
        public ProkaAppCore ()
        {
            Data    = new AppData();
            Doc     = new AppDocument(Data);
        }
    }
}
```

UIBlocks.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
using System.Windows;
using System.Windows.Input;
using System.Windows.Media;
//
namespace ProkaApp.UI
{
    #region ELEMENTS
    //
    public delegate void TimeSpanChangedHandler(object sender, object arg);
    //
    #region BASE
    abstract class NUIElement : FrameworkElement
    {
    }
    #endregion
    /*************************************************************************/
    #region TEMPORAL
    class TimeLine : FrameworkElement
    {
        public event TimeSpanChangedHandler TimeSpanChanged;
        public bool IsTimeSpanChanged { get; private set; }


        //
        #region TimeUnit DependencyProperty
        public DrawingMode TimeUnit
        {
            get { return (DrawingMode)GetValue(TimeUnitProperty); }
            set { SetValue(TimeUnitProperty, value); }
        }
        //
        public static readonly DependencyProperty TimeUnitProperty =
            DependencyProperty.Register("TimeUnit", typeof(DrawingMode),
            typeof(TimeLine), new FrameworkPropertyMetadata(DrawingMode.Days,
                FrameworkPropertyMetadataOptions.AffectsMeasure |
                FrameworkPropertyMetadataOptions.AffectsRender,
                new PropertyChangedCallback(OnTimeUnitChange)));
        //
        private static void OnTimeUnitChange(DependencyObject d,
                                             DependencyPropertyChangedEventArgs e)
        {
            TimeLine tl = d as TimeLine;
            DrawingMode dm = (DrawingMode)e.NewValue;
            switch (dm)
            {
                case DrawingMode.Days:
                    tl.ItemWidth = UnitDays;
                    break;
                case DrawingMode.Weeks:
                    tl.ItemWidth = UnitWeeks;
                    break;
                case DrawingMode.Months:
                    tl.ItemWidth = UnitMonths;
                    break;
                case DrawingMode.Year:
                    tl.ItemWidth = UnitYears;
                    break;
                default: break;
            }
            if (tl.PTimeSpan.Days < 0)
            {
                tl.Width = 0;
            }
            else
            {
                tl.Width = (tl.PTimeSpan.Days + 1) * tl.ItemWidth;
            }
        }
        #endregion

        #region FirstDay DependencyProperty
        public DateTime FirstDay
        {
            get { return (DateTime)GetValue(FirstDayProperty); }
```

```
    set { SetValue(FirstDayProperty, value); }
}

public static readonly DependencyProperty FirstDayProperty =
    DependencyProperty.Register("FirstDay", typeof(DateTime), typeof(TimeLine),
    new FrameworkPropertyMetadata(DateTime.Today.AddMonths(-1).Date,
        FrameworkPropertyMetadataOptions.AffectsMeasure |
        FrameworkPropertyMetadataOptions.AffectsRender,
        new PropertyChangedCallback(OnFirstDayChanged)));

private static void OnFirstDayChanged(DependencyObject d,
                                      DependencyPropertyChangedEventArgs de)
{
    if (d != null)
    {
        TimeLine tl = d as TimeLine;
        tl.PTimeSpan = tl.LastDay - ((DateTime)de.NewValue).Date;
        if (tl.PTimeSpan.Days < 0)
        {
            tl.Width = 0;
        }
        else
        {
            tl.Width = (tl.PTimeSpan.Days + 1) * tl.ItemWidth;
        }
        tl.IsTimeSpanChanged = true;
    }
}
#endregion

#region LastDay DependencyProperty
public DateTime LastDay
{
    get { return (DateTime)GetValue(LastDayProperty); }
    set { SetValue(LastDayProperty, value); }
}

public static readonly DependencyProperty LastDayProperty =
    DependencyProperty.Register("LastDay", typeof(DateTime), typeof(TimeLine),
    new FrameworkPropertyMetadata(DateTime.Today.AddMonths(3).Date,
        FrameworkPropertyMetadataOptions.AffectsRender |
        FrameworkPropertyMetadataOptions.AffectsMeasure,
        new PropertyChangedCallback(OnLastDayChanged)));

private static void OnLastDayChanged(DependencyObject d,
                                     DependencyPropertyChangedEventArgs de)
{
    if (d != null)
    {
        TimeLine tl = d as TimeLine;
        tl.PTimeSpan = ((DateTime)de.NewValue).Date - tl.FirstDay.Date;
        if (tl.PTimeSpan.Days < 0)
        {
            tl.Width = 0;
        }
        else
        {
            tl.Width = (tl.PTimeSpan.Days + 1) * tl.ItemWidth;
        }
        tl.IsTimeSpanChanged = true;
    }
}
#endregion
//

public TimeSpan PTimeSpan { get; private set; }
public DateTime Current { get; private set; }
public int DayCount { get; private set; }
public double ItemWidth { get; private set; }
//
private string[] months = {"", "Januar", "Februar", "Marts",
                           "April", "Maj", "Juni", "Juli",
                           "August", "September", "Oktober",
                           "November", "December" };
```

```
private const double UnitDays = 32;
private const double UnitWeeks = 16;
private const double UnitMonths = 8;
private const double UnitYears = 3;
//
private System.Globalization.CultureInfo cultureDK;
private Typeface txtArial;
private FlowDirection flowToRight;
private Brush transparent;
private Brush blackBrush;
private Pen blackPen;
private Size elementSize;
//
private DrawingVisual dv;
List<Visual> uilist = new List<Visual>();
//
public TimeLine()
{
    //this.SnapsToDevicePixels = false;
    try
    {
        // Event handlers
        this.MouseDown += new MouseButtonEventHandler(TimeLine_MouseDown);
        this.MouseMove += new MouseEventHandler(TimeLine_MouseMove);
        //
        ItemWidth = UnitDays;
        cultureDK = new System.Globalization.CultureInfo("da-DK");
        txtArial = new Typeface("Arial");
        flowToRight = FlowDirection.LeftToRight;
        transparent = Brushes.Transparent;
        blackBrush = Brushes.Black;
        blackPen = new Pen(Brushes.Black, 1);
        //
        PTimeSpan = LastDay.Date - FirstDay.Date;
        elementSize = new Size();
        elementSize.Width = PTimeSpan.Days * ItemWidth;
        this.Width = elementSize.Width;
        this.Height = 40;
        //
        Current = DateTime.Today.Date;
        //
        uilist = new List<Visual>();
        dv = new DrawingVisual();
        //
        this.ToolTip = FindResource("timelineTip");
        IsTimeSpanChanged = false;
    }
    catch (Exception exc)
    {
        MessageBox.Show(exc.StackTrace, exc.Message);
    }
}

void TimeLine_MouseMove(object sender, MouseEventArgs e)
{
    Point pos = e.GetPosition(this);
}

void TimeLine_MouseDown(object sender, MouseButtonEventArgs e)
{
    if (e.ClickCount == 2)
    {
        DateTimeConfig cfg = new DateTimeConfig();
        cfg.FirstDate = this.FirstDay;
        cfg.LastDate  = this.LastDay;
        System.Windows.Forms.DialogResult res = cfg.ShowDialog();
        if (res == System.Windows.Forms.DialogResult.OK)
        {
            this.FirstDay = cfg.FirstDate;
            this.LastDay  = cfg.LastDate;
        }
    }
}
//
```

```
protected override int VisualChildrenCount
{
    get
    {
        return uilist.Count;
    }
}
//
protected override Visual GetVisualChild(int index)
{
    if (index < 0 || index > uilist.Count)
        throw new ArgumentOutOfRangeException("index");
    return uilist[index];
}
//
protected override void OnRender(DrawingContext drawingContext)
{
    base.OnRender(drawingContext);
    //Perform drawing
    switch (this.TimeUnit)
    {
        case DrawingMode.Days:
            this.DrawModeDays(drawingContext);
            break;
        case DrawingMode.Months:
            this.DrawModeMonths(drawingContext);
            break;
        case DrawingMode.Weeks:
            this.DrawModeWeeks(drawingContext);
            break;
        case DrawingMode.Year:
            this.DrawModeYear(drawingContext);
            break;
        default: break;
    }
    if (IsTimeSpanChanged)
    {
        IsTimeSpanChanged = false;
        if (TimeSpanChanged != null)
        {
            TimeSpanChanged(this, null);
        }
    }
}
//
#region DRAWDAYS
private void DrawModeDays(DrawingContext drawingContext)
{
    double txtSize = 12;
    //
    FormattedText mon =
          new FormattedText("Ma", cultureDK, flowToRight, txtArial, txtSize, Brushes.Black);
    FormattedText tue =
          new FormattedText("Ti", cultureDK, flowToRight, txtArial, txtSize, Brushes.Black);
    FormattedText wed =
          new FormattedText("On", cultureDK, flowToRight, txtArial, txtSize, Brushes.Black);
    FormattedText thr =
          new FormattedText("To", cultureDK, flowToRight, txtArial, txtSize, Brushes.Black);
    FormattedText fri =
          new FormattedText("Fr", cultureDK, flowToRight, txtArial, txtSize, Brushes.Black);
    FormattedText stu =
          new FormattedText("Lø", cultureDK, flowToRight, txtArial, txtSize, Brushes.Black);
    FormattedText sun =
          new FormattedText("Sø", cultureDK, flowToRight, txtArial, txtSize, Brushes.Black);
    List<FormattedText> dayNamesList = new List<FormattedText>(7);
    dayNamesList.Add(mon);
    dayNamesList.Add(tue);
    dayNamesList.Add(wed);
    dayNamesList.Add(thr);
    dayNamesList.Add(fri);
    dayNamesList.Add(stu);
    dayNamesList.Add(sun);
    //
    List<DayOfWeek> dayOfTheWeekList = new List<DayOfWeek>(7);
```

```
            dayOfTheWeekList.Add(DayOfWeek.Monday);
            dayOfTheWeekList.Add(DayOfWeek.Tuesday);
            dayOfTheWeekList.Add(DayOfWeek.Wednesday);
            dayOfTheWeekList.Add(DayOfWeek.Thursday);
            dayOfTheWeekList.Add(DayOfWeek.Friday);
            dayOfTheWeekList.Add(DayOfWeek.Saturday);
            dayOfTheWeekList.Add(DayOfWeek.Sunday);
            //
#if DEBUG
            DateTime start = DateTime.Now;
#endif
            // Build a list off all days for the specified period

            int daysCount = this.PTimeSpan.Days + 1;
            List<DateTime> li = new List<DateTime>(daysCount);
            for (int i = 0; i < daysCount; i++)
            {
                DateTime current = FirstDay.AddDays(i).Date;
                li.Add(current);
            }
            //
            int xoffset = 32;
            // Mark today
            Pen noPen = new Pen(Brushes.Transparent, 0);
            Point p = new Point(0, 20);
            int iToday = li.IndexOf(DateTime.Today.Date);
            if (iToday > 0)
            {
                xoffset = iToday << 5;
                p.X = xoffset;
                drawingContext.DrawRectangle(Brushes.LightGreen, noPen,
                                             new Rect(p, new Size(32, 20)));
            }
            // Get all saturdays
            var saturdays = from sat in li where sat.DayOfWeek == DayOfWeek.Saturday select sat;
            //
            Size ss = new Size(ItemWidth * 2, 20);
            // Mark all saturdays with LightYellow
            foreach (var item in saturdays)
            {
                int sindex = li.IndexOf(item);
                xoffset = sindex << 5;
                p.X = xoffset;
                drawingContext.DrawRectangle(Brushes.LightYellow, noPen, new Rect(p, ss));
            }
            // If the first item is sunday mark with LightYellow
            if (li[0].DayOfWeek == DayOfWeek.Sunday)
            {
                p.X = 0;
                ss = new Size(32, 20);
                drawingContext.DrawRectangle(Brushes.LightYellow, noPen, new Rect(p, ss));
            }
            // Draw border
            Point zero = new Point(0, 0);
            drawingContext.DrawRectangle(transparent, blackPen, new Rect(zero,
                                         new Point(Width, Height)));
            zero.Y += 20;
            drawingContext.DrawLine(blackPen, zero, new Point(Width, 20));
            //
            Point p1 = new Point(xoffset, 20);
            Point p2 = new Point(xoffset, 40);
            // draw first
            Point txtPoint = new Point(8, 24);
            int ifirst = dayOfTheWeekList.IndexOf(FirstDay.DayOfWeek);
            drawingContext.DrawText(dayNamesList[ifirst], txtPoint);
            txtPoint.X += ItemWidth;
            //
            for (int i = 1; i < daysCount; i++)
            {
                DayOfWeek dow = li[i].DayOfWeek;
                int index = dayOfTheWeekList.IndexOf(dow);
                // draw vertical lines
                xoffset = i << 5;
                p1.X = xoffset;
```

```
                p2.X = xoffset;
                drawingContext.DrawLine(blackPen, p1, p2);
                // draw day text
                drawingContext.DrawText(dayNamesList[index], txtPoint);
                txtPoint.X += ItemWidth;
            }
            // Get all mondays
            IEnumerable<DateTime> mondays = from g in li
                                            where g.DayOfWeek == DayOfWeek.Monday select g;
            Point ptxt = new Point(4, 4);
            Point plineA = new Point(0, 0);
            Point plineB = new Point(0, 20);
            List<DateTime> ld = new List<DateTime>(mondays);
            // test if first item on the list is monday
            int im = li.IndexOf(ld[0]);
            xoffset = im << 5;
            if (im != 0)
            {
                // draw vertical line only if the item is not the first item on the list
                plineA.X = xoffset;
                plineB.X = xoffset;
                drawingContext.DrawLine(blackPen, plineA, plineB);
            }
            ptxt.X = xoffset + 6;
            FormattedText txt = new FormattedText(ld[0].ToShortDateString(), cultureDK, flowToRight,
                                                  txtArial, 12, Brushes.Black);
            drawingContext.DrawText(txt, ptxt);
            // start drawing from the second item
            for (int i = 1; i < ld.Count; i++)
            {
                int index = li.IndexOf(ld[i]);
                xoffset = (index << 5);
                plineA.X = xoffset;
                plineB.X = xoffset;
                drawingContext.DrawLine(blackPen, plineA, plineB);
                ptxt.X = xoffset + 6;
                FormattedText formTxt = new FormattedText(ld[i].ToShortDateString(), cultureDK,
                            flowToRight, txtArial, 12, Brushes.Black);
                drawingContext.DrawText(formTxt, ptxt);
            }
#if DEBUG
            TimeSpan time;
            DateTime stop = DateTime.Now;
            time = stop - start;
            System.Diagnostics.Debug.WriteLine(String.Format("mS: {0}", time.Milliseconds));
#endif
        }
        #endregion
        //
        #region DRAWWEEKS
        private void DrawModeWeeks(DrawingContext dc)
        {
            // TODO Mark weekends and draw text for the first week
            int daysCount = this.PTimeSpan.Days + 1;
            List<DateTime> listOfDays = new List<DateTime>(daysCount);
            for (int i = 0; i < daysCount; i++)
            {
                DateTime current = FirstDay.AddDays(i).Date;
                listOfDays.Add(current);
            }
            IEnumerable<DateTime> mondays = from md in listOfDays
                                            where md.DayOfWeek == DayOfWeek.Monday select md;
            Point zero = new Point(0, 20);
            // Mark today
            int iToday = listOfDays.IndexOf(DateTime.Today.Date);
            double xoffset;
            if (iToday > 0)
            {
                xoffset = iToday << 4;
                zero.X = xoffset;
                dc.DrawRectangle(Brushes.LightGreen, new Pen(transparent, 0),
                                 new Rect(zero, new Size(ItemWidth, 20)));
            }
            // Draw border
```

```
        zero.X = 0;
        zero.Y = 0;
        dc.DrawRectangle(transparent, blackPen, new Rect(zero, new Point(Width, Height)));
        zero.Y = 20;
        dc.DrawLine(blackPen, zero, new Point(Width, 20));
        //
        Point pA = new Point(0, 0);
        Point pB = new Point(0, 40);
        Point ptxt = new Point(4, 4);
        int index;
        DateTimeHelper dth = new DateTimeHelper();
        foreach (var item in mondays)
        {
            index = listOfDays.IndexOf(item);
            pA.X = index << 4;
            pB.X = index << 4;
            dc.DrawLine(blackPen, pA, pB);
            if (index > 0)
            {
                int nr = dth.WeekNumber(item);
                string weeknr = String.Format("Uge {0}", nr);
                FormattedText txt = new FormattedText(weeknr, cultureDK, flowToRight,
                                                      txtArial, 12, blackBrush);
                ptxt.X = (index << 4) + 56 - (txt.Width / 2); // 56 = (ItemWidth * 7) / 2
                ptxt.Y = 10 - (txt.Height / 2);
                dc.DrawText(txt, ptxt);
                // draw week numbers
                txt = new FormattedText(listOfDays[index].ToShortDateString(), cultureDK,
                                        flowToRight, txtArial, 12, blackBrush);
                ptxt.X = (index << 4) + 4;
                ptxt.Y = 30 - (txt.Height / 2); ;
                //double dummy = txt.Width;
                dc.DrawText(txt, ptxt);
            }
        }
    }
    #endregion
    //
    #region DRAWMONTHS
    private void DrawModeMonths(DrawingContext dc)
    {
        if (PTimeSpan.Days < 30)
        {
            MessageBox.Show("DrawModeMonths::Timespan er for lille!", "Information",
                MessageBoxButton.OK, MessageBoxImage.Information);
            return;
        }
        int daysCount = this.PTimeSpan.Days + 1;
        List<DateTime> listOfDays = new List<DateTime>(daysCount);
        for (int i = 0; i < daysCount; i++)
        {
            DateTime current = FirstDay.AddDays(i).Date;
            listOfDays.Add(current);
        }
        IEnumerable<DateTime> foo = from md in listOfDays where md.Day == 1 select md;
        List<DateTime> bar = new List<DateTime>(foo);
        int index = listOfDays.IndexOf(bar[0]);
        Point zero = new Point(0, 0);
        // Mark today
        int iToday = listOfDays.IndexOf(DateTime.Today.Date);
        double xoffset;
        if (iToday > 0)
        {
            xoffset = iToday << 3;
            zero.X = xoffset;
            zero.Y = 20;
            dc.DrawRectangle(Brushes.LightGreen, new Pen(transparent, 0),
                             new Rect(zero, new Size(ItemWidth, 20)));
        }
        // Draw border
        zero.X = 0;
        zero.Y = 0;
        dc.DrawRectangle(transparent, blackPen, new Rect(zero, new Point(Width, Height)));
        zero.Y = 20;
```

```
dc.DrawLine(blackPen, zero, new Point(Width, 20));
// Draw months
Point point0 = new Point(0, 0);
Point point1 = new Point(0, 20);
FormattedText txt;
string str;
index = listOfDays.IndexOf(bar[0]);
if (index > 0)
{
    str = String.Format("{0} {1}", months[listOfDays[0].Month], listOfDays[0].Year);
    txt = new FormattedText(str, cultureDK, flowToRight, txtArial, 12, blackBrush);
    if (txt.Width < index * ItemWidth)
    {
        point0.X = ((index * ItemWidth) / 2) - (txt.Width / 2);
        point0.Y = 4;
        dc.DrawText(txt, point0);
    }
    else
    {
        txt = new FormattedText(str, cultureDK, flowToRight, txtArial, 10, blackBrush);
        if (txt.Width < index * ItemWidth)
        {
            point0.X = ((index * ItemWidth) / 2) - (txt.Width / 2);
            point0.Y = 6;
            dc.DrawText(txt, point0);
        }
    }
}
foreach (var item in bar)
{
    index = listOfDays.IndexOf(item);
    point0.X = index << 3;
    point1.X = index << 3;
    point0.Y = 0;
    dc.DrawLine(blackPen, point0, point1);
    str = String.Format("{0} {1}", months[item.Month], item.Year);
    txt = new FormattedText(str, cultureDK, flowToRight, txtArial, 12, blackBrush);
    point0.X += 120 - (txt.Width + 10) / 2;
    point0.Y += 4;
    dc.DrawText(txt, point0);
}
// draw weeks
IEnumerable<DateTime> mondays = from dl in listOfDays
              where dl.DayOfWeek == DayOfWeek.Monday select dl;
List<DateTime> mondayList = new List<DateTime>(mondays);
// get poistion of the first item on the mondays list
index = listOfDays.IndexOf(mondayList[0]);
// if there is enough space to draw first item text
if (index >= 5)
{
    // draw first item on the alldays list
    str = listOfDays[0].ToShortDateString().Substring(0, 5);
    txt = new FormattedText(str, cultureDK, flowToRight, txtArial, 12, blackBrush);
    point0.X = 4;
    point0.Y = 24;
    dc.DrawText(txt, point0);
}
else if (index >= 2)
{
    str = listOfDays[0].ToShortDateString().Substring(0, 5);
    txt = new FormattedText(str, cultureDK, flowToRight, txtArial, 10, blackBrush);
    point0.X = 4;
    point0.Y = 24;
    dc.DrawText(txt, point0);
}
// draw a vertical line to mark a new week
if (index != 0)
{
    xoffset = index << 3;
    point0.X = xoffset; point0.Y = 20;
    point1.X = xoffset; point1.Y = 40;
    dc.DrawLine(blackPen, point0, point1);
}
str = mondayList[0].ToShortDateString().Substring(0, 5);
```

```
            txt = new FormattedText(str, cultureDK, flowToRight, txtArial, 12, blackBrush);
            point0.X = (index << 3) + 4;
            point0.Y = 24;
            dc.DrawText(txt, point0);
            // draw others
            point1.Y = 40;
            for (int i = 1; i < mondayList.Count; i++)
            {
                index = listOfDays.IndexOf(mondayList[i]);
                xoffset = index << 3;
                point0.X = xoffset;
                point1.X = xoffset;
                point0.Y = 20;
                dc.DrawLine(blackPen, point0, point1);
                string s = mondayList[i].ToShortDateString().Substring(0, 5);
                txt = new FormattedText(s, cultureDK, flowToRight, txtArial, 12, blackBrush);
                point0.X = xoffset + 4;
                point0.Y += 4;
                dc.DrawText(txt, point0);
            }
        }
        #endregion
        //
        #region DRAWYEAR
        private void DrawModeYear(DrawingContext dc)
        {
#if DEBUG
            DateTime start = DateTime.Now;
#endif
            int daysCount = this.PTimeSpan.Days + 1;
            List<DateTime> AllDays = new List<DateTime>();
            for (int i = 0; i < daysCount; i++)
            {
                DateTime current = FirstDay.AddDays(i).Date;
                AllDays.Add(current);
            }
            IEnumerable<DateTime> FirstMonth = from mon in AllDays where mon.Day == 1 select mon;
            List<DateTime> AllFirstInTheMonth = new List<DateTime>(FirstMonth);
            //
            int index;
            double xoffset;
            string caption;
            FormattedText txt;
            Point point0 = new Point(0, 0);
            Point point1 = new Point(0, 40);
            // Mark today
            int iToday = AllDays.IndexOf(DateTime.Today.Date);
            if (iToday > 0)
            {
                xoffset = iToday * ItemWidth;
                point0.X = xoffset;
                point0.Y = 20;
                dc.DrawRectangle(Brushes.LightGreen, new Pen(transparent, 0),
                                 new Rect(point0, new Size(ItemWidth, 20)));
            }
            // Draw border
            point0.X = 0;
            point0.Y = 0;
            dc.DrawRectangle(transparent, blackPen, new Rect(point0, new Point(Width, Height)));
            point0.X = Width;
            point0.Y = 20;
            point1.Y = 20;
            dc.DrawLine(blackPen, point0, point1);
            // Draw first month
            if (AllDays[0] == AllFirstInTheMonth[0])
                index = 0; // Do nothing
            else
            {
                index = AllDays.IndexOf(AllFirstInTheMonth[0]);
                xoffset = index * ItemWidth;
                caption = String.Format("{0}", months[AllDays[0].Month]);
                double txtSize = 12;
                txt = new FormattedText(caption, cultureDK, flowToRight, txtArial, txtSize,
                                        blackBrush);
```

```
                while (xoffset < txt.Width)
                {
                    txtSize--;
                    txt = new FormattedText(caption, cultureDK, flowToRight, txtArial, txtSize,
                                            blackBrush);
                }
                point0.X = (xoffset / 2) - (txt.Width / 2);
                point0.Y = 24;
                dc.DrawText(txt, point0);
                // set year caption
                caption = String.Format("{0}", AllDays[0].Year);
                txt = new FormattedText(caption, cultureDK, flowToRight, txtArial, txtSize,
                                        blackBrush);
                point0.X = (xoffset / 2) - (txt.Width / 2);
                point0.Y = 4;
                dc.DrawText(txt, point0);
            }
            // Draw months
            point1.Y = 40;
            foreach (var item in AllFirstInTheMonth)
            {
                index = AllDays.IndexOf(item);
                xoffset = index * ItemWidth;
                point0.X = xoffset;
                point0.Y = 0;
                point1.X = xoffset;
                dc.DrawLine(blackPen, point0, point1);
                caption = months[item.Month];
                txt = new FormattedText(caption, cultureDK, flowToRight, txtArial, 12, blackBrush);
                point0.X = xoffset + 45 - (txt.Width / 2);
                point0.Y = 24;
                dc.DrawText(txt, point0);
                //
                caption = String.Format(" {0}", item.Year);
                txt = new FormattedText(caption, cultureDK, flowToRight, txtArial, 12, blackBrush);
                point0.X = xoffset + 45 - (txt.Width / 2);
                point0.Y = 4;
                dc.DrawText(txt, point0);
            }
#if DEBUG
            DateTime stop = DateTime.Now;
            TimeSpan ts = stop - start;
            System.Diagnostics.Debug.WriteLine(String.Format("DrawModeYear render: {0} mS",
                                               ts.Milliseconds));
#endif
        }
        #endregion
        //
        internal Point GetPoint(DateTime date)
        {
            int index = (date.Date - this.FirstDay.Date).Days;
            Point p = new Point(index * this.ItemWidth, 0);
            return p;
        }
        //
        internal double GetLength(int duration)
        {
            if (duration < 0)
                throw new ArgumentOutOfRangeException("duration");

            return duration * ItemWidth;
        }
    }

    public enum DrawingMode
    {
        Days, Weeks, Months, Year
    }
    #endregion
    /*************************************************************************/
    #region SPATIAL
    class SpatialLine : FrameworkElement
    {
    }
```

```
#endregion
/*************************************************************************/
#region CONTENT
class NContent : FrameworkElement
{
    private const double _thickness = 1;
    private List<Visual> visuals;
    private System.Globalization.CultureInfo cultureDK;
    private FlowDirection flowRight = FlowDirection.LeftToRight;
    private Typeface txtFace;
    private FormattedText txt;
    private Brush blackBrush;
    private Brush transparent;
    private Pen blackPen;
    private Pen dummyPen;

    public NContent()
    {
        visuals     = new List<Visual>();
        cultureDK   = new System.Globalization.CultureInfo("da-DK");
        txtFace     = new Typeface("Arial");
        blackBrush  = Brushes.Black;
        transparent = Brushes.Transparent;
        blackPen    = new Pen(blackBrush, 1);
        dummyPen    = new Pen(Brushes.Transparent, 0);
    }
    //
    public void DrawElement(Point p, Size s, Color c, string caption)
    {
        Brush itemColorBrush = new SolidColorBrush(c);
        //
        DrawingVisual dv = new DrawingVisual();
        using (DrawingContext dc = dv.RenderOpen())
        {
            Point ptxt = p;
            dc.DrawRoundedRectangle(itemColorBrush, dummyPen, new Rect(p, s), 4, 8);
            txt = new FormattedText(caption, cultureDK, flowRight, txtFace, 12, blackBrush);
            ptxt.X = p.X + s.Width + 10;
            ptxt.Y = ptxt.Y + (s.Height / 2) - (txt.Height / 2);
            dc.DrawText(txt, ptxt);
        }
        visuals.Add(dv);
        AddVisualChild(dv);
        AddLogicalChild(dv);
    }
    //
    public void RedrawElement(int index, Size s, Point p, Color c, string caption)
    {
        Brush itemColorBrush = new SolidColorBrush(c);
        DrawingVisual item   = visuals[index] as DrawingVisual;
        using (DrawingContext dc = item.RenderOpen())
        {
            Point ptxt = p;
            p.Y -= s.Height / 2;        // Y placement of the upper left corner
            dc.DrawRoundedRectangle(itemColorBrush, dummyPen, new Rect(p, s), 4, 8);
            txt = new FormattedText(caption, cultureDK, flowRight, txtFace, 12, blackBrush);
            ptxt.X = p.X + s.Width + 10;
            ptxt.Y -= txt.Height / 2;
            dc.DrawText(txt, ptxt);
        }
    }
    //
    public void InvalidateContent()
    {
        foreach (DrawingVisual item in visuals)
        {
            RemoveLogicalChild(item);
            RemoveVisualChild(item);
        }
        visuals.Clear();
    }
    //
    protected override int VisualChildrenCount
    {
```

```
            get
            {
                return visuals.Count;
            }
        }
        //
        protected override Visual GetVisualChild(int index)
        {
            if (index < 0 || index >= visuals.Count)
                throw new ArgumentOutOfRangeException("index");
            return visuals[index];
        }
        //
        public override string ToString()
        {
            return "ProkaApp.UI.NContent";
        }
        protected override void OnMouseMove(MouseEventArgs e)
        {
            base.OnMouseMove(e);
        }
    }
    #endregion
    #endregion
}
```

VandloebView.xaml

```
<Window x:Class="ProkaApp.VandloebView"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Vandløb oversigt" Height="300" Width="800"
    Loaded="OnInit">
    <Window.Resources>
        <DataTemplate x:Key="StationTemplate">
            <Grid ShowGridLines="False">
                <Grid.RowDefinitions>
                    <RowDefinition />
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="120"/>
                    <ColumnDefinition Width="120"/>
                    <ColumnDefinition Width="120"/>
                </Grid.ColumnDefinitions>
                <Border Grid.Column="0" BorderBrush="Black" BorderThickness="0, 0, 1, 1"
                        Margin="0, 3, 2, 0" Padding="2">
                    <TextBlock Text="{Binding Path = Station}" TextAlignment="Center" />
                </Border>
                <Border Grid.Column="1" BorderBrush="Black" BorderThickness="0, 0, 1, 1"
                         Margin="0, 3, 2, 0" Padding="2">
                    <TextBlock Text="{Binding Path = VersNavn}" TextAlignment="Center" />
                </Border>
                <Border Grid.Column="2" BorderBrush="Black" BorderThickness="0, 0, 1, 1"
                        Margin="0, 3, 0, 0" Padding="2">
                    <TextBlock Text="{Binding Path = TopoStationerId}" TextAlignment="Center" />
                </Border>
                <!---->
            </Grid>
        </DataTemplate>
        <DataTemplate x:Key="VlIdentTemplate">
            <TextBlock Text="{Binding Path = VlNavn}" />
        </DataTemplate>
        <DataTemplate x:Key="StationTemplateNoGrid">
            <TextBlock Text="{Binding Path = Station}" />
        </DataTemplate>
        <Style x:Key="listHeaderStyle" TargetType="Label">
            <Setter Property="Width" Value="120" />
            <Setter Property="HorizontalContentAlignment" Value="Center" />
            <Setter Property="Background" Value="AliceBlue" />
            <Setter Property="BorderBrush" Value="Black" />
            <Setter Property="BorderThickness" Value="0,0,1,0" />
        </Style>
```

```
    </Window.Resources>
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="*" />
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="200"/>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="Auto"/>
        </Grid.ColumnDefinitions>
        <StackPanel Grid.Row="0" Grid.Column="0" Background="AliceBlue" >
            <Label HorizontalAlignment="Center">Vandløb navn</Label>
        </StackPanel>
        <StackPanel Grid.Row="0" Grid.Column="1" Orientation="Horizontal" Background="AliceBlue">
            <Label Style="{StaticResource listHeaderStyle}" Margin="2, 0, 0, 0">Station</Label>
            <Label Style="{StaticResource listHeaderStyle}">VersNavn</Label>
            <Label Style="{StaticResource listHeaderStyle}">TopostationerID</Label>
        </StackPanel>
        <StackPanel Grid.Row="0" Grid.Column="2" Background="AliceBlue" />
        <ListBox Grid.Column ="0" Grid.Row="1" Name="ListVandloeb"
                 ItemsSource ="{Binding Path=VlIdent}"
                 ItemTemplate="{StaticResource VlIdentTemplate}"
                 SelectionChanged="ListVandloeb_SelectionChanged">
        </ListBox>
        <GridSplitter Grid.Row="1" Grid.Column="0" Width="2"/>
        <ListBox Grid.Column="1" Grid.Row="1" Name="ListStation"
                 ItemsSource ="{Binding Path=Topostationer}"
                 ItemTemplate="{StaticResource StationTemplate}"/>
        <GroupBox Grid.Row="1" Grid.Column="2" Header="Station data" Name="groupBox1"
                  Height="Auto" Width="Auto" BorderBrush="Black" BorderThickness="1"
                  FontFamily="Arial">
            <StackPanel Orientation="Vertical">
                <!---->
            </StackPanel>
        </GroupBox>
        <StatusBar Grid.Row="2" Grid.Column="0" Grid.ColumnSpan="3">
            <StackPanel Orientation="Horizontal">
                <TextBlock Text="Vandløb indlæst: " Margin="2, 0, 2, 0"/>
                <TextBlock Text="{Binding Items.Count, ElementName=ListVandloeb}"
                     Margin="0, 0, 20, 0" />
                <TextBlock Text="Antal stationer: " />
                <TextBlock Text="{Binding Items.Count, ElementName=ListStation}"
                     Margin="0, 0, 20, 0" />
            </StackPanel>
        </StatusBar>
    </Grid>
</Window>
```

VandloebView.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
//
using System.Data;

namespace ProkaApp
{
   /// <summary>
   /// Interaction logic for VandloebView.xaml
   /// </summary>
```

```
    public partial class VandloebView : Window
    {

        private DataSet _data;
        public  DataSet Data
        {
            get { return _data; }
            set
            {
                _data = value;
            }
        }

        public VandloebView()
        {
            InitializeComponent();
        }

        private void comboBox1_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            object o = e.AddedItems;
        }

          private void ListVandloeb_SelectionChanged(object sender, SelectionChangedEventArgs e)
          {
              string navn;
              System.Collections.IList objectlist = e.AddedItems;
              if (objectlist.Count == 1)
              {
                  object[] items = (objectlist[0] as DataRowView).Row.ItemArray;
                  navn = String.Format("{0}", items[1]);
                  (this.Tag as ProkaApp.BusinessLogic.AppDocument).HentStationer(navn);
              }
          }

          private void OnInit(object sender, RoutedEventArgs e)
          {
              ListVandloeb.DataContext = _data;
              ListStation.DataContext  = _data;
          }
    }
}
```